UNIVERSITY OF PADOVA

DOCTORAL THESIS

PH.D. COURSE IN INFORMATION ENGINEERING
INFORMATION AND COMMUNICATIONS TECHNOLOGY

Artificial Intelligence for Data Analysis and Signal Processing

---

Author:
Matteo GADALETA

Supervisor:
Prof. Michele ROSSI

Coordinator:
Prof. Andrea NEVIANI

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Engineering

in the

Department of Information Engineering

September 2018

UNIVERSITY OF PADOVA

Abstract

Department of Information Engineering

Artificial Intelligence for Data Analysis and Signal Processing

Matteo GADALETA

Artificial intelligence, or AI, currently encompasses a huge variety of fields, from areas such as logical reasoning and perception, to specific tasks such as game playing, language processing, theorem proving, and diagnosing diseases. It is clear that systems with human-level intelligence (or even better) would have a huge impact on our everyday lives and on the future course of evolution, as it is already happening in many ways. In this research AI techniques have been introduced and applied in several clinical and real world scenarios, with particular focus on deep learning methods. A human gait identification system based on the analysis of inertial signals has been developed, leading to misclassification rates smaller than 0.15%. Advanced deep learning architectures have been also investigated to tackle the problem of atrial fibrillation detection from short length and noisy electrocardiographic signals. The results show a clear improvement provided by representation learning over a knowledge-based approach. Another important clinical challenge, both for the patient and on-board automatic alarm systems, is to detect with reasonable advance the patterns leading to risky situations, allowing the patient to take therapeutic decisions on the basis of future instead of current information. This problem has been specifically addressed for the prediction of critical hypo/hyperglycemic episodes from continuous glucose monitoring devices, carrying out a comparative analysis among the most successful methods for glucose event prediction. This dissertation also shows evidence of the benefits of learning algorithms for vehicular traffic anomaly detection, through the use of a statistical Bayesian framework, and for the optimization of video streaming user experience, implementing an intelligent adaptation engine for video streaming clients. The proposed solution explores the promising field of deep learning methods integrated with reinforcement learning schema, showing its benefits against other state of the

art approaches. The great knowledge transfer capability of artificial intelligence methods and the benefits of representation learning systems stand out from this research, representing the common thread among all the presented research fields.

UNIVERSITÀ DI PADOVA

Sommario in lingua italiana

Dipartimento di Ingegneria dell'Informazione

Artificial Intelligence for Data Analysis and Signal Processing

Matteo GADALETA

L'intelligenza artificiale, o AI, comprende attualmente una grande varietà di campi, da concetti come il ragionamento logico e la percezione, a obiettivi specifici come la risoluzione di giochi complessi, l'elaborazione del linguaggio, la dimostrazione di teoremi matematici e la diagnosi di malattie. È chiaro che sistemi con un'intelligenza a livello umano (o anche superiore) avrebbero un enorme impatto sulla nostra vita quotidiana e sul futuro corso dell'evoluzione, come si sta già osservando sotto numerosi aspetti. In questa tesi vengono proposte tecniche di AI applicate a diversi scenari, sia clinici che reali, con particolare interesse a metodi di *deep learning*. In particolare, è stato sviluppato un sistema di identificazione dell'andatura umana basato sull'analisi di segnali inerziali, che porta a errori di classificazione inferiori allo 0,15%. Sono state inoltre studiate avanzate architetture di deep learning per affrontare il problema della rilevazione della fibrillazione atriale, analizzando segnali elettrocardiografici rumorosi e di breve durata. I risultati mostrano un netto miglioramento fornito da metodi di *representation learning* rispetto ad un approccio *knowledge-based*. Un'altra importante sfida clinica, sia per il paziente che per i sistemi automatici di avvertimento, è quella di rilevare con ragionevole anticipo i segnali che portano a situazioni rischiose, consentendo al paziente di prendere decisioni terapeutiche sulla base di informazioni future. Questo problema è stato specificamente contestualizzato nella previsione di episodi ipo/iperglicemici critici, analizzando segnali acquisiti da dispositivi di monitoraggio continuo del glucosio ed effettuando un'analisi comparativa tra i metodi di maggior successo per la previsione di eventi glicemici. In questa tesi vengono inoltre mostrati i vantaggi degli algoritmi di apprendimento automatico per il rilevamento di anomalie nel traffico veicolare, attraverso l'uso di un approccio bayesiano, e per l'ottimizzazione dell'esperienza utente durante la visione di un flusso video, implementando

un motore di adattamento intelligente per i client di video streaming. La soluzione proposta esplora la promettente integrazione tra metodi di deep learning e schemi di apprendimento con rinforzo, mostrando i suoi vantaggi rispetto ad altri approcci allo stato dell'arte. La grande capacità di trasferimento delle conoscenze dei metodi di intelligenza artificiale, e i vantaggi dei sistemi di apprendimento automatico, evincono da questa ricerca, rappresentando il filo conduttore tra i campi di ricerca presentati.

Acknowledgements

Undertaking this Ph.D. has been a truly learning and enriching experience for me and it would not have been possible without the support and guidance that I received from many people.

First and foremost, I would like to thank my research advisor Prof. Michele Rossi for introducing me to this exciting field of science and for his dedicated help, advice, inspiration, encouragement and continuous support throughout my Ph.D.

I wish to extend my sincere gratitude to all the members of the Signet group and all the people at the University of Padova who supported me and contributed to this work, with special regards to Enrico Grisan and Andrea Facchinetti from the Bioengineering Group.

I gratefully acknowledge The Scripps Research Translational Institute, for providing me financial support and Director of Digital Medicine Steven Steinhubl, to allow me to carryout part of my research in this esteemed institution. I also thank all the people who welcomed me and made me feel at home during this period.

A special thank goes to Giorgio Quer for his invaluable input, support and friendship throughout the last year of this journey, and for his help and assistance during the period in the US, both personal and professional.

Contents

List of Abbreviations

Adam	**Ada**ptive **M**oment Estimation
AF	**A**trial **F**ibrillation
AI	**A**rtificial **I**ntelligence
ARIMA	**A**uto**R**egressive **I**ntegrated **M**oving **A**verage
BG	**B**lood **G**lucose
BN	**B**ayesian **N**etwork
BTT	**B**ackpropagation **T**hrough **T**ime
BVI	**B**ayesian **V**ariational **I**nference
CDF	**C**umulative **D**istribution **F**unction
CDN	**C**ontent **D**elivery **N**etwork
CHO	Carbohydrates (**C**arbon **H**ydrogen **O**xygen)
CNN	**C**onvolutional **N**eural **N**etwork
CGM	**C**ontinuous **G**lucose **M**onitoring
CRC	**C**linical **R**esearch **C**enter
CT	**C**lassification **T**ree
DAG	**D**irected **A**cyclic **G**raphs
DASH	**D**ynamic **A**daptive **S**treaming over **H**TTP
DL	**D**eep **L**earning
DQN	**D**eep **Q**-**N**etwork
DTW	**D**ynamic **T**ime **W**arping
DWT	**D**iscrete **W**avelet **T**ransform
ECG	**E**lectro**C**ardio**G**raphy
FESTIVE	**F**air, **E**fficient, and **S**table adap**TIVE** algorithm
FIR	**F**inite **I**mpulse **R**esponse
FN	**F**alse **N**egative
FP	**F**alse **P**ositive
GI	**G**lycemic **I**ndex
GMM	**G**aussian **M**ixture **M**odel
HMM	**H**idden **M**arkov **M**odel
MDP	**M**arkov **D**ecision **P**rocess
MMSE	**M**inimum **M**ean **S**quare **E**rror
MPC	**M**odel **P**redictive **C**ontrol
MPD	**M**edia **P**resentation **D**escription
MSS	**M**icrosoft **S**mooth **S**treaming
IDNet	**ID**entification **Net**work

IoT	**I**nternet **o**f **T**hings
IQR	**I**nterquartile **R**ange
***k*-NN**	***k*-N**earest **N**eighbours
LDA	**L**inear **D**iscriminant **A**nalysis
LOO	**L**eave **O**ne **O**ut
LSTM	**L**ong-**S**hort **T**erm **M**emory
MLP	**M**ulti **L**ayer **P**erceptron
NB	**N**aive **B**ayes
OCC	**O**ne-**C**lass **C**lassification
OSVM	**O**ne-class **S**upport **V**ector **M**achine
PANDA	**P**robe **AND** **A**dapt
PCA	**P**rincipal **C**omponent **A**nalysis
PDF	**P**robability **D**ensity **F**unction
PPG	**P**hoto**P**lethysmo**G**raphy
PRC	**P**recision-**R**ecall **C**urve
QoE	**Q**uality **o**f **E**xperience
RAHAS	**R**ate **A**daptation **H**euristic for **A**daptive video **S**treaming
RBF	**R**adial **B**asis **F**unction
ReLU	**Re**ctified **L**inear **U**nit
RF	**R**andom **F**orest
RMSE	**R**oot **M**ean **S**quare **E**rror
RNN	**R**ecurrent **N**eural **N**etwork
ROC	**R**eceiver **O**perating **C**haracteristic
RL	**R**einforcement **L**earning
SAECG	**S**ignal **A**veraged **ECG**
SD	**S**uccessive **D**ifferences
SMBG	**S**elf-**M**onitoring **B**lood **G**lucose
SPRT	**S**equential **P**robability **R**atio **T**est
SSGPE	**S**um of **S**quares of the **G**lucose **P**rediction **E**rror
SSIM	**S**tructural **SIM**ilarity
SV	**S**upport **V**ector
SVM	**S**upport **V**ector **M**achine
SVR	**S**upport **V**ector **R**egression
T1D	**T**ype-**1** **D**iabetes
TOD	**T**emporal **O**utlier **D**iscovery
TN	**T**rue **N**egative
TP	**T**rue **P**ositive
TPR	**T**rue **P**ositive **R**ate
VI	**V**alue **I**teration

Introduction

The term "artificial intelligence" (AI) was first coined by John McCarthy more than five decades ago. Since then, there have been countless debates and discussions for a universally accepted definition of the term, perhaps because of its reference to the word "intelligence", which is itself an abstract quantity. A good definition of AI can be found in [1], where the author defines it as the *simulation of human intelligence on a machine, so as to make the machine efficient to identify and use the right piece of knowledge at a given step of solving a problem*. Thus, an intelligent system is a system capable of thinking and acting *rationally*. A system can act rationally only after acquiring adequate knowledge from the real world. Hence, a prerequisite for rational actions lies on the capacity of building up knowledge from real world information, ability known as *perception*. The learning capability of a machine is then strictly related to its perception.

The subject of AI spans a wide horizon. It has been enriched with studies from numerous disciplines, from psychology, philosophy, computer science, cognitive science, engineering and mathematics. AI deals with various fields of interest, such as knowledge representation schemes, methods for resolving uncertainty of data, techniques for intelligent search, schemes for automated machine learning and many others. Artificial intelligence has already shown its decision-making skills in the recognition and interpretation of patterns in many different fields. Among the application areas of AI, we have image recognition, natural language processing, expert systems, game-playing, theorem-proving, robotics and many others. In this study the focus is on the emerging area of AI for signal processing, with particular emphasis on deep learning techniques and biomedical applications. The adoption of AI in future clinical practice has the potential to be transformational.

This dissertation is organized in two parts. In the first part (Chapters 1–3) some human sensing applications are explored, including several sources of data: motion data from inertial sensor, electrocardiographic signals and glucose monitoring readings. The second part (Chapters 4–5) addresses two

real-world scenarios, focusing on a vehicular traffic anomaly detection system, and the optimization of video streaming user experience. Learning based artificial intelligence techniques represent the common thread among these topics, and connect them from several points of view, also emphasizing the great knowledge transfer capability of artificial intelligence methods.

In chapter 1 a user identification framework from smartphone-acquired motion signals is presented. The main goal is to recognize a target user from their way of walking, using the accelerometer and gyroscope signals provided by a commercial smartphone worn in the front pocket of the user's trousers. The framework features several innovations including: *i)* a robust and orientation-independent walking cycle extraction block (Sec. 1.2), *ii)* a novel feature extractor based on convolutional neural networks (Sec. 1.3), *iii)* a one-class support vector machine to classify walking cycles (Sec. 1.4), and the coherent integration of these into *iv)* a multi-stage authentication technique. The algorithm exploits a deep learning approach as universal feature extractors for gait recognition, and combines classification results from subsequent walking cycles into a multi-stage decision making framework (Sec. 1.5). Experimental results and a comparison of this approach against state of the art techniques are shown in Sec. 1.5.1.

Chapter 2 focuses on the automatic classification of atrial fibrillation (AF) events in *short* and *noisy* single-lead electrocardiographic signals (ECG) acquired from wireless sensors. AF is the most common serious abnormal heart rhythm, affecting about 34 million people in the world, it is often asymptomatic, so it may not be diagnosed and the affected subjects may be unaware of their condition. Since approximately 10 to 20% of ischemic strokes are associated with AF first diagnosed at the time of stroke, the early detection of asymptomatic AF is very important, as it can help prevent strokes by instituting appropriate preventive anticoagulation. For the considered AF detection task, five different classification architectures based on deep convolutional neural networks have been analyzed, following the most recent developments in the deep learning (DL) field (Sec. 2.1.4). These designs, which were originally proposed for the analysis of images, are here adapted to one dimensional ECG signals, and their classification performance is compared with that of a state of the art classifier using expert features (Sec. 2.1.3).

This chapter is organized as follows. Sec. 2.1 introduces the dataset used for the study, along with technical details on data processing, feature extraction and training procedure. In Sec. 2.2 the performance metrics used for the analysis are defined and explained. In Sec. 2.3 the obtained results are shown and some final remarks are highlighted in Sec. 2.4.

One of the most important objective of any diabetes therapy is to maintain the blood glucose concentration within the euglycemic range, avoiding or at least mitigating critical hypo/hyperglycemic episodes. Modern Continuous Glucose Monitoring (CGM) devices bear the promise of providing the patients with an increased and timely awareness of glycemic conditions as these get dangerously near to hypo/hyperglycemia. However, the challenge, both for the patient and on-board automatic alarm systems, is to detect with reasonable advance the patterns leading to risky situations, allowing the patient to take therapeutic decisions on the basis of future (predicted) instead of current glucose concentration. In the last years, several real-time short-term prediction algorithms such a scope have been developed. Nevertheless, despite some of them have shown to be potentially effective, current CGM sensors still do not embed any predictive alert feature. One of the main reason is that a technically sound performance comparison of these algorithms on the same dataset is still missing, which could allow to better evidence their pros and cons. This issue is addressed in Chapter 3. The aim of this study is to fill the aforementioned gap, by carrying out a comparative analysis among the most common methods for glucose event prediction. Both regression (Sec. 3.1.1) and classification (Sec. 3.1.2) algorithms have been implemented and analyzed, including static and dynamic training approaches (Sec. 3.1.4). The dataset consists of 89 CGM time series measured in diabetic subjects for 7 subsequent days. Performance metrics, specifically defined to assess and compare the event prediction capabilities of the methods, have been introduced in Sec. 3.1.3 and analyzed in Sec. 3.2.

Chapter 4, unlike the previous chapters which are more focused on human sensing applications, concerns the automated and runtime analysis of vehicular data from large scale traffic monitoring networks. This problem is tackled through localized and small-size Bayesian networks (BNs), which are utilized to capture the spatio-temporal relationships underpinning traffic

data from nearby road links. A dedicated BN is set up, trained, and tested for each road in the monitored geographical map. The joint probability distribution between the cause nodes and the effect node in the BN is tracked through a Gaussian Mixture Model (GMM), whose parameters are estimated via Bayesian variational inference operating on unlabeled data. Forecasting and anomaly detection are performed on statistical measures derived at runtime by the trained GMMs. The design choices lead to several advantages: the approach is scalable as a small-size BN is associated with and independently trained for each road and the localized nature of the framework allows flagging atypical behaviors at their point of origin in the monitored geographical map. Technical details are presented in Sec. 4.2, The effectiveness of the proposed framework is tested using a large dataset from a real network deployment, comparing its prediction performance with that of selected regression algorithms from the literature, while also quantifying its anomaly detection capabilities (Sec. 4.3).

Chapter 5 covers the topic of reinforcement learning, and its promising integration with deep learning techniques. The ever-increasing demand for seamless high-definition video streaming, along with the widespread adoption of the Dynamic Adaptive Streaming over HTTP (DASH) standard, has been a major driver of the large amount of research on bitrate adaptation algorithms. The complexity and variability of the video content and of the mobile wireless channel make this an ideal application for learning approaches. In this study, the D-DASH framework is presented, which combines deep learning and reinforcement learning techniques to optimize the Quality of Experience (QoE) of DASH (Sec. 5.2 and Sec. 5.3). Different learning architectures are proposed and assessed, combining feed-forward and recurrent deep neural networks with advanced strategies (Sec. 5.4 and Sec. 5.5). D-DASH designs are thoroughly evaluated against prominent algorithms from the state of the art, both heuristic and learning-based, evaluating performance indicators such as image quality across video segments and freezing/rebuffering events (Sec. 5.6).

Chapter 1

Human Gait Identification with Inertial Sensors

Wearable technology is advancing at a very fast pace. Many wearable devices, such as smart watches and wristbands are currently available in the consumer market and they often possess miniaturized inertial motion sensors (accelerometer and gyroscope) as well as other sensing hardware capable of gathering biological signs such as photoplethysmographic signals, skin temperature and so forth. Other wearables, such as commercial physiological monitors, deliver a number of vitals via their wireless interfaces, including electrocardiogram, heart rate, chest motion, etc. The same holds true for recent smartphones, that allow for the collection of user's feedback and for the realtime assessment of their health condition. They also feature sophisticated sensing technology, among which inertial sensors have been considered. With sensing technology growing rapidly, two major problems are related to the analysis of wearable data and to the identification of the mobile users who provide it, so that it can be assessed with reasonably high accuracy whether the data sources are genuine. Notably, certifying the data sources is a necessary step toward the widespread use of this technology in the medical field and, in this chapter, it is developed the user identification technology required to make this possible. A great deal of work has been carried out on gait recognition in the last decade [2]. In general, biometric gait recognition can be grouped into three main categories: 1) computer vision based, 2) floor sensor based and 3) wearable sensor based [3]. Most of the recent work belongs to the first category, where image and video analysis are performed to infer the user identity [4–8]. Nevertheless, user identification from wearables is a sensible approach in those scenarios where the deployment of cameras in not possible.

In this chapter the IDNet (IDentification Network) framework, a new system for the identification of mobile users from smartphone-acquired motion data, is proposed and detailed. As shown in [9, 10], modern phones possess highly accurate inertial sensors, which allow for non-obtrusive gait biometrics. IDNet leverages deep Convolutional Neural Networks (CNN) [11] and tools from machine learning, such as Support Vector Machines (SVM) [12], combining them in an innovative fashion. Specifically, IDNet encompasses algorithms for 1) walking cycle segmentation, 2) feature extraction and, finally, 3) user identification. CNNs are used as *universal* feature extractors to discriminate gait signatures from different subjects. Single- as well as multi-stage classifiers are finally combined with CNNs to identify the user through the accumulation of scores from subsequent walking cycles. As shown in Sec. 1.3, this solution recognize the target user with high accuracy and outperforms state of the art techniques such as [13–18]. Some key points of the proposed approach are:

- The design and validation of an original preprocessing techniques that includes: a robust algorithm for the extraction of walking cycles and an original transformation to move smartphone acquired motion signals into an orientation invariant reference system. Subsequent processing is carried out within this reference system, as this considerably improves results, see Sec. 1.2. As opposed to making motion data orientation independent, previous studies either use data acquired from a sensor in a known and fixed position [16, 17, 19, 13, 14, 20–22], or use orientation independent features at the cost of losing information about the direction of the forces [23].

- The design of a new CNN-based feature extraction tool, which is trained only once on a representative set of users and then used at runtime as a *universal* feature extractor, see Sec. 1.3. Note that with CNNs, statistical features are automatically extracted as part of the CNN training phase (automatic feature engineering) as opposed to the selection of predefined and often arbitrary features, as commonly done in the literature [15, 16, 19, 14].

- The combination of CNN-extracted features with a one-class SVM (OSVM) classifier [24], which is solely trained on the target subject, see Sec. 1.4. The resulting SVM scores are then accumulated across multiple walking

cycles to get higher accuracies, through a new multi-stage identification framework, see Sec. 1.5.

- The coherent integration of these techniques into the IDNet framework, that uses smartphone-acquired accelerometer and gyroscope motion data. The integration of gyroscope data provides further performance improvements, as shown in Sec. 1.3.2.

- The experimental validation of IDNet, proving its superiority against solutions from the state of the art, see Sec. 1.3, and achieving identification errors below 0.15%, see Sec. 1.5.

Part of the results presented in this chapter has been published in [25].

1.1 Related Work

Interest in gait analysis began in the 60's, when walking patterns from healthy people, termed as normal patterns, were investigated by Murray et al. [26]. These measurements were performed through the analysis of photos acquired using interrupted light photography. Murray compared normal gait parameters against those from pathologic gaits [27] and showed that gait is unique to each individual. Since then, human identification through gait recognition has been enjoying a growing interest. Most recent works are based on computer vision [28, 3]. Currently, multi-view gait recognition problem and condition invariance (e.g., clothing or carried items, walking speed, view angle, etc.) are of special interest [8]. Many new approaches have been studied to improve recognition performance, such as 3D body estimation [5], complete canonical correlation analysis [6], sparse coding and hypergraph learning [7]. However, mobile devices are becoming increasingly sophisticated and can provide high quality inertial measurements. Multiple activities can thus be analyzed using wearable sensors data, and exploited, e.g., for task identification [29]. A thorough review of the latest developments in this area can be found in Sprager's work [2].

The interest in this study is in human gait identification through smartphone inertial sensors. Ailisto et al. [30] were the first to look at this problem and they did it through accelerometer data. In their paper, they used a triaxial accelerometer worn on a belt with fixed orientation: the x-axis pointed forward, the y-axis to the left and the z-axis was aligned with the direction of

gravity. Only data points from the x and z axes were used for identification purposes. Gait cycle extraction was performed through a simple peak detection method, and a template was built for each subject. User identification employed a template matching technique, for which different methods were explored: temporal correlation, frequency domain analysis and data distribution statistics.

In [31], Derawi et al. proposed more robust preprocessing, cycle detection and template comparison algorithms. Data were acquired using a mobile phone worn on the hip, and only the vertical z-axis was considered for motion analysis. Dynamic Time Warping (DTW) [32] was used as the distance measure, to ensure robustness against non-linear temporal shifts. This scheme was also tested in [21], where majority voting and cyclic rotation were compared as inference rules. In a further study [22], Hidden Markov Models (HMM) were explored. Accelerometer data were split into windows of fixed length, which were then utilized to train the HMMs. Good identification results were obtained, but at the cost of long measurement phases (30 seconds).

Classification algorithms based on machine learning were also investigated. Either gait cycles extraction [33] or fixed windows lengths [14] are possible signal segmentation methods. After that, a *feature extraction* technique is applied to each segment and statistical measures such as mean, standard deviation, root mean square, zero-crossing rate or histogram bin counts, are commonly used. However, more advanced features are required for better results, like cepstral coefficients, which are widely used for speech recognition [14], or features extracted through frequency analysis, i.e., using Fourier [13] or wavelet transforms [33]. Supervised algorithms are typically used for classification, including k-Nearest Neighbours (k-NN) [14, 16, 18, 19], Support Vector Machines (SVM) [15, 19, 33], Multi Layer Perceptrons (MLP) [15, 19] and Classification Trees (CT) [15, 19].

Accelerometer based gait analysis has also interest in the medical field. Using time-frequency analysis, Huang et al. showed that signals acquired by a waist-worn device on a patient with cervical disc herniation differed before and after the surgery [20]. In [19], classification algorithms were used to discriminate a group of subjects with non-specific chronic low back pain from healthy subjects. Complex parameters, e.g., dynamic symmetry and cyclic

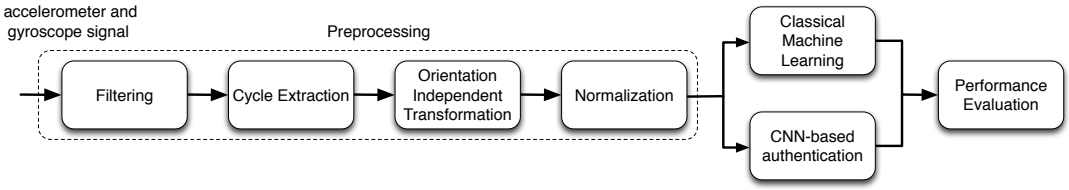


FIG. 1.1: Signal processing workflow.

stability of gait, were extracted by Jiang et al. [34]. However, their evaluation requires to place sensors on the legs, and fine gait details are difficult to extract from signals acquired by a single waist-worn sensor.

In most of the related work the acquisition system was placed according to a *controlled and well known orientation* on the subject body. In real scenarios, this is however unlikely to be the case. It is thus important to implement an algorithm whose performance is invariant to the smartphone orientation, which is somewhat unconstrained (and unknown). This makes the phone reference system with good probability misaligned with respect to the direction of motion and the definition of subject specific and time invariant templates impossible. To deal with this, two different approaches can be used. The first consists in the extraction of features that are rotation invariant (e.g., correlation matrices of Fourier transforms [23] or gait dynamic images [35]). The second relies on the transformation of inertial signals [15], projecting them into a new *orientation invariant* three-dimensional reference system, which is extracted directly from the data. Here, the latter approach has been adopted. Another distinctive feature of the presented work is the use of an original processing pipeline exploiting automatic feature extraction through CNNs and a scoring algorithm combining OSVM and multi-step decision analysis.

1.2 Signal Processing Framework

The aim of IDNet is to correctly recognize a subject from his/her way of walking, through the acquisition of inertial signals from a standard smartphone. The proposed processing workflow is shown in Fig. 1.1. Walking data is first acquired, then some preprocessing is performed entailing:

1. pre-filtering to remove motion artifacts (Sec. 1.2.1),

2. the extraction of walking cycles (Sec. 1.2.2),

3. a transformation to move the raw walking data into a new *orientation independent* reference system (Sec. 1.2.3),

4. a normalization to represent each walking cycle (accelerometer and gyroscope data) through fixed length, zero mean and unit variance vectors (Sec. 1.2.4).

After this, the walking cycles are ready to be processed to identify the user. The standard approach, called "Classical Machine Learning" entails the computation of a number of pre-established statistical features, the most informative of which are selected and used to train a classifier. Various machine learning techniques are usually exploited to this purpose, and are trained through a supervised approach. Hence, the classification performance is assessed and the whole process is usually iterated for a further feature selection phase. In this way, the features that are used for the classification task are progressively refined until a final feature set is attained. Note that statistical features are often assessed by the designer through educated guesses and a trial and error approach.

As opposed to this, IDNet advocates the use of convolutional neural networks (see Sections 1.3 and 1.4). These have been successfully used by the video processing community [36] but they have been rarely exploited for the analysis of inertial data from wearable devices. One of the main advantages of this approach is that statistical features are automatically assessed by the CNN as a result of a supervised training phase. In Sec. 1.3, the CNN is trained to act as a universal feature extractor, whereas in Sec. 1.4 a OSVM is trained as the final classifier. Once the CNN is trained, the proposed system operates assuming that the smartphone only has access to the walking patterns of the target user (i.e., the legitimate user) and the SVM is solely trained using his/her walking data. The system is based on the premise that, at runtime, the CNN should be capable of producing discriminant features for unseen users and the OSVM, once trained on the target, should reliably detect impostors, although their walks were not used for training. The processing blocks are described in higher details in the following subsections.

Notation: $\boldsymbol{x} \in \mathbb{R}^n$ represents a column vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T$ with elements $x_i \in \mathbb{R}$, where $(\cdot)^T$ is the transpose operator. $|\boldsymbol{x}| = n$ returns the number of elements in vector $\boldsymbol{x}$. $\overline{x} = (\sum_{i=1}^{n} x_i)/n$, whereas $\|\boldsymbol{x}\| = (\sum_{i=1}^{n} x_i^2)^{1/2}$ is the L2-norm operator. If $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, their inner product is defined as $\boldsymbol{x} \cdot \boldsymbol{y} =$

$\boldsymbol{x}^T\boldsymbol{y}$ and their entrywise product as $\boldsymbol{x} \circ \boldsymbol{y} = (x_1y_1, x_2y_2, \ldots, x_ny_n)^T$. Vector $\mathbf{1}_n = (1, 1, \ldots, 1)^T$ with $|\mathbf{1}_n| = n$. Matrices are denoted by uppercase and bold letters. For example, if $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{R}^n$, a $3 \times n$ matrix is defined as $\boldsymbol{M} = [\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}]^T$, whose rows contain the three vectors. In addition, element (i, j) of matrix $\boldsymbol{X}$ is denoted by $X_{i,j} \in \mathbb{R}$. $\vec{\boldsymbol{r}}$ represents a 3D vector $\vec{\boldsymbol{r}} = (r_1, r_2, r_3)^T$ and $\hat{\boldsymbol{r}}$ is the corresponding 3D versor $\hat{\boldsymbol{r}} = \vec{\boldsymbol{r}}/\|\vec{\boldsymbol{r}}\|$. For any two 3D vectors $\vec{\boldsymbol{r}}$ and $\vec{\boldsymbol{s}}$, their cross-product is indicated as $\vec{\boldsymbol{r}} \times \vec{\boldsymbol{s}} = (r_2s_3 - r_3s_2, r_3s_1 - r_1s_3, r_1s_2 - r_2s_1)^T$. The gravity vector is referred to as $\vec{\boldsymbol{\rho}}$. $u(i)$ represents a time series, where $i = 1, 2, \ldots$ is the discrete time index. For acceleration data, the boldface letter $\boldsymbol{a}$ is reserved for vectors, $a(i)$ for time series and $\boldsymbol{A}$ for matrices. The same notation is adopted for the gyroscope data, using $\boldsymbol{g}$, $g(i)$ and $\boldsymbol{G}$, respectively for vectors, time series and matrices. Finally, for any time series $a(i)$ and $N_\mathrm{s} \geq 1$, it is defined $\boldsymbol{a}(i, N_\mathrm{s}) = \texttt{vrect}(a, i, N_\mathrm{s}) \triangleq (a(i), \ldots, a(i + N_\mathrm{s} - 1))^T$, where $\boldsymbol{a}(i, N_\mathrm{s})$ is a column vector containing samples $a(i), \ldots, a(i + N_\mathrm{s} - 1)$.

1.2.1 Data Acquisition and Filtering

A proper dataset is key to the successful design and testing of identity recognition algorithms. Some datasets are publicly available. The largest one was acquired by the Institute of Scientific and Industrial Research (ISIR) at Osaka University (OU) [37]. It contains motion data collected from 744 subjects using four motion sensors: three inertial sensors were placed on a belt, with triaxial accelerometer and gyroscope, and a smartphone was worn in the center back waist, and only measured triaxial accelerometer data. Despite the high number of participants, the main problem with this dataset is that motion data was acquired in a controlled environment, and for each subject only two short data sequences are available, which are not enough for deep network training. Furthermore, smartphone's gyroscope data is not provided. Other datasets are available, but featuring a much smaller number of participants. Casale et al. collected accelerometer data from a smartphone positioned in the chest pocket from 22 users walking over a predefined path [38]. In [39], a motion capture suit was used to acquire data from 40 subjects walking in a small area at different speeds and with direction changes. However, due to the acquisition environment and conditions, these data are more suitable for human gait modeling rather than for user identification. Frank et al. collected data from a mobile phone in the pocket of 20 individuals at McGill University, performing two separate 15 minute walks on two different days [40].

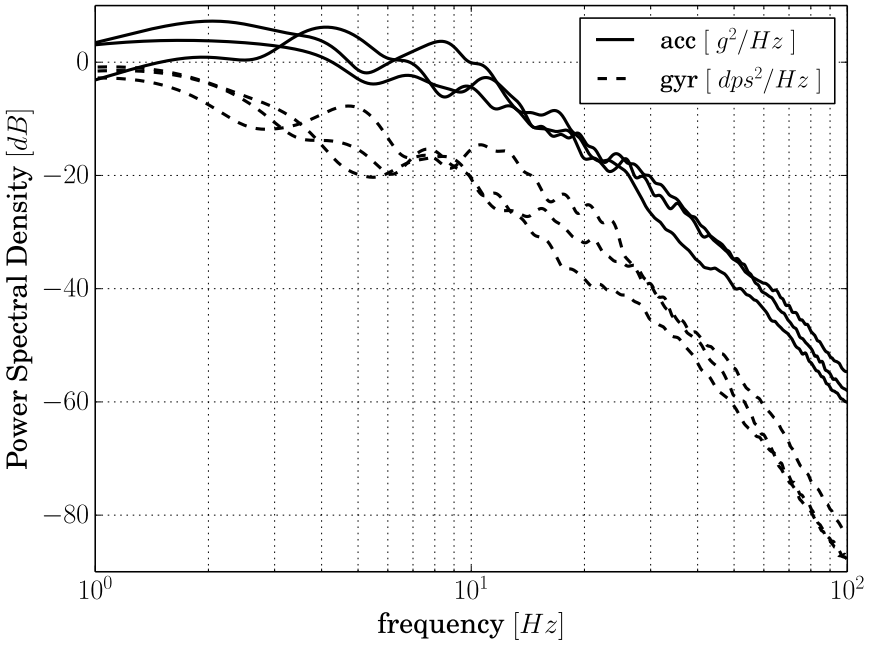


FIG. 1.2: Power spectral density of accelerometer (continuous lines, one for each axis) and gyroscope (dashed lines) data.

Also in this case gyroscope data is not provided. All these databases do not meet the requirements. In fact, for a proper training long data collection phases are necessary, preferably from different days and with devices freely worn in the user's front pockets. Hence, a dedicated data acquisition phase has been performed specifically for the purpose of this study.

In particular, motion traces from 50 subjects have been acquired, during a period of six months using Android smartphones worn in the right front pocket of the users' trousers. The following devices were used: Asus Zenfone 2, Samsung S3 Neo, Samsung S4, LG G2, LG G4 and a Google Nexus 5. Several acquisition sessions of about five minutes were performed for each subject, in variable conditions, e.g., with different shoes and clothes. Each subject has been asked to walk as she/he felt comfortable with, to mimic real world scenarios. For the data acquisition, an Android inertial data logger application has been developed, which saves accelerometer, gyroscope and magnetometer signals into non-volatile memory and then automatically transfers them to an Internet server for further processing. The magnetometer signal is not used for identification purposes. In general, IDNet can be used carrying the device in other positions but each requires a dedicated training.

In Fig. 1.2, the power of accelerometer and gyroscope signals at different frequencies is shown through the Welch's method [41], considering a full

walking trace and setting the Hanning window length to 1 s, with half window overlap. Most of the signal power is located at low frequencies, mostly below 40 Hz (where the power is at least 30 dB smaller than the maximum). The raw inertial signals were acquired using an average sample frequency ranging between 100 and 200 Hz (depending on the smartphone model), which is more than appropriate to capture most of the walking signal's energy.

At the first block of IDNet processing chain, due to the non-uniform sampling performed by the smartphone operating system, a cubic Spline interpolation is applied to represent the input data through evenly spaced points (200 points/second). Hence, a low pass Finite Impulse Response (FIR) filter with a cutoff frequency of $f_{c1} = 40$ Hz is used for denoising and to reduce the motion artifacts that may appear at higher frequencies. In fact, given the power profile of Fig. 1.2, the selected cutoff frequency only removes noise and preserves the relevant (discriminative) information about the user's motion.

In the following, $a_x(i)$ and $g_x(i)$ are the filtered and interpolated acceleration and gyroscope time series along axis x, where $i = 1, 2, \ldots$ is the sample number. The same notation holds for axes y and z.

1.2.2 Extraction of Walking Cycles

A template-based and iterative method that solely considers the accelerometer's magnitude signal is used for the extraction of walking cycles. This signal is in fact inherently invariant to the rotation of the smartphone and, as such, allows for the precise assessment of walking cycles regardless of how the user carries the device in her/his front pocket. For each sample $i = 1, 2, \ldots$ the acceleration magnitude is computed as:

$$a_{\text{mag}}(i) = (a_x(i)^2 + a_y(i)^2 + a_z(i)^2)^{1/2}. \tag{1.1}$$

To identify the template, a reference point in $a_{\text{mag}}(i)$ has to be located. To do so, inspired by [17], $a_{\text{mag}}(i)$ is first filtered through a low-pass filter with cutoff frequency $f_{c2} = 3$ Hz. Thus, the first minimum is detected, which corresponds to the heel strike [42], and the corresponding index is called $\tilde{i}$. This minimum is then refined by looking at the original signal $a_{\text{mag}}(i)$ within an interval centered on $\tilde{i}$ spanning one second of data, and picking

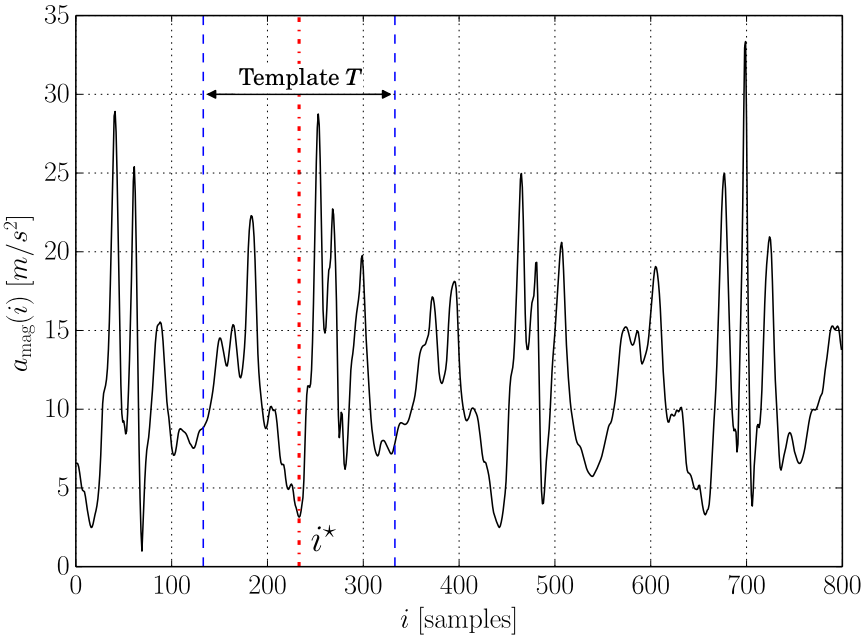


FIG. 1.3: Template extraction using the accelerometer magnitude $a_{\text{mag}}(i)$. The first template is the signal between the blue dashed vertical lines. The red dashed line in the center corresponds to i^*.

the minimum value of $a_{\text{mag}}(i)$ in this interval. This identifies a new index $i^\star$ for which $a_{\text{mag}}(i^\star)$ is a local minimum. As an example, this minimum is shown through a red vertical (dashed-dotted) line in Fig. 1.3. As a second step, a window of one second centered in $i^\star$ is considered, which is represented through two vertical blue (dashed) lines in Fig. 1.3. Now, the samples of $a_{\text{mag}}(i)$ falling between the two blue lines define the first *gait template*, called $\boldsymbol{T}$, with $|\boldsymbol{T}| = N_{\text{s}}$ samples, where N_{s} corresponds to the number of samples measured in one second. The extracted template is then iteratively refined and, at the same time, used to identify subsequent walking cycles. To this end, for any two real vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ of the same size n, the correlation distance is defined as follows:

$$\texttt{corr\_dist}(\boldsymbol{u},\boldsymbol{v}) = 1 - \frac{(\boldsymbol{u}-\overline{u}\mathbf{1}_n)\cdot(\boldsymbol{v}-\overline{v}\mathbf{1}_n)}{\|(\boldsymbol{u}-\overline{u}\mathbf{1}_n)\|\,\|(\boldsymbol{v}-\overline{v}\mathbf{1}_n)\|}\,. \tag{1.2}$$

The template $\boldsymbol{T}$ is then processed with the acceleration magnitude through the following Eq. (1.3), leading to a further metric $\varphi(i)$, where $i = 1, 2, \ldots$ is the sample index:

$$\begin{aligned} \boldsymbol{a}_{\text{mag}}(i, N_{\text{s}}) &= \texttt{vrect}(a_{\text{mag}}, i, N_{\text{s}})\,, \\ \varphi(i) &= \texttt{corr\_dist}(\boldsymbol{T}, \boldsymbol{a}_{\text{mag}}(i, N_{\text{s}}))\,,\ i = 1, \ldots\,. \end{aligned} \tag{1.3}$$

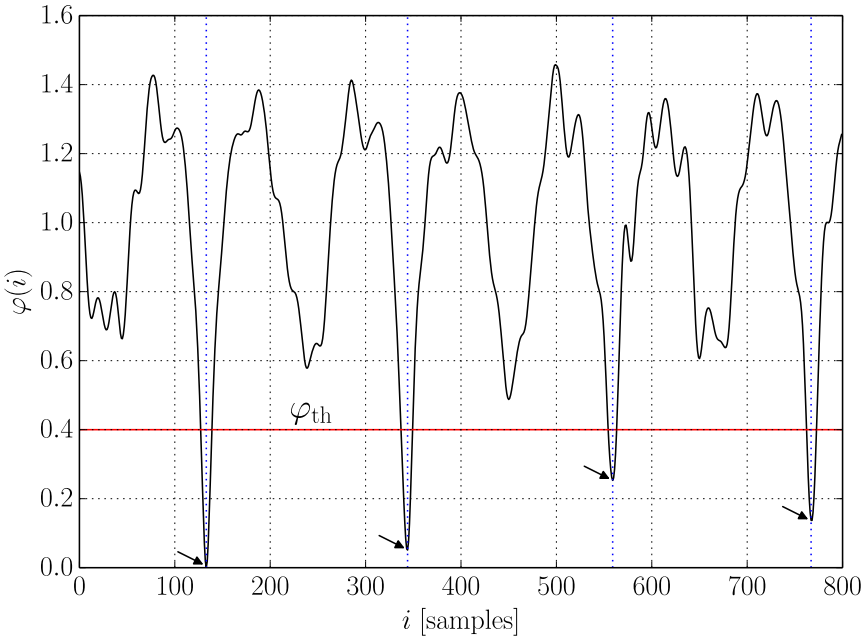


FIG. 1.4: Correlation distance $\varphi(i)$ between $a_{\text{mag}}(i)$ and the template $\boldsymbol{T}$ of Fig. 1.3. Local minima identify the beginning of walking cycles.

As can be seen from Fig. 1.4, the function $\varphi(i)$ exhibits some local minima, which are promptly located by comparing $\varphi(i)$ with a suitable threshold φ_{th} and performing a fast search inside the regions where $\varphi(i) < \varphi_{\text{th}}$. The indices corresponding to these minima determine the optimal alignments between the template $\boldsymbol{T}$ and $a_{\text{mag}}(i)$. In particular, the second of these identifies the beginning of the second gait cycle. From these facts we have that:

1. the samples between the second and the third minima correspond to the second gait cycle. It is thus possible to locate accelerometer and gyroscope vectors for this walking cycle, which are respectively defined as: $\boldsymbol{a}_x$, $\boldsymbol{a}_y$, $\boldsymbol{a}_z$ and $\boldsymbol{g}_x$, $\boldsymbol{g}_y$, $\boldsymbol{g}_z$, still expressed in the (x, y, z) coordinate system of the smartphone. The number of samples does not necessarily match the template length and usually differs from cycle to cycle, as it depends on the length and duration of walking steps.

2. A second template $\boldsymbol{T}'$ is obtained by reading N_{s} samples starting from the second minimum.

At this point, a new template is obtained through a weighted average of the old template $\boldsymbol{T}$ and the new one $\boldsymbol{T}'$:

$$\boldsymbol{T} = \alpha \boldsymbol{T} + (1 - \alpha)\boldsymbol{T}' \,, \tag{1.4}$$

where $\alpha = 0.9$ has been been used for the results shown in this study. The new template $\boldsymbol{T}$ is then considered for the extraction of the next walking cycle and the procedure is iterated. Note that this technique makes it possible to obtain an increasingly robust template at each new cycle.

A template matching approach exploiting a similar rationale was used in [17], where the authors employed the Pearson product-moment correlation coefficient between template and $a_{\text{mag}}(i)$. The main differences between [17] and the proposed approach are: the template $\boldsymbol{T}$ is extracted in a neighborhood of $i^\star$, using a fixed number of samples N_s, whereas they take the samples between two adjacent minima of $\varphi(i)$ (which may then differ in size for different cycles). Furthermore, in Eq. (1.4), a discrete-time filter is utilized to refine the template $\boldsymbol{T}$ at each walking cycle, making it more robust against speed changes. In previous work [17], the template is instead kept unchanged up to a point when minima cannot be longer detected, and a new template is to be obtained.

Finally, a normalization phase is required to represent all the cycles through the same number of points N, as this is required by the following feature extraction and classification algorithms. Before doing this, however, a transformation of accelerometer and gyroscope signals is performed to express these inertial signals in a *rotation invariant* reference system, as described next.

1.2.3 Orientation Independent Transformation

To evaluate the new orientation invariant coordinate system, three orthogonal versors $\hat{\boldsymbol{\xi}}$, $\hat{\boldsymbol{\zeta}}$, $\hat{\boldsymbol{\psi}}$ are to be found, whose orientation is independent of that of the smartphone and aligned with gravity and the direction of motion. Specifically, the aim is to express accelerometer and gyroscope signals in a coordinate system that remains fixed during the walk, with versor $\hat{\boldsymbol{\zeta}}$ pointing up (and parallel to the user's torso), versor $\hat{\boldsymbol{\xi}}$ pointing forward (aligned with the direction of motion) and $\hat{\boldsymbol{\psi}}$ tracking the lateral movement and being orthogonal to the other two. This entails inferring the orientation of the mobile device carried in the front pocket from the acceleration signal acquired during the walk. To this end, a technique similar to those of [43, 44] has been adopted.

Gravity is the main low frequency component in the accelerometer data, and will be the starting point of the transform. Moreover, although it is a

constant vector, it continuously changes when represented in the (x, y, z) coordinate system of the smartphone, due to the user's mobility and the subsequent change of orientation of the device. So, even the gravity vector $\vec{\rho}$ is not constant when expressed through the smartphone coordinates. The mean direction of gravity within the current walking cycle is considered as the first axis of the new reference system. Let n_k be the number of samples in the current cycle k, with $k = 1, 2, \ldots$. $\boldsymbol{a}_x$, $\boldsymbol{a}_y$ and $\boldsymbol{a}_z$ represent the acceleration samples in the current cycle k along the three axes x, y and z, with $|\boldsymbol{a}_x| = |\boldsymbol{a}_y| = |\boldsymbol{a}_z| = n_k$, whereas $\boldsymbol{g}_x$, $\boldsymbol{g}_y$ and $\boldsymbol{g}_z$ indicate the gyroscope samples in the same cycle k, with $|\boldsymbol{g}_x| = |\boldsymbol{g}_y| = |\boldsymbol{g}_z| = n_k$. The gravity vector $\vec{\boldsymbol{\rho}}_k$ within cycle k is estimated as:

$$\vec{\boldsymbol{\rho}}_k = (\overline{a}_x, \overline{a}_y, \overline{a}_z)^T . \tag{1.5}$$

The first versor of the new system $\hat{\boldsymbol{\zeta}}$ is obtained as:

$$\hat{\boldsymbol{\zeta}} = \frac{\vec{\boldsymbol{\rho}}_k}{\|\vec{\boldsymbol{\rho}}_k\|} . \tag{1.6}$$

Now, the acceleration matrix is defined as $\boldsymbol{A} = [\boldsymbol{a}_x, \boldsymbol{a}_y, \boldsymbol{a}_z]^T$ of size $3 \times n_k$, whose rows corresponds to $\boldsymbol{a}_x$, $\boldsymbol{a}_y$ and $\boldsymbol{a}_z$. Likewise, the gyroscope matrix is $\boldsymbol{G} = [\boldsymbol{g}_x, \boldsymbol{g}_y, \boldsymbol{g}_z]^T$, whose rows corresponds to $\boldsymbol{g}_x$, $\boldsymbol{g}_y$ and $\boldsymbol{g}_z$. The projected acceleration and gyroscope vectors along axis $\hat{\boldsymbol{\zeta}}$ are:

$$\boldsymbol{a}_\zeta = \boldsymbol{A} \cdot \hat{\boldsymbol{\zeta}} , \; \boldsymbol{g}_\zeta = \boldsymbol{G} \cdot \hat{\boldsymbol{\zeta}} , \tag{1.7}$$

where the new vectors have the same size n_k. By removing this component from the original accelerometer signal, the latter is projected on a plane that is orthogonal to $\hat{\boldsymbol{\zeta}}$. This is the horizontal plane (parallel to the floor). This *flattened* acceleration data is represented through a new matrix $\boldsymbol{A}^f = [\boldsymbol{a}_x^f, \boldsymbol{a}_y^f, \boldsymbol{a}_z^f]^T$ of size $3 \times n_k$, where $\boldsymbol{a}_x^f$, $\boldsymbol{a}_y^f$ and $\boldsymbol{a}_z^f$ are vectors of size n_k that describe the acceleration on the new plane:

$$\boldsymbol{A}^f = \boldsymbol{A} - \hat{\boldsymbol{\zeta}} \boldsymbol{a}_\zeta^T . \tag{1.8}$$

Analyzing this flattened acceleration, it can be observed that, during a walking cycle, it is unevenly distributed on the horizontal plane. Also, the acceleration points on this plane are dispersed around a preferential direction,

which has the highest excursion (variance). Here, it is assumed that the direction with the largest variance in the measurement space contain the dynamics of interest, i.e., it is parallel to the direction of motion, as it was also observed and verified in previous research [43]. Given this, this direction is considered as the second axis (versor $\hat{\boldsymbol{\xi}}$) of the new reference system. This is done by applying the Principal Component Analysis (PCA) [45] on the projected points, which finds the direction along which the variance of the measurements is maximized. The detailed procedure is as follows:

1. Find the empirical mean along each direction x, y and z (rows 1, 2 and 3 of the flattened acceleration matrix $\boldsymbol{A}^f$). Store the mean in a new vector $\boldsymbol{u}$ of size 3×1., i.e.:

$$u_i = \frac{1}{n_k} \sum_{j=1}^{n_k} A^f_{i,j}\,,\; i = 1, 2, 3\,. \tag{1.9}$$

2. Subtract the empirical mean vector $\boldsymbol{u}$ from each column of matrix $\boldsymbol{A}^f$, obtaining the new matrix $\boldsymbol{A}^f_{\text{norm}}$:

$$\boldsymbol{A}^f_{\text{norm}} = \boldsymbol{A}^f - \boldsymbol{u}(\mathbf{1}_{n_k})^T\,. \tag{1.10}$$

3. Compute the sample 3×3 autocovariance matrix $\boldsymbol{\Sigma}$:

$$\boldsymbol{\Sigma} = \frac{\boldsymbol{A}^f_{\text{norm}}(\boldsymbol{A}^f_{\text{norm}})^T}{n_k - 1}\,. \tag{1.11}$$

4. The eigenvalues and the corresponding eigenvectors of $\boldsymbol{\Sigma}$ are evaluated. The eigenvector $\vec{v}$ associated with the maximum eigenvalue identifies the direction of maximum variance in the dataset (i.e., the first principal component of the PCA transform).

Hence, versor $\hat{\boldsymbol{\xi}}$ is evaluated as:

$$\hat{\boldsymbol{\xi}} = \frac{\vec{v}}{\|\vec{v}\|}\,. \tag{1.12}$$

Accelerometer and gyroscope data are then projected along $\hat{\boldsymbol{\xi}}$ through the following equations: $\boldsymbol{a}_\xi = \boldsymbol{A} \cdot \hat{\boldsymbol{\xi}}$ and $\boldsymbol{g}_\xi = \boldsymbol{G} \cdot \hat{\boldsymbol{\xi}}$. Being $\hat{\boldsymbol{\xi}}$ placed on a plane that is orthogonal to $\hat{\boldsymbol{\zeta}}$, these two versors are also orthogonal. The third axis

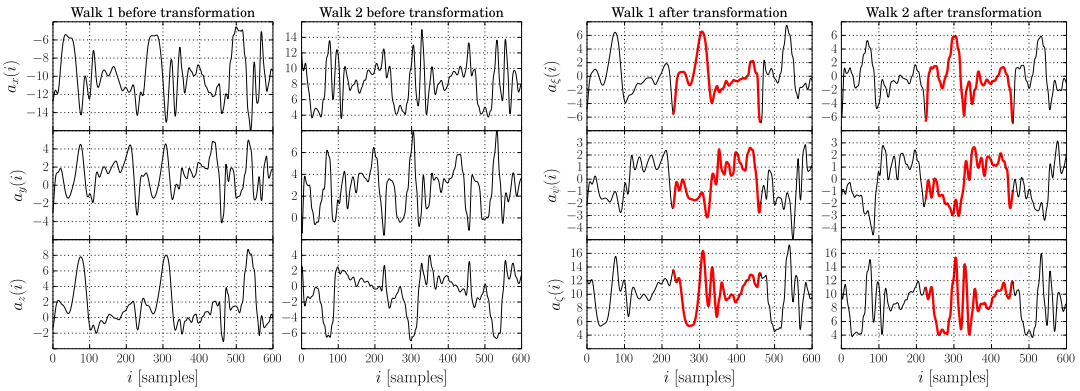


FIG. 1.5: Raw accelerometer data from two different walks, acquired from a smartphone worn in the right front pocket with different orientations. Accelerometer data in the smartphone reference system (x, y, z) (left), and after the transformation (ξ, ψ, ζ) (right). IDNet implements a PCA-based transformation that makes walking data rotation invariant, i.e., subject-specific gait patterns emerge in the new coordinate system (see the red colored patterns in the right plots).

is then obtained through a cross product:

$$\hat{\boldsymbol{\psi}} = \hat{\boldsymbol{\zeta}} \times \hat{\boldsymbol{\xi}}\,, \tag{1.13}$$

and the new accelerometer and gyroscope data along this axis are respectively obtained as: $\boldsymbol{a}_\psi = \boldsymbol{A} \cdot \hat{\boldsymbol{\psi}}$ and $\boldsymbol{g}_\psi = \boldsymbol{G} \cdot \hat{\boldsymbol{\psi}}$. The transformed vectors $(\boldsymbol{a}_\xi, \boldsymbol{a}_\psi, \boldsymbol{a}_\zeta)$ and $(\boldsymbol{g}_\xi, \boldsymbol{g}_\psi, \boldsymbol{g}_\zeta)$, along with the magnitude vectors $\boldsymbol{a}_{\text{mag}}$ and $\boldsymbol{g}_{\text{mag}}$ are the output of the Orientation Independent Transformation block of Fig. 1.1.

An example of this transform is shown in Fig. 1.5, where accelerometer and gyroscope data from two different walks from the same subject are plotted. These signals were acquired carrying the phone in the right front pocket of the subject's trousers using two different orientations. As highlighted in the figure, this projection makes walking data rotation invariant. In fact, subject-specific gait patterns emerge in the new coordinate system (see the red colored patterns in the right plots).

1.2.4 Normalization

Each gait cycle has a different duration, which depends on the walking speed and stride length. So, considering the accelerometer and gyroscope data collected during a full walking cycle, acceleration and gyroscope vectors have a

variable size, which are now expressed in the new orientation invariant coordinate system discussed in Section 1.2.3. However, since feature extraction and classification algorithms require N-sized vectors for each cycle, where N has to be fixed, a further adjustment is necessary. A further Spline interpolation has been used to cope with this cycle length variability, representing all walking cycles through vectors of $N = 200$ samples each. This specific value of N was selected to avoid aliasing. In fact, assuming a maximum cycle duration of $\tau = 2$ seconds, which corresponds to a very slow walk, and a signal bandwidth of $B = 40$ Hz, a number of samples $N > 2B\tau = 160$ samples/cycle is required. Amplitude normalization was also implemented, to obtain vectors with zero mean and unit variance, as this leads to better training and classification performance. This results in a total of eight N-sized vectors for each walking cycle, which are inputted into the feature extraction and classification algorithms of the following sections.

1.3 Convolutional Neural Network

In this section, the chosen Convolutional Neural Network (CNN) architecture for IDNet (Sec. 1.3.1) is detailed, along with its optimization, training and quantitative comparison against the most common classifiers from the literature (Sec. 1.3.2).

1.3.1 CNN Architecture

CNNs are feed-forward deep neural networks differing from fully connected multilayer networks for the presence of one or more convolutional layers. At each convolutional layer, a number of *kernels* is defined. Each of them has a number of weights, which are convolved with the input in a way that the same set of weights, i.e., the same kernel, is applied to all the input data, moving the convolution operation across the input span. Note that, as the same weights are reused (shared weights), and each kernel operates on a small portion of the input signal, it follows that the network connectivity structure is sparse. This leads to advantages such as a considerably reduced computational complexity with respect to fully connected feed forward neural networks. For more details the reader is referred to [46]. CNNs have been proven to be excellent feature extractors for images [47] and here their effectiveness for motion data is also shown. The CNN architecture designed to

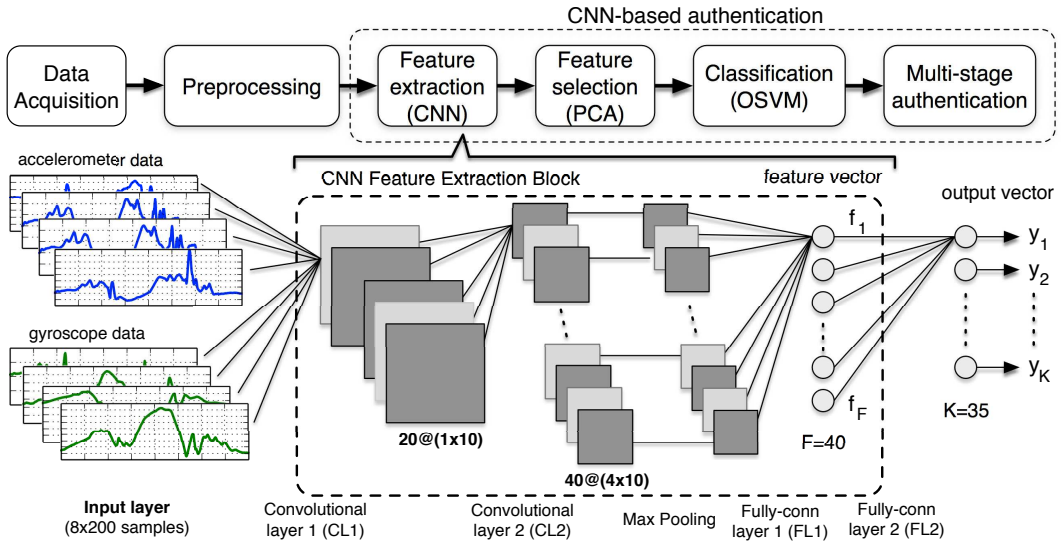


FIG. 1.6: IDNet framework. CL1 and CL2 are convolutional layers, FL1 and FL2 are fully connected layers. X@(Y×Z) indicates the number of kernels, X, and the size of the kernel matrix, Y×Z.

this purpose is shown in Fig. 1.6. It is composed of a cascade of two convolutional layers, followed by a pooling and a fully-connected layer. The convolutional layers perform a dimensionality reduction (or feature extraction) task, whereas the fully-connected one acts as a classifier. Accelerometer and gyroscope data from each walking cycle is processed according to the algorithms of Sec. 1.2. The input matrix for a generic walking cycle is referred to as $\boldsymbol{X} = (\boldsymbol{a}_\xi, \boldsymbol{a}_\psi, \boldsymbol{a}_\zeta, \boldsymbol{a}_{\text{mag}}, \boldsymbol{g}_\xi, \boldsymbol{g}_\psi, \boldsymbol{g}_\zeta, \boldsymbol{g}_{\text{mag}})^T$, where all the vectors are normalized to N samples (see Sec. 1.2.4). In detail, we have (CL = Convolutional Layer, FL = Fully-connected Layer):

- **CL1** The first convolutional layer implements one dimensional kernels (1x10 samples) performing a first filtering of the input and processing each input vector (rows of $\boldsymbol{X}$) separately. This means that at this stage it is not captured any correlation among different accelerometer and gyroscope axes. The activation functions are linear and the number of convolutional kernels is referred to as N_{k1}.

- **CL2** The second convolutional layer is intended to seek discriminant and class-invariant features. Here, the cross-correlation among input vectors is considered (kernels of size 4x10 samples) and the output activation functions are non-linear hyperbolic tangents. Max pooling is applied to the output of CL2 to further reduce its dimensionality and

increase the spatial invariance of features [48]. N_{k2} is the number of convolutional kernels used for CL2.

- **FL1** This is a fully connected layer, i.e., each output neuron of CL2 is connected to all input neurons of this layer (weights are not shared). Hyperbolic tangent activation functions are used at the output neurons. FL1 output vector is termed $\boldsymbol{f} = (f_1, \ldots, f_F)^T$, and contains the F features extracted by the CNN.

- **FL2** Each output neuron in this layer corresponds to a specific class (one class per user), for a total of K neurons, where K is the number of subjects considered for the training phase. The K dimensional output vector $\boldsymbol{y} = (y_1, \ldots, y_K)^T$ is obtained by a *softmax* activation function, which implies that $y_j \in (0, 1)$, $j = 1, \ldots, K$ and $\sum_{j=1}^{K} y_j = 1$ (stochastic vector). Also, y_j can be thought of as the probability that the current data matrix $\boldsymbol{X}$ belongs to class (user) j.

The network is trained in a supervised manner for a total of K subjects solving a multi-class classification problem, where each of the input matrices $\boldsymbol{X}$ in the dataset is assigned to one of K mutually exclusive classes. The *target* output vector $\boldsymbol{t} = (t_1, \ldots, t_K)^T$ has binary entries and is encoded using a 1-of-K coding scheme, i.e., they are all zero except for that corresponding to the subject that generated the input data.

1.3.2 CNN Optimization and Results

In this section, some approaches for the optimization of the CNN are presented, quantifying its classification performance and comparing it against classification techniques from the literature. As said above, the output of layer FL2 is the stochastic vector $\boldsymbol{y}$, whose j-th entry y_j, $j = 1, \ldots, K$, can be seen as the probability that the input pattern belongs to user j, i.e., $y_j = y_j(\boldsymbol{w}, \boldsymbol{X}) = \text{Prob}(t_j = 1 | \boldsymbol{w}, \boldsymbol{X})$, where $\boldsymbol{w}$ is the vector containing all the CNN weights, $\boldsymbol{X}$ is the current input matrix (walking cycle) and $t_j = 1$ if $\boldsymbol{X}$ belongs to class j and $t_j = 0$ otherwise. If $\mathcal{X}$ is the set of all training examples, let's define the batch set as $\mathcal{B} \subset \mathcal{X}$. Let $\boldsymbol{X} \in \mathcal{B}$ and denote the corresponding output vector by $\boldsymbol{y}(\boldsymbol{w}, \boldsymbol{X})$ and its j-th entry by $y_j(\boldsymbol{w}, \boldsymbol{X})$. The corresponding target vector is $\boldsymbol{t}(\boldsymbol{X}) = (t_1(\boldsymbol{X}), \ldots, t_K(\boldsymbol{X}))^T$. The CNN is then trained

through a stochastic gradient descent algorithm which minimizes a *categorical cross-entropy loss function* $L(\boldsymbol{w})$, defined as [12, Eq. (5.24) of Sec. 5.2]:

$$L(\boldsymbol{w}) = -\sum_{\boldsymbol{X}\in\mathcal{B}}\sum_{j=1}^{K} t_j(\boldsymbol{X})\log(y_j(\boldsymbol{w},\boldsymbol{X}))\,. \tag{1.14}$$

During training, Eq. (1.14) is iteratively minimized, by rotating the walking cycles (training examples) in the batch set $\mathcal{B}$ so as to span the entire input set $\mathcal{X}$. Training continues until a stopping criterion is met (see below).

Walking patterns from K subjects are used to train the CNN, and the same number of cycles N_c is considered for each of them, for a total of KN_c training cycles. N_t randomly chosen walking cycles from each subjects are used to obtain a test set $\mathcal{P}$. The remaining cycles are split into training $\mathcal{T}$ and validation $\mathcal{V}$ sets, with $|\mathcal{P}| = KN_\mathrm{t}$, $|\mathcal{T}| = KN_\mathrm{c}$, $\mathcal{X} = \mathcal{P}\cup\mathcal{T}\cup\mathcal{V}$, where all the sets have null pairwise intersection and are built picking input patterns from $\mathcal{X}$ evenly at random. Set $\mathcal{V}$ is used to terminate the training phase, and termination occurs when the loss function $L(\boldsymbol{w})$ evaluated on $\mathcal{V}$ does not decrease for twenty consecutive training epochs. After that, the network weights which led to the minimum validation loss are used to assess the CNN performance on set $\mathcal{P}$. This is done through an *accuracy* measure, defined as the number of walking cycles correctly classified by the CNN divided by the total number of cycles in $\mathcal{P}$. The following graphs show the mean accuracy obtained averaging the test set performance over ten different networks, all of them trained through the just explained approach by considering $K = 35$ subjects from the dataset and $N_\mathrm{t} = 100$ cycles per subject.

As a first set of results, the impact of F (neurons in layer $FL1$) and of the number of convolutional kernels in CL1 and CL2 are analyzed. Since the last layer FL2 acts as a classifier, F can be seen as the number of features extracted by the CNN. In general, a too small F can lead to poor classification results; too many features, instead, would make the state space too big to be effectively dealt with (curse of dimensionality) and may lead to overfitting issues [49]. Besides F, it has been investigated the right number of kernels to use within each convolutional layer. Three networks are considered by picking different $(N_{\mathrm{k}1}, N_{\mathrm{k}2})$ pairs. For network 1 $(N_{\mathrm{k}1} = 10, N_{\mathrm{k}2} = 20)$ is used, network 2 has $(N_{\mathrm{k}1} = 20, N_{\mathrm{k}2} = 40)$ and network 3 has $(N_{\mathrm{k}1} = 30, N_{\mathrm{k}2} = 50)$. In Fig. 1.7, the accuracy performance of these networks as a function of F is shown. From this plot, it can be seen that at least $F = 20$ neurons have

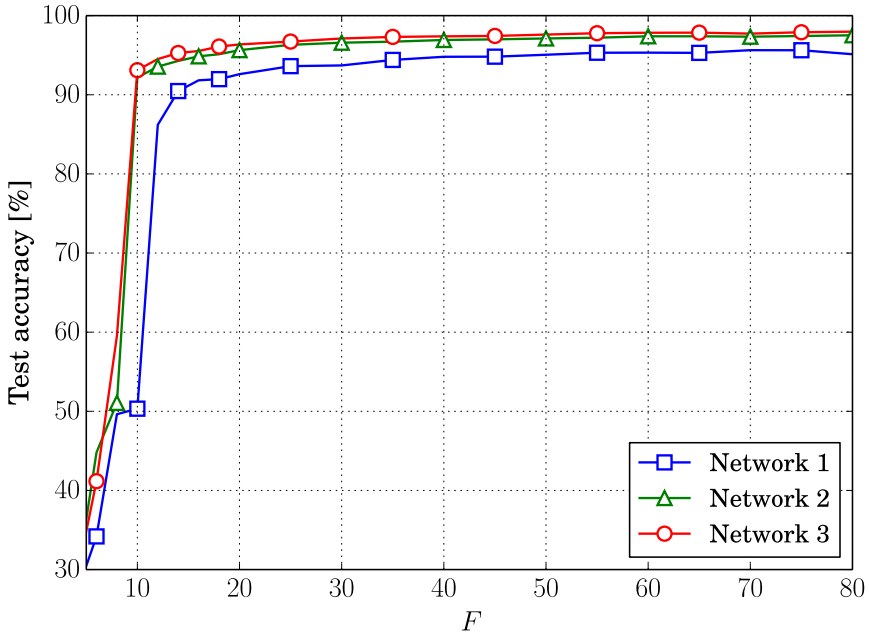


FIG. 1.7: CNN test accuracy *vs* number of features F in layer FL1. Three curves are shown for three different network configurations (number of kernels in layers CL1 and CL2).

to be used at the output of FL1 and that the accuracy performance stabilizes around $F = 40$, leading to negligible improvements as N grows beyond this value. As for the number of kernels, small networks (network 1) perform worse than bigger ones (networks 2 and 3), but increasing the number of kernels beyond that used for network 2 does not lead to appreciable improvements. Hence, $F = 40$ is used for the results of this study, with $(N_{k1} = 20, N_{k2} = 40)$.

A key performance comparison is shown in Fig. 1.8, where the accuracy is plotted against N_c for a CNN classifier and four selected classification algorithms from the literature, i.e., Classification Trees (CT) [50], Naive Bayes (NB) classifiers [51], k-Nearest Neighbors (k-NN) [52] and Support Vector Machines (SVM) [53].[1] These approaches were used in a large number of papers including [14, 16, 15, 19, 33]. For their training, 112 features were extracted from the signal samples in $\boldsymbol{X}$, including their variance, mean trend, windowed mean difference, variance trend, windowed variance difference, maxima and minima, spectral entropy, zero crossing rate and bin counts. These features, were then utilized to train the selected classifiers in a supervised manner. Note that, while the CNN automatically extracts its features

[1] For SVM, a linear kernel has been considered, as it outperformed polynomial and radial basis function ones. A one-versus-all strategy was used solve the considered multiclass problem for the binary classifers.

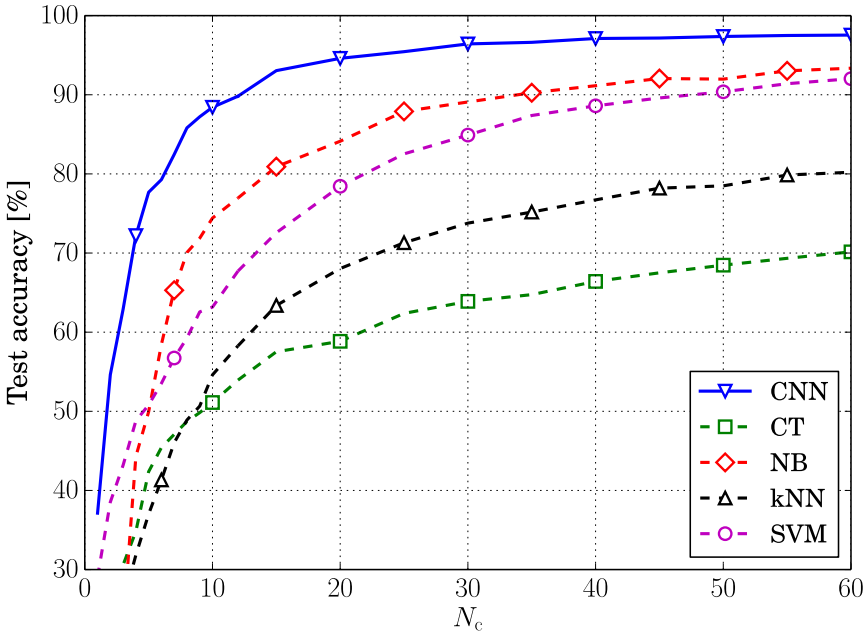


FIG. 1.8: CNN test accuracy *vs* number of walking cycles N_c used for training. Results for CT, NB, *k*-NN and SVM classifiers from the literature are also shown.

(vector $\boldsymbol{f}$), with previous techniques these are manually selected based on human expertise and statistical knowledge.

From Fig. 1.8 it can be observed that the CNN-based classification approach surpasses all the previous classifiers from the literature, delivering better accuracies across the entire range of N_c. Also, the accuracy increases with an increasing N_c until it saturates and no noticeable improvements are observed. While a higher N_c is always beneficial, a higher number of cycles also entails a longer acquisition time, which we would rather avoid. For this reason, $N_c = 40$ has been used for the following results, as it provided a good trade-off between accuracy and complexity across all the experiments.

To illustrate the superiority of CNN features with respect to manually extracted ones, in the following an instructive experiment is performed. The CNN is considered as a feature extraction block, by removing the output vector $\boldsymbol{y}$ and using the inner feature vector $\boldsymbol{f}$ to train the above classifiers from the literature (CT, NB, *k*-NN and SVM). The corresponding accuracy results are provided in Fig. 1.9. All the classifiers perform better when trained using CNN features, with typical improvements in the test accuracy of more than 10%. For instance, for a *k*-NN classifier trained with $N_c = 30$ cycles per subject, the accuracy increases from 71% (manually extracted features) to 94% (CNN features). The best performance is provided by the combined use of CNN features and SVM.

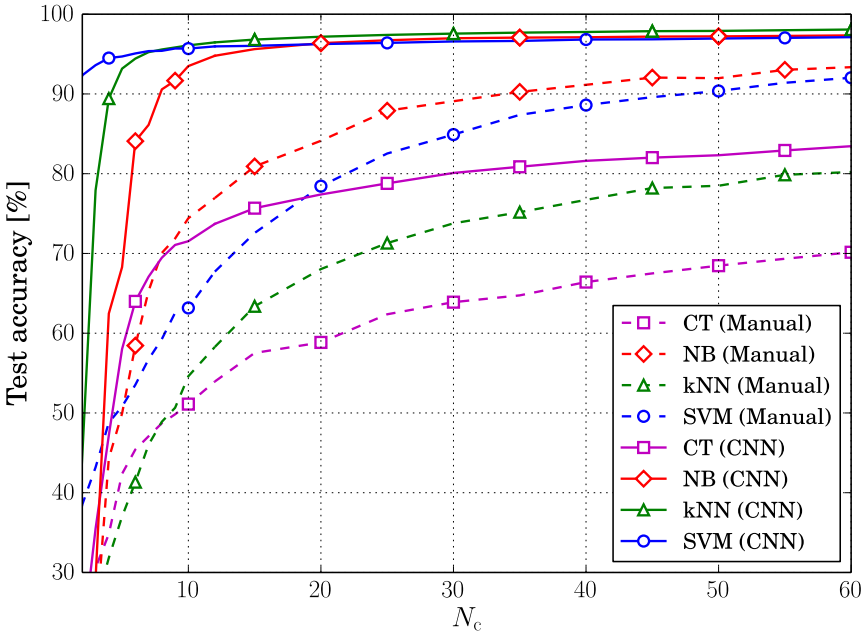


FIG. 1.9: Test accuracy of CT, NB, *k*-NN and SVM classifiers. "CNN" indicates training with CNN-extracted features, whereas "Manual" means standard feature extraction.

A last consideration is in order. Most of the previous papers only used accelerometer data, but the results show that using both gyroscope and accelerometer provides further improvements, see Fig. 1.10.

1.4 One-Class Support Vector Machine Training

In this section, the IDNet CNN-based identification chain is further extended through the design of an SVM classifier which is trained solely using the motion data of the target subject. This is referred to as One-Class Classification (OCC) and it is important for practical applications where motion signals of the target user are available, but those belonging to other subjects are not. More importantly, with this approach the classification framework can be extended to users that were not considered in the CNN training.

1.4.1 Revised Classification Architecture

Due to the generalization property of deep convolutional networks, once trained, the CNN can be used as a universal feature extractor, providing meaningful features even for subjects that were not included in the training. To take advantage of this, the output neurons of FL2 is discarded and

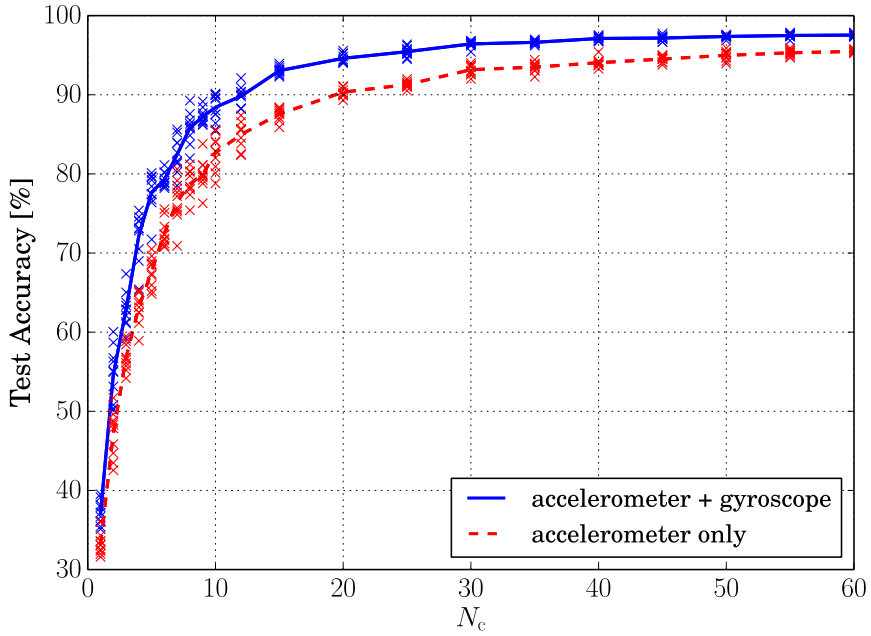


FIG. 1.10: Impact of gyroscope data. Lines represent the mean accuracy (averaged over ten networks), whereas markers indicate the results of the ten network instances.

the CNN can be used as a dimensionality reduction tool that, given an input matrix $\boldsymbol{X}$, returns a user dependent feature vector $\boldsymbol{f}$. The CNN is then trained only once considering the optimizations of Sec. 1.3.2. All its weights and biases are then precomputed and will not be modified at classification time. Considering the diagram of Fig. 1.6, the output of the CNN is the feature vector $\boldsymbol{f}$. An additional feature selection block lead to a reduced number of features, from F to $S \leq F$ (dimensionality reduction). PCA is used to accomplish this task and the new feature vector is called $\boldsymbol{s}$. Hence, we have $\boldsymbol{s} = \Upsilon(\boldsymbol{f})$, where $\Upsilon(\cdot) : \mathbb{R}^F \rightarrow \mathbb{R}^S$ is the PCA transform.

A One-class Support Vector Machine (OSVM) is then used as the classification algorithm (Sec. 1.4.2). It defines a *boundary* around the feature (training) vectors belonging to the target subject. At runtime, as a new walking cycle is processed, the OSVM takes the feature vector $\boldsymbol{s}$ and outputs a *score*, which is a distance measure between the current feature vector and the SVM boundary [12, Chapter 7]. As discussed shortly, this score relates to the likelihood that the current walking cycle belongs to the target user.

1.4.2 One-Class SVM Design

Next, the OSVM block of Fig. 1.6 is designed. It differs from a standard binary SVM classifier as the SVM boundary is built solely using patterns from the positive class (target user). The strategy proposed by Schölkopf is to map the data into the feature space of a kernel, and to separate them from the origin with maximum margin [54]. The corresponding minimization problem is similar to that of the original SVM formulation [53]. Using an appropriate hyperplane (in the space transformed by a suitable kernel function) it is possible to discriminate the target vectors. The OSVM takes as input the reduced feature vector $\boldsymbol{s} = (s_1, \ldots, s_S)^T$, and it use the following Radial Basis Function (RBF) kernel, that for any $\boldsymbol{s}, \boldsymbol{s}' \in \mathbb{R}^S$ is defined as:

$$\Psi(\boldsymbol{s}, \boldsymbol{s}') = \left(\Phi(\boldsymbol{s}) \cdot \Phi(\boldsymbol{s}')\right) = \exp\left(-\gamma \left\|\boldsymbol{s} - \boldsymbol{s}'\right\|^2\right), \tag{1.15}$$

where $\Phi(\boldsymbol{s})$ is a feature map and γ is the RBF kernel parameter, which intuitively relates to the radius of influence that each training vector has for the space transformation. ℓ is the number of training points (feature vectors), $\boldsymbol{\omega}$ and b are the hyperplane parameters in the transformed domain (through Eq. (1.15)) and $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_\ell)^T$ is the vector of slack variables, which are introduced to deal with outliers. Given this, the following quadratic program is defined to separate the feature vectors in the training set, $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_\ell$, from the origin:

$$\begin{aligned} \min_{\boldsymbol{\omega}, \boldsymbol{\varepsilon}, b} \quad & \frac{1}{2}\|\boldsymbol{\omega}\|^2 + \frac{1}{\nu\ell}\sum_{j=1}^{\ell}\varepsilon_j - b \\ \text{subject to} \quad & \left(\boldsymbol{\omega} \cdot \Phi(\boldsymbol{s}_j)\right) \geq b - \varepsilon_j\,,\ \varepsilon_j \geq 0\,,\ j = 1, \ldots, \ell \end{aligned} \tag{1.16}$$

$\nu \in (0, 1)$ is one of the most important parameters and sets an upper bound on the fraction of outliers and a lower bound on the fraction of Support Vectors (SV) [54]. The decision function for a generic feature vector $\boldsymbol{s}$ is defined as $d(\boldsymbol{s}) \in \{-1, +1\}$, is obtained solving Eq. (1.16), and only depends on the training vectors through the following relations:

$$\begin{aligned} d(\boldsymbol{s}) &= \operatorname{sgn}\left(h(\boldsymbol{s})\right), \\ h(\boldsymbol{s}) &= \sum_{j=1}^{\ell} \alpha_j \Psi(\boldsymbol{s}_j, \boldsymbol{s}) - b\,. \end{aligned} \tag{1.17}$$

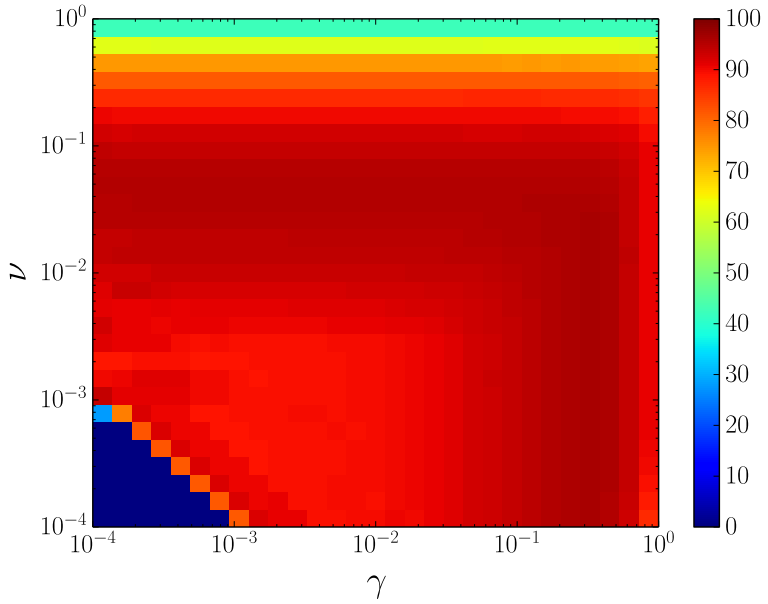


FIG. 1.11: OSVM: F-measure as a function of γ and ν.

Now, $\alpha_j \geq 0, \forall\, j$, and only some of the training vectors have $\alpha_j > 0$. These are the *support vectors* associated with the classification problem and are the only ones who count in the definition of the SVM boundary. $h(\boldsymbol{s})$ is the *score* associated with vector $\boldsymbol{s}$. It weighs the distance from the SVM boundary, i.e., is greater than zero if $\boldsymbol{s}$ resides inside the boundary, zero if it lies on it and negative otherwise.

Hence, the SVM is trained using a set of ℓ feature vectors from the target user, obtaining the SVM boundary (and the related decision function) through Eq. (1.17). After training, the performance of the obtained SVM classifier is tested considering feature vectors from the positive class C_1 (target user) and the negative one C_0 (any other user). Note that the vectors used for this test were not considered during the SVM training.

As it is customary for binary classification approaches, the two most important metrics to assess the goodness of a classifier are the *precision* and the *recall*. The precision is the fraction of true positives, i.e., the fraction of patterns identified of the target class that in fact belong to the target user, while the recall corresponds to the fraction of target patterns that are correctly classified out of the entire positive class of samples [55]. Often, these two metrics are combined into their harmonic mean, which is called F-measure and is used as the single quality parameter.

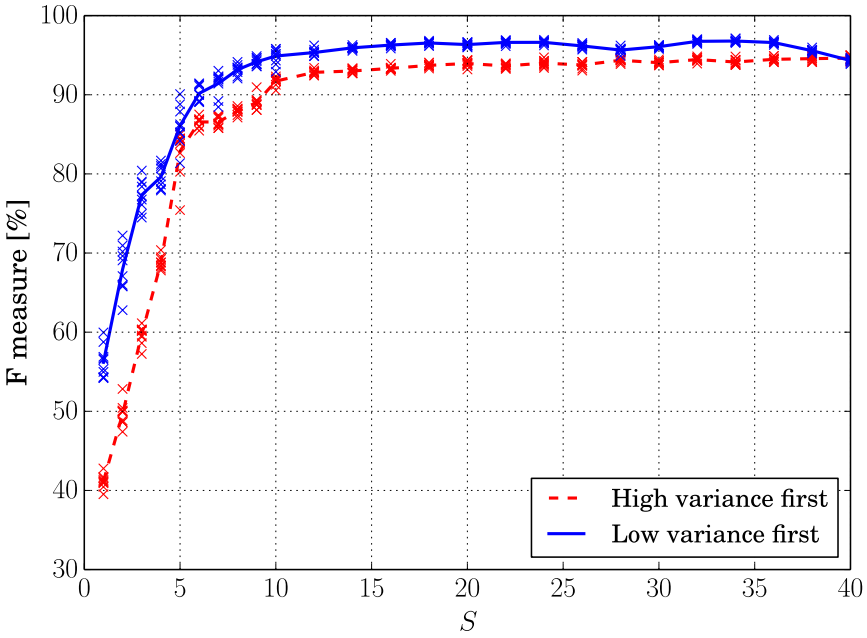


FIG. 1.12: OSVM: F-measure as a function of the number of retained PCA features S. The number of CNN-extracted features is $F = 40$.

In Fig. 1.11, the F-measure is plotted as a function of the two SVM parameters γ and ν. As seen from this plot, the area where the classifier's performance is maximum is quite ample. This is good as it means that even selecting γ and ν once for all at design stage, the performance of the SVM classifier is not expected to change much if the signal statistics changes or a new target user is considered. In other words, this relatively weak dependence on the parameters entails an intrinsic robustness for the classifier. For the results that follow the parameters have been set to $\gamma = 0.3$ and $\nu = 0.02$.

Two last considerations are in order. The first relates to the PCA transformation $\mathrm{Y}(\cdot)$ and in particular to how many and which principal components have to be retained for the output feature vectors. In fact, as pointed out in [56], two options are possible to go from the CNN-extracted feature vector $\boldsymbol{f}$ to $\boldsymbol{s}$. The first is to retain the $S \leq F$ entries of the transformed vector (expressed in the PCA basis) that correspond to the principal components with highest variance, whereas a second option is to retain those with the smallest. Fig. 1.12 shows the F-measure of the OSVM classifier as a function of S for $F = 40$ (number of CNN-extracted features). From this plot it can be observed that picking $S < F$ in general provides better results and also that considering the principal components with lowest variance provides better results for this class of problems. This is in accordance with [56].

The last consideration regards the amount of feature vectors belonging

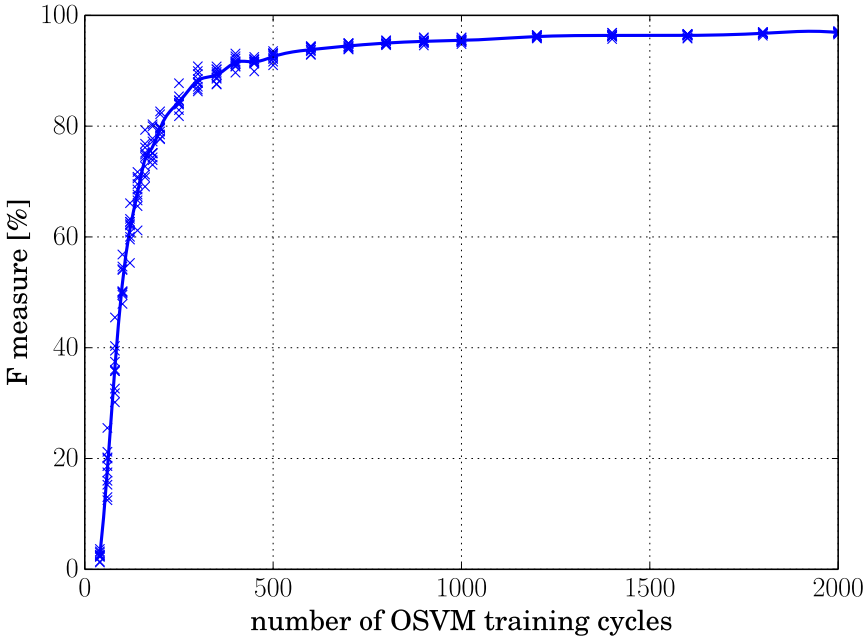


FIG. 1.13: F-measure as a function of the number of walking cycles used to train the OVSM classifier.

to the target user that should be used for the OSVM training. Note that this number is related to the walking time required for a new subject to train his/her personal identification system. To perform this analysis, a fixed number of cycles were randomly extracted from the whole target dataset and were used to train the OSVM. The remaining walking cycles were used as the positive test set. In Fig. 1.13 the F-measure as a function of this number of cycles is shown. From these results, it follows that increasing the number of cycles beyond $1,000$ leads to little improvement. This number corresponds to about 15 minutes of walking activity, distributed among different acquisition sessions. Multiple sessions are recommended to account for some statistical variation due to wearing different clothes.

Once all the model's parameters are defined, the OSVM score can be analyzed. Let $p_\theta(h(\boldsymbol{s})) = p(h(\boldsymbol{s}) \mid \boldsymbol{s} \in C_\theta)$ be the estimated probability density function (pdf) of the OSVM score $h(\boldsymbol{s}) \in \mathbb{R}$, provided that the walking cycle belongs to a user of class C_θ with $\theta \in \{0, 1\}$. Empirical pdfs $p_\theta(h(\boldsymbol{s}))$ from the dataset are provided in Fig. 1.14.

1.5 Sequential Analysis

The so far discussed processing pipeline returns a score for each walking cycle. However, as seen in Fig. 1.14, when a score falls near the point where

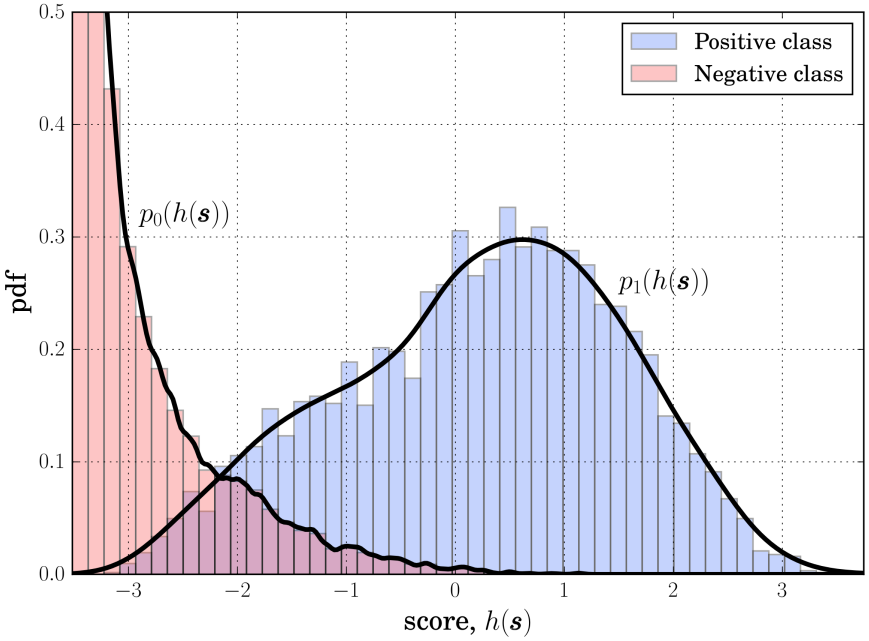


FIG. 1.14: Empirical pdf of the OSVM scores for class C_1 ($p_1(h(\boldsymbol{s}))$) and C_0 ($p_0(h(\boldsymbol{s}))$).

the two pdfs intersect, there is a high uncertainty about the identity of the user who generated it. In IDNet, this indetermination is reduced by jointly considering the scores from successive walking cycles. Let $\mathcal{O} = (o_1, o_2, \dots)$ be a sequence of subsequent OSVM scores from the same subject, where $o_i = h(\boldsymbol{s}_i) \in \mathbb{R}$ and $i = 1, 2, \dots$ is the walking cycle index. From previous analysis, o_i can be thought of as a random process having probability density function $p_\theta(h(\boldsymbol{s}_i)) = p_\theta(o_i)$, $\theta \in \{0, 1\}$, and the objective is to reliably estimate θ from the scores in $\mathcal{O}$. Toward this, it is assumed that subsequent scores belong to the same user and that they are independent and identically distributed (i.i.d), i.e., they are independently drawn from $p_\theta(\cdot)$, with θ unknown. For the estimation of θ the Wald's sequential probability ratio test (SPRT) [57, 58] is used. The two hypotheses are defined as $\{H_1 : \theta = 1\}$, meaning that the sequence $\mathcal{O}$ belongs to the target user (class C_1), and $\{H_0 : \theta = 0\}$, meaning that another user generated it (class C_0). Hence, the most likely hypothesis can be assessed through the SPRT. That is, by measuring new scores it is possible to decrease the uncertainty about θ. Considering n samples $(o_1, o_2, \dots, o_n)$, the final decision takes on two values $D_n = 0$ or $D_n = 1$, where $D_n = j$, $j \in \{0, 1\}$ means that hypothesis H_j is accepted and therefore the alternative hypothesis is rejected. Owing to the previous assumptions (i.i.d. scores, generated by the same subject), for

n scores $\mathcal{O}_n = (o_1, o_2, \ldots, o_n)$ the joint pdf is:

$$\tilde{p}_\theta(\mathcal{O}_n) = \prod_{j=1}^{n} p_\theta(o_j), \quad \theta \in \{0, 1\}\,. \tag{1.18}$$

Defining $\lambda_j = p_1(o_j)/p_0(o_j)$, the likelihood ratio of the sequence $\mathcal{O}$ truncated at index n, $\mathcal{O}_n$, is

$$\frac{\tilde{p}_1(\mathcal{O}_n)}{\tilde{p}_0(\mathcal{O}_n)} = \prod_{j=1}^{n} \frac{p_1(o_j)}{p_0(o_j)} = \prod_{j=1}^{n} \lambda_j\,, \tag{1.19}$$

and applying the logarithm, we get:

$$\Lambda_n = \log\left(\frac{\tilde{p}_1(\mathcal{O}_n)}{\tilde{p}_0(\mathcal{O}_n)}\right) = \sum_{j=1}^{n} \log\left(\lambda_j\right)\,. \tag{1.20}$$

Waiting a further step $n+1$ before making a decision, from Eq. (1.20) the new log-likelihood Λ_{n+1} is conveniently obtained as $\Lambda_{n+1} = \Lambda_n + \log(\lambda_{n+1})$. The SPRT test starts from time 1, obtaining one-class OSVM scores $o_1, o_2, \ldots$ for each successive walking cycle. After n cycles, the cumulative log-likelihood ratio is $\Lambda_n = \Lambda_{n-1} + \log(\lambda_n)$, with $\Lambda_0 = 0$. Two suitable thresholds A and B are defined and the test continues to the next cycle $n+1$ if $A < \Lambda_n < B$, H_1 is accepted if $\Lambda_n \geq B$, whereas H_0 is accepted if $\Lambda_n \leq A$. Moreover, defining α as the probability of accepting H_1 when H_0 is true and β that of accepting H_0 when H_1 is true, A and B can be approximated as: $A = \log(\beta/(1-\alpha))$ and $B = \log((1-\beta)/\alpha)$, see [57].

1.5.1 Experimental Results

The motion data from $K = 35$ subjects was used to train the CNN feature extractor, with $N_c = 40$, $F = 40$ and $S = 20$. One user out of the remaining 15 was considered as the *target* user and 14 as the negatives for the final tests. The following results are obtained through a leave-one-out cross-validation approach for the sessions of the target user, i.e., out of twelve sessions, eleven are used for training and one for the final tests. The session that is left out is rotated and the final results are averaged across all trials. The results of the multi-stage framework are shown in Fig. 1.15. False positive rates (i.e., a user is mistakenly recognized as the target) and false negative ones (i.e., the target is not recognized) are smaller than 0.15% for an appropriate choice of the SPRT thresholds (α and β). Also, a reliable identification requires fewer

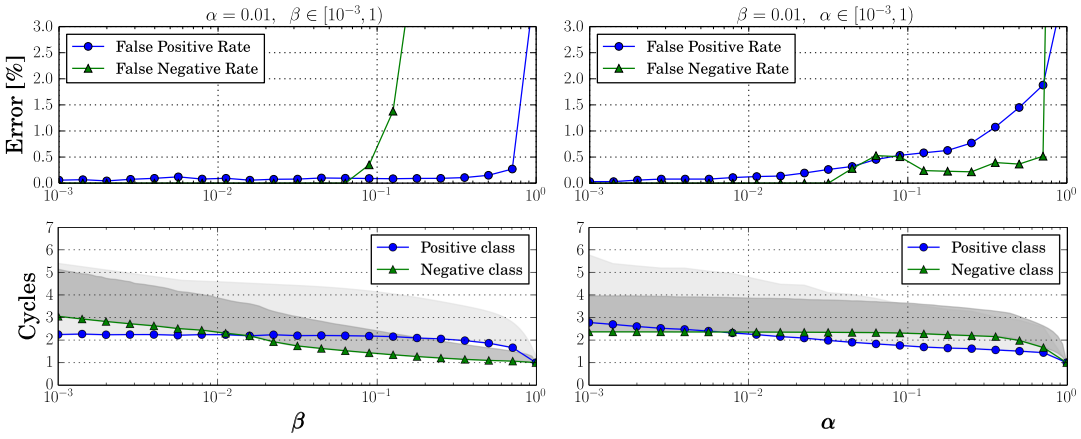


FIG. 1.15: Results of the multi-stage identification framework. False positive and negative rates are shown in the top graphs, the number of walking cycles required to make a final decision on the user's identity is shown in the bottom ones. Upper shaded areas extend for a full standard deviation from the mean and include about 80% of the events.

than five walking cycles in 80% of the cases. This means that the framework is very accurate and at the same time fast. The best results that were obtained in previous papers lead to error rates ranging from 5 to 15% [13–18]. Nevertheless, a direct comparison with these approaches is very difficult to carry out due to the different datasets (e.g., number of subjects and walking time), acquisition settings (e.g., smartphone or sensors location). The reader can refer to Section 1.3.2 for a fair comparison between the single-step classification framework and classical feature extraction techniques on our dataset.

As for the given assumptions, in light of the small number of cycles required, it is reasonable to presume that the same subject generates the scores in $\mathcal{O}$. For the i.i.d. assumption, the decision framework has been extended to the first-order autoregressive model of [58, Chapter 3, p. 158], which allows tracking the correlation across successive cycles. However, this did not lead to any appreciable performance improvement and only implied a higher complexity.

1.6 Discussion

In this chapter a user identification framework for inertial signals acquired from smartphones has been presented. Various schemes performing manual feature extraction and using the selected features for user classification have appeared in the recent literature. In sharp contrast with these, IDNet

exploits convolutional neural networks, as they allow for an automatic feature engineering and have excellent generalization capabilities. These deep neural networks are then used as universal feature extractors to feed classification techniques, combining them with one-class support vector machines and a novel multi-stage decision algorithm. With this framework, the neural network is trained once for all and subsequently utilized for new users. The one-class classifier is solely trained using motion data from the target subject; it returns a score weighing the dissimilarity of newly acquired data with respect to that of the target. Subsequent scores are then accumulated through a multi-stage decision approach. Experimental results show the superiority of IDNet against prior work, leading to misclassification rates smaller than 0.15%, achieved in fewer than five walking cycles for most of the experiments. Design choices and the optimization of the various processing blocks were discussed and compared against existing algorithms.

Chapter 2

ECG Signal Analysis for Early Diagnosis of Heart Diseases

Deep learning (DL) has amply demonstrated its knowledge extraction capabilities in the identification of features for the classification of clinical images [59]. For example, DL has been shown to possess classification skills on par with board-certified ophthalmologists for detecting diabetic retinopathy and macular edema from retinal fundus images [60]. In [61], DL reached a level of accuracy on a par with dermatologists in the identification and classification of skin cancers.

While DL has been extensively utilized for image processing, the same does not hold true for biomedical time series, for which the use of these techniques is still mostly unexplored. With this study, the purpose is to fill this gap by carrying out a comprehensive analysis among advanced representation learning methods for the classification of short one-dimensional (1D) signals. Since 1D signal analysis bears relevant differences with respect to image processing, the techniques have to be adapted to this domain.

The main focus of this study is on an important clinical problem, the detection of atrial fibrillation (AF), which is the most common serious abnormal heart rhythm, affecting about 34 million people in the world (between 2 and 3% of the population in Europe and North America) [62]. AF is characterized by rapid and irregular beating of the atria, which is often asymptomatic, so it may not be diagnosed and the affected subjects may be unaware of this condition. Since approximately 10 to 20% of ischemic strokes are associated with AF first diagnosed at the time of stroke, the early detection of asymptomatic AF is of crucial importance, as it can help prevent strokes by instituting appropriate preventive anticoagulation [63].

The detection of AF often requires: 1) long-term monitoring, up to a few

weeks, since a subject may be affected by paroxysmal AF and experience an AF event only occasionally; and 2) accurate electrocardiography (ECG) monitoring, since an AF event may be easily confounded with noise, so accuracy in the ECG monitoring is of great importance. These two conditions make the detection of AF a relatively time-intensive and expensive procedure using current methods.

On the other hand, wireless and wearable devices make it easier to monitor ECG for long time spans, although, with current technology, the quality of the signals collected in free-living conditions is not always good enough to permit an accurate detection of AF events. In order to test the accuracy of AF detection algorithms with short and noisy ECG signals, PhysioNet launched the Computing in Cardiology Challenge 2017 [64], making a large dataset of labeled ECG records publicly available. Several approaches have been proposed to detect AF events and other arrhythmias using this dataset. Among the most successful studies, [65] proposes a set of high-level and clinically meaningful features to train two different classifiers: one evaluates the trace globally, and the other analyzes per-beat features as a sequence. The two classifiers are then combined for the final evaluation [65]. In [66], DL features are also combined with expert features to train a gradient boosting classifier. In [67], the authors split the multi-class classification problem into hierarchical binary classifiers based on several manual features to improve the classification performance. In [68], a subset of significant features have been selected from a very large set of hand-crafted features, and used to train a random forest classifier for the final task.

A substantial knowledge on the signals and clinical expertise is often required to achieve a good performance, including morphological information, time-frequency representation and statistical analysis. Although some DL approaches have been proposed in the past [69, 70], an exhaustive study on pure representation learning techniques, and the comparison among their modern implementations and algorithms based on expert features is still missing. This work is an attempt to fill this gap, by adapting the most successfully DL techniques to detect AF from the data collected by wireless sensors, comparing different DL solutions against standard approaches that exploit *manually engineered features*.

The study starts with an overview of state of the art signal processing techniques, that are used as a benchmark for comparison. These methods are

based on the selection of a set of *expert features*, that are suggested by clinical studies and connected to the rhythm / morphology of the ECG, and on the automatic detection of such features in the signal [71–75]. These approaches, referred in the following to as *feature engineering*, require a specific domain knowledge to transform raw (unprocessed) data into a suitable representation that is able to generalize on unseen data, and used to train a classifier. A detailed review of past research is provided in Section 2.4.1.

As a promising alternative, in the present study the use of DL for the detection of AF events is investigated, exploiting advanced convolutional neural network (CNN) architectures [47, 76–79]. CNNs have been widely investigated in the computer vision field, but an in-depth analysis targeted at biomedical 1D signal processing is still missing. Here, effective CNN architectures for ECG analysis are investigated, focusing on AF detection. Notably, with DL techniques the features are automatically identified, without requiring any prior expert knowledge.

Some of the main advances introduced by this study are summarized next.

- A uniform mathematical framework is reviewed and provided for the most common feature engineering approaches used in the literature for ECG signal analysis.
- Five different DL methods are proposed, originally used within the image processing field, providing insightful implementation details on the exploitation of such models to analyze short intervals (with variable length) of 1D ECG signals.
- The five DL methods are used to discriminate between short ECG signals corresponding to normal sinus rhythm, atrial fibrillation, other arrhythmias and noise; comparing their performance in terms of detection accuracy (precision and recall), complexity and training time.
- The capability of representation learning techniques to extract meaningful information directly from raw data is quantified, showing the benefits against standard methods based on manually engineered (expert) features.

This chapter is structured as follows. The dataset is presented in section 2.1 (Section 2.1.1), along with the pre-processing to remove noise and

artifacts (Section 2.1.2), the state of the art approach using expert features (Section 2.1.3), a novel approach involving DL techniques (Section 2.1.4) and the training method for the considered classifiers (Sections 2.1.5 and 2.1.6). The performance metrics are defined in Section 2.2, whereas the numerical results are shown in Section 2.3. In Section 2.4, the literature on AF detection is analyzed, including a discussion on the value and the potential of DL-based feature extraction techniques.

2.1 Methods

The proposed analysis of ECG signals is organized into three phases: 1) preprocessing of the signal, 2) feature extraction, and 3) classification. Next, these phases are described and two approaches for feature extraction are proposed: 2.a) a classical feature engineering approach, and 2.b) a novel representation learning approach. For both approaches, several solutions are investigated and compared.

2.1.1 Single-lead ECG dataset

The data used in this study has been made publicly available as part of the 2017 PhysioNet and Computing in Cardiology Challenge. All the traces have been acquired using AliveCor's single-channel ECG wireless sensors (generations one to three), which can record short ECG signals by touching two electrodes with the fingertips. To use such device, the subject should hold one electrode in each hand, creating a single-lead equivalent ECG. Some of the ECGs were inverted, since the device does not have a preferred orientation [64].

This dataset is divided into a public set with 8,528 ECG records and a hidden set of 3,628 records (not publicly available), sampled at $f_s = 300$ Hz with different duration (from 9 to 61 seconds). Each record has been classified using ten different algorithms and a set of at least three experts[1] into one of these four classes: normal sinus rhythm (5,154 samples), AF (771 samples), other type of arrhythmias (2,557 samples) and noisy data (46 samples) [2]. Example ECG traces from these categories are shown in Fig. 2.1.

[1] The details of the labeling procedure can be found in [64].

[2] The number of samples is referred to the public dataset only. The hidden set was not released, and it was used by us only once to test the algorithm.

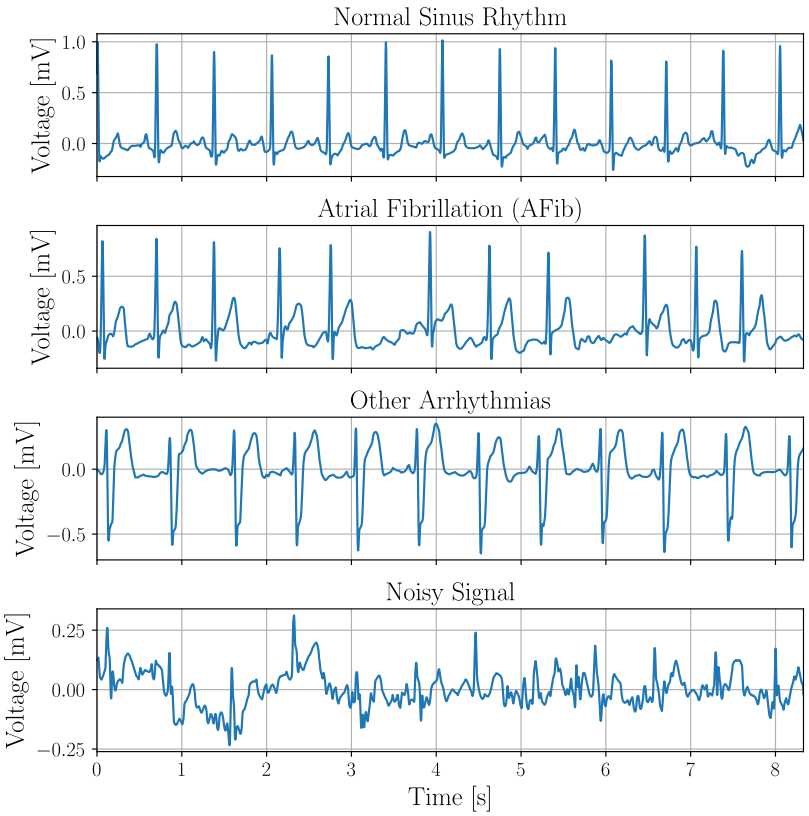


FIG. 2.1: Some examples of signals belonging to the four classes of the dataset: normal sinus rhythm, atrial fibrillation, other arrhythmias and noise.

2.1.2 Pre-processing

The ECG signal intervals have different lengths and they are affected by noise and artifacts. Before the feature extraction phase, they need to be pre-processed to improve the performance of the classifier. Three operations should be considered: 1) a baseline wander removal, 2) an element-wise normalization, and 3) a random cropping.

Baseline wander removal

The baseline wander is a low frequency artifact, present in most ECG traces, which can be the result of movement and breathing. These fluctuations may be larger than the useful signal, particularly in free-living conditions. Several approaches have been proposed, e.g., simple Butterworth filter [80], spline approximation [81], and moving median subtraction [82]. In this study, a high-pass filter based on wavelet transformation [83] is used, as it excels in the preservation of the signal's morphological traits [84]. First, the acquired ECG signal is decomposed using a discrete wavelet transform (DWT) at a

predefined decomposition level L. Then, the approximation coefficients associated with the last level L (corresponding to the lowest frequency) are set to zero. Finally, the inverse DWT is used to obtain the filtered signal. The decomposition level L corresponds to a cut-off frequency f_c, obtained as

$$L = \log_2(f_s/f_c) - 1\,, \tag{2.1}$$

where f_s is the sampling frequency. The frequency of the baseline wander for ECG signals is usually between 0.1 Hz (in the absence of movement) and 0.65 Hz (during a stress test) [83]. In this study, $f_c = 0.3$ Hz is considered, which leads to $L = 9$ for $f_s = 300$ Hz. The wavelet used is Daubechies 9.

Element-wise normalization

The ECG signal is usually affected by many sources of noise (electrode contact noise, baseline drift, motion artifacts, instrumentation noise, power line and electromyography interference), and other unpredictable factors due to the subject's position or the electrodes' coupling. Amplitude normalization typically consists of a linear rescaling leading to unbiased unit norm signals. Since this normalization is independently applied to each trace, it is referred to as *element-wise normalization*, to distinguish it from the batch normalization used in the DL architectures.

Signal cropping

All the architectures considered in this study are optimized for a fixed size input. If the original ECG signal is longer that 30 seconds, it is cropped at random to an interval of 30 seconds, while if the original ECG signal is shorter, the interval is padded with zeros to match the desired input dimension.

2.1.3 Feature extraction: feature engineering approach

In this section, the process to extract the expert features from the ECG signal is detailed, as summarized in Fig. 2.2.

RR interval-based features

The first phase is the detection of the R-peaks, corresponding to the highest electrical activity during the ventricular depolarization. A method based

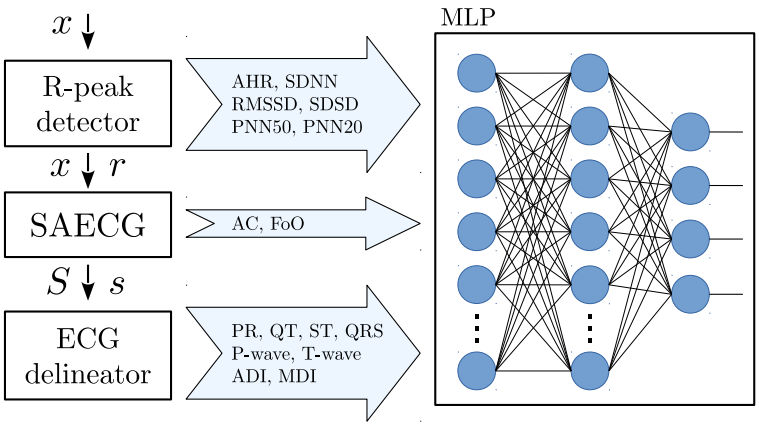


FIG. 2.2: Diagram of the expert feature based model.

on two moving average filters has been used, leading to high accuracy results with a low computational effort [85]. The output is a sequence of time instants representing the R-peaks location, as shown in Fig. 2.3.

The following heart rate variability features have been considered:

- *AHR* - the average heart rate, obtained from the mean time between two consecutive R-peaks (RR interval);
- *SDNN* - the standard deviation of the RR intervals;
- *RMSSD* - the root mean square of the differences between adjacent RR intervals (SD);
- *SDSD* - the standard deviation of SD;
- *PNN*50 - the fraction of successive RR intervals that differ by more than 50 ms;
- *PNN*20 - the fraction of successive RR intervals that differ by more than 20 ms.

Signal averaged ECG features

The signal averaged ECG (SAECG) is used to cope with the noise present in single-lead ECG recordings, a denoising method that has already shown its effectiveness in ECG signal processing [86]. Given an ECG signal $\boldsymbol{x}$ with N samples, the ECG trace $\boldsymbol{x} = [x(1), \ldots, x(N)] \in \mathbb{R}^N$ is divided into R intervals, where each interval contains a single QRS complex. Hence, the intervals are aligned (based on the location of the R peaks) and averaged to obtain a single characteristic ECG shape. The algorithm entails the following steps.

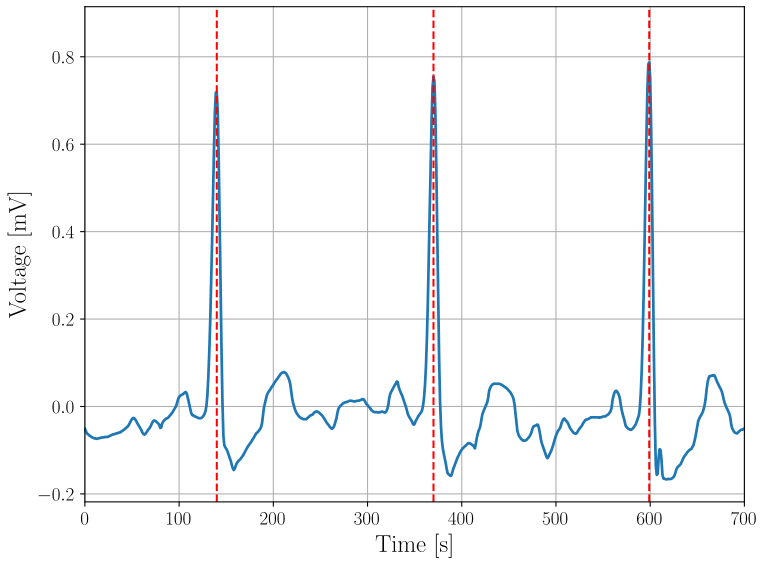


FIG. 2.3: Example ECG trace from the dataset. The vertical red lines represent the detected R-peaks.

1. The R-peak sequence $\boldsymbol{r} = [r_i, \ldots, r_R]$, $\boldsymbol{r} \in \mathbb{N}^R$ should be identified, where r_i is the index of the i^{th} peak in $\boldsymbol{x}$, and R is the total number of peaks.

2. A set of R intervals $\mathcal{Q} = \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_R\}$ is extracted from the ECG trace $\boldsymbol{x}$, a single interval per peak in $\boldsymbol{r}$. The samples in the i-th interval are:

$$\boldsymbol{q}_i = [q(r_i - M), \ldots, q(r_i + M)]\,,\ i = 1, \ldots, R\,, \tag{2.2}$$

   where $\boldsymbol{q}_i$ has size $2M + 1$, is centered around the i-th ECG beat, and extends M samples to the left and right sides of it. M is a predefined beat length, that is here set to $M = 0.7 \times \text{RR}$, i.e., 70% of the average RR interval.

3. The Pearson correlation coefficient is evaluated for each pair of elements in set $\mathcal{Q}$, obtaining a correlation matrix $\boldsymbol{C} = [C(i,j)]$.

4. The interval $\boldsymbol{q}_{i^*}$ with the highest average correlation among all the other intervals in $\mathcal{Q}$ is selected as:

$$i^* = \underset{i}{\texttt{arg}\,\texttt{max}}\ \frac{1}{R-1} \sum_{j=1, j\neq i}^{R} C(i,j)\,. \tag{2.3}$$

   $\boldsymbol{q}_{i^*}$ is then added to a new (initially empty) set $\mathcal{S}$, and removed from $\mathcal{Q}$.

5. The representative interval $\boldsymbol{s} = [s(1), \ldots, s(2M+1)]$, $\boldsymbol{s} \in \mathbb{R}^{2M+1}$, is obtained by averaging all the intervals in set $\mathcal{S}$.

6. A correlation vector $\boldsymbol{c_s}$ is evaluated between $\boldsymbol{s}$ and all the elements in set $\mathcal{Q}$. Hence, the interval $\boldsymbol{q}_{i^*} \in \mathcal{Q}$ with the highest correlation with $\boldsymbol{s}$ is selected.

7. If the correlation between $\boldsymbol{q}_{i^*}$ and $\boldsymbol{c_s}$ is greater than a predefined threshold γ, $\boldsymbol{q}_{i^*}$ is added to the set $\mathcal{S}$ and removed from $\mathcal{Q}$, and the procedure is repeated from point 5).

The outputs of this method are the set $\mathcal{S} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{R_\mathcal{S}}\}$, which contains all the $R_\mathcal{S} = |\mathcal{S}|$ beats ($R_\mathcal{S} \leq R$) that are considered for the evaluation, and the representative beat $\boldsymbol{s}$. A correlation threshold $\gamma = 0.7$ has been chosen to exclude artifacts and false positive beats, while considering intervals with reasonable morphological differences.

Two features are extracted during this process:

- *AC* - the average correlation, evaluated over all pair of elements in set $\mathcal{S}$.

- *FoO* - the fraction of intervals that were originally in $\boldsymbol{x}$, but that are not included in set $\mathcal{S}$, which are considered as outliers.

Following this, a wavelet-based ECG delineator is implemented to identify the start and end of the P-wave (P_s and P_e, respectively), the T-wave (T_s and T_e), and the QRS complex (QRS_s and QRS_e), see Fig. 2.4.

The wavelet transform provides a description of the signal in the time-frequency domain, allowing the analysis of the temporal features of a signal at different resolutions [87]. It can be observed that carrying out a wavelet-based analysis on the raw signal sometimes produces inaccurate localization of the points of interest (start and end points of ECG waves). The SAECG technique allows for an increased accuracy and the use of the representative signal $\boldsymbol{s}$ reduces the effect of sensor noise and artifacts. As shown in Fig. 2.4, by analyzing the signal $\boldsymbol{s}$, we can determine:

- $PR_i = QRS_s - P_s$, the period between the beginning of atrial (P-wave onset) and ventricular depolarization (QRS complex onset), which represents the time needed by the action potential to propagate from the sinoatrial (SN) node to the atrioventricular (AV) node.

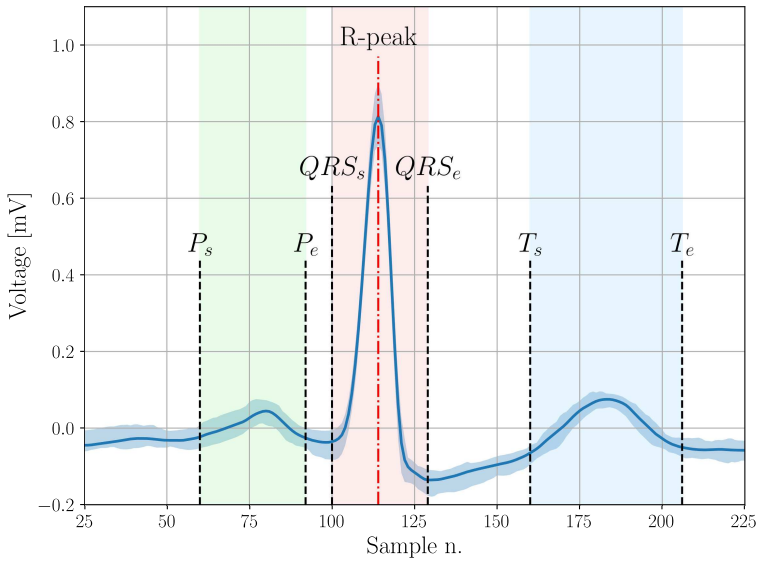


FIG. 2.4: SAECG example where the points extracted by the ECG delineator are indicated in the figure and the shaded area represents the interquartile range of the aligned beats.

- $PR_s = QRS_s - P_e$, the time from the end of the atrial depolarization (end of the P-wave) and the beginning of the ventricular depolarization (QRS onset). During this period, no electrical activity is measured in normal conditions.

- $QRS = QRS_e - QRS_s$, the QRS complex duration, which represents the time for a complete ventricular depolarization.

- $QT = T_e - QRS_s$, the time from the beginning of the QRS complex to the end of the T-wave. The T-wave is related to the ventricular repolarization, the duration of which depends on the cells resistivity. If some cells present a higher resistivity than normal, for example due to previous ischemic episodes, a prolonged QT interval would be measured.

- $ST = T_s - QRS_e$, the period during which the ventricles are completely depolarized (end of the QRS complex, also known as *J-point*), waiting for the subsequent repolarization, which starts with the T-wave.

Furthermore, the amplitude of the P and T waves are also included in the feature set.

The main issue with the evaluation of the SAECG is the loss of information on inter-beat morphology variability, which represents an important factor to consider for arrhythmia detection. To overcome this limitation, a set $\mathcal{S}$ is utilized to evaluate an amplitude dispersion index (ADI) [88], both for

HR analysis	Delineation	SAECG	P-wave	T-wave
AHR	PR_i	AC	amplitude	amplitude
$SDNN$	PR_s	FoO	ADI_{max}	ADI_{max}
$RMSSD$	QRS		ADI_{avg}	ADI_{avg}
$SDSD$	QT			
$PNN50$	ST			
$PNN20$				

TABLE 2.1: Set of features for the feature engineering approach.

the P-wave and the T-wave. For the P-wave, the SAECG signal $\boldsymbol{s}$ is first considered and the set of N_P indexes corresponding to the P-wave are selected, where $N_P = P_e - P_s$ is the total number of samples in the P-wave. Thus, a new set of vectors is built, $\mathcal{P} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{R_\mathcal{S}}\}$, with the same cardinality of $\mathcal{S}$, whose elements only contain P-wave samples,

$$\boldsymbol{p}_i = [s_i(P_s), s_i(P_s+1), \ldots, s_i(P_e)]\,,\ i = 1, \ldots, R_\mathcal{S}\,. \tag{2.4}$$

That is, $\boldsymbol{p}_i$ is a masked version of $\boldsymbol{s}_i$, which only contains the portion of $\boldsymbol{s}_i$ between the start (P_s) and end (P_e) indexes of the P-wave. Then, a dispersion measure for each $\boldsymbol{p}_i$ is evaluated as

$$AD(j) = \underset{i}{\texttt{iqr}}(\ \boldsymbol{p}_i(j)\ ),\ j = 1, \ldots, N_P\,, \tag{2.5}$$

where $\texttt{iqr}(\cdot)$ is the inter-quartile range operator. Finally, all the $R_\mathcal{S}$ elements in $\mathcal{P}$ are considered to calculate the maximum and the average value of the dispersion, i.e.,

$$ADI_{\text{max}} = \underset{j}{\texttt{max}}\, AD(j),\ j = 1, \ldots, N_P\,, \tag{2.6}$$

$$ADI_{\text{avg}} = \frac{1}{N_P}\sum_j AD(j),\ j = 1, \ldots, N_P\,. \tag{2.7}$$

A similar procedure is used to calculate the above measures for the T-wave. In the final feature vector, ADI_{max} and ADI_{avg} are included for both the P-wave and the T-wave.

The full set of features considered for the feature engineering approach is summarized in Tab. 2.1.

Classifier

A multilayer perceptron (MLP) neural network has been trained to compare the discriminative power of expert features against those learned by the deep neural network architectures of the following sections. The considered MLP has 2 fully connected hidden layers with 256 neurons each, and rectified linear unit (ReLU) is used as activation function [89].

2.1.4 Feature Extraction: Deep-learning architectures

Most of the existing convolutional neural networks implementations are designed to cope with bi-dimensional data (images). Two main workflows can be identified to adapt these implementations to the analysis of single-lead (1D) ECG signals.

W1) The data can be mapped into a time-frequency representation and then treated as a bi-dimensional signal, as usually done with speech analysis. The choice of this input transformation highly impacts the final performance.

W2) All the operations can be performed in the original one-dimensional space, giving more flexibility to the network, at the price of a more difficult training.

In this study, W2 is used in this respect, since the ECG analysis requires a fine temporal resolution, and time-frequency representations typically entail a loss in that domain. Furthermore, the data dimensionality would make the training intractable using (W1) with a high time resolution, and a heavy redesign would be required, which is left to future investigations.

In the following, I delve into the details on the proposed DL architectures, which are shown in Fig. 2.5.

Alexnet

The architecture in Fig. 2.5-(a) was the first deep convolutional network to be successfully trained on a very large dataset (several millions of images) for the classification of a thousand different classes [47]. It consists of five convolutional layers, followed by two fully-connected ones, where the non-linear activation functions at the output of each layer are ReLU. Max-pooling is

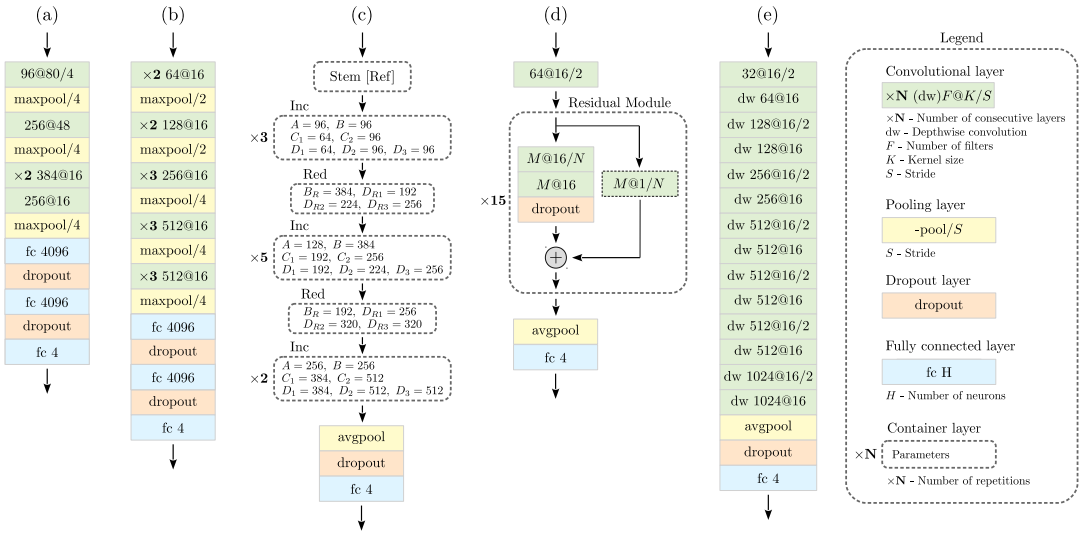


FIG. 2.5: Network schemes for all the considered architectures: AlexNet (a), VGG (b), Inception (c), ResNet (d) and MobileNet (e). The notation used is detailed in the legend on the right. Refer to Fig. 2.6 for the implementation of the Inception modules.

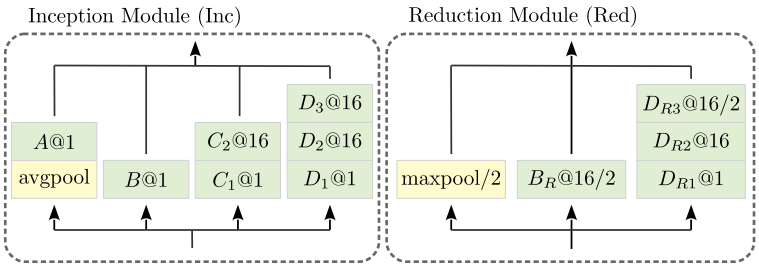


FIG. 2.6: Basic modules used by the Inception network.

used as a dimensionality reduction method and to increase the spatial invariance of features. *Dropout* is applied at the output of the fully-connected layers during training to mitigate *overfitting* (regularization technique). It simply drops the contribution of each neuron of the previous layer, with a given probability [90]. The original implementation makes use of 11×11, 5×5 and 3×3 convolution kernels, which have been respectively substituted by 1×80, 1×48 and 1×16 in the mono-dimensional version. After an exploratory phase, it has been found that a mono-dimensional kernel of size 1×16 is a good compromise between number of parameters and computational power, and it is adopted here to replace the bi-dimensional 3×3 kernel, used in the vast majority of the modern convolutional network implementations.

In Fig. 2.5-(a), it can be observed that the convolutional kernel size K decreases layer after layer. At the same time, the number of filters increases,

and the data dimensionality is consistently reduced to obtain a more abstract and compact representation of the data after each layer.

Visual Geometry Group (VGG)

The architecture in Fig. 2.5-(b) increases the depth of the network while keeping the same basic structure of the previous model. In the proposed implementation, a total of 13 convolutional layers with constant kernel size K are considered. It has been shown that stacking several smaller convolutions the receptive field considerably increases with the depth, leading to better performance than with a single large convolution [76]. The drawback is a significant increase in the complexity (i.e., in the number of network parameters, the weights), with a higher risk of overfitting. Substantial dimensionality reduction is performed using max pooling.

Inception

The architecture in Fig. 2.5-(c) is designed to deal with two of the issues encountered with deeper networks. (I1) Overfitting, due to a high number of parameters. (I2) Higher complexity, since the computational requirements increase quadratically with the number of filters [91]. It has been observed that part of these computations may not be required and this leads to sparse parameter matrices (with many weights either zero or close to zero). Unfortunately, the numerical computation on sparse data is inefficient. The Inception network is an attempt to approximate sparse convolutions through more efficient and dense blocks. In particular, the computation is split into different parts, each one with its specific purpose. In this implementation, four parallel branches are considered for each layer (Fig. 2.6), the outputs of which are concatenated to form the input of the next layer, as detailed in Fig. 2.5. A key task is performed by 1×1 convolutions, which are used to efficiently map the local relations among the filters onto a reduced space. This procedure can be intuitively represented as a compression, and the key point is to approximate sparse operations in the original space with dense operations in the compressed space. Each of the four branches performs a different computation. In the *inception module* (Fig. 2.6, from left to right), the network entails an average pooling, to increase the spatial invariance, a pure 1×1 convolution, a single and a two-layer convolution to process local correlations. Even though a 1×1 convolution is used in each branch, it is worth noting that,

after the final concatenation, the input and the output dimensionality are exactly matched, having chosen appropriately the parameters of the inception modules (Fig. 2.5). The dimensionality reduction task is indeed carried out by a separate *reduction module* (Fig. 2.6), which reduces the input dimension and at the same time increases the number of filters, following the typical behavior of a deep network. This implementation is based on [78]. Finally, an average pooling along the feature maps before the last classification block is included, as it has been shown to yield better results [92].

ResNet

The architecture in Fig. 2.5-(d) is designed to address the following issues of very deep models: (I1) The gradient becames exponentially small with the network (backpropagation) depth, preventing the parameters to be effectively updated (*vanishing gradients* [93]). (I2) The model has too many parameters, so the network becomes harder to train, leading to sub-optimal results or lack of convergence (*degradation problem*). Residual layers have been introduced to mitigate these problems by adding an identity mapping on each layer. In this way, instead of learning the direct relation between the input and the output, these modules learn the residual representation, hence the name. If a layer, for any reason, is not able to be effectively trained, the identity mapping allows for the propagation of the previously processed information, making it possible to optimize extremely deep architectures without suffering from the degradation problem. Furthermore, these modules also address the issue of vanishing gradients. The presence of *shortcut connections* allows the gradients to easily propagate back through the network. This implementation is inspired by [94], where the authors obtained promising results for ECG analysis. The number of filters (parameter M in Fig. 2.5-(d)) is doubled every 4 layers, and the signal dimensionality is halved every 2. A 1×1 convolution is optionally used among the shortcut connection to match the input and output dimensions, if necessary.

MobileNet

Two final issues are addressed by the architecture in Fig. 2.5-(e). (I1) The large number of parameters, and (I2) the high computation demand. MobileNet exploits *depthwise separable convolutions* to address them [79]. Let consider a standard convolution with F_I input channels, F_O output channels, and K total

kernel parameters. The total number of parameters required for this operation is $F_I \times F_O \times K$. MobileNet splits this computation into two steps. (S1) A *depthwise convolution* is applied to the input data, which performs a spatial convolution independently over each channel, entailing $F_I \times K$ weights. (S2) A *pointwise convolution*, i.e., a 1×1 convolution, is used to project the output into the new channel space, considering also the relations among the channels, with $F_I \times F_O$ weights. The total number of parameters is $F_I(K + F_O)$, with a reduction in computation of a factor $1/F_O + 1/L$, where L represents the dimensionality of the input feature maps [79]. As for the residual network implementation, no pooling is used between the layers; instead, a stride operation is used to reduce the dimensionality, which further decreases the computational requirements [95].

2.1.5 Classification

A final classification module is concatenated to each of the above described neural network architectures. This block consists of a linear fully-connected layer with C neurons, where C corresponds to the number of classes. A `softmax` block is applied to the output of this linear layer: since the classes are mutually exclusive, the `softmax` activations return a C-dimensional output vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_C)$, whose element $y_j \in [0, 1]$ represents the estimated probability that the input ECG sample belongs to class j, with $j = 1, 2, \ldots, C$ and $\sum_{j=1}^{C} y_j = 1$ (stochastic vector).

2.1.6 Training procedure

The training procedure described in this section is adopted for the MLP with expert-based features, and for the classification approach based on representation learning through deep networks. The ECG signal classifier is obtained concatenating the following blocks: 1) expert features: the MLP is concatenated with the classification block of Section 2.1.5, 2) deep networks: each of the CNN architectures of Fig. 2.5 is concatenated with the classification block of Section 2.1.5. Learning, in both cases, consists of training the network parameters (weights) of the classifiers through gradient descent.

Specifically, a stochastic gradient descent optimization method with Nesterov accelerated gradient [96] is used to minimize a weighted cross-entropy loss function, iteratively updating the weight vector $\boldsymbol{w}$ of the network. Given

an input ECG sequence $\boldsymbol{x}$, the classifier acts as a function mapping $\boldsymbol{x}$ into a stochastic vector $\boldsymbol{y}(\boldsymbol{x}, \boldsymbol{w}) = (y_1, y_2, \ldots, y_C)$, whose j-th entry $y_j(\boldsymbol{x}, \boldsymbol{w})$ represents the estimated probability that the input $\boldsymbol{x}$ belongs to class j, with $j = 1, \ldots, C$, where C is the number of classes. Since the dataset is labeled, each input example $\boldsymbol{x}$ has an associated (ground truth) class label c, which has been assessed as described in Sec. 2.1.1. The loss function for a given generic sample $\boldsymbol{x}$ of class c, is:

$$f(\boldsymbol{x}, c, \boldsymbol{w}) = \alpha_c \left[-\log \left(\frac{e^{y_c(\boldsymbol{x}, \boldsymbol{w})}}{\sum_j e^{y_j(\boldsymbol{x}, \boldsymbol{w})}} \right) \right], \tag{2.8}$$

where α_c is the class weight, which is used to cope with imbalanced datasets and is here computed as the fraction of samples of class c over the total number of samples in the training set.

If $\mathcal{X}$ is the set of all training examples, let's define the batch set as $\mathcal{B} \subset \mathcal{X}$, with cardinality $B = |\mathcal{B}|$. The optimization method minimizes a further loss function, obtained averaging $f(\boldsymbol{x}, c, \boldsymbol{w})$ over the batch set $\mathcal{B}$, and adding a weight regularization term $\|\boldsymbol{w}\|^2$:

$$F(\mathcal{B}, \boldsymbol{w}) = \frac{1}{B} \sum_{(\boldsymbol{x}, c) \in \mathcal{B}} f(\boldsymbol{x}, c, \boldsymbol{w}) + \beta \|\boldsymbol{w}\|^2, \tag{2.9}$$

where the regularization parameter β is implemented as a weight decay during the updates.

Batch normalization [97] has been implemented after each convolution and before the ReLU activation function, as it increases the training stability and also acts as a regularizer.

2.2 Performance metrics and statistical analysis

For all the learning procedures, the public available dataset of 8,528 ECG records has been uniquely used, while no access is allowed to the hidden test set.

A 5-fold cross validation approach has been used for the presented numerical results: the public dataset has been randomly split into 5 subsets, maintaining for each the original distribution between the classes. One subset is used as a test set (subset $\mathcal{T}$), while the remaining ones constitute the training and validation subsets.

Each test sample $\boldsymbol{x} \in \mathcal{T}$ is associated with a ground truth label $c(\boldsymbol{x}) \in \mathcal{C} = \{C_S, C_A, C_O, C_N\}$, corresponding to normal sinus rhythm, AF, other arrhythmias, or noise, respectively. The output of the classifier is another label $c_p(\boldsymbol{x}) \in \mathcal{C}$, which represents the predicted class.

Precision and recall have been considered as the performance metrics. The **precision** estimates the fraction of examples correctly predicted as a specific class c_p (true positives) among all the examples predicted as c_p, and it can be interpreted as the probability that a new example with predicted class c_p, actually belongs to that class. It is defined as

$$\texttt{precision}(\rho) = \frac{\sum_{\boldsymbol{x}\in\mathcal{T}} \mathbb{1}(c(\boldsymbol{x}) = \rho) \cdot \mathbb{1}(c_p(\boldsymbol{x}) = \rho)}{\sum_{\boldsymbol{x}\in\mathcal{T}} \mathbb{1}(c_p(\boldsymbol{x}) = \rho)}, \tag{2.10}$$

where $\rho \in \{C_S, C_A, C_O, C_N\}$, and $\mathbb{1}(\cdot)$ is the indicator function, with $\mathbb{1}(Z) = 1$ if Z is true, and $\mathbb{1}(Z) = 0$ otherwise.

The **recall** estimates the fraction of samples belonging to a specific class that are correctly classified (sensitivity). It can be seen as the probability that a new example of class c will be correctly classified. It is defined as:

$$\texttt{recall}(\rho) = \frac{\sum_{\boldsymbol{x}\in\mathcal{T}} \mathbb{1}(c(\boldsymbol{x}) = \rho) \cdot \mathbb{1}(c_p(\boldsymbol{x}) = \rho)}{\sum_{\boldsymbol{x}\in\mathcal{T}} \mathbb{1}(c(\boldsymbol{x}) = \rho)}. \tag{2.11}$$

The **F1-measure** combines precision and recall through an harmonic mean, providing a good way to compare different algorithms:

$$\texttt{F1}(\rho) = \frac{2\sum_{\boldsymbol{x}\in\mathcal{T}} \mathbb{1}(c(\boldsymbol{x}) = \rho) \cdot \mathbb{1}(c_p(\boldsymbol{x}) = \rho)}{\sum_{\boldsymbol{x}\in\mathcal{T}} \left[\mathbb{1}(c_p(\boldsymbol{x}) = \rho) + \mathbb{1}(c(\boldsymbol{x}) = \rho)\right]}. \tag{2.12}$$

In order to have a global performance assessment metric, a `score` has been also defined, which is evaluated by averaging the given metric $m \in \{\texttt{precision}, \texttt{recall}, \texttt{F1}\}$ among the normal sinus rhythm, AF and other arrhythmias classes, i.e.,

$$\texttt{score}(m) = \frac{\sum_{\rho\in\{\mathcal{S},\mathcal{A},\mathcal{O}\}} m(\rho)}{3}. \tag{2.13}$$

Noisy signals have been excluded for two reasons: (R1) the number of noisy examples are extremely limited (only 46 samples to divide among training, validation and test set), and (R2) noise detection can be performed as a separated task with a dedicated algorithm, which is out of the scope of this study.

2.3 Results

The validation subset is used to compare the performance of the classification for the five DL architectures and the MLP classifier with expert features. Each neural network has been trained for a total of 200 epochs, with a step-based learning rate annealing policy, starting from $\lambda = 0.1$, and reducing this value by a factor of 3 every 25 epochs[3].

Precision and recall for each class and for each method are shown in Fig. 2.7. The performance of the feature engineering approach is significantly lower than that of all the considered DL architectures. Focusing on the AF class, a lower precision and a higher recall can be observed, corresponding to a higher misclassification rate. These findings cannot be generalized, as new expert features can be derived and added to the MLP framework, but they provide an idea of the advantages of DL architectures. In particular, they suggest that *predefined features* may not be appropriate to extract information in a realistic environment and in free-living conditions, i.e., in the presence of time-varying noise and artifacts due to the movement of the sensor. Data-driven techniques, instead, excel in such complex scenarios, where the diversity of the acquired signals could heavily affect the classification performance, being able to automatically learn the best way to fit the training data.

The performance of the classifiers is influenced by the amount of data available for each class, due to the fact that the neural networks effectiveness depend on the training data dimensionality. As expected, considering the limited data available, the learned features are not able to accurately classify noisy signals. Some improvements may be obtained using data augmentation techniques, and these improvements are left for future work.

The `score` values for precision, recall and F1-measure are reported in Fig. 2.8. From this figure, it can be observed that the classification performance increases using deeper architectures, as noticed by comparing AlexNet with VGG, or with more advanced structures, like Inception and ResNet. On the downside, deeper architectures entail a higher number of parameters and an increase in computational complexity, as highlighted in Fig. 2.9, where

[3] In a preliminary analysis it has been determined that starting with a high learning rate drives to better performance in this case. Gradient clipping has been applied to have a more stable training.

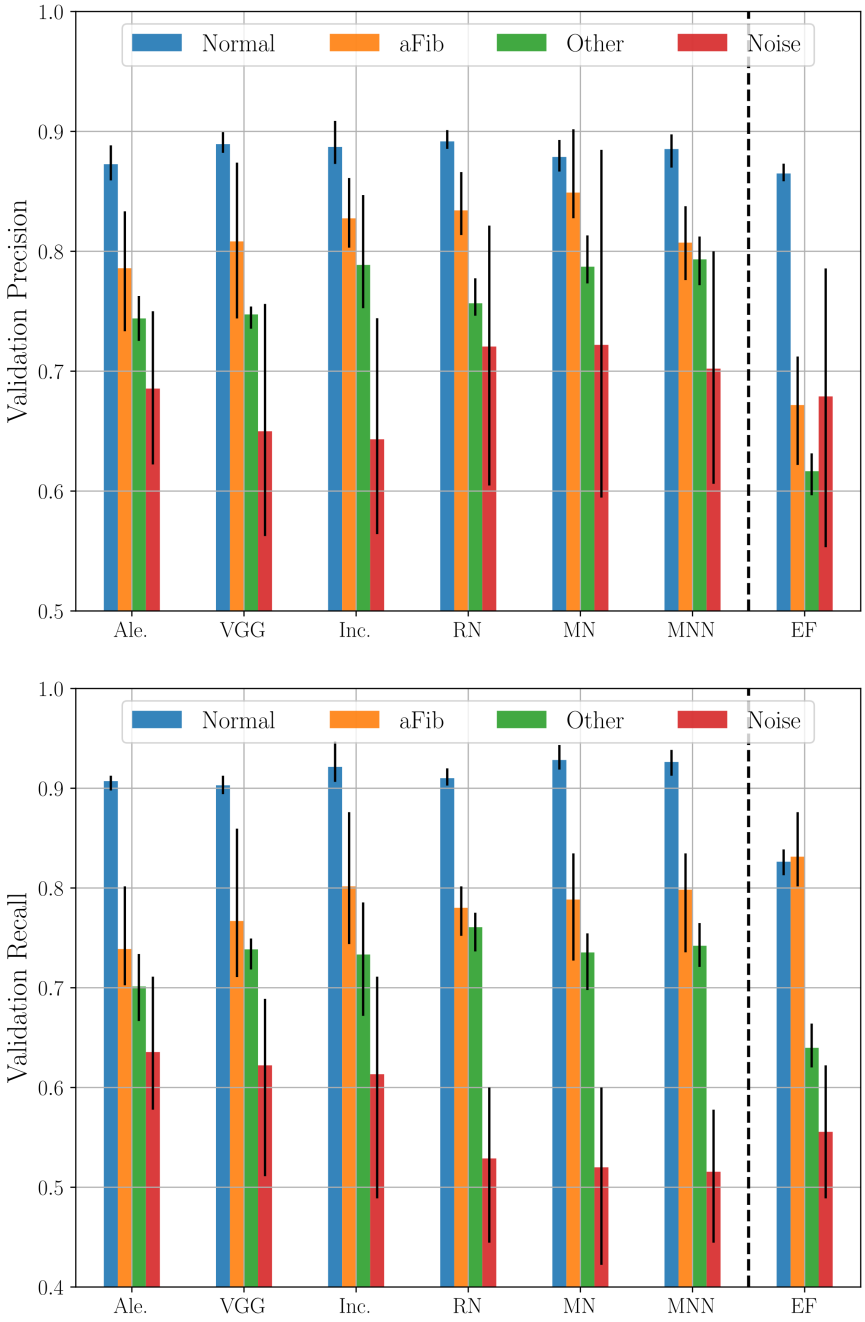


FIG. 2.7: Validation precision (top) and recall (bottom) achieved by all the models considered in this study (Ale.=Alexnet, VGG, Inc.=Inception, RN=ResNet, MN=MobileNet, MNN=MobileNet with element-wise input normalization, EF=Expert Features). All the obtained cross-validation values lie between the lines on the top of the bars; the bars represent the average value among the runs.

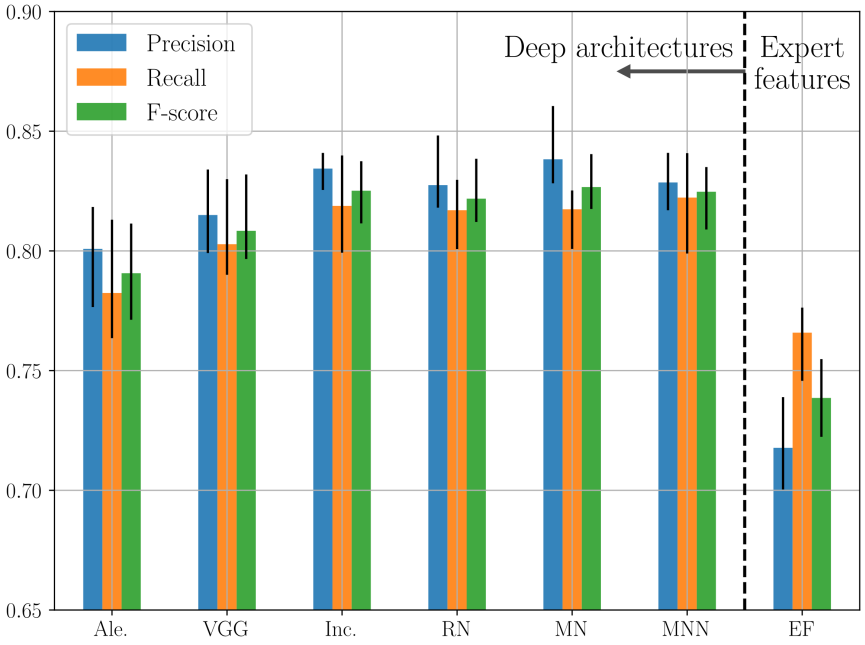


FIG. 2.8: Validation score achieved by all the models considered in this study (Ale.=Alexnet, VGG, Inc.=Inception, RN=ResNet, MN=MobileNet, MNN=MobileNet with element-wise input normalization, EF=Expert Features). All the obtained cross-validation values lie between the lines on the top of the bars; the bars represent the average value among the runs.

the number of parameters and the average training time for each epoch are shown.

A remarkable result is achieved by depth-wise separable convolutions in MobileNet, which performs on par with Inception in terms of accuracy, while leading to substantial reductions in training time and number of parameters, as shown in Fig. 2.9. For this architecture, the effects of element-wise normalizations (described in Sec. 2.1.2) are negligible, as shown in Fig. 2.7 and Fig. 2.8.

Fig. 2.10 shows the score value for the F1-measure (on the validation set) as a function of the number of epochs during the deep network training. These values can be used to quantify the effectiveness of each architecture in extracting relevant information. The VGG network is slower in converging to its optimal weight vector, due to the very high number of parameters, while ResNet performs significantly better in terms of convergence time.

To summarize, in Fig. 2.11 the five DL techniques are compared in terms of score, data efficiency (average score after 20 epochs of training), memory efficiency (number of parameters) and computational efficiency (training time). The results are normalized such that for each dimension the worst technique has a point in the inner circle, while the best one has a point in

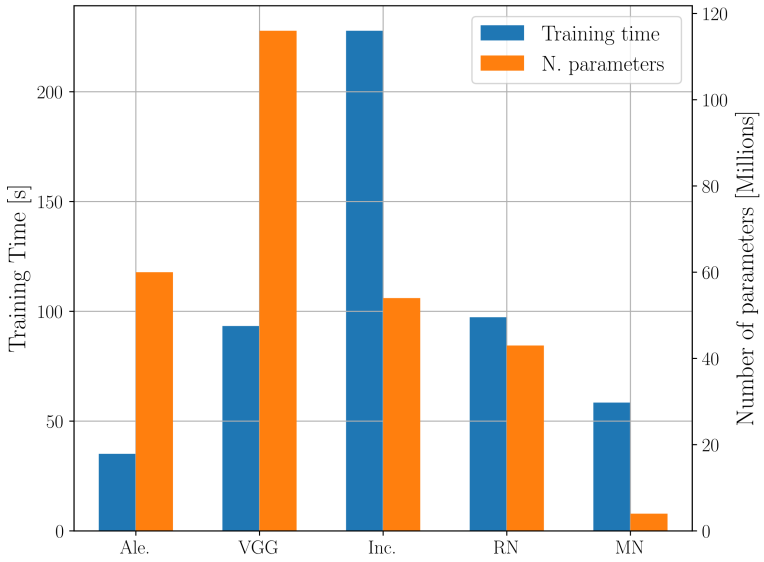


FIG. 2.9: Number of parameters and training time for all the deep learning architectures considered in this study (Ale.=Alexnet, VGG, Inc.=Inception, RN=ResNet, MN=MobileNet, MNN=MobileNet with element-wise input normalization, EF=Expert Features).

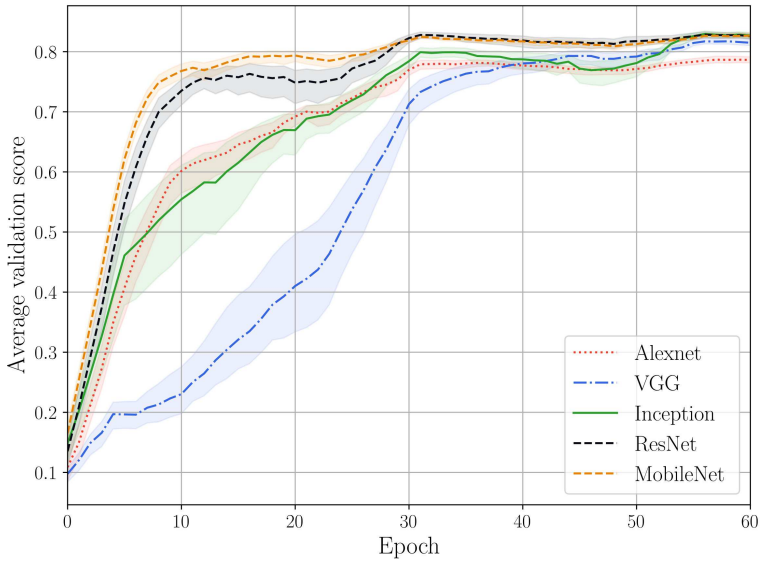


FIG. 2.10: Validation score, averaged among the classes, during the training. The shaded area represents the variance considering all the cross-validation runs.

the outer circle. The AlexNet implementation has the lowest computational complexity, but it does not perform well along the other dimensions. MobileNet performs better than the other techniques in terms of score, number of parameters and data efficiency, while in terms of computational complexity it is only second to AlexNet (although not by much). According to the

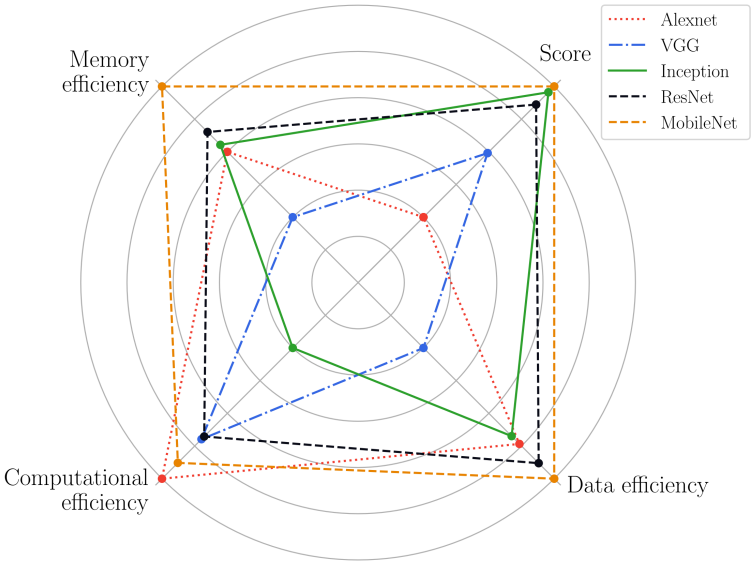


FIG. 2.11: Overall graphical comparison among the DL architectures.

discussed results, MobileNet has been chosen as the representative DL architecture for the final testing phase.

First, the same test as the other participants in the Physionet Computing in Cardiology challenge has been performed. The algorithm has been first trained using the entire public dataset, following the same guidelines described in Sec. 2.1. Then, its performance has been tested on the hidden dataset (see Sec. 2.1.1 for more details) considering the score defined in Eq. (2.13). The proposed method achieves a $\texttt{score}(\texttt{F1}) = 0.80$, very close to the best algorithm of the challenge, which performs a $\texttt{score}(\texttt{F1}) = 0.83$. The limited availability of training data negatively affects representation learning, as pointed out in [66], where the coexistence of expert features along with DL representations improves the performance of this specific task. By achieving a performance very close to the best result in the Physionet challenge, it has been shown that an approach purely based on representation learning can reach the performance of approaches that also exploit expert based features, thus requiring preliminary clinical expertise. This is an encouraging result especially for the analysis of less investigated physiological signals, for which a limited number of expert based features have been recognized.

Finally, the deep architecture based on MobileNet has been compared to the expert-based feature approach. In this last comparison, due to the unavailability of the hidden test set, the cross-validation approach is used to

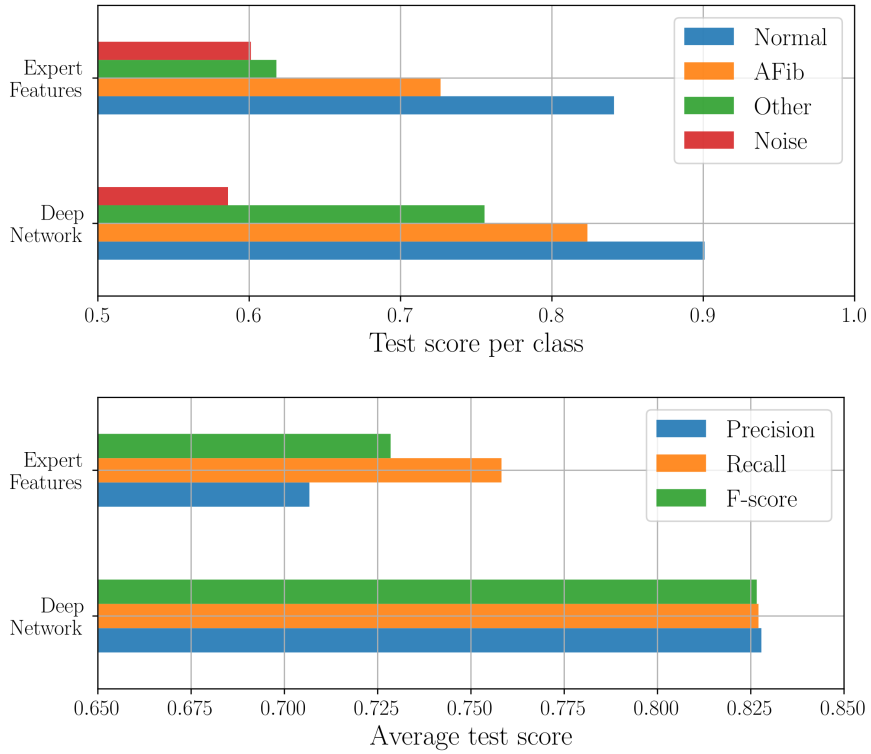


FIG. 2.12: Final performance achieved by the DL (MobileNet) and expert feature approaches. On the top figure the F1-measure for each class is presented. On the bottom, the average precision, recall and F1 scores are shown.

perform the test on all the samples in the public dataset, leading to a greater reliability of the results, while keeping a strict separation between training and testing samples. The models are trained considering a higher number of epochs (2,000), and with a more relaxed learning rate annealing (which is halved every 150 epochs).

The performance of MobileNet and of the feature engineering approach is shown in Fig. 2.12, where the F1-measure for each class and the corresponding average score are shown. Once again, the selected DL architecture (MobileNet) provides a significant improvement over both metrics.

Ultimately, let's focus on the simpler problem of AF detection, so as to reduce the problem to a binary classification task, where the goal is to determine whether the input ECG signal $\boldsymbol{x}$ presents AF or not. Under this assumption, each sample in the testing set is classified as AF if the corresponding estimated probability is greater than or equal to a threshold τ. By varying τ, a distinct value pairs for precision and recall can be measured. This procedure generates a curve in a precision-recall plane (PRC), shown in Fig. 2.13 for both the deep network and the expert features based classifier. The isocurves for three example values of the F1-measure are also shown in the figure for

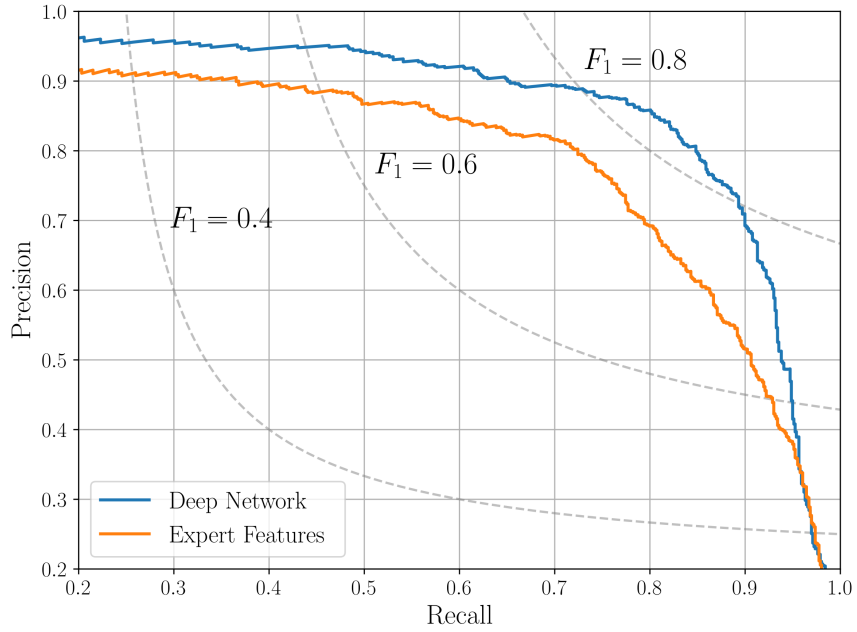


FIG. 2.13: Precision-recall curve (aFib class) for the final deep network implementation and the expert feature approach. The isocurves for some levels of F1-measure are also shown for increased readability.

increased readability. With this graph, it is possible to choose an operational point for the classifiers, based on the desired trade-off between precision and recall.

2.4 Discussion

AF is the most common significant heart arrhythmia in the population and is associated with a 5-fold increase in the risk of a stroke and with a 2-fold risk of mortality. But it is also especially well suited for screening, as there are preventative therapies available that have been shown to decrease the risk of these complications by 65 percent. In addition, a gold-standard for diagnosis exists in the form of an ECG. While in the past, the performance of an ECG was limited to the patient care setting, today there is an ever-growing number of options available for an individual to obtain their own ECG, making the need for precise automated detection of AF from individually-obtained ECGs uniquely important.

2.4.1 AF detection: Feature engineering approach

In a clinical environment, a 12-lead ECG is commonly used to observe AF and other arrhythmias for short intervals. A single-lead ECG sensor can detect AF over a longer period, monitoring the subject for, currently, up to two weeks in free-living conditions. The amount of data produced is substantial and can not be analyzed by an expert, so several techniques for an automated detection have been developed. The first efforts for automatic arrhythmia detection were based on the analysis of the heart rhythm using classical signal processing techniques, such as adaptive filtering [98], autocorrelation [99] and sequential hypothesis testing [100]. Frequency domain analysis has also been investigated, while non-linear dynamic time warping techniques have been used in [101].

Beside the heart rhythm, morphological characteristics such as the shape and position of the P and T waves (associated with the atrial depolarization and ventricular repolarization, respectively), can also contribute to arrhythmia detection [102, 103]. The classical approach for detecting AF involves the selection of a set of *features* (suggested by clinical experts) and the automatic detection of such features in the signal. Besides the heart rhythm [104], the QRS complex can be analyzed [71, 72], or more advanced features [73–75]. These approaches, being based on *feature engineering*, require a specific domain expertise to transform raw data (i.e., unprocessed data) into a suitable representation that is able to generalize on unseen data, and used to train a classifier.

2.4.2 AF detection: Representation learning approach

In this chapter, a promising approach based on artificial intelligence (AI) algorithms has been investigated. The concept of *representation learning* encompasses any method that is able to automatically discover useful representation of data for classification, detection or regression, without the need for feature engineering. This approach is particularly effective in the presence of non-clinical signals, such as the photoplethysmography (PPG) signal [105], or in the presence of short and noisy ECG signals as those that are considered in this study.

DL methods have been the most successful AI algorithms of this kind. Very complex non-linear functions can be learned by representing data through

multiple levels of abstraction, and composing non-linear transformations in deep layered structures. DL techniques have significantly improved the state of the art in speech recognition, visual object recognition, object detection and other domains [106]. On the downside, these methods require a large amount of data to achieve good results.

Four important observations can be made as a results of this analysis. First, all the proposed DL approaches resulted in superior performance compared to the expert-based approach. This is due to the fact that the signals analyzed are taken in free-living conditions, in the presence of sensor noise or external noise and artifacts. DL has shown to be very effective in these cases. Second, the analysis showed that the performance (in terms of precision and recall) vary among the considered DL networks, highlighting the advantage of deeper networks in terms of accuracy. Third, computational complexity and training time are important considerations in the choice of an algorithm, especially if the learning phase needs to be periodically repeated in the presence of new data (or new sensor characteristics). In this context, it has been showed how depthwise separable convolutions, used in the presented implementation of MobileNet, provide a significant reduction in terms of complexity. Finally, the sole use of DL learning methods leads to results very close to approaches where they are combined with complex feature engineering [66]. The availability of a larger dataset may further reduce this gap.

The final selection of MobileNet and its comparison with the feature engineering approach highlighted the superiority of the DL approach for the considered AF detection task. It should also be highlighted another important aspect, which is the flexibility of DL methods. In fact, while an expert based approach required a careful choice of features, the DL approach was automatically able to select the most important features for the targeted detection problem. Furthermore, its application to a new physiological signal with completely different characteristics, e.g., like the PPG signal, would be straightforward in the presence of a large annotated dataset.

On the other side, the main limitations of an approach based on DL should be also noted. First, it needs a large set of labeled data for proper training. For example, the limited accuracy to detect noise has been highlighted in this chapter, probably due to the limited number of noise samples in the dataset.

Second, the results of the DL approach are not easy to interpret, since the output of the algorithm is a set of probabilities (one for each class), but it is not clear which morphological features in the signal were triggering the decision.

Chapter 3

Prediction of Adverse Glycemic Events from CGM Signal

Diabetes is a metabolic disorder characterized by chronic hyperglycemia resulting from a deficient insulin production or utilization. This is due to the destruction of beta cells in the pancreas (type 1 diabetes), which requires daily administration of insulin, or an ineffective use of insulin (type 2 diabetes) [107]. Insulin-dependent diabetes requires a daily management, which commonly consists in diet, physical exercise and exogenous insulin administration [108, 109], which are tuned on the basis of self-monitoring blood glucose (SMBG) measurements usually performed 3-4 times per day.

In recent years, blood glucose (BG) monitoring has been revolutionized by the advent of Continuous Glucose Monitoring (CGM) sensors, consisting of wearable subcutaneous needle-based and minimally-invasive devices that allow measuring the BG concentration almost continuously (1-5 min sampling period) for several consecutive days/weeks. Thanks to this, and to the availability of acoustic/visual alerts for hypo/hyperglycemia, CGM sensors have become a key tool to effectively improve diabetes management and glucose control [110]. However, avoiding hypoglycemic and hyperglycemic events, or at least mitigating their frequency and duration, still remains an open challenge [111]. In particular, the real-time prediction of future levels of BG concentration from its past history could allow the patient to take therapeutic actions on the basis of predicted glycemic levels instead of the current glycemic status, possibly mitigating and/or avoiding imminent critical events [112, 113]. Most of the modern sensors include hardware capable of performing heavy computations. The relatively low sensors' sampling frequency allows for near-realtime operation. Nevertheless, most of

the wearable sensors usually delegate this task to an additional, more powerful device, to enhance their battery time and to limit the memory requirements [114].

Several algorithms for the real-time prediction of hypoglycemic and hyperglycemic events from CGM data have been proposed in the literature. Glucose prediction by using as input only the past history of the CGM signal has been explored with auto-regressive models [115–117], artificial neural networks [118], and kernel-based methods [119]. Since glucose dynamics are influenced by the quantity of ingested carbohydrates (CHO), injected insulin, physical activity, etc., glucose prediction algorithms that consider some (or even all) of these signals have been proposed, e.g., autoregressive-moving average with exogenous inputs models [120–122], random forests [123], support vector based algorithms [124], Gaussian processes [125], linear multi-step predictors [126], neural networks [127–131] and multi-model systems [132]. An increasing spread of data-driven approaches can be observed, and the availability of high quality measurements is of primary importance. In [133], for example, the authors make use of artificially generated data to obtain a large dataset. Nevertheless, real data is always preferable and leads to a more accurate prediction analysis. Despite these attempts, commercial CGM sensors still do not embed any prediction algorithm for the early detection of hypoglycemia and hyperglycemia [110, 111], at variance with the "artificial pancreas" systems that couple a sensor with an insulin/glucose pump connected in close loop. However, basic methods are typically used for these devices. For example, a simple linear projection algorithm is embedded in Medtronic systems, both sensor-augmented pumps [114] and hybrid closed loop systems [134], to predict hypoglycemia and suspend the basal insulin delivery trying to mitigate hypoglycemic states (*"suspend before low"*). One of the reasons for this, is that prediction algorithms have been developed and tested using different datasets (either real or simulated), and that an in depth comparison of their performance on the *same* dataset is still missing [135].

The main contribution of this study is to fill the aforementioned gap, offering a performance comparison among regression and classification algorithms for hypoglycemic and hyperglycemic event prediction based on CGM data. Two different approaches have been implemented and analyzed: *static* and *dynamic* methods, which are detailed in Section 3.1. Numerical results

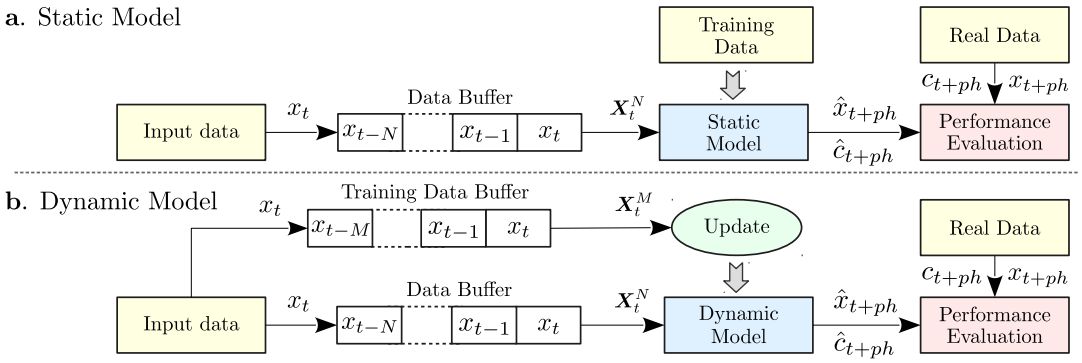


FIG. 3.1: System diagram. Static (a) and Dynamic (b) models.

are presented in Section 3.2. Most of the previous works focus on the prediction performance of the algorithms. Here, the event detection problem has been investigated, introducing a specific analysis for this purpose, and showing that common prediction metrics are not always in accordance with event detection capabilities. Finally, in Section 3.3 some concluding remarks are provided.

Part of the results presented in this chapter has been published in [136].

3.1 Methods

The present work is meant to be a comparative analysis among the most common machine learning algorithms for glucose events prediction. Only the Continuous Glucose Monitoring (CGM) readings are considered for this study, whereas exogenous inputs or any additional information are not taken into account as they are not available in the dataset.

In this study, two different approaches have been analyzed and tested (see Fig. 3.1), defined as follows:

Static - A single model is trained once, in an offline fashion, using a subset of the available data. The data may belong to subjects different from the one considered in the test phase. The model is never updated during the test and a fixed amount N of previous samples are used as input.

Dynamic - The model is updated with each new sample. M previous samples are used to update the model, and N previous samples are used to estimate future glucose levels or events. Note that $M > N$ in order to have enough training samples for updating the model. In this case, the data used for updating the model and the data used for the test belong to the same subject.

In this study, several machine learning techniques have been analyzed. These models can be divided in two main categories: *regression* and *classification* algorithms, detailed in the following sections.

3.1.1 Regression algorithms

The aim of this type of algorithms is to predict the future value of the BG concentration, that is considered to be a real number, from measurements acquired in the past. In general, they implement a function r such that:

$$r(\boldsymbol{X}_t^N, \boldsymbol{\beta}_t) = \hat{x}_{t+ph} \qquad \hat{x}_{t+ph} \in \mathbb{R}\,, \tag{3.1}$$

where $\boldsymbol{X}_t^N = [x_{t-N+1}, \ldots, x_t] \in \mathbb{R}^N$ is a vector containing the last N CGM readings, ph is the prediction horizon, and $\boldsymbol{\beta}_t$ represents a vector containing algorithm-specific parameters. This parameter vector is unknown and is to be found during the training process. The sub-index t represents the discrete time progression. In general, $\boldsymbol{\beta}_t$ is not constant. In particular, if a dynamic approach is used, the model parameters continuously change at each iteration; when considering a static model, instead, the parameters are optimized in the training phase, and then remain fixed ($\boldsymbol{\beta}_t = \boldsymbol{\beta}$). As performance metric, let's define the regression error, corresponding to a prediction horizon ph, as:

$$e_t^{ph} = x_{t+ph} - \hat{x}_{t+ph}\,. \tag{3.2}$$

Furthermore, a Root Mean Square Error (RMSE) can be associated with an entire signal of T samples as follows:

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(e_t^{ph}\right)^2}\,. \tag{3.3}$$

Sum of Squares of the Glucose Prediction Error (SSGPE), defined in Eq. (3.4), is also considered as a relative error estimation metric.

$$SSGPE = \sqrt{\frac{\sum_{t=1}^{T}\left(e_t^{ph}\right)^2}{\sum_{t=1}^{T}\left(x_{t+ph}\right)^2}}\,. \tag{3.4}$$

The methods considered in this study that fall into this category are: linear, quadratic and spline fitting, Linear Regression, Bayesian Regression [137], Support Vector Regression (SVR) [138] and Autoregressive Integrated Moving Average (ARIMA) models.

3.1.2 Classification algorithms

In order to perform a classification of the events corresponding to specific levels of BG concentration, the BG has been quantized into a finite number of classes. Following a well-accepted and commonly used definition [139], the following states have been considered: severe hyperglycemic (SHyper), hyperglycemic (Hyper), normal (Norm), hypoglycemic (Hypo) and severe hypoglycemic (SHypo). The function q that maps the CGM readings x_t onto the class set $\mathbb{C} = \{\text{SHyper, Hyper, Norm, Hypo, SHypo}\}$ is defined as follows:

$$c_t = q(x_t) = \begin{cases} \text{SHypo} & \text{if } x_t \leqslant 50 \\ \text{Hypo} & \text{if } 50 < x_t \leqslant 70 \\ \text{Norm} & \text{if } 70 < x_t < 180 \\ \text{Hyper} & \text{if } 180 \leqslant x_t < 250 \\ \text{SHyper} & \text{if } x_t \geqslant 250\,. \end{cases} \tag{3.5}$$

All the measurement units are mg/dL and have been omitted for the sake of conciseness. The aim of the classification algorithms is to predict the future state given the past CGM readings. In particular, all the classification models presented in this study implement a function f such as:

$$f(\boldsymbol{X}_t^N, \boldsymbol{\theta}_t) = \hat{c}_{t+ph} \qquad \hat{c}_{t+ph} \in \mathbb{C}\,, \tag{3.6}$$

where $\boldsymbol{X}_t^N$ is a vector containing the previous N glucose readings, $\boldsymbol{\theta}_t$ is the unknown parameters vector and ph is the prediction horizon. As for Eq. (3.1), upon completion of the training phase, $\boldsymbol{\theta}_t$ is a constant vector or it changes at each iteration for the static and dynamic approach, respectively.

An *accuracy* measure is used as the performance metric for the classification algorithms, which is here computed as the percentage of classes that are correctly predicted. Note that the accuracy metric is evaluated including

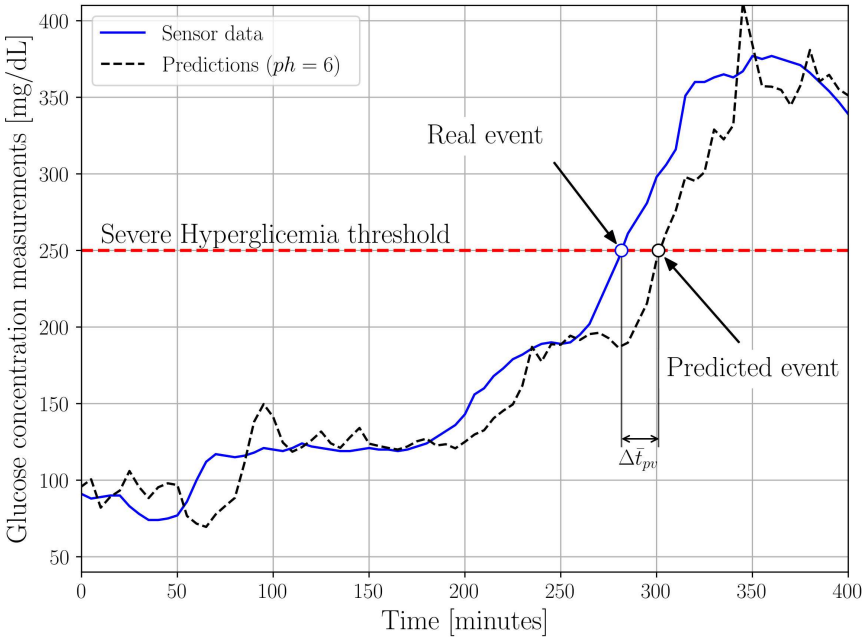


FIG. 3.2: Example of severe hyperglycemic event onset. A Linear Regression model has been considered.

the samples belonging to all the considered classes (hypo, hyper and normal states). Therefore, the result may be affected by an imbalanced dataset.

In this study, the following classification algorithms have been considered: Support Vector Machine (SVM) [53], k-Nearest Neighbor (kNN) [52], Naive Bayes (NB) [51], Classification Tree (CT) [50], Random Forest (RF) [140], AdaBoost [141], Linear Discriminant Analysis (LDA).

3.1.3 Events detection

The ultimate goal of this study is to present a performance evaluation framework involving a large number of prediction and classification models, trying to figure out the most appropriate approach to solve glucose event prediction problems. Let's consider as *events* all the time instants $\bar{t}$ whose corresponding samples verify one of the following conditions:

$$
\begin{array}{lll}
x_{\bar{t}-1} > 50 & \text{and } x_{\bar{t}} \leqslant 50 & \text{Severe Hypoglycemia Event} \\
x_{\bar{t}-1} > 70 & \text{and } x_{\bar{t}} \leqslant 70 & \text{Hypoglycemia Event} \\
x_{\bar{t}-1} < 180 & \text{and } x_{\bar{t}} \geqslant 180 & \text{Hyperglycemia Event} \\
x_{\bar{t}-1} < 250 & \text{and } x_{\bar{t}} \geqslant 250 & \text{Severe Hyperglycemia Event}
\end{array}
$$

Four different sets can be created according to this criterion, with each set containing all the events that meet one of the above defined conditions. Due to the presence of measurement noise and signal fluctuations, multiple consecutive events of the same type could be observed within a short time frame. To avoid this behavior, which is quite unrealistic, a settling time of 30 minutes is also considered [142, 143]. That is, if a specific event occurs, additional events of same type are not considered for the following 30 minutes. The same procedure can also be applied to the predicted samples $\hat{x}_t$, and a similar approach is used for the predicted classes $\hat{c}_t$, where the event is mapped onto a class change.

In the following, I refer to a specific set, containing a generic type of event. The considerations that follow apply to all events.

Let $\mathbb{V} = \{\bar{t}_1, \ldots, \bar{t}_V\}$ be the generic set containing the V time instants associated with the real events, obtained analyzing the CGM measurements x_t, and let $\mathbb{P} = \{\bar{t}_1, \ldots, \bar{t}_P\}$ be the set containing the P predicted events, found using the predicted BG concentration values $\hat{x}_t$ (or $\hat{c}_t$ in case of classification methods). Each element $\bar{t}_p \in \mathbb{P}$ is individually analyzed and marked as either a True Positive (TP) or a False Positive (FP), according to the following criterion.

Let's define $\Delta\bar{t}_{pv} = \bar{t}_p - \bar{t}_v$ as the distance between a generic predicted event time $\bar{t}_p$ and a real event $\bar{t}_v$ (see Fig. 3.2). ph is the temporal prediction step into the future. For each $\bar{t}_p \in \mathbb{P}$, if there exists $\bar{t}_v \in \mathbb{V}$ such as $-k < \Delta\bar{t}_{pv} < ph$, $\bar{t}_p$ is considered to be a TP and $\bar{t}_v$ is removed from the set $\mathbb{V}$ and no longer considered in the following steps, to prevent that the same event be considered multiple times. A TP condition is indeed assigned to a predicted event that, in general, actually occurs in the future, within a tolerance range. If the condition above is not met, $\bar{t}_p$ is considered to be a FP, i.e., the predicted event will not occur or has already occurred. $k \in \mathbb{N}$ is a positive constant that prevents the association of events that are too distant apart. When there are multiple events meeting the above condition, only the one with the minimum $|\Delta\bar{t}_{pv}|$ distance is considered. After having checked all the elements in set P, all the remaining events in the set $\mathbb{V}$, if any, are tagged as False Negatives (FN). In fact, at this point, the set $\mathbb{V}$ contains the observed events that have not been correctly predicted. In Tab. 3.1, a list of possible conditions associated with a predicted event is shown for clarity. In Fig. 3.2 an example of the onset of a severe hyperglycemic event is shown,

$\Delta\bar{t}_{pv} < -k$	The event is out of the analysis range.
$-k \leqslant \Delta\bar{t}_{pv} < 0$	The real event occurs later than the predicted time $\bar{t}_p$, but still within the tolerance range.
$\Delta\bar{t}_{pv} = 0$	The real event occurs exactly when predicted.
$0 < \Delta\bar{t}_{pv} < ph$	The real event $\bar{t}_v$ occurs earlier than the predicted time $\bar{t}_p$, but still in the future.
$\Delta\bar{t}_{pv} \geqslant ph$	The real event has already happened. Its prediction has failed.

TABLE 3.1: List of possible conditions associated with a predicted event.

along with the predictions of a linear regressor with a prediction horizon $ph = 6$.

Once all the TP, FP and FN events have been found, Precision, Recall and F-measure can be evaluated as follows [144]:

$$\texttt{Precision} = \frac{N_{TP}}{N_{TP} + N_{FP}} \tag{3.7}$$

$$\texttt{Recall} = \frac{N_{TP}}{N_{TP} + N_{FN}} \tag{3.8}$$

$$\texttt{F-measure} = 2 \cdot \frac{\texttt{Precision} \cdot \texttt{Recall}}{\texttt{Precision} + \texttt{Recall}} \tag{3.9}$$

where N_{TP}, N_{FP} and N_{FN} are respectively the total number of TP, FP and FN events. With these three metrics, an overall assessment of the algorithm performance is possible, along with a probabilistic interpretation. The Precision estimates the fraction of correctly predicted events (i.e., the true positives) among all the predicted events (including the false positive outcomes). It can be interpreted as the probability that a predicted event will actually happen in the near future. The Recall, instead, estimates the fraction of real events that is correctly predicted by the algorithm. It can be seen as the probability that a real event is detected in advance. Both these metrics are extremely important and should be jointly considered to provide a complete performance analysis. The F-measure combines these two metrics with an harmonic mean, providing a good way to compare different algorithms' outcomes.

Note that the metrics defined in Eqs. (3.7), (3.8) and (3.9) refer to a specific set $\mathbb{V}$, that is related to a particular event type. When another event is considered, these results may vary considerably. To provide an overall assessment, all the different conditions can be aggregated and all the events can be considered at the same time. In this study, the focus is both on the overall

performance and on the specific hyperglycemic event, which is considered to be of great importance for diabetic patients.

A compact and clear overview of the event prediction capabilities of a method can be assessed with these metrics. Nevertheless, there is still a limitation, due to the loss of information about the time difference $\Delta\bar{t}_{pv}$ between the predicted and the real events, referred to as *prediction distance* in the following. Therefore, an additional statistical analysis is devoted to the comparison of the prediction distances achieved by each of the selected methods.

To provide more clarity, it is remarked the clear difference between *signal prediction* and *event prediction*, and all the corresponding metrics. The former is related to the capability of the algorithm to predict the future value of the signal, or the corresponding class in case of classifiers, given some historical data. The event prediction analysis, instead, is specifically intended to evaluate the ability to predict the occurrence of a specific condition.

3.1.4 Training process

A different training process is used for the static and dynamic approaches. The main difference between them is that the static model is trained only once, the dynamic model, instead, is updated at each iteration. The input data for each of these models, however, is of the same type, i.e., they both take the past N CGM readings as input and output the predicted BG concentration (for regressors) or the corresponding class (for classifiers) at a preset prediction horizon ph.

Let consider a generic vector $\boldsymbol{X}^M = [x_1, \ldots, x_M]$ containing M measurements, and let ph be the prediction horizon, with $M > N + ph$. A series of input samples $\boldsymbol{i_n^N} \in \mathbb{R}^N$, and the corresponding output label o_n, can be extracted from this vector, according to the following criterion:

$$\begin{aligned} \text{Regression} \quad & \boldsymbol{i_n^N} = [x_{n-N+1}, \ldots, x_n] \quad & o_n = x_{n+ph} \\ \text{Classification} \quad & \boldsymbol{i_n^N} = [x_{n-N+1}, \ldots, x_n] \quad & o_n = q(x_{n+ph}) \\ & \forall n \in \{N, \ldots, M - ph\} \end{aligned} \tag{3.10}$$

All the pairs $(\boldsymbol{i_n^N}, o_n)$ generated according to Eq. (3.10) are included in a set used for training or testing the analyzed models. In the following, the specific training procedure for static and dynamic approaches is detailed.

Static model

Data acquired from K different subjects is considered in this phase. Here, a Leave One Patient Out validation procedure is used to evaluate the performance metrics. In particular, at each round of the validation a subject is considered as a test subject and referred to as the *target user*. All the data belonging to the other $K-1$ subjects is used to generate a single training set according to Eq. (3.10). Then, this set is employed to find the parameters $\boldsymbol{\beta}$ or $\boldsymbol{\theta}$ in Eqs. (3.1) and (3.6). Once the model is trained, its parameters remain fixed. A test set, created with the target user data, is used to evaluate the performance metrics, both for prediction capabilities (residuals for regressors and accuracy for classifiers) and event detection (Precision, Recall, F-measure). This procedure is repeated for each subject in the dataset.

Dynamic model

For the dynamic model approach, only the data belonging to the target user is considered, both for training and testing procedures. Let us consider a generic time instant t, and a vector containing the current (time t) and the previous $M-1$ measurements $\boldsymbol{X}_t^M = [x_{t-M+1}, \dots, x_t] \in \mathbb{R}^M$. A training set is generated from this vector using Eq. (3.10), and it is used to find the model parameters $\boldsymbol{\beta}_t$ or $\boldsymbol{\theta}_t$ associated with the current time instant. Then, the last N samples $\boldsymbol{X}_t^N$ are employed to make a prediction $\hat{x}_{t+ph}$, or $\hat{c}_{t+ph}$ in case of classifiers, where $N < M$ represents the number of inputs for the selected model. Finally, the prediction is compared to the real value x_{t+ph} or c_{t+ph}, to evaluate the performance metrics.

This procedure is repeated for each time instant t for which both the future value x_{t+ph} and $\boldsymbol{X}_t^M$ are available. At each iteration, the new measurement is added to the vector $\boldsymbol{X}_t^M$, and the oldest one is dropped, in order to always have a training vector of fixed length, containing the last M readings.

Note that: **1)** the model is trained using only the past data, which is always available in a real implementation of the algorithm, and **2)** each model is used to predict a single value.

3.2 Results

Data consist of CGM monitoring for 7 consecutive days of 89 Type-1 Diabetes (T1D) patients, which are extracted from a larger datasets. CGM traces were measured by the Dexcom G4 Platinum CGM sensor (Dexcom Inc., San Diego, CA), which has a sampling period of 5 minutes. The CGM sensor has been inserted at the beginning of day 1 of the study and calibrated accordingly to manufacturer instructions (i.e., twice at the beginning of the monitoring, and then every 12 hours). In this period, patients were admitted to the clinical research center (CRC) at days 1, 4, and 7, each time for a time period of 12 hours. During CRC admission, subjects underwent a delayed-and-increased insulin bolus at meal-time in order to simulate rapid glucose fluctuations, aimed at testing CGM sensors in more challenging conditions. During the rest of the monitoring, patients were asked to behave as usual (free-living conditions). No additional signals, e.g., CHO content of the meals or injected insulin, are available. Additional clinical information, e.g., the cohort, are specified in [145].

Since during acquisition on real subjects, failures and irregular sampling may happen, the CGM data was first interpolated using a third-order spline for obtaining a regular sampling period, and then the value of any missing data reading was imputed. In case of too many consecutive missing samples, data is discarded and not considered in the corresponding period of time. All the presented results are shown using boxplots, which provide a statistical and compact view of the results. The boxes represent the Interquartile Range (IQR), the line is the median value, and the whiskers have a coverage factor of 90%. A time window of $N = 5$ samples has been considered for all the models. A further increase in the window size N did not show significant improvements.

In Fig. 3.3, the RMSE values for an increasing prediction horizon ph are shown. As expected, the error considerably increases for long-term predictions. SVR and Linear Regression algorithms have been used here, but other methods exhibit a similar behavior. For the sake of comparison, the results reported in [118, 127, 128] are also shown in the plot, bearing in mind that the dataset used is different. Note that the algorithms implemented in [127] and [128] make use of additional information on ingested carbohydrates, which allows them to achieve better performance. If available, these additional sources of information are expected to lead to improved results. A

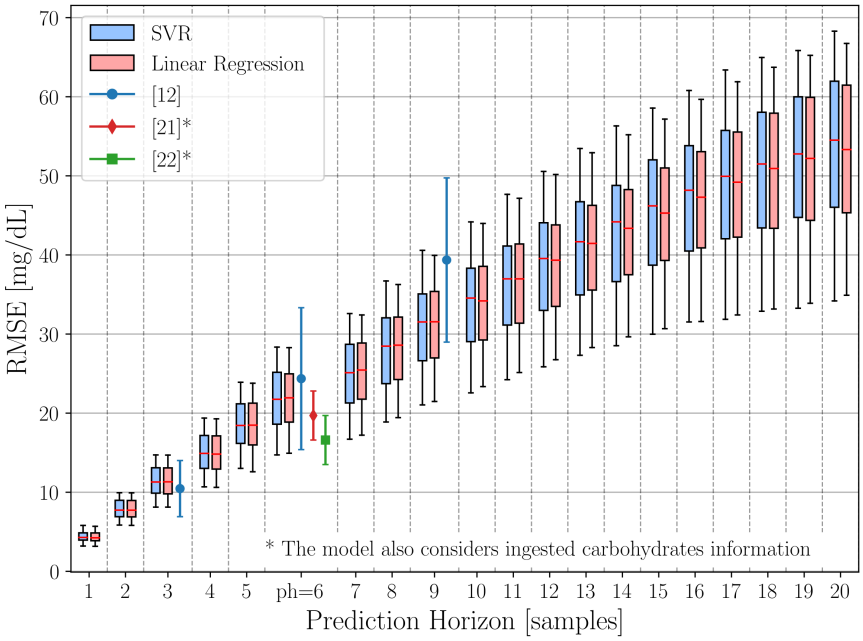


FIG. 3.3: RMSE for increasing prediction horizon. As comparison, the results reported in [118, 127, 128] have been included.

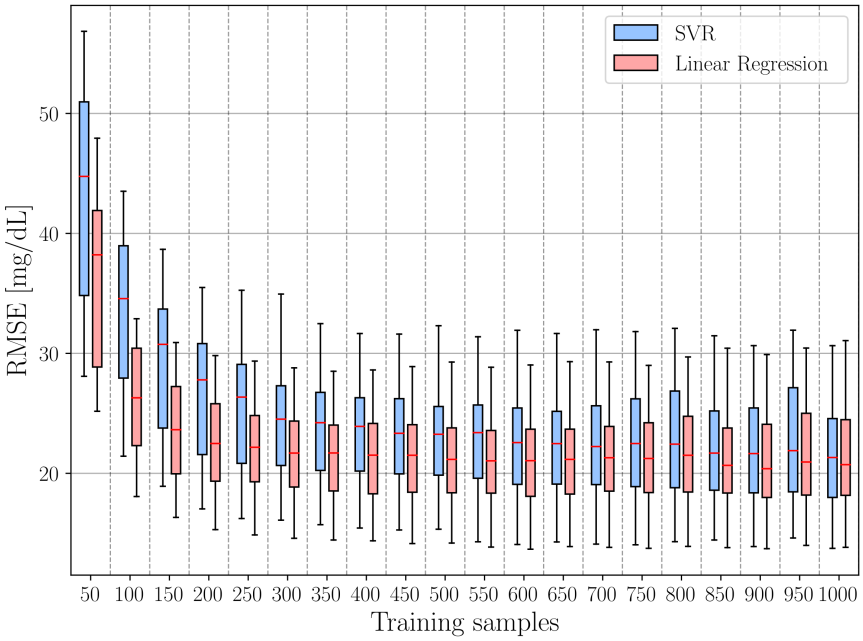


FIG. 3.4: RMSE for increasing training samples of the dynamic model.

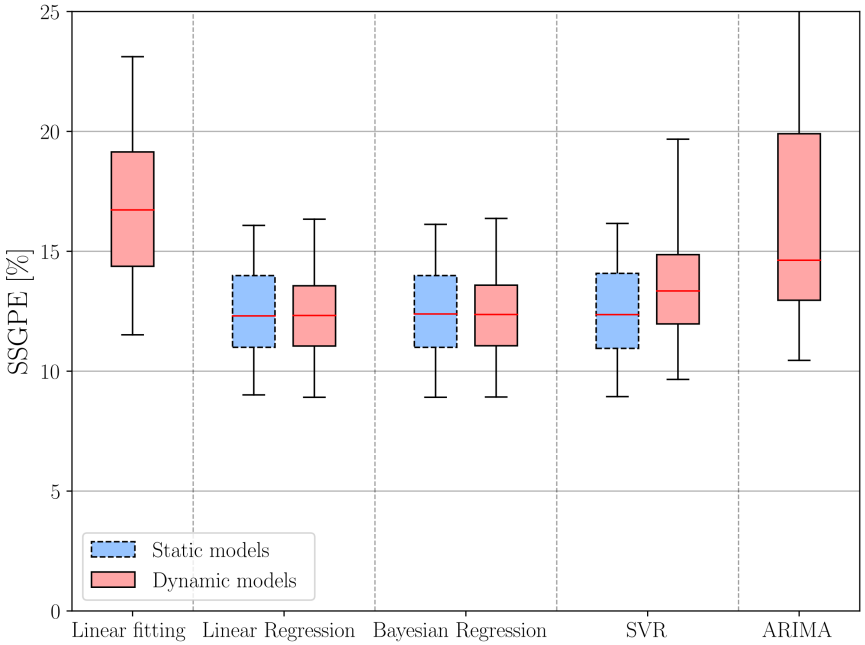


FIG. 3.5: SSGPE for each regression method.

prediction horizon of $ph = 6$ samples (corresponding to 30 minutes) is considered for the following analyses, as it provides a good trade-off between error and prediction time.

The dimension M of the training data buffer (see Fig. 3.1) for the dynamic approach is an important parameter to take into account. The RMSE has been measured varying the number of historical samples in the training set. The result is shown in Fig. 3.4, where the RMSE performance of SVR and Linear Regression dynamic models are shown as examples. A saturation effect is observed for a buffer size larger than $M = 400$ samples, beyond which no additional improvements are observed. This value, which approximately corresponds to 30 hours of training data, is then chosen to train the final models.

The signal prediction performance of all the methods considered in this study is shown in Fig. 3.5 (regressors) and Fig. 3.6 (classifiers). SSGPE has been chosen as representative metric for regression methods, and accuracy for classifiers, as defined in Section 3.1.

Note that only a dynamic implementation has been considered for linear fitting and ARIMA models. This is why the results for their static versions do not appear in Fig. 3.5. Quadratic and Spline fitting provide poor results, and for this reason have also been omitted from the plot.

Some of the algorithms require to set a number of hyper-parameters, which

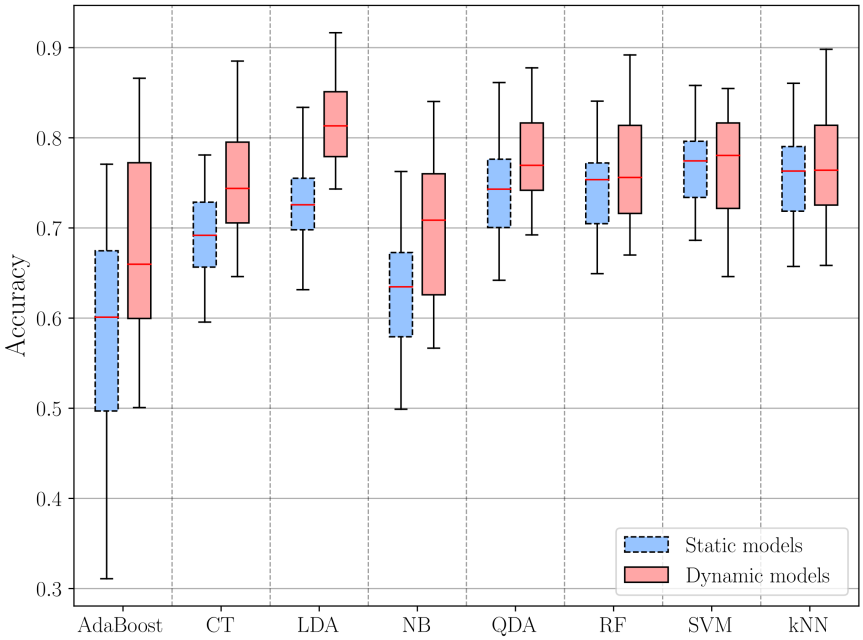


FIG. 3.6: Accuracy for each classification method.

can heavily affect the performance of the learning algorithms. Prior to the proper training of the algorithms requiring such hyper-parameters, an optimization phase is always required. This pre-train explores different set of hyper-parameters, looking at the performance of the algorithm on a validation set (separated from the training and testing data). Once the pre-training phase is finished, the hyper-parameters value is kept fixed during the subsequent training and testing phases. Several ARIMA models of different orders have been validated, considering all the possible combinations up to a maximum of the second order. The model that provided the best results has an autoregressive term $ar = 2$, a differencing term $i = 0$ and a moving average term $ma = 0$. All the presented results refer to these parameters. Radial Basis Function (RBF) kernel is considered for kernel based methods, i.e., SVM and SVR: a grid search procedure has been applied during pre-training to optimize the penalty parameter of the error term and the RBF kernel coefficient. A number of 3, 5 and 10 neighbors have been considered for the kNN algorithm, the results that are shown use a value of 10. Finally, a maximum of 50 weak classifiers have been used for the AdaBoost method. Please refer to Table 3.2 for further details on the setting of hyper-parameters.

Regression methods (Fig. 3.5) exhibit very similar performance both for static and dynamic approaches, except for the Linear Fitting method, which provides higher errors. As for the classification methods, all the dynamic

Method	Parameters settings
Bay. Reg.	All the shape and the inverse scale parameters for the Gamma distribution priors have been set to 10^{-6}.
SVR	A radial basis function kernel has been considered. The penalty parameter of the error term has been set to 5^3 and the kernel coefficient to 5^{-6}.
ARIMA	A second order auto-regressive model has been considered.
AdaBoost	A maximum of 50 estimators and a learning rate of 1 have been considered [146].
CT	Gini impurity has been used to measure the quality of a split [147].
LDA	Singular value decomposition has been used as the solver method.
NB	The likelihood of the features is assumed to be Gaussian.
RF	10 estimators have been considered. Gini impurity has been used to measure the quality of a split.
SVM	A radial basis function kernel has been considered. The penalty parameter of the error term has been set to 10^5 and the kernel coefficient to 1^{-7}.
kNN	10 neighbors have been considered.

TABLE 3.2: Hyper-parameters settings.

models perform better, in term of accuracy, than their static counterparts. LDA is the model with the highest average accuracy across all classes. As a general consideration, AdaBoost and NB perform worse than other classification methods, on average.

Analyzing signal prediction performance is certainly a good comparison strategy. Nevertheless, this approach could be misleading when event detection capabilities need to be assessed. An event, as defined in Sec. 3.1.3, only occurs when a predefined condition is met, such as the overcoming of a threshold. However, the metrics that are commonly used to evaluate prediction algorithms usually assume that all samples are equally important, but this could lead to inappropriate results, especially when the dataset is highly imbalanced, due, for example, by the prevalence of normal states. In this case, a small prediction error will be driven by the ability of a method to follow the normal range, considering the hypo- and hyper-glycemic events as outliers. For this reason, and also to allow a direct comparison between regression and classification algorithms, a more insightful analysis has been devoted to event detection performance, as detailed in Section 3.1.3. A simple prediction method that considers the last measurement as the predicted value is typically used as a lower bound on the performance evaluation.

		Severe Hypoglycemia		Hypoglycemia		Hyperglycemia		Severe Hyperglycemia	
		Static	Dynamic	Static	Dynamic	Static	Dynamic	Static	Dynamic
Lin. Fit.	Recall	n/a	0.96 ± 0.11	n/a	0.93 ± 0.14	n/a	0.92 ± 0.08	n/a	0.92 ± 0.10
	Precision	n/a	0.24 ± 0.11	n/a	0.38 ± 0.16	n/a	0.51 ± 0.11	n/a	0.43 ± 0.13
	F-measure	n/a	0.37 ± 0.14	n/a	0.52 ± 0.17	n/a	0.65 ± 0.10	n/a	0.58 ± 0.13
Lin. Reg.	Recall	0.60 ± 0.33	0.62 ± 0.36	0.62 ± 0.22	0.51 ± 0.29	0.89 ± 0.08	0.78 ± 0.20	0.80 ± 0.17	0.61 ± 0.30
	Precision	0.49 ± 0.28	0.55 ± 0.34	0.47 ± 0.20	0.41 ± 0.24	0.66 ± 0.11	0.63 ± 0.17	0.61 ± 0.17	0.50 ± 0.27
	F-measure	0.51 ± 0.27	0.57 ± 0.34	0.53 ± 0.20	0.44 ± 0.25	0.75 ± 0.09	0.69 ± 0.17	0.68 ± 0.16	0.54 ± 0.27
Bay. Reg.	Recall	0.61 ± 0.32	0.59 ± 0.38	0.62 ± 0.22	0.50 ± 0.30	0.89 ± 0.09	0.77 ± 0.21	0.80 ± 0.17	0.60 ± 0.29
	Precision	0.50 ± 0.28	0.54 ± 0.35	0.47 ± 0.20	0.41 ± 0.25	0.66 ± 0.11	0.62 ± 0.18	0.61 ± 0.17	0.49 ± 0.26
	F-measure	0.52 ± 0.27	0.55 ± 0.35	0.53 ± 0.20	0.44 ± 0.26	0.75 ± 0.09	0.68 ± 0.18	0.68 ± 0.16	0.53 ± 0.27
SVR	Recall	0.49 ± 0.33	0.44 ± 0.39	0.75 ± 0.22	0.52 ± 0.29	0.87 ± 0.09	0.73 ± 0.19	0.83 ± 0.16	0.66 ± 0.27
	Precision	0.43 ± 0.30	0.39 ± 0.34	0.51 ± 0.19	0.45 ± 0.26	0.64 ± 0.11	0.60 ± 0.17	0.62 ± 0.17	0.53 ± 0.24
	F-measure	0.43 ± 0.27	0.39 ± 0.33	0.59 ± 0.19	0.46 ± 0.25	0.73 ± 0.09	0.65 ± 0.17	0.70 ± 0.15	0.58 ± 0.24
ARIMA	Recall	n/a	0.53 ± 0.38	n/a	0.42 ± 0.29	n/a	0.68 ± 0.25	n/a	0.53 ± 0.27
	Precision	n/a	0.49 ± 0.36	n/a	0.37 ± 0.25	n/a	0.59 ± 0.22	n/a	0.46 ± 0.25
	F-measure	n/a	0.50 ± 0.36	n/a	0.39 ± 0.26	n/a	0.63 ± 0.23	n/a	0.49 ± 0.25
AdaBoost	Recall	0.28 ± 0.34	0.05 ± 0.12	0.54 ± 0.30	0.23 ± 0.23	0.82 ± 0.14	0.31 ± 0.19	0.29 ± 0.29	0.21 ± 0.18
	Precision	0.11 ± 0.17	0.08 ± 0.17	0.28 ± 0.21	0.24 ± 0.25	0.53 ± 0.17	0.22 ± 0.14	0.32 ± 0.30	0.21 ± 0.21
	F-measure	0.12 ± 0.15	0.06 ± 0.13	0.34 ± 0.21	0.22 ± 0.23	0.63 ± 0.14	0.25 ± 0.16	0.24 ± 0.20	0.20 ± 0.17
CT	Recall	0.95 ± 0.12	0.21 ± 0.31	0.89 ± 0.16	0.53 ± 0.30	0.94 ± 0.06	0.70 ± 0.18	0.92 ± 0.10	0.48 ± 0.32
	Precision	0.13 ± 0.07	0.12 ± 0.17	0.23 ± 0.11	0.29 ± 0.17	0.32 ± 0.09	0.32 ± 0.11	0.28 ± 0.09	0.22 ± 0.15
	F-measure	0.23 ± 0.11	0.14 ± 0.19	0.35 ± 0.14	0.36 ± 0.20	0.47 ± 0.10	0.44 ± 0.13	0.43 ± 0.10	0.30 ± 0.20
LDA	Recall	0.96 ± 0.12	0.10 ± 0.20	0.77 ± 0.21	0.25 ± 0.22	0.93 ± 0.06	0.71 ± 0.20	0.88 ± 0.10	0.41 ± 0.30
	Precision	0.27 ± 0.13	0.18 ± 0.33	0.31 ± 0.16	0.29 ± 0.26	0.61 ± 0.13	0.51 ± 0.17	0.65 ± 0.15	0.30 ± 0.21
	F-measure	0.41 ± 0.16	0.12 ± 0.22	0.42 ± 0.18	0.25 ± 0.21	0.73 ± 0.11	0.59 ± 0.17	0.74 ± 0.13	0.34 ± 0.24
NB	Recall	0.72 ± 0.26	0.20 ± 0.25	0.66 ± 0.20	0.44 ± 0.23	0.34 ± 0.14	0.23 ± 0.19	0.05 ± 0.09	0.15 ± 0.20
	Precision	0.34 ± 0.21	0.12 ± 0.15	0.38 ± 0.18	0.32 ± 0.22	0.34 ± 0.13	0.21 ± 0.16	0.05 ± 0.09	0.13 ± 0.16
	F-measure	0.44 ± 0.21	0.14 ± 0.17	0.45 ± 0.16	0.36 ± 0.21	0.33 ± 0.12	0.22 ± 0.17	0.05 ± 0.09	0.13 ± 0.17
QDA	Recall	0.88 ± 0.20	0.10 ± 0.19	0.79 ± 0.18	0.35 ± 0.26	0.93 ± 0.08	0.67 ± 0.19	0.81 ± 0.15	0.44 ± 0.27
	Precision	0.30 ± 0.14	0.10 ± 0.19	0.33 ± 0.16	0.29 ± 0.22	0.52 ± 0.13	0.41 ± 0.16	0.39 ± 0.14	0.25 ± 0.16
	F-measure	0.43 ± 0.17	0.10 ± 0.19	0.45 ± 0.17	0.31 ± 0.23	0.66 ± 0.12	0.50 ± 0.16	0.51 ± 0.14	0.31 ± 0.18
RF	Recall	0.93 ± 0.20	0.19 ± 0.27	0.86 ± 0.16	0.46 ± 0.28	0.92 ± 0.09	0.60 ± 0.20	0.91 ± 0.10	0.38 ± 0.28
	Precision	0.18 ± 0.09	0.12 ± 0.17	0.30 ± 0.14	0.30 ± 0.19	0.43 ± 0.11	0.34 ± 0.14	0.44 ± 0.11	0.21 ± 0.15
	F-measure	0.29 ± 0.13	0.14 ± 0.19	0.43 ± 0.17	0.36 ± 0.21	0.58 ± 0.11	0.43 ± 0.15	0.59 ± 0.11	0.26 ± 0.19
SVM	Recall	0.98 ± 0.08	0.31 ± 0.34	0.86 ± 0.19	0.62 ± 0.29	0.95 ± 0.06	0.81 ± 0.18	0.93 ± 0.08	0.58 ± 0.34
	Precision	0.32 ± 0.13	0.13 ± 0.15	0.36 ± 0.17	0.32 ± 0.17	0.58 ± 0.13	0.47 ± 0.17	0.60 ± 0.14	0.34 ± 0.20
	F-measure	0.47 ± 0.15	0.18 ± 0.19	0.49 ± 0.18	0.41 ± 0.21	0.71 ± 0.11	0.59 ± 0.17	0.72 ± 0.12	0.43 ± 0.25
kNN	Recall	0.96 ± 0.11	0.11 ± 0.20	0.86 ± 0.18	0.28 ± 0.24	0.92 ± 0.07	0.38 ± 0.21	0.85 ± 0.13	0.24 ± 0.21
	Precision	0.26 ± 0.12	0.15 ± 0.29	0.37 ± 0.17	0.25 ± 0.22	0.56 ± 0.12	0.28 ± 0.17	0.55 ± 0.15	0.17 ± 0.16
	F-measure	0.40 ± 0.15	0.12 ± 0.21	0.50 ± 0.17	0.25 ± 0.21	0.69 ± 0.10	0.32 ± 0.18	0.66 ± 0.13	0.20 ± 0.17

TABLE 3.3: Events detection metrics for all the analyzed methods, static and dynamic implementations, and for each event type. The reported uncertainty refers to the standard deviation.

Since this method is not able to predict any event, as defined in this study, it has no prediction capabilities and it has not been considered in the results. The event-based performance metrics are included in Fig. 3.7, where an exhaustive comparison among all the methods is shown. This figure shows the Recall (blue boxes), the Precision (red boxes) and the F-measure (green boxes), dividing the results into two groups: regressors and classifiers (separated through a vertical dashed line). The results in the same group refer to the static implementation (dashed boxes) and to the dynamic model (solid line boxes). As a general behavior, a high recall and a low precision can be observed. This is typically due to a tendency to predict events that will not really happen. This effect is particularly emphasized for most of the static

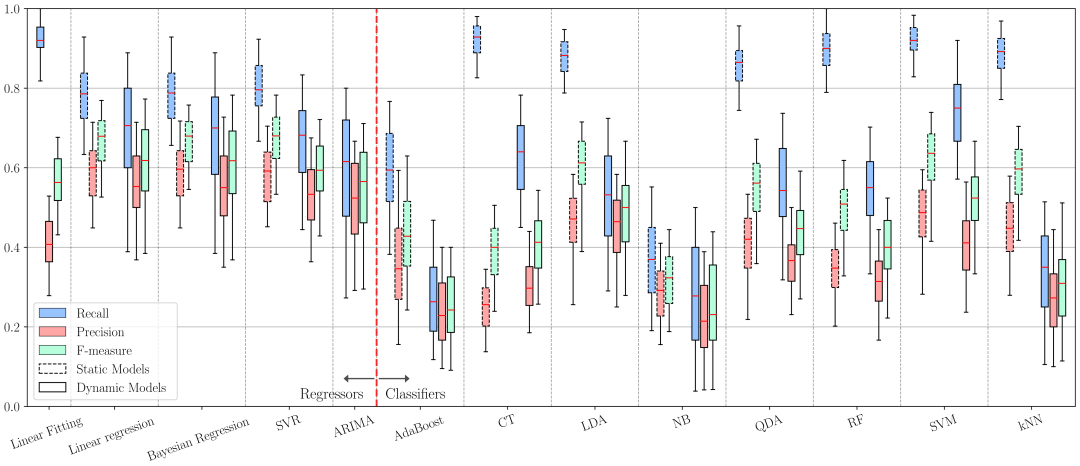


FIG. 3.7: Events detection metric for all the analyzed algorithms (prediction horizon $ph = 6$).

classifiers and for linear fitting. Considering the F-measure as the most representative score, static models always perform better, or at least are comparable with the corresponding dynamic implementations. It is worth noting that the signal prediction performance results show a different behavior, especially for classifiers, where the dynamic implementations show even better performance. This result empirically demonstrates that the signal prediction performance metrics, evaluated considering the entire signals, could be very imbalanced and they are not always a good indicator of event prediction capabilities, which require a more thorough investigation.

The classifier showing the best performance is SVM, followed by LDA and kNN. Regression methods, even if with a lower recall, provide a significantly higher precision, which leads to an improved F-measure. In particular, the static implementation of SVR, Linear and Bayesian Regression provide the best results across all methods.

It is remarked that these results refer to all the possible events, including hypoglycemic and hyperglycemic conditions. It is also interesting to evaluate the performance when only a specific event is considered. For a more detailed analysis, Table 3.3 reports all the event detection metrics, along with their standard deviations, separating the results over the four considered event types. Under this assumption, the classification problem needs to be revised. The class space set $\mathbb{C}$, introduced in Section 3.1.2, is now reduced to a binary set $\mathbb{C}_H = \{\text{Hyper, Norm}\}$, and also the mapping function defined in Eq. (3.5) is modified accordingly. All the classifiers are retrained solving this reduced binary classification problem. The regressors, instead, remain

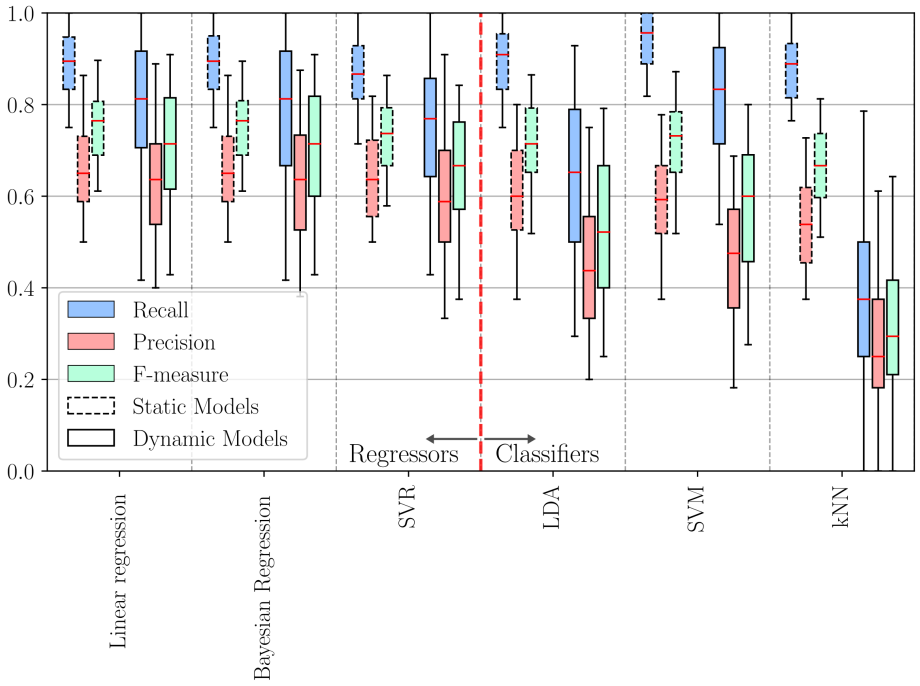


FIG. 3.8: Events detection metric for all the analyzed algorithms only considering hyperglycemic events (prediction horizon $ph = 6$).

the same, but their performance is now evaluated on the specific event only, to obtain a fair comparison.

In this study, the main focus is on hyperglycemic events. The frequent occurrence of these events in the dataset makes them suitable for a more in-depth analysis. Furthermore, in a preventive scenario, where a patient takes pre-emptive actions based on the predicted glucose level, the hyperglycemic condition is of primary importance. In Fig. 3.8, the event detection performance of the three best regressors and classifiers is shown, when only the hyperglycemic events are considered. A significant overall improvement can be observed. In particular, the classifiers exhibit improved performance when specialized to detect a specific event (hyperglycemia in this case). Also in this case the static models provide better results than their dynamic counterparts.

The best methods remain the Linear and Bayesian Regression, which perform almost the same and have a median F-measure value of 0.76. The best performing classifier is the SVM, which has a median F-measure value of 0.73, and the highest median recall value of about 0.95.

The last analysis concerns the prediction distance, i.e., the period between the real event and the predicted one. Statistical results are shown in Fig. 3.9. Since the prediction horizon in this implementation is $ph = 6$ samples, the

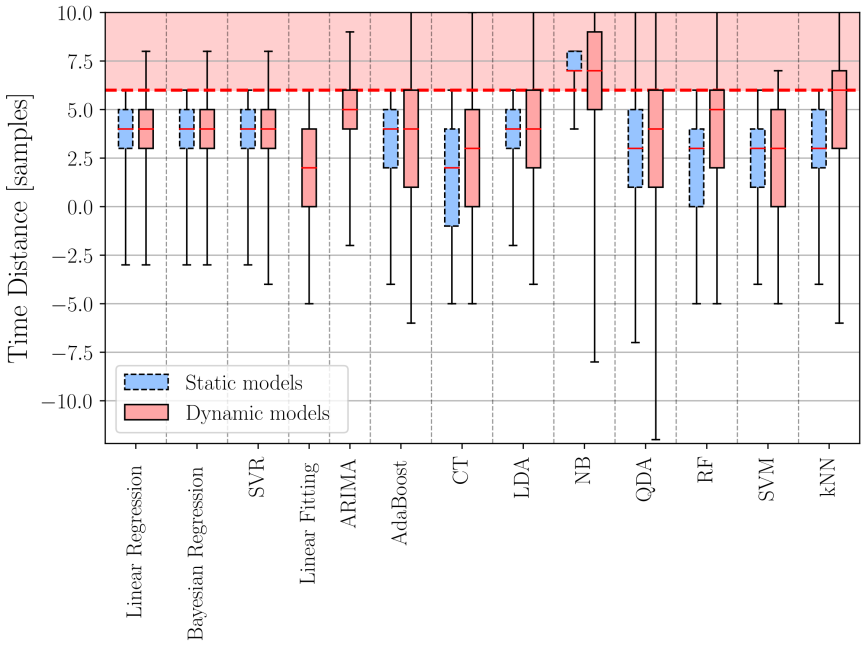


FIG. 3.9: Temporal distance between real and predicted events (prediction horizon $ph = 6$).

dashed red line in the figure corresponds to the actual time of the event. Even if the prediction horizon is 30 minutes, events are not always detected that early. Most of the times, even when an event is correctly predicted, it actually happens sooner than expected.

Let consider the best performing regressor and classifier, i.e., the Linear or Bayesian Regression method and the SVM. The former has a median prediction time of only 2 samples, which means that half of the correctly predicted events actually happen at least 10 minutes later. Furthermore, the IQR goes from 5 to 15 minutes, i.e., half of the times the event will happen after a time period within this range. The SVM, instead, has much better performance, with an IQR ranging from 10 to 25 minutes. Therefore, even if the SVM F-measure is slightly lower, this classifier is able to predict the events earlier than the regression methods. This makes a specialized classifier the best choice for glucose event prediction algorithms, and a good performance benchmark for future developments.

3.3 Discussion

A comparison between a number of selected regression and classification algorithms for the prediction of hyper and hypoglycemic events based on

CGM signals has been carried out in this study. The value of the glucose concentration in the blood depends on many external factors that affect glucose-insulin dynamics, such as the amount of CHO of a meal or of a snack, the insulin injected, physical activity, etc.. In addition, CGM devices are often affected by a high level of noise and error, mainly due to the interaction with the human body, that is dynamic by nature. The lack of additional information from the user, which plays a fundamental role, makes the prediction a difficult task. Two different approaches have been investigated: static and a dynamic implementations. Both signal prediction and event detection metrics are considered in this study. The former, typically used in the literature, quantifies the capability of the model to make good predictions; the latter, instead, refers to the capacity of the model to correctly predict future hyper/hypoglycemic events. Static methods exhibit better performance for most of the analyses considered in this work, with particular focus on F-measure values, as shown in Tab. 3.3. However, prediction metrics (RMSE and SSGPE) are not always in agreement with the event detection capabilities of the algorithms, which require a specific analysis. The best results, in terms of event prediction capabilities, have been obtained with Linear and Bayesian Regression methods. All the classifiers show some improvement when trained for a specific and single event, such as hyperglycemia. In particular, Support Vector Machine (SVM) performs nearly the same as the best regressor, under this working assumption. An additional analysis based on the prediction time shows that classification methods also tend to predict the events sooner with respect to regressors. Based on these considerations, a specialized SVM yields the best overall performance.

Chapter 4

A Bayesian Framework for Vehicular Monitoring Networks

Smart cities are witnessing a digital revolution involving a constellation of sensing technologies, which are being employed to gather a range of environmental, parking and traffic data. Application examples for such data abound: park monitoring networks are being installed in major cities such as Los Angeles (CA) [148], San Francisco (CA) [149], and Barcelona [150], among others. Parking sensor data can be utilized to identify free parking lots and, moreover, to pinpoint atypical parking patterns [151, 152]. Solutions to manage traffic flows are also being developed. For example, in [153, 154] video processing techniques are used to estimate the traffic density based on video frames from surveillance cameras deployed on traffic light poles, devising smart control strategies for traffic lights. Other works address traffic forecasting, see, e.g., [155]. A common trait of these systems is that data is generated in large amounts and, in turn, its manual inspection is impractical. Moreover, the patterns of interest are often hidden and difficult to observe through a mere visual inspection, even by skilled personnel. Machine learning tools are deemed a natural means to efficiently and effectively process these data.

In this work, a Bayesian framework for vehicular traffic monitoring networks is proposed. Its core idea is that information from road links that are in close proximity is likely to be highly correlated with that in the current (target) road link, at any time interval. Moreover, temporal correlation is also relevant, i.e., past observations from nearby links also tend to be correlated with the current reading at any target road. Owing to these facts, in this study the spatio-temporal evolution of vehicular streams has been

modeled among multiple road links in large-scale scenarios through *localized* and *small-size* Bayesian Networks (BNs). These, are implemented as Directed Acyclic Graphs (DAG), representing conditional independence relations among random variables. Specifically, a dedicated BN is configured, trained, and tested *for each target road* in the monitored (urban) geographical map. The joint probability distribution between the cause nodes (data utilized for forecasting) and the effect node (data to be predicted at any "current" time, belonging to the target road link) is described through a Gaussian Mixture Model (GMM) whose parameters are estimated via Bayesian Variational Inference (BVI) operating on unlabeled data. Optimal forecasting follows from the criterion of Minimum Mean Square Error (MMSE). Moreover, *anomaly detection* is also performed by devising a probabilistic score associated with the marginal conditional distribution of the effect node. The so obtained GMMs are time-dependent, i.e., several GMMs can be estimated for the same target road for different days of the weeks and/or hours of the day. Also, the proposed framework is *distributed*, lightweight, and capable of operating in realtime and, in turn, it appears to be a promising candidate to deal with Internet of Things (IoT) applications in large-scale networks, where new data is to be processed on-the-fly. The key features of the model are: i) the approach is scalable as a BN is associated with and independently trained for each road, ii) spatio-temporal information is considered (for increased robustness and accuracy of the statistical model so obtained) and iii) the localized nature of the framework allows flagging atypical behaviors at their point of origin in the monitored geographical map. In addition, the approach is here validated against a number of popular regression schemes from the literature, to quantify its predictive power, testing it on data from a large real-world deployment, featuring readings from 686 measurement points for a full year. Finally, the framework's capability of detecting anomalies is quantified in the presence of injected noise, which is precisely controlled in terms of power, location (road link) and position in time. The numerical results reveal that a localized and small size DAG, implemented for each road in the monitored area suffices to obtain very good prediction and anomaly detection accuracies. This means that large monitoring networks can be tackled by training independent and small-size DAGs, a process that can be efficiently parallelized across disjoint processors, ensuring scalability as the size of the network increases.

This chapter is organized as follows. In Section 4.1, the state of the art on anomaly detection and prediction in vehicular data is reviewed. In Section 4.2, the Bayesian framework is formulated, detailing the dataset (Section 4.2.1), the BN/GMM models (Section 4.2.2), the use of real data for training/validation (Section 4.2.3). Numerical results are provided in Section 4.3, assessing the prediction (Section 4.3.1) and anomaly detection (Section 4.3.2) performance of the new scheme. Future research directions are discussed in Section 4.4.

Part of the results presented in this chapter has been published in [156].

4.1 State of the Art Analysis

A substantial amount of research has been carried out to provide anomaly detection techniques in a wide range of application domains such as cyber intrusion detection, fraud detection, medical anomaly detection, industrial damage detection, image processing, textual anomaly detection, sensor networks, etc., and has been reviewed in several surveys [157–160]. The reader is directed to these sources for a detailed and comprehensive treatment of the anomaly detection problem. Next, this study is solely focused on works dealing with traffic analysis.

In [161], the authors introduce a *road segment based* anomaly detection problem, observing the road segment whose traffic condition deviates the most from the expected behavior. This work departs from other scientific papers [162, 163], which are region/grid specific, and not road based. First, a deviation-based method is put forward to quantify the anomaly by means of a score in the range $[0, 1]$. Second, a diffusion-based algorithm exploiting a *heat diffusion model* is proposed to infer the major anomaly causes on the transportation network, given that an abnormal traffic trend in a road segment can trigger another abnormal traffic trend in a road segment located nearby. This model does not include historical information streams from adjacent road links, but represents the expected behavior of the road segment as a normal random variable. In [164], a more sophisticated scheme is presented. There, a Temporal Outlier Discovery (TOD) framework is proposed to quantify the anomaly based on drastic changes in the agglomerated temporal information of the *entire dataset*. Specifically, at each time step, every road segment checks its similarity with respect to every other segment,

and historical similarity values are stored in a temporal neighborhood vector. Anomaly detection for each road is accomplished by jointly considering mobility data from all the streets in the data set. While this should be quite robust in terms of detection capability, as it is expected to be reliable even when all the neighborhood of the current road is experiencing a traffic anomaly, it is *hardly scalable* and *difficult to train and use at runtime* as the number of roads to monitor increases. It is therefore deemed impractical for the large-scale network that is considered in this study. [165] adopts a Bayesian framework. The authors focus on the short-term traffic flow *forecasting* task, which amounts to determine the traffic condition of a *target road* in the near future, usually within a time range of $5 - 30$ minutes. Historical information streams from the adjacent road links and the target link are taken into account by means of Bayesian networks, where the joint probability distribution between the *cause* nodes (directional information streams from the adjacent road links) and the *effect* node (traffic condition of the target link in the next time interval, to be predicted) is represented through a Gaussian Mixture Model (GMM), and forecasting is performed by computing the expectation of the Probability Density Function (PDF) associated with the predicted traffic condition. Also, the authors improve the analysis carried out in [166, 167] by including historical information streams from the adjacent road links and the target link. However, besides forecasting, the authors do not provide any reasonable anomaly detection criterion to state whether the distribution associated with the predicted traffic condition deviates from the expected behavior. In a sense, the authors do not exploit Bayesian networks to their full extent. Furthermore, the scheme is tested on a limited dataset (only $2,400$ sample points from real world traffic data), which make their numerical evaluation quite preliminary.

Here, a Bayesian framework for vehicular traffic monitoring networks is proposed. This approach bears similarities with [165], as both use a BN and an associated GMM. Nevertheless, the effectiveness of localized Bayesian networks is demonstrated in large and real datasets, extending their capabilities to anomaly detection, validating the framework with a large real-world deployment and comparing it against a number of regression approaches from the literature, to test its ability to capture the spatio-temporal structure underpinning real data.

4.2 Bayesian Framework

As briefly discussed above, in vehicular traffic monitoring networks it makes sense to exploit information from multiple road links at past time intervals to forecast the traffic flow in any target road link at any (current) time interval. In the following, a (any) *target* road is considered in the topology, interchangeably referring to it as the *current* road. The objective of the BN is to jointly track current and past speed values for the current road, along with past speed values of *upstream* roads. To cut down the number of connections among multiple road links in the associated DAG, the target road link and its upstream road links are considered in the DAG construction, while neglecting downstream links. This choice allows reducing the number of possible connections in the DAG and, in turn, its complexity. This can be justified from the way BNs represent conditional independence relations among random variables: *a node is independent of its ancestors given its parents, where the ancestor/parent relationship is with respect to some fixed topological ordering of the nodes* [168].

4.2.1 Traffic Readings

Mobility data was acquired from 686 Bitcarrier sensor nodes scanning Bluetooth and Wi-Fi signals of mobile devices traveling on road links [169] for a full year. This sensing system has been deployed by Worldsensing in a major city (details are not provided to protect the company's industrial plans). Readings were taken with a time granularity of 5 minutes (the time slot duration), and each reading from any of the sensors corresponds to the average traveling speed (in [km/h]) gathered during the corresponding time slot (with timestamps in UTC format). For 686 Bitcarrier sensor nodes, this amounts to a total of $54,591,660$ data points in the considered time period from January 2016 to December 2016. To account for missing/unavailable data points, the entire dataset is pre-processed via linear interpolation, jitter removal, and low-pass filtering. After this pre-processing phase, a time series (one per sensor) with one point every 5 minutes is obtained. The Bayesian learning routines operate on these time slotted signals.

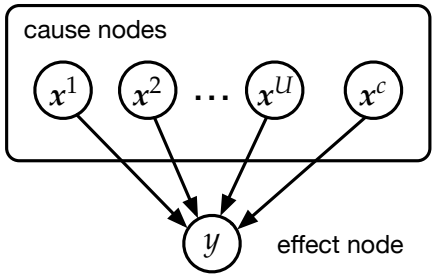


FIG. 4.1: Bayesian network associated with any target node in the physical network topology.

4.2.2 Probabilistic inference via GMM

As per the above discussion, it is assumed that the traffic flow in the current road link at current time interval is independent of other road links, given the traffic flow in the current link and in its upstream road links at past time intervals. Taking advantage of conditional independence relations, the trends of the current link can be statistically analyzed, computing the marginal conditional distribution of the effect (i.e., current) node, as described below.

At any current time t, let $\boldsymbol{z} = (\boldsymbol{x}, y)$ denote a multidimensional random vector: $\boldsymbol{x}$ is a random vector containing the random variables (r.v.s) associated with the cause nodes in the DAG and y denotes a (scalar) r.v. representing the speed value of the current node (to be estimated). In the BN, past measures are considered: if t is the current time slot, the memory spans those samples in $\{t-1, \ldots, t-W\}$, where W is the memory size. Let U be the number of upstream roads in the physical network topology. For each upstream road it is defined a cause node u in the DAG with r.v.s. $\boldsymbol{x}^u = (x^u_{t-1}, x^u_{t-2}, \ldots, x^u_{t-W})$, where $u = 1, \ldots, U$ and x^u_{t-i} represents the speed reading for road u at time $t-i$, with $i = 1, \ldots, W$. For the current road, a further cause node is defined in the DAG, with associated vector $\boldsymbol{x}^c = (x^c_{t-1}, x^c_{t-2}, \ldots, x^c_{t-W})$. Hence, $\boldsymbol{x}$ is obtained as the concatenation of $\boldsymbol{x}^u$ ($u = 1, \ldots, U$) and $\boldsymbol{x}^c$, i.e., $\boldsymbol{x} = (\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^U, \boldsymbol{x}^c)$. The r.v. y contains the speed at the current time t for the current road c. A diagram of the just described Bayesian network is shown in Fig. 4.1. For the numerical results in this study a memory of $W = 5$ time slots is considered, i.e., 25 minutes of historical data.

A GMM is adopted to approximate the joint probability distribution of

z. Besides forecasting, *anomaly detection* is also performed by taking into account a probabilistic score associated with the marginal Cumulative Distribution Function (CDF) of the effect node, as detailed shortly in Section 4.3.2.

The GMM is defined as:

$$P(\boldsymbol{z}|\Theta) = \sum_{m=1}^{M} \alpha_m P_m(\boldsymbol{z}|\boldsymbol{\theta}_m)\,, \tag{4.1}$$

where M is the number of Gaussians in the mixture, whose parameters are $\Theta = \{\alpha_1, \ldots, \alpha_M, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M\}$. α_m are scalars such that $\sum_{m=1}^{M} \alpha_m = 1$. Each $P_m(\cdot|\boldsymbol{\theta}_m)$ is a Probability Density Function (PDF) characterized by $\boldsymbol{\theta}_m = (\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$, $m = 1, \ldots, M$, i.e., $P_m(\boldsymbol{z}|\boldsymbol{\theta}_m) = G(\boldsymbol{z}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$. In this study, the parameters are estimated via Bayesian Variational Inference (BVI) with unlabeled data [168]. BVI can be seen as an extension of the Expectation-Maximization (EM) algorithm from a maximum a posteriori estimation of the single most probable value of each parameter to a complete Bayesian estimation, which computes (an approximation to) the entire posterior distribution of the parameters and latent variables. The marginal conditional distribution of the effect node is computed as:

$$P(y|\boldsymbol{x}) = \frac{P(\boldsymbol{x}, y)}{P(\boldsymbol{x})} = \frac{P(\boldsymbol{x}, y)}{\sum_y P(\boldsymbol{x}, y)}\,, \tag{4.2}$$

where the sum is over the domain of r.v. y and $P(\boldsymbol{x}, y) = P(\boldsymbol{z}|\Theta)$. Exploiting the properties of Gaussian PDFs [170], a concise optimal forecasting relationship can be derived, which allows new data to be processed on-the-fly:

$$\begin{aligned} \widehat{y} &= \int y P(y|\boldsymbol{x}) dy \\ &= \int y \left[\sum_{m=1}^{M} \beta_m G(y|\boldsymbol{x}; \mu_{m,y|\boldsymbol{x}}, \Sigma_{m,y|\boldsymbol{x}}) \right] dy = \\ &= \sum_{m=1}^{M} \beta_m \left[\int y G(y|\boldsymbol{x}; \mu_{m,y|\boldsymbol{x}}, \Sigma_{m,y|\boldsymbol{x}}) dy \right] = \\ &= \sum_{m=1}^{M} \beta_m \mu_{m,y|\boldsymbol{x}}\,. \end{aligned} \tag{4.3}$$

Specifically, for $m = 1 \ldots, M$,

$$\begin{aligned}
\mu_{m,y|\boldsymbol{x}} &= \mu_{m,y} - \boldsymbol{\Sigma}_{m,y\boldsymbol{x}}\boldsymbol{\Sigma}_{m,\boldsymbol{xx}}^{-1}(\boldsymbol{\mu}_{m,\boldsymbol{x}} - \boldsymbol{x}) \\
\Sigma_{m,y|\boldsymbol{x}} &= \Sigma_{m,yy} - \boldsymbol{\Sigma}_{m,y\boldsymbol{x}}\boldsymbol{\Sigma}_{m,\boldsymbol{xx}}^{-1}\boldsymbol{\Sigma}_{m,\boldsymbol{x}y} \\
\boldsymbol{\mu}_m &= (\boldsymbol{\mu}_{m,\boldsymbol{x}}, \mu_{m,y}) \\
\boldsymbol{\Sigma}_m &= \begin{pmatrix} \boldsymbol{\Sigma}_{m,\boldsymbol{xx}} & \boldsymbol{\Sigma}_{m,\boldsymbol{x}y} \\ \boldsymbol{\Sigma}_{m,y\boldsymbol{x}} & \Sigma_{m,yy} \end{pmatrix} \\
\beta_m &= \frac{\alpha_m G(\boldsymbol{x}; \boldsymbol{\mu}_{m,\boldsymbol{x}}, \boldsymbol{\Sigma}_{m,\boldsymbol{xx}})}{\sum_{n=1}^{M} \alpha_n G(\boldsymbol{x}; \boldsymbol{\mu}_{n,\boldsymbol{x}}, \boldsymbol{\Sigma}_{n,\boldsymbol{xx}})} \,.
\end{aligned} \tag{4.4}$$

For the numerical results in this study, M is set to 20 and the GMM parameters Θ are initialized via K-means clustering [168].

4.2.3 Data Matrices and Typical Weekly Profiles

Upon collecting and pre-processing the raw data from the sensor nodes, two additional data objects are defined: 1) the *data matrix* and 2) the *typical weekly profile*. These are defined for *each* target road as follows. 1) The data matrix is the collection of readings gathered from the target road (effect node in the DAG) for all times t and from the cause nodes in the previous W time slots (i.e., variable $\boldsymbol{z}$, see Section 4.2.2). 2) The typical weekly profile is a polished time series, which is *anomaly-free* and will be used in Section 4.3.2 as a ground truth signal to quantify the Bayesian framework's ability to detect anomalies. This profile is obtained as follows: for each time t in a certain day of the week d (e.g., Monday), a window of data points is considered before and after it (the window size is equal to 15 minutes, for a total of 30 minutes with the current time t being in the center of it). All the data points belonging to this time window for this *same* day of the week are collected for an entire year (e.g., all Mondays in a year for a window of 30 minutes centered on t) and the PDF associated with the readings in this time window is computed. For each time t, it is finally evaluated the median of this PDF, which becomes the new datapoint for the considered target road at time t and day d. The typical weekly profile may be seen as a sort of expected behavior for the traffic on each road across the entire year.

4.3 Numerical Results

In this section, the performance of the Bayesian framework of Section 4.2 (referred to in the following as `Bayesian net`) is assessed from two points of view: 1) its **forecasting capability** (Section 4.3.1) and 2) its **anomaly detection accuracy** (Section 4.3.2).

1) Forecasting: for the forecasting capability, `Bayesian net` is evaluated in terms of the error (*residual*) between the original and the predicted data points, i.e., $\text{Res}(y_t, \widehat{y}_t) = y_t - \widehat{y}_t$, where y_t is the speed reading at time t for any of the deployed sensors with $t = 1, \ldots, T$, with T being the number of readings for that sensor in a full year. The following state of the art algorithms are also considered in the performance assessment: Last Sample ("Last sample"), Bayesian Regression ("Bayes regr"), Lasso Regression ("Lasso regr"), Linear Regression ("Linear regr"), Ridge Regression ('Ridge regr"), and Regression Tree ("Tree regr"). Numerical results are obtained using 75% of the data matrix (*training set*) to estimate the GMM parameters for each road in the physical network topology, whereas the remaining 25% (*validation set*) is employed to test the obtained GMM. The forecasting capability of the above schemes is obtained aggregating the results of the data points in the validation set for each sensor node, computing various statistics for the residuals. The prediction performance is shown and discussed in Section 4.3.1.

2) Anomaly detection: the typical weekly profile of Section 4.2.3 is utilized as a ground-truth signal to evaluate the anomaly detection accuracy, as follows. For each sensor node, artificial anomalies, consisting of random noise sequences whose duration is $D = 5$ time slots (5 minutes each), are injected in random non-overlapping positions of the sensor's weekly profile. This artificial signal is a D-sequence of additive, zero-mean white Gaussian noise samples with standard deviation σ_n. For the anomaly detection, a probabilistic *score* is obtained from the marginal distribution of the effect node applied to the noisy weekly profile. Hence, the anomaly rating is compared against a sensor-specific threshold ζ. This threshold is set according to a network requirement quantifying the percentage of anomalies that a sensor is expect to flag in a day in the considered vehicular network. To evaluate the anomaly detection accuracy, the results of all sensor nodes have been aggregated, each evaluated considering a noisy version of its weekly profile. Numerical results are discussed in Section 4.3.2.

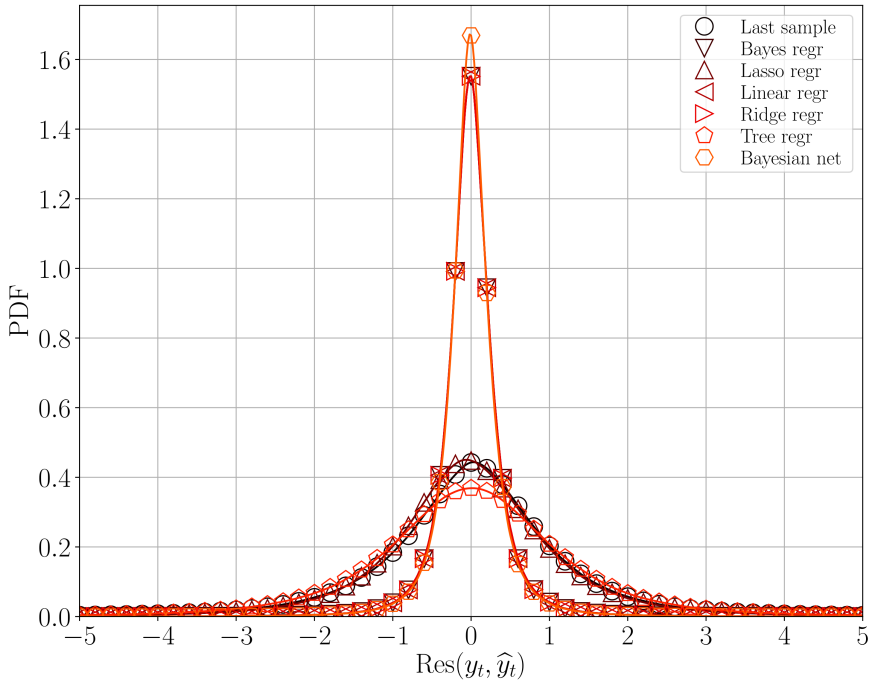


FIG. 4.2: PDF of the residual Res$(y_t, \widehat{y}_t)$.

4.3.1 Forecasting Capability

In Fig. 4.2 it is shown the PDF of the residual Res$(y_t, \widehat{y}_t)$. The `Bayesian net` has a greater forecasting capability that the other schemes. In fact, its PDF is more sharply peaked around zero, meaning a smaller difference between the actual and the predicted samples, $|y_t - \widehat{y}_t|$. As a consequence, the Root Mean Squared Error (RMSE), defined as RMSE $= (\sum_{t=1}^{T}(y_t - \widehat{y}_t)^2/T)^{1/2}$, is also smaller.

Fig. 4.3 shows the (empirically measured) complementary CDF of the residual Res$(y_t, \widehat{y}_t)$, i.e., $P[\text{Res}(y_t, \widehat{y}_t) \geq R]$, where R is kept fixed for all sensor nodes. `Bayesian net` reports a lower number of events for which Res$(y_i, \widehat{y}_i) \geq R$ and, in turn, its curve in Fig. 4.3 decreases the fastest, reaching a minimum of 10^{-6} for $R = 11$ km/h. These results indicate that a Bayesian approach is an appropriate tool to perform forecasting, achieving competitive performance with the best algorithms from the literature.

4.3.2 Anomaly Detection Accuracy

Next, the typical weekly profiles are used as ground-truth signals to evaluate the anomaly detection accuracy. This makes it possible to work with *labeled* time series and, in turn, to precisely quantify the classification performance

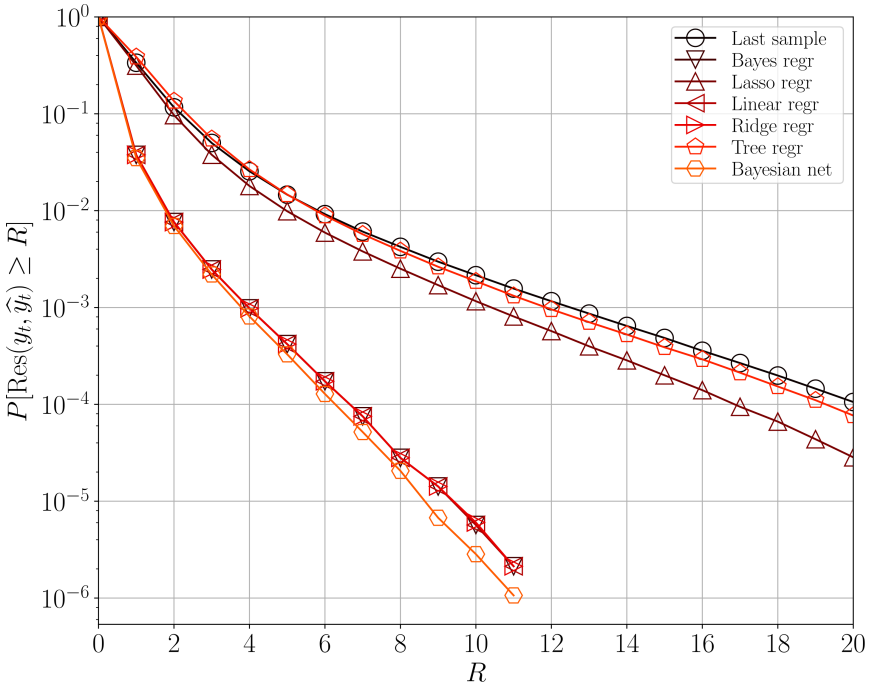


FIG. 4.3: Complementary CDF of the residual $\mathrm{Res}(y_t, \widehat{y}_t)$.

of `Bayesian net` in terms of: i) `Precision`, ii) True Positive Rate (`TPR`) and iii) `F measure`. In Fig. 4.4, a segment from a typical weekly profile is shown for sensor (target) node with ID 1000003 and cause nodes 1000003, 1000090, 1000197 (the speed traces in this plot are synchronized in time).
According to the DAG construction method of Section 4.2.2, the target node 1000003 is also among the cause nodes in the DAG, and it contains the past readings $\{x^c_{t-1}, \ldots, x^c_{t-W}\}$. Instead, the effect node contains y_t, i.e., the speed measured at the target road at time t.

Let's now consider Fig. 4.4 to illustrate how anomalies are injected and detected. For each sensor node, random artificial anomalies of length D time slots are injected in random non-overlapping positions, as explained above. Hence, a probabilistic *score* is computed from the marginal CDF of the effect node (that is computed taking the noisy profile as the input sequence). Whenever the anomaly rating exceeds the (sensor-specific) threshold ζ, the corresponding time slot is flagged as containing an anomaly (see the circular markers in the top subplot of Fig. 4.4). In the bottom subplot, it is shown the score for the trace in the top subplot, which is defined as:

$$\mathrm{SCORE}_t = \begin{cases} \log_{10}(C_t) - \log_{10}(0.5) & C_t < 0.5 \\ -\log_{10}(1 - C_t) + \log_{10}(0.5) & C_t \geq 0.5\,, \end{cases} \tag{4.5}$$

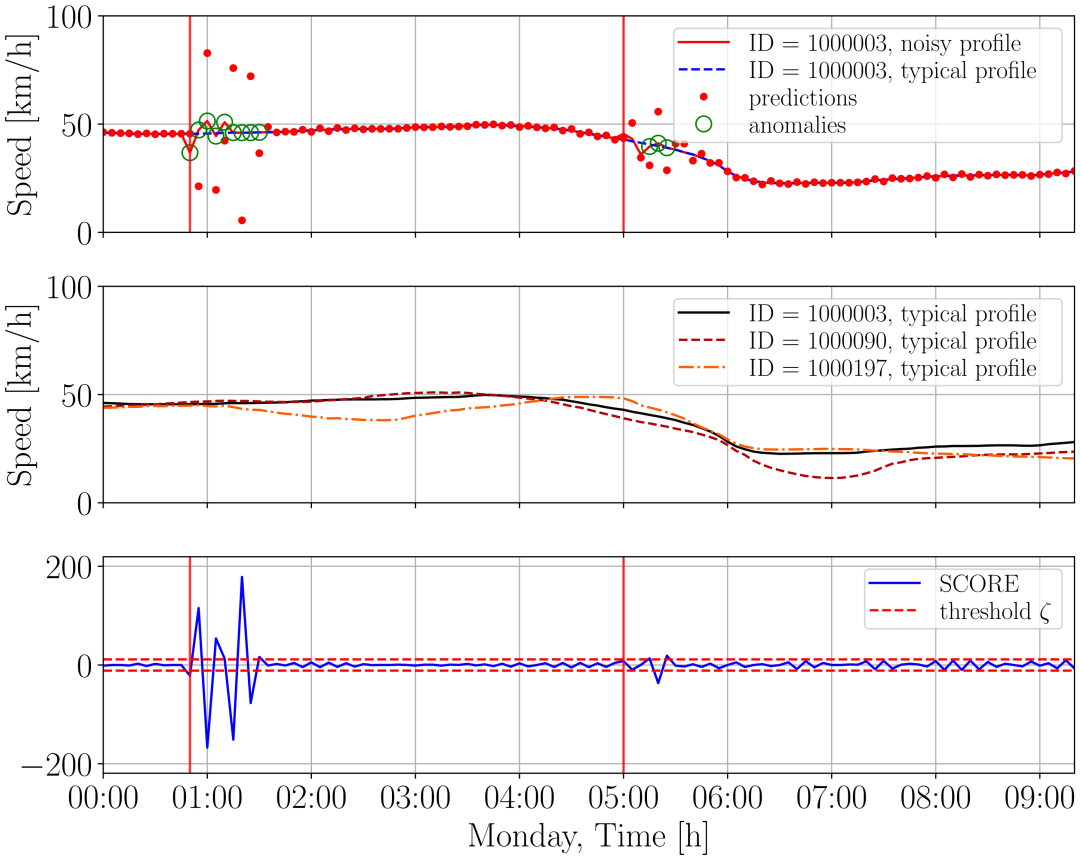


FIG. 4.4: Anomaly detection example for sensor node with ID 1000003 and cause nodes 1000003, 1000090, 1000197.

where $C_t = \text{CDF}(y_t|\boldsymbol{x}_t)$ is the Cumulative Distribution Function computed for y_t and conditioned on the past readings ($\boldsymbol{x}_t$ in the DAG). Using Eq. (4.5), the further the current speed y_t is from the median of the PDF $P(y_t|\boldsymbol{x}_t)$, the greater is the score. A high score means that the speed value y_t is atypical with respect to what would be predicted by the (marginalized) PDF. In this example, $\sigma_n = 5$ km/h is used, which is the maximum noise level that was considered in the experiments. As for the threshold ζ, for Fig. 4.4 $\log_{10}(\text{no. of anomalies}/\text{no. of samples}) = -3$ (application *requirement*) has been set, where "no. of samples" is the total number of data points in the validation set for each sensor node; ζ is numerically found to meet this. For a given threshold ζ, anomalies are detected (circular markers in the top subplot of Fig. 4.4) by assessing whether $|\text{SCORE}_t| \geq \zeta$. Note that in Fig. 4.4 single target road is considered and, as such, the given application requirement is used to compute a single threshold ζ for that road. However, for an entire network, this same requirement is employed to derive one threshold per DAG (i.e., one for each target road in the physical topology).

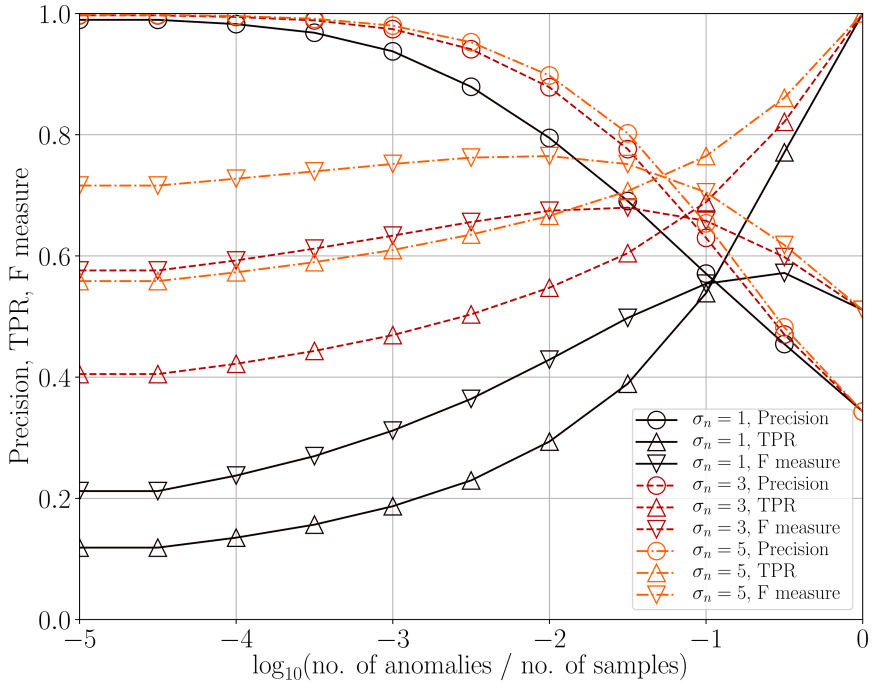


FIG. 4.5: BN performance, when used as an anomaly classifier.

Referring to α as the total number of (artificial) anomalies that were injected, the number of time slots that may be possibly affected is $S = \alpha(D + W)$. This is because anomalies are non-overlapping, each anomaly lasts D time slots and its effect could propagate for W further time slots due to the memory in the DAG, i.e., $D + W$ is the support of a single anomaly. Given this, let's define $\mathcal{S}$ (with $|\mathcal{S}| = S$) as the set of time slots that could possibly contain an anomalous reading, as per the previous reasoning. From this definition, it follows that the maximum number of True Positives (TP) is $|\mathcal{S}| = S$. The number of False Negatives (FN), False Positives (FP), and True Negatives (TN) are also considered, and $\sum X$ represents the total number of time slots that are flagged as being of type X, with $X \in \{\texttt{TP}, \texttt{FP}, \texttt{TN}, \texttt{FN}\}$. For instance, in the example of Fig. 4.4, we have $\alpha = 2$, $D = 5$, $W = 5$, $S = 20$, $\sum \texttt{TP} = 13$, $\sum \texttt{FN} = 7$, $\sum \texttt{FP} = 0$, and $\sum \texttt{TN} = T' - S$, where T' is the number of time slots in the graphs. For the following results, $\alpha = 70$ has been used, which corresponds to an average of 10 artificial anomalies that are added per day.

Fig. 4.5 shows the classification performance of the proposed score-based anomaly detector in terms of: i) `Precision`, ii) `TPR` and iii) `F measure` (F).

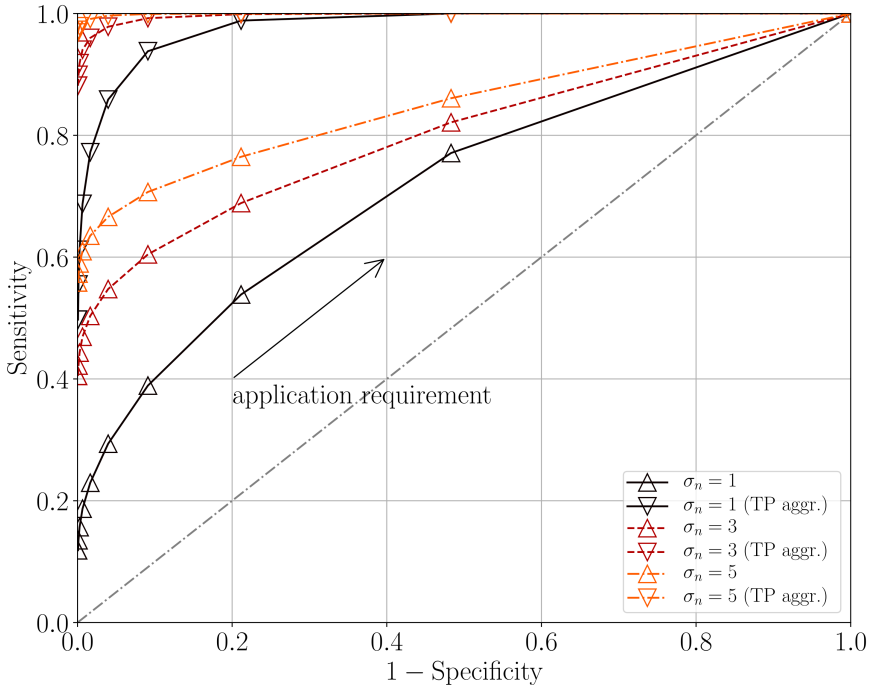


FIG. 4.6: The ROC space.

These metrics are defined as follows:

$$\texttt{Precision} = \frac{\sum \texttt{TP}}{\sum(\texttt{TP} + \texttt{FP})} \tag{4.6}$$

$$\texttt{TPR} = \frac{\sum \texttt{TP}}{\sum(\texttt{TP} + \texttt{FN})} \tag{4.7}$$

$$F = \frac{2\sum \texttt{TP}}{(2\sum \texttt{TP} + \sum(\texttt{FP} + \texttt{FN}))} \tag{4.8}$$

In Fig. 4.5, these metrics are plotted as a function of the application requirement (i.e., $\log_{10}$(no. of anomalies/no. of samples)), which is reported in the abscissa. As an example, a requirement equal to zero means that *all* the time samples are flagged as containing an anomaly and, as such, the true positive rate is $\texttt{TPR} = 1$. However, in this case, the `Precision` is heavily impacted by the number of false positives (FP), which is at least $T - S$, where S is the maximum number of time slots affected by real anomalies (true positives) and T corresponds to the number of time slots in the time series. As expected, the anomaly detection accuracy increases with an increasing noise level, approaching $F = 0.8$ for $\sigma_n = 5$ km/h (when the requirement on the x-axis is -2). Also, a higher `Precision` entails a smaller TPR and vice-versa.

In Fig. 4.6, the Receiver Operating Characteristic (ROC) space is shown, obtained plotting the TPR (i.e., `Sensitivity`) against the False Positive Rate,

$\texttt{FPR} = \sum \texttt{FP} / \sum(\texttt{FP} + \texttt{TN})$ (i.e., $1 - \texttt{Specificity}$) varying the application requirement as a free parameter. This space shows the discrimination capability of the score-based anomaly classifier by varying the requirement. Ideally, we would like to get $\texttt{TPR} \to 1$ and $\texttt{FPR} \to 0$, which means that desirable working points lie in the upper-left corner of the ROC space. As expected, the anomaly detection accuracy increases with an increasing noise level (increasing σ_n). Moreover, the performance of the proposed Bayesian framework can be further improved through the following TP aggregation criterion ("TP aggr."). As discussed above, each anomaly has an associated support of $D + W$ time slots. Hence, whenever the score exceeds the threshold at any given time instant, $D + W$ data points per anomaly instance are counted as true positives if at least one alarm is raised within the real (and known) support of the injected anomaly. With aggregation, the ROC curves effectively move towards the upper-left corner of the space, leading to some major improvement. For the example in Fig. 4.4, this strategy leads to $\sum \texttt{TP} = 20$, $\sum \texttt{FN} = 0$, $\sum \texttt{FP} = 0$, and the total number of true negatives is $\sum \texttt{TN} = T' - S$, where T' is the number of time slots in the plot. The rationale about this approach is that, if there is at least one alarm within the support of an anomaly instance, in practice, this may be sufficient to declare the entire anomaly instance as detected.

4.4 Discussion

With this work it has been demonstrated that localized Bayesian networks are an efficient and lightweight means to tackle prediction and anomaly detection problems in large vehicular networks. Nevertheless, a few research avenues remain open. An automated procedure could be set up to adapt the memory size W and the considered upstream nodes in the DAG in a sensor-specific fashion. Suitable dimensionality reduction tools could be used to reduce the complexity associated with the BN training task. A more refined Bayesian model could be defined, associating a state to each sensor according to the locally experienced traffic condition (e.g., normal, congested) and specializing the GMMs to it.

Chapter 5

Deep Reinforcement Learning for DASH Video Streaming

Video streaming has been the dominant source of Internet traffic for the last few years; right now, videos make up 55% of all mobile traffic, and this figure is predicted to rise to 75% in the next five years [171]. Since its inception in 2011, Dynamic Adaptive Streaming over HTTP (DASH) [172] has become the dominant standard for video transmission, as it relies on the existing HTTP server and Content Delivery Network (CDN) infrastructure and is not affected by firewalls and NATs. The DASH standard leaves complete freedom to the client in the choice of the adaptation policy: videos are divided into short segments (usually a few seconds long), which are encoded at different compression levels to generate an *adaptation set* of *representations* at different bitrates. The server makes all the segments in the adaptation set available to the client as HTTP resources, as well as a Media Presentation Description (MPD) file containing all the information about the video segments and their URLs. The client sequentially downloads each segment, switching between representations according to its adaptation logic in an attempt to optimize the Quality of Experience (QoE) for the current video and network conditions.

The research on DASH adaptation algorithms is still ongoing, and most commercial systems employ very basic heuristic approaches, leading to annoying quality variations [173, 174] and to an inefficient use of network resources. In order to maximize the user QoE, the adaptation logic needs to take into account both the video content, which affects the perceived quality of the downloaded representations, and the playout buffer state. In fact, a major factor in video QoE is rebuffering [175], i.e., the temporary freezes in the video playout as the client waits until the next segment is downloaded [176].

In this challenging scenario, Reinforcement Learning (RL) has emerged as an elegant and viable solution to the video adaptation problem. RL-based algorithms learn from past experience by trial-and-error, and gradually converge to the optimal policy [177]. The biggest design issue in these systems is that the state space of the corresponding Markov Decision Process (MDP) is very large. On the other hand, the number of states needs to be small to ensure quick convergence and make online solutions react timely to changes in the environment statistics.

In this work, deep neural networks are used to learn excellent video adaptation strategies, while compactly and effectively capturing the experience acquired from the environment. The use of deep neural networks leads to several advantages, such as: i) the ability to deal with very large state spaces efficiently, effectively coping with the curse of dimensionality issue of RL algorithms, ii) the possibility of compactly representing the acquired experience through a set of weights, iii) the attainment of much better QoE performance, that is here quantified in terms of both the instantaneous visual quality of each segment and the quality variations across video segments, as well as the frequency of freezing/rebuffering events. Here, D-DASH is hence proposed, a framework for DASH video streaming that, combining deep neural networks with a carefully designed reinforcement learning mechanism, yields better QoE than prominent state of the art rate-adaptation algorithms. More specifically, the key points of this study are the following.

- Existing RL-based techniques for video adaptation have been reviewed, emphasizing their limitations in terms of training effort (time and required memory space) and QoE.

- The DASH video streaming problem has been formulated within a Deep Q-learning framework, detailing the design choices, that include: the choice of a reward function that effectively takes into account video quality variations and freezing/rebuffering events, the use of a learning architecture featuring two twin neural networks and a replay memory to get an improved stability and a faster convergence, and the use of a pretraining phase (on synthetic traces) to speed up the convergence of D-DASH when confronted with real traces.

- Deep forward and recurrent neural network architectures have been evaluated, assessing their training cost (time and memory) and their

attained QoE.

- A thorough numerical validation of D-DASH is provided, considering real and simulated traces and comparing it against prominent algorithms from the state of the art.

The results shed some light on the importance of tracking a certain amount of channel memory in the learning architecture, especially for complex network scenarios, and the superiority of the proposed deep Q-learning designs: in all the experiments, these have shown a higher average quality with smaller fluctuations across video segments, and a much lower percentage of rebuffering events.

The rest of the chapter is organized as follows: Section 5.1 presents the state of the art in DASH adaptation logic, with a focus on RL-based algorithms. Section 5.2 defines the system model, detailing the decision making instants, the playout buffer behavior and the reward function. Section 5.3 uses these notions within a decision making framework, starting from a Markov decision process approach to then delve into standard and deep Q-learning based ones. Section 5.4 provides some preliminaries on the neural network structures that will be considered in this framework and Section 5.5 presents the D-DASH deep Q-learning system. Section 5.6 describes the simulation settings/parameters, quantifies the performance of D-DASH over synthetic and real channel traces, and compares it against selected algorithms from the literature. Finally, Section 5.7 concludes the work.

Part of the results presented in this chapter has been published in [178].

5.1 Related work

In this section, it is provided an overview of the most relevant works on DASH client-side adaptation strategies in the literature. As measuring QoE itself is a subject of intensive research in the field, there is not a single set of reference performance metrics [179]. In-depth reviews of the factors that impact QoE and the way they are measured in different streaming systems can be found in [180, 181]. Among such elements, in this work following three factors are considered. The first, and most obvious, is the *instantaneous picture quality*, which can be assessed through different techniques. Notable ones are: objective metrics such as the bitrate, no-reference metrics, which

gauge the video distortion solely from the received frames (i.e., no external quality reference is provided) [182], or full-reference metrics such as Structural Similarity Index (SSIM) [183], a perception-based metric which measures the distortion between the input and output of the video encoder at the transmitter side. These approaches are adopted by various DASH adaptation algorithms in the literature [184, 181, 185]. The second factor, which is often the most pressing concern in adaptive video streaming, is represented by *rebuffering events*: their length and frequency strongly affect user QoE, as demonstrated by Hoßfeld *et al.* in [175]. The third factor is *video quality stability*: users notice frequent transitions in the video quality and might be annoyed by them [186]; in DASH video streaming, quality changes are not due to packet loss, but to switches between different adaptations (i.e., sudden changes in the quality of the video).

The literature on adaptation logics for DASH is vast, and the reader is referred to [187] for a comprehensive overview of existing techniques. For the purpose of this work, adaptation logics can be divided into two broad categories: 1) heuristics and 2) dynamic programming based. One of the earliest adaptation algorithms to go beyond simple buffer-based control loops was the Fair, Efficient, and Stable adapTIVE algorithm (FESTIVE) [188]: this scheme does not address QoE explicitly, but rather uses a stability cost function and a limit on the frequency of bitrate increases to privilege stability over instantaneous video quality. FESTIVE has the added advantage of being fairer than commercial protocols in terms of QoE, as well as of reducing the number of rebuffering events because of its conservative approach when increasing the bitrate.

Probe and Adapt (PANDA) [189] is another landmark algorithm in the DASH adaptation literature. It uses a proactive probing strategy to evaluate the channel capacity and changes its rate estimate according to an additive-increase approach to prevent fluctuations due to non-persistent cross-traffic and on-off effects. In stationary conditions, its bitrate estimate equals the TCP throughput. PANDA then uses a hysteresis threshold to avoid frequent switches between adjacent representations. A similar but simpler heuristic, called QoE-driven Rate Adaptation Heuristic for Adaptive video Streaming (QoE-RAHAS), was presented by Petrangeli *et al.* in 2014 [190].

Both PANDA and FESTIVE tend to be extremely conservative in their decisions, and they often end up underutilizing the available capacity unless

the network conditions are extremely stable. Heuristics are hard-pressed to efficiently exploit the available capacity and still limit rebuffering events and quality switches, unless they are designed with knowledge of the network statistics, thus enabling the adoption of a dynamic programming approach to optimally solve the adaptation problem.

Li *et al.* use finite-horizon dynamic programming [189], modeling the adaptation problem as a fairness problem between future segments, with an added penalty for quality switches. Adding the adaptation logic on top of PANDA's switching strategy, the authors manage to significantly improve video quality without losing the algorithm's buffer and quality stability properties. In order to reduce the computational load, the authors assume that the available capacity remains constant throughout the optimization horizon. However, this assumption could prove to be unrealistic in wireless channels, which can suffer from outages and quick capacity variations due to mobility, fading and cross-traffic.

Yin *et al.* developed a system [191] that uses Model Predictive Control (MPC) to make decisions based on a look-ahead approach, with a QoE model similar to that considered in this work. More specifically, they rely on a throughput predictor to make optimal choices for a finite time horizon of five future segments. This approach is equivalent to dynamic programming and has the same critical dependency on the quality of the throughput predictor: if the predicted channel statistics are inaccurate, the adaptation logic will make suboptimal decisions.

An interesting paper by Bokani *et al.* [192] uses a Markov Decision Process (MDP) model to determine the optimal adaptation policy with dynamic programming. The main issue of this solution is the computational load: the model is too complex to be solved at runtime. The authors propose several solutions to mitigate the complexity issue, but the performance of these heuristics is either unsatisfactory or requires to store a large amount of offline computational results in the device's memory. A more recent work by Zhou *et al.* [193] proposes a pseudo-greedy heuristic to tackle the same problem, with the same kind of limitations. Other works use MDPs in more specific situations, such as streaming for multihomed hosts [194] and cloud-assisted adaptive streaming [195].

5.1.1 Reinforcement Learning and DASH

There are several works in the literature that use RL to overcome dynamic programming's two biggest drawbacks: computational load, and the need to know the statistics of the network and video content in advance. RL-based algorithms learn the network statistics from experience, and their computational complexity is low. The main limitation of RL is the amount of experience it needs to make good decisions: as the number of states of the MDP grows, so does the necessary training time. Specifically, adaptive RL algorithms need to have a very coarse state granularity to effectively react to changes in the environment. Hence, there is a fundamental trade-off between adaptability and descriptive power: to be adaptable, the algorithm will need to visit and update each state frequently, i.e., to have a small number of states. However, having fewer states means that the knowledge of the environment is inevitably poorer (and, probably, so will be the algorithm's choices).

Two works by Claeys *et al.* represent the two opposite extremes in this trade-off: the algorithm in [196] has a complex reward function and a 6-dimensional state definition, and its training requirements are daunting, while the algorithm presented in [197] is lean and converges very quickly, but has a very rough state definition and a suboptimal performance. Another lean Q-learning algorithm that uses a similar MDP model was presented by Martín *et al.* in 2015 [198].

In 2015, Van Der Hooft *et al.* presented an interesting hybrid between RL and standard algorithms [199]: their system is based on Microsoft Smooth Streaming (MSS), but adapts the parameters of the heuristic to reflect the network conditions, improving its performance. The algorithm is still not QoE-aware, and the simple buffer-based MSS heuristic is suboptimal, but hybrid solutions such as this might be an interesting approach to overcome some of RL's main drawbacks.

In another recent work [200], the authors try to overcome the basic trade-off between efficiency and adaptability by parallelizing the learning process: since the problem has a well-known structure, experience in one situation can be used to learn how to act in others. This ad-hoc generalization scheme relies on the specific structure of the problem, but it is a step in the right direction.

As it will become apparent in the next sections, the D-DASH framework

makes it possible to better exploit the observed data, learning from experience faster than other algorithms in the literature and quickly adapting the video encoding policy to the current working context.

5.2 System model

As mentioned above, a DASH client that downloads a video sequence segment by segment has been considered in this study. The decision-making process follows a slotted time model, where $t = 1, 2, \ldots$ is the video segment number. For any given segment t, the client is free to choose which representation to download from a given adaptation set A. Each representation, in turn, is uniquely associated with a certain video quality level of the segment, indicated by q_t. The aim is to find the policy that maximizes the QoE perceived by the user. In the following of this section the assumptions are presented regarding the video streaming service and then it is introduced the reward function that accounts for the QoE factors described in the previous section. Along the way, the main notation used in the chapter will also be introduced.

5.2.1 Video streaming services

Each video segment is characterized by a certain quality-vs-rate trend, which is described by means of the function $F_t(q_t)$ that gives the size of the segment t with quality q_t. $F_t(\cdot)$ is assumed to be known before downloading segment t, as it can be made available to the client as part of the MDP or predicted from previous scenes, since the correlation between subsequent segments is usually high.

Denoting by C_t the average channel capacity experienced during the downloading of the segment t, the total downloading time τ_t can be easily obtained as

$$\tau_t = \frac{F_t(q_t)}{C_t}. \tag{5.1}$$

T is the (constant) playout duration of a video segment. Moreover, let's define the *buffer* (time) for segment t, denoted by B_t, as the lapse of time between the starting of the download of segment t and the instant the segment is due to start its playout at the client. Rebuffering events (during which

the video playout freezes) occur whenever the playout buffer empties before the next segment has been completely downloaded, i.e., when $\tau_t > B_t$. Conversely, when $\tau_t < B_t \leq T$ the download of the segment t is completed before its planned playout time and the download of the next segment can start immediately, thus adding an extra $B_t - \tau_t$ time budget to the buffer of segment $t+1$. Accordingly, the rebuffering time for a generic segment t is given by:

$$\phi_t = \max(0, \tau_t - B_t)\,, \tag{5.2}$$

while the buffer for the next segment is computed as

$$B_{t+1} = T + \max(0, B_t - \tau_t)\,. \tag{5.3}$$

The buffer is limited to 20 seconds, as most commercial video streaming systems limit video buffering to save memory and network capacity.

5.2.2 Reward function

As discussed in Section 5.1, the QoE of a video client depends on the visual quality of the current segment, the quality variation between segments, and the playout freezing events due to rebuffering. In the following, it is introduced a reward function that captures these aspects and, in turn, can be used to derive policies that maximize the QoE of video streaming customers.

In this work, the SSIM has been chosen as the instantaneous quality metric of a video sequence, so that q_t is the average SSIM of the video frames in segment t. SSIM is one of the most common objective metrics in the literature and has been shown to correlate well with the perceived QoE [201]. As it is a full-reference metric (i.e., its computation requires a full knowledge of the uncompressed segment [202]), it cannot be calculated by the streaming client, but its values when varying the video representation can be conveniently pre-computed, stored on the video server, and included as a field in the MPD. This puts the computational burden for calculating the SSIM onto the server side, even though, as reported in [203] the $F_t(q_t)$ characteristic of a video sequence can be automatically estimated from the size of the *encoded frames* by using a properly-trained deep neural network, thus making it possible to skip the computationally intensive frame-by-frame comparison with the original video sequence at the server. Other approaches are possible, for example computing the quality q_t of video frames at the client through a

no-reference metric [182]. Although not considered in this study, this is also viable and would require the implementation of an additional block at the client to estimate the quality of the received frames. The impact of inaccurate quality estimates, via no-reference metrics, is left to future investigations.

Finally, the reward function for segment t is defined as follows:

$$r(q_{t-1}, q_t, \phi_t, B_{t+1}) = \\ = q_t - \beta \|q_t - q_{t-1}\| - \gamma \phi_t - \delta [\max(0, B_{\text{thr}} - B_{t+1})]^2 . \tag{5.4}$$

The first term on the right-hand side accounts for the benefit of a higher quality q_t of the video, while the following two negative terms are penalty factors due to *quality variations* in consecutive frames and *rebuffering events*, respectively. The right-most term is a further penalty that is applied whenever the buffer level is below a (preset) threshold B_{thr} and it has been introduced to further reduce the chance of highly-damaging rebuffering events. The coefficients β, γ, and δ are weighting factors that regulate the relative importance of the three penalty terms. Note that, neglecting the right-most penalty term (i.e., setting δ to zero), the reward function (5.4) is the same used in [200] and [191], and similar to that proposed and validated by De Vriendt *et al.* in [186].

The weights β, γ and δ are here used to select different points in the trade-off between a high instantaneous quality, a constant quality level, and a smooth playback. The desired operational point might depend on several factors, including user preferences and video content, and tuning these parameters is outside the scope of this work.

5.3 Machine learning optimization framework

This section presents the machine-learning based approaches considered in this study to find video adaptation policies that try to maximize the perceived QoE. First, the problem is formulated as an MDP, and then the Q-learning and Deep-Q learning algorithms are introduced and detailed.

5.3.1 Markov Decision Process model

If video content is modeled as a sequence of scenes with exponentially distributed duration, like in [204], the adaptive video streaming service can be modeled as an MDP (see, e.g., [205]), with action space A, state space S, and

reward function $\rho : S \times S \times A \to \mathbb{R}$. By a slight abuse of notation, but for the sake of a compact notation, q_t is also used to indicate the action of downloading a segment t with visual quality q_t. The action $q_t \in A$, taken when the system is in a given state $s_t \in S$, determines the statistical distribution of the next state s_{t+1} and the reward $\rho(s_t, s_{t+1}, q_t)$ attained in step t. A policy is a function $\Pi : S \to A$ that maps states into actions. The *long-term utility* achieved by an admissible policy Π when starting from state s_0 is defined as

$$R(s_0; \Pi) = \lim_{h \to +\infty} \mathrm{E}\left[\sum_{t=0}^{h} \lambda^t \rho(s_t, s_{t+1}, \Pi(s_t)) \middle| s_0, \Pi \right] \tag{5.5}$$

where $\lambda \in [0, 1)$ is a discount factor that ensures convergence, and $P(s_{t+1}|s_t)$ is the one-step conditional transition probability of the state process $\{s_t\}$. An equivalent recursive formulation is given by:

$$R(s_0; \Pi) = \sum_{s_1 \in S} P(s_1|s_0)\rho(s_0, s_1, \Pi(s_0)) + \lambda R(s_1; \Pi). \tag{5.6}$$

The optimal policy $\Pi^*(\cdot)$ is finally found as:

$$\Pi^*(s) = \arg\max_{\Pi} R(s; \Pi), \quad \forall s \in S. \tag{5.7}$$

The action space, the state space, and the utility function need to be mapped to the parameters of the proposed system in order to apply this approach to this problem.

The action space clearly corresponds to the set A of possible representations of the video segments that can be chosen by the client. Accordingly, the action at step t corresponds to the choice of the quality q_t of the next segment to be downloaded. Therefore, from a state $s_t \in S$, an agent implementing a policy $\Pi(\cdot)$ will require the downloading of the segment t with quality $q_t = \Pi(s_t)$. The state space should be as small as possible, to reduce the complexity of the MDP, but at the same time each state should contain enough information to permit a precise evaluation of the utility function for each possible action $q_t \in A$. Considering the reward function Eq. (5.4) as a natural option for the utility function of the MDP, the state should hence include the video quality q_{t-1} of the last (the $(t-1)$-th) downloaded segment, the current buffer B_t state, the quality-rate characteristic $F_t(q_t)$ of the next segment (the t-th), and the future channel capacity C_t from which, given

the chosen action q_t, it is possible to determine the download time τ_t of the next segment, the rebuffering time f_t and the next buffer state B_{t+1} using Eq. (5.1), Eq. (5.2), and Eq. (5.3), respectively. However, the future capacity C_t of the channel can only be known in a statistical sense, from the past observations. Let's then define the vector $\mathbf{C}_t = [C_{t-n}, C_{t-n+1}, \ldots, C_{t-1}]$ of the previous n channel capacity samples, where n is assumed to be larger than the coherence time of the channel. In this case, the process $\mathbf{C}_t$ exhibits the Markov property, since the knowledge of samples that are further away in the past do not add any information to that contained in the current channel vector $\mathbf{C}_t$. Summing up, the state of the MPD at step t can be described by the 4-tuple $s_t = (q_{t-1}, \mathbf{C}_t, \phi_t, B_t)$.

Given the state s_t and the action q_t, the utility function of the MDP can be finally defined as

$$\rho(s_t, s_{t+1}, q_t) = r(q_t, q_{t-1}, \phi_t, B_{t+1}). \tag{5.8}$$

The problem Eq. (5.7) can then be solved through dynamic programming, e.g., Value Iteration (VI) [205], but the computational complexity becomes rapidly unmanageable as the size of the problem grows. A possible approach to deal with the curse of dimensionality is to adopt RL tools, such as Q-learning [206], which gradually converges to the optimal solution through trial-and-error, as explained next.

5.3.2 Q-learning

Q-learning is a RL algorithm introduced by Watkins in 1992 [206]. It works by maintaining a table of estimates $Q(s, q)$ of the expected long-term reward (given by (5.6) in the problem formulation) for each state-action pair (s, q).

The Q-learning algorithm can use various exploration policies to decide the next action based on the Q-values: the most common are the ε-greedy policy and Softmax. Both strategies are non-deterministic; the ε-greedy policy chooses the action with the highest Q-value with probability $1 - \varepsilon$, and a sub-optimal action at random with probability ε. Softmax chooses an action according to the softmax distribution of Q-values:

$$p(q_t|s_t) = \frac{\exp\left(-\frac{Q(s_t,q_t)}{\xi}\right)}{\sum_{q\in A}\exp\left(-\frac{Q(s_t,q)}{\xi}\right)}, \tag{5.9}$$

where the parameter ξ sets the greediness of the algorithm. Greedier algorithms make suboptimal choices less often, but run a higher risk of getting stuck in local minima since they explore the state space less frequently.

In standard Q-learning, the Q-value $Q(s_t, q_t)$ is updated if the learning agent takes action q_t in state s_t. The future reward over the infinite time horizon is approximated as $\max_{q \in A} Q(s_{t+1}, q)$, i.e., the best Q-value of the future state s_{t+1}. If the Q-value matches the real expected long-term reward, Q-learning coincides with the Bellman formulation, which exactly solves the problem. In practice, since the Q-values are to be learned at runtime, they only provide an approximation of the real long-term rewards, but it has been proved that they converge to the optimal rewards for sensible exploration policies. The Q-learning update function is given by:

$$\hat{R}(s_t, q_t) \quad = \rho(s_t, s_{t+1}, q_t) + \max_q \lambda Q(s_{t+1}, q) \tag{5.10}$$

$$Q(s_t, q_t) \quad \leftarrow Q(s_t, q_t) + \alpha(\hat{R}(s_t, q_t) - Q(s_t, q_t)) \,, \tag{5.11}$$

where the learning rate α sets the aggressiveness of the update and is usually decremented over time as the learning agent gets closer to convergence. The choice of the maximum Q-value in the bootstrap approximation Eq. (5.10) makes Q-learning an *off-policy* learning algorithm, since the greedy policy used in the long-term reward estimation (update policy) usually differs from the actual policy the learner uses (exploration policy). As the Q-values converge, the exploration policy should become greedier until it reaches convergence.

Limits of the Q-learning approach

The Q-learning approach is powerful, yet with some limitations: the algorithm provably converges to the optimal policy if its parameters are chosen correctly [206], but the convergence speed is an issue in complex problems. In [207], Kearns and Singh proved that, for an MDP with N states and A actions, the number of samples necessary for the expected reward to converge within ε of the optimal policy with probability $1 - p$ is bounded by:

$$O\left(N\,A\,\varepsilon^{-2}\left(\log\left(\frac{N}{p}\right) + \log\left(\log\left(\varepsilon^{-1}\right)\right)\right)\right). \tag{5.12}$$

For fixed values of ε and p, the number of training steps is $O(NA\log(N))$,

but the constant factor can be significant. Due to the curse of dimensionality, the number of states of the MDP tends to be very large for all but the most trivial problems, making standard Q-learning need a huge amount of training samples to reach convergence and obtaining a good trade-off between precision and adaptability.

Depending on the definition of the video adaptation MDP, standard Q-learning algorithms can either be fast to converge and adaptable in an online setting [197, 198] or efficient after convergence [196]: the number of states necessary to accurately represent the environment makes Q-learning slow and unwieldy.

The objective is to concoct RL algorithms that converge quickly and that are capable of generalizing based on experience, i.e., that can cope with *previously unseen* channel/quality patterns, and that approximate well the optimal policies that would be obtained by solving the MDP. To achieve this, referring to s' as the state of the system upon taking action q and by r the actual reward accrued from that action, we advocate using the 4-tuple (s, q, r, s') to update $Q(s, q)$ and, along the same lines of [200], to concurrently improve the Q-values associated with other states and actions. As explained below, this is obtained through Deep Q-learning, as it provides a natural and effective way to generalize the knowledge acquired during specific transitions and reuse it for other states and actions.

5.3.3 Deep-learning integration

Considering the scenario that tackled in this study, Q-learning has two main drawbacks:

- *Continuous state space*. The quantities defining the state space, namely, the buffer state and the channel capacity take value in some real interval. The definition of an MDP of manageable size involves a quantization of buffer and channel capacity according to a predefined number of levels (dictated by the quantization step). The smaller the quantization step, the better the approximation of the actual (continuous) variable and the more accurate the fit between the MDP and the process it represents. However, the number of states grows very quickly as the quantization step gets smaller. The best compromise between representation accuracy and number of states is often difficult to find;

- *Curse of dimensionality*. This is a direct consequence of the previous point, since the number of samples required for Q-learning to converge grows very quickly with the number of states, according to Eq. (5.12). Here, the problem is not only related to the convergence time, but also to the data availability: optimal policies may be hard to attain due to the need for too many data samples, which may not be available in practical settings.

With *deep Q-learning* these issues are addressed by approximating the action-value function $Q(s,q)$ through a neural network that, for any given state and action pair (s,q), returns the corresponding (estimated) Q-value $Q(s,q)$. The network weights, once trained, will encode the mapping and replace the tables used by Q-learning. This allows the model to be fed with continuous variables, avoiding the quantization problem, and has the further desirable property that neural networks, if properly trained, are able to *generalize*, providing correct answers (excellent Q-value approximations) even for points (s,q) that were never processed in the training phase. In other words, neural networks act as universal approximators. As quantified below, in Sec. 5.6.4, this amounts to a reduction of the number of training samples that are required to reach a certain performance level. With this approach, the RL logic remains unchanged, and there is no restriction on the neural network architecture to use. The network's weights are updated to approximate the optimal action-value function $Q^*(s,q)$ using typical gradient descent optimization methods: the numerical results shown in this study have been obtained by using the Adaptive Moment Estimation (Adam) method [208], in conjunction with back-propagation. The use of neural networks within the proposed architecture for video-streaming control is further discussed in Sec. 5.6.4.

5.4 Neural network architectures: preliminaries

A typical neural network consists of many simple units, called neurons, connected in several ways which depend on the implemented architecture. Input neurons get activated directly by the environment state variables, while other neurons are activated through weighted connections from neurons residing in previous layers. Given the architecture, i.e., the way neurons are

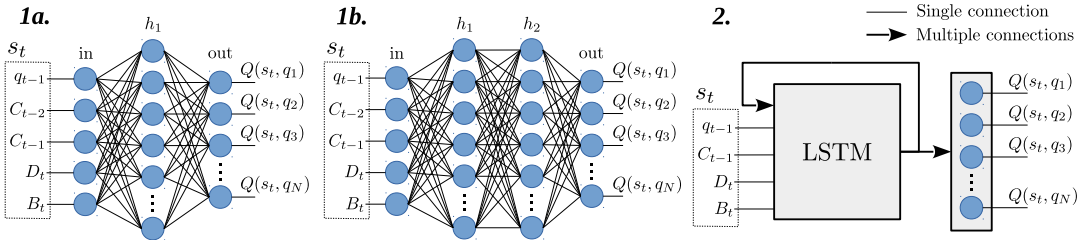


FIG. 5.1: Network architectures considered in this study: an MLP network with one (***1a***) and two (***1b***) hidden layers; a RNN based on an LSTM cell (**2**). For this plot, the channel memory was assumed $n = 2$, i.e., $\boldsymbol{C}_t = [C_{t-2}, C_{t-1}]$.

connected, and the activation functions, i.e., how the weighted input is reshaped by each neuron (and subsequently sent forward to the next layer), the whole system is completely determined by the connection weights and biases, which are both included in a single vector $\boldsymbol{w}$ in the following. Any type of neural network can be used to approximate the action-value function. In this study, two different network architectures have been explored, specifically Multilayer Perceptron (MLP) and Long-Short Term Memory (LSTM) networks.

An MLP is a fully connected feed-forward network with one or more hidden layers (see Fig. 5.1, subfigures (1a) and (1b)). The output of any neuron in any layer $\ell \geq 1$ is obtained by first computing a *weighted* sum of all the outputs from the previous layer, and then evaluating it through a non-linear activation function (the hyperbolic tangent, in this case). The final output vector, from the neurons in the last layer, is obtained through a cascade of operations of this type, by passing the output vector from any layer $\ell \geq 1$ to all neurons in the next layer $\ell + 1$. This network has no memory, and this means that given an input vector, the final output vector only depends on the network's weights. In this case, to keep track of some memory, such as the temporal evolution of video samples up to a certain number of instants (video segments) in the past, the input vector has to be extended to contain this information. This entails a redefinition of the system state (to account for *current* and *past* samples), which corresponds to a larger number of neurons and to a higher complexity (in terms of training time and memory space).

On the other hand, Recurrent Neural Networks (RNNs) implement some internal feedback mechanics that introduce memory, i.e., given an input vector, the network output depends on the network's weights *and* on the previous inputs. In this study, an RNN network based on LSTM cells [209] is implemented as sketched in Fig. 5.1(2). A schematic diagram of the LSTM

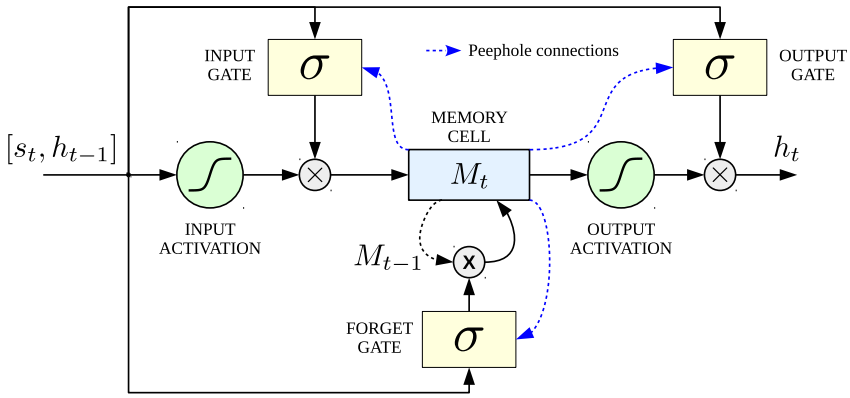


FIG. 5.2: Schematic diagram of an LSTM cell.

internal structure is shown in Fig. 5.2. The memory is implemented through a Memory Cell that allows storing or forgetting information about past network states. This is made possible by structures called gates that handle access to the Memory Cell. Gates are composed of a cascade of a network with sigmoidal activation function (σ) and a point-wise multiplication block. There are three gates in an LSTM cell: **1.** the *input gate*, that controls the new information that need to be stored in the Memory Cell, **2.** the *forget gate*, that manages the information to keep in the memory and what to forget, and **3.** the *output gate*, associated with the output of the cell (h_t). In addition, all the data that pass through a gate is reshaped by an activation function (usually an hyperbolic tangent). Optionally, peephole connections can be added to allow all gates inspect the current cell state, even when the output gate is closed [210]. Backpropagation Through Time (BTT) is usually used in conjunction with optimization methods to train RNNs [211].

5.5 Deep Q-learning for DASH adaptation

Conventional machine learning techniques are often limited in their ability to analyze data in their natural form. Usually, a good representation of the environment requires complex analysis and considerable expertise. This phase is commonly referred to as *feature engineering* and aims at finding a suitable representation of the raw data through a reduced set of features (*feature vector*), from which the learning system can extract useful environment information.

Representation learning consists of a set of mechanisms to automate this process: the learning machine is fed with raw data and discovers the best representation for detection or classification on its own. Deep-learning methods

are representation learning techniques which use multiple successive layers of artificial neurons; each layer is composed of simple (but non-linear) modules that transform the input representation into a slightly more abstract one, which is then used as the input for the next layer. With a complex enough deep structure, even very complex functions can be learned successfully. A great deal of work has been carried out on deep-learning architectures in the last decade [106, 212]. For further details, see, e.g., [213].

The presented D-DASH framework combines a Q-learning approach with a deep-learning framework to obtain optimal policies for the DASH protocol adaptation engine, as described in Sec. 5.3. Learning systems of this kind, referred to as *Deep Q-networks (DQNs)*, have been used in many complex systems in different research fields with state of the art performance, although their development is quite recent.

The main difference between standard Q-learning, as described in Sec. 5.3.2, and DQN, is in the way of estimating the Q-value of each state-action pair, generalized by the function $Q(s,q)$, i.e., an approximation of the optimal action-value function $Q^*(s,q)$. While standard Q-learning keeps a table of values and updates each state-value pair separately, DQN uses a deep-learning approach to approximate the Q-function. This can be achieved in two different ways:

1. a single deep network, fed with the current system state, is used to simultaneously estimate the Q-values for all possible actions;

2. one separate deep network is trained *for each* possible action, approximating a sub-space of the whole action-state set.

In this study the first approach has been utilized, as it has the advantage of providing the entire set of Q-values (always needed to make the final decision) with a single computation.

Considering the MDP defined in Sec. 5.3.1, a loss function $\tilde{L}$ at iteration t is evaluated using the following 4-tuple $e_t = (s_t, q_t, r_t, s_{t+1})$, which here is referred to as the *agent's experience* at time t. The loss function can be derived from the Bellman equation in Eq. (5.11) as follows [214]:

$$\tilde{L}_t(s_t, q_t, r_t, s_{t+1}|\boldsymbol{w}_t) = \left(r_t + \lambda \max_q \hat{Q}(s_{t+1}, q|\bar{\boldsymbol{w}}_t) - Q(s_t, q_t|\boldsymbol{w}_t)\right)^2, \quad (5.13)$$

where r_t is the reward accrued for segment t, for which the Eq. (5.4) is used. Note that the framework is rather general and any QoE function $S(q_t)$ can be plugged in without requiring any change to the model. Two deep neural networks, with the same architecture, are considered. A first network, with weight vector $\boldsymbol{w}_t$, is updated for each new segment (at every time step t), and is used to build the Q-value map $Q(s_t, q_t|\boldsymbol{w}_t)$. A second neural network, referred to as the *target network*, is accounted to increase the stability of the learning system [214], and its weight vector $\bar{\boldsymbol{w}}_t$ is updated every K steps (segments), by setting it equal to that of the first network and keeping it fixed for the next $K-1$ steps, i.e., $\bar{\boldsymbol{w}}_t = \boldsymbol{w}_t$ every K steps. The target network is used to retrieve the mapping $\hat{Q}(s_{t+1}, q|\bar{\boldsymbol{w}}_t)$ in Eq. (5.13). Another improvement is given by the implementation of a technique called *experience replay* [215]. The agent's experience $e_t = (s_t, q_t, r_t, s_{t+1})$ is stored into a *replay memory* $R = \{e_1, \ldots, e_t\}$ after each iteration. In this way, a new loss function L_t that also accounts for the past experience can be considered. Specifically, a subset $R_M = \{e_1, \ldots, e_M\}$ of M samples, with $e_j \in R$, $j = 1, 2, \ldots, M$, is extracted uniformly at random from the replay memory R, and L_t is finally evaluated as an empirical mean over the samples in set R_M:

$$L_t(\boldsymbol{w}_t) = \frac{1}{M} \sum_{e_j \in R_M} \tilde{L}_t(e_j|\boldsymbol{w}_t) \,. \tag{5.14}$$

This leads to three main advantages: greater data efficiency, uncorrelated subsequent training samples and independence between current policy and samples [214].

The whole process can be divided into two consecutive phases, which take a different but fixed number of iterations: namely, the *update phase*, and the *test phase*.

Update phase. The exploration parameter, namely ε in the case of an ε-greedy policy or ξ for softmax (see Sec. 5.3.2), is gradually reduced. It is recalled that a smaller value for this parameter means that the policy tends to prefer the action that is considered to be optimal at the current training stage. Furthermore, at each iteration the network's weights are updated to minimize the loss function in Eq. (5.14). The Adam method is used as the gradient descent optimization algorithm: it implements an adaptive learning rate to provide a faster and more robust convergence [208].

Test phase. The exploration parameter is set to zero, thus obtaining a *greedy*

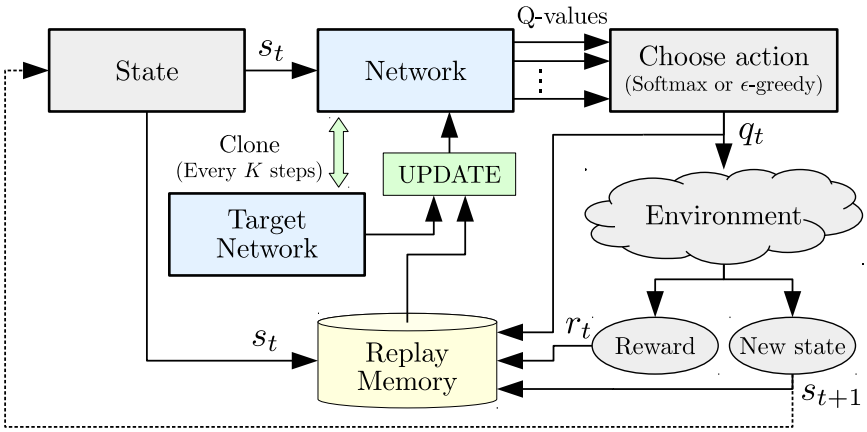


FIG. 5.3: Schematic diagram of an update iteration.

policy implementing the actions that are deemed optimal given the current system state and the mapping $Q(s_t, q_t|\boldsymbol{w}_t)$ from the first neural network. For this phase, the weights $\boldsymbol{w}_t$ are frozen and are no longer updated for the whole duration of the test. The target network is not used in the test phase and all the performance evaluations are based on the results obtained during this second phase.

A schematic diagram of an update iteration is shown in Fig. 5.3. First, the current environment state s_t is fed to the deep neural network, which outputs an estimate of the Q-value for each possible action $q \in A$, i.e., the various representations in the adaptation set A. Then, an action q_t is chosen according to either an ε-greedy or softmax policy. Upon taking action q_t, the system moves to the new state s_{t+1} and a new reward r_t is evaluated according to Eq. (5.4). The newly acquired experience $e_t = (s_t, q_t, r_t, s_{t+1})$ is stored into the replay memory R. Hence, a batch of M samples is extracted, uniformly at random, from the replay memory and is used to update the network's weights through the Adam optimization method. The loss function in Eq. (5.14) is minimized, using the mapping from the target network, i.e., $\hat{Q}(s_{t+1}, q|\bar{\boldsymbol{w}}_t)$, whose weights $\bar{\boldsymbol{w}}_t$ are updated every K steps.

Two different types of deep network architectures have been tested in this study (see Fig. 5.1): **1.** a fully connected feed-forward network, and in particular an MLP network with one (Figure *1a*) and two (Figure *1b*) hidden layers, respectively referred to as MLP_1 and MLP_2 in the following, and **2.** a recurrent neural network based on a LSTM cell [209]. An LSTM with deep hidden-to-output function has been implemented [216]. Accordingly, the LSTM output is processed by another fully connected layer to provide the final Q-values. It is remarked that the number of previous components

considered in the capacity vector $\boldsymbol{C}_t$ of the input state differs in the two cases: feed-forward networks (MLP_1 and MLP_2) require to be fed with the whole vector $\boldsymbol{C}_t$, whereas recurrent networks only need the last channel capacity sample C_{t-1} as input, as the feedback loop inside the LSTM cell keeps track of the channel memory. The dimension of vector $\boldsymbol{C}_t$ has to be fixed beforehand for the feed-forward networks, and cannot be adapted after the definition of the network. In this study, 2 history samples are considered for MLP_1 and MLP_2, and 5 history samples for a *"long history"* (lh) version of MLP_2, referred to as MLP_{lh}.

The LSTM has the advantage of automatically learning what to store inside the memory cell, and what to forget. Here, in addition to standard LSTM, an LSTM cell with peephole connections is also considered, referred to as LSTM_{ph}.

Due to the continuous changes of the target function and training data, no serious overfitting issues are expected. However, a dropout regularization technique has been applied to MLP_2 (termed MLP_{do}) to verify this assumption. Dropout is a well-known and simple method to prevent neural networks from overfitting [217]. It amounts to randomly dropping neurons and their connections, with a certain probability. A 20% dropout probability is considered in the training of MLP_{do}.

5.6 Simulation and results

In this work, segments belonging to the same representation set are assumed to have a constant size and variable quality; this is a common assumption in the relevant literature. In real systems, the encoding parameters of the video are often constant, with segments of varying size [218]; however, the performance of the learning algorithms should not be significantly affected by this factor, and the model presented in Sec. 5.2 is fully general.

To evaluate the performance of the proposed algorithms, extensive trace-based simulations have been carried out, where the different components of the system have been modeled in a realistic manner from real data traces. More specifically, the video model was derived from real videos from the EvalVid database.[1] The video traces were characterized in terms of their quality-rate function where, as previously explained, SSIM [183] is used as

[1] http://www2.tkn.tu-berlin.de/research/evalvid/cif.html

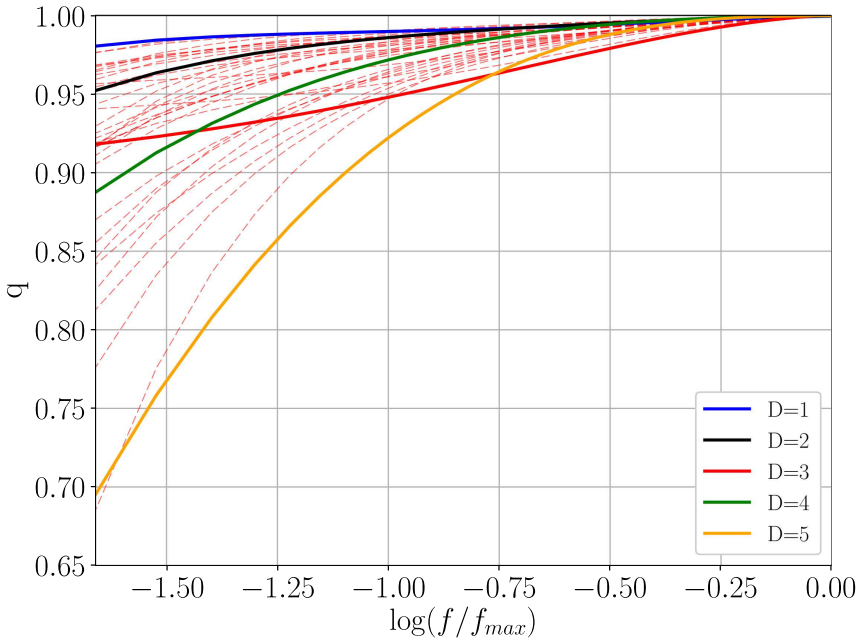


FIG. 5.4: Reference quality-rate curves.

the instantaneous video quality metric. Then, the SSIM-rate characteristics of a large number of videos in the dataset is derived, from which a limited number of reference SSIM-rate curves have been extracted. Following [219], such curves have been represented as 4th-degree polynomials function of the bitrate, so that the quality (SSIM) of a given video sequence encoded at rate $f/f_{max} \in (0,1]$ is given by

$$q \simeq d_0 + \sum_{k=1}^{4} d_k (\log(f/f_{\max}))^k, \tag{5.15}$$

where $f_{\max}$ is the full-quality video segment size, and $f \leq f_{\max}$ is the actual segment size. The vector $\mathbf{d} = [d_0, \ldots, d_4]$ offers then a synthetic representation of the complexity of a video scene. The chosen reference curves are shown in Fig. 5.4, and the values of each reference curve's vector $\mathbf{d}$ are listed in Table 5.1. These reference curves were combined into scenes with exponentially distributed duration, a model validated by Rose in [204], with an average of 10 seconds (i.e., 5 segments). In this way, a large number of realistic video traces for the tests can be generated. In the numerical results, 8 different values for the segment size f have been considered: if q is the segment quality, f is defined as $f = F_t(q)$, where $F_t(\cdot)$ is the inverse of Eq. (5.15), see also Table 5.2 for the considered adaptation set. In the learning system, each video segment is characterized by a vector $\mathbf{d}_t$, which is summarized by

PARAMETER	VALUES
T	2 s
$f_{\max}$	20 Mb
$F_t(\cdot)/T$	$\{0.25, 0.5, 1, 2, 3, 4, 6, 10\}$ Mb/s
C_t (pretrain)	$\{0.4, 0.75, 1.5, 2.5, 3.5, 4.5, 5.75, 7.25, 9, 12.5\}$ Mb/s
$D_t = 1$ (Akiyo)	$\mathbf{d}_t = [0.99947, -0.01015, -0.02888, -0.02427, 0.00415]$
$D_t = 2$ (News)	$\mathbf{d}_t = [0.99970, -0.01064, -0.02291, -0.02531, 0.00074]$
$D_t = 3$ (Bridge - far)	$\mathbf{d}_t = [1.00033, -0.01051, -0.05385, -0.08211, 0.01361]$
$D_t = 4$ (Harbor)	$\mathbf{d}_t = [0.99977, -0.00505, 0.00554, -0.01726, 0.00022]$
$D_t = 5$ (Husky)	$\mathbf{d}_t = [0.99984, 0.00998, 0.07590, -0.01138, 0.00040]$

TABLE 5.1: Simulation parameters

an index D_t, as shown in Table 5.2.

To model the transport capacity of the HTTP connection three datasets have been considered: a) *real* traces of HTTP video streaming sessions over LTE [220]; b) *synthetic* traces obtained through the network simulation platform `ns3`; c) *Markovian* traces obtained by means of a simple Markov model. The learning algorithms were first trained on the *Markovian* channel model (*pretrain* phase), then shown a small number of realistic traces (either *real* or *synthetic*) as an actual training (*train* phase). The neural networks' weights were then frozen and their exploration parameter was set to zero to carry out a fair comparison against state of the art algorithms from the literature using other traces from the same dataset (*test* phase). The *pretrain* phase lasted for 500 synthetic videos, while the *train* phase only lasted for 40 actual videos. The final *test* phase was performed over 100 actual video episodes; each video episode in all phases lasted 400 segments of $T = 2$ s each. The purpose of the *pretrain* phase is to teach the learning agent the basic mechanics of video streaming in a highly variable and realistic network scenario. The rationale is that the solution so obtained should represent a sensible starting point for the learning process when confronted with real traces, leading to a much shorter training time.

The Markov model for the *pretrain* phase used the same settings as in [200]; the capacity evolution was controlled by a random walk model among a set of levels, with a state change probability of 0.5. The possible capacity levels C_t are listed in Table 5.1. It is remarked that the Markovian traces were only used in the pre-training phase, while the performance evaluation was carried out by considering realistic traces, either from real HTTP measurements or generated by ns3.

Each component of the reward function in (5.4) was scaled into the range between 0 and 1 to avoid numerical convergence issues.

5.6.1 Algorithm settings

The algorithms from the literature selected as benchmarks were the parallel Q-learning and the simple rate-based heuristic from [200], MPC [191], which is a dynamic programming-like solution, and FESTIVE [188], which is among the most conservative heuristic-based algorithms, thus generally guaranteeing good stability and a low number of rebuffering events. One of FESTIVE's aims is providing a stable quality for multiple clients, which is beyond the scope of this work, so its performance should be evaluated accordingly. Another point that bears consideration is the computational complexity of MPC: the authors themselves state that a real-time implementation would require pre-computed tables, which leads to memory occupation. The PANDA [189] and QoE-RAHAS [190] algorithms have been also implemented, but their performance was significantly worse than all the other algorithms' for any configuration of their parameters in the simulation environment, so they are not show them in any of the plots.

The standard Q-learning algorithm used a Softmax policy; the other learning algorithms were tested with both policies, and the best-performing one was chosen in each case. The policy for each algorithm is listed in Table 5.2.

The hyperparameters of the learning algorithms were optimized by performing an exhaustive search and selecting the combinations that performed best in the *pretrain* phase; the values of the parameters for all algorithms are listed in Table 5.2, except for the exploration parameter (either ε or ξ, depending on the exploration mode), which followed the profile shown in Fig. 5.5 over the three phases. The settings and model for FESTIVE and the standard Q-learning algorithm are the same used in [188] and [200] respectively, except where otherwise stated. In Fig. 5.5, each video episode has $M = 400$ segments, the standard Q-learning algorithm uses preset thresholds to quantize the video quality and capacity measurements, which are listed in Table 5.2.

The MPC parameters have been adapted to reflect the definition of QoE and provide a fair comparison, as its basic model is very similar to ours and using the same QoE function provides a more meaningful comparison.

ALGORITHM	PARAMETER	VALUE
FESTIVE	target buffer	15 s
	buffer randomness	0.25 s
	α	10
	k_S (switch memory)	10
	k_C (throughput memory)	5
MPC	maximum buffer size	30 s
	γ	2
	μ	50
	K	5
All learners	β	2
	maximum buffer size	20 s
	γ	50
	δ	0.001
	B_{thr} (safe buffer)	10 s
Q-learning	α	0.1
	policy	Softmax
	q_t (SSIM)	$\{0.84, 0.87, 0.9, 0.92, 0.94, 0.96, 0.98, 0.99, 0.995\}$
	C_t (Mb/s)	$\{0.5, 1, 2, 3, 4, 5, 6, 8, 10\}$
MLP_1	hidden neurons N_h	256
	learning rate	10^{-3}
	batchsize M	1000
	K	20
	policy	Softmax
	λ	0.9
MLP_2	hidden neurons N_{h1}, N_{h2}	128, 256
MLP_{do}	learning rate	10^{-4}
MLP_{lh}	batchsize M	1000
	K	20
	policy	ε-greedy
	λ	0.9
LSTM	number of units N_c	128
LSTM_{ph}	learning rate	10^{-3}
	batchsize M	100
	K	200
	policy	Softmax
	λ	0.5

TABLE 5.2: Adaptation algorithm parameters

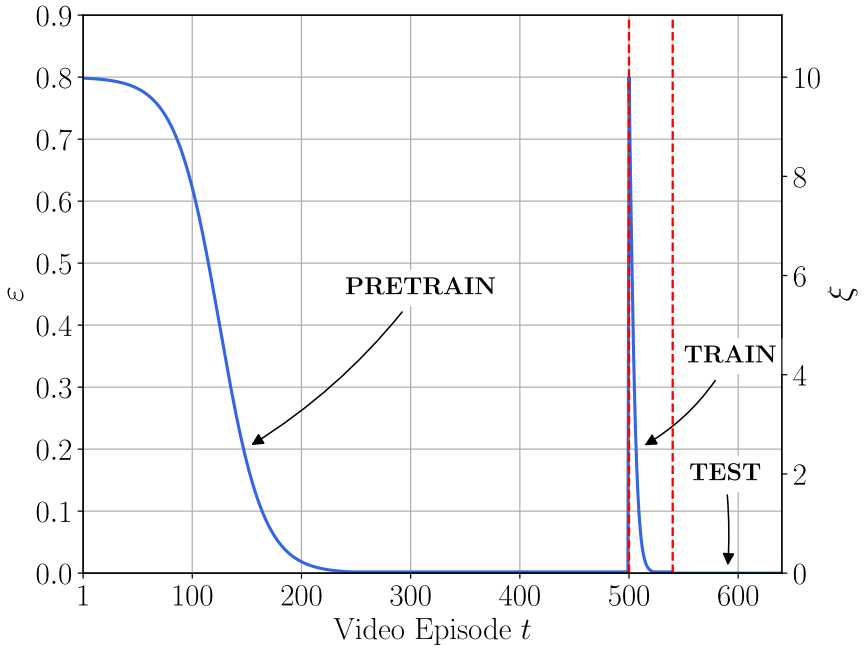


FIG. 5.5: Profile of the exploration rate during the training and testing phases.

5.6.2 Results: real traces

The simulations aimed to assess the performance of the D-DASH framework with regard to the three major video quality metrics, i.e., instantaneous quality, its stability and the frequency of rebuffering events. D-DASH based algorithms have been compared against existing approaches from the literature, while also investigating their convergence speed.

Fig. 5.6 shows a smoothed version of the reward over the three phases of the simulation for the MLP_1 and MLP_2, using standard Q-learning and the rate-based heuristic as a comparison. The tests were performed by freezing the learner state and setting a greedy policy (i.e., always taking the action with the highest expected reward) after each video episode. The benefits of the Deep-Q approach appear evident at a first glance; standard Q-learning gets a lower reward at convergence, as well as taking more than 200 episodes to overtake the rate-based heuristic and failing to adapt quickly enough to the real traces in the *train* phase.

The two D-DASH algorithms achieve higher rewards after just a few videos: MLP_2 converges in the first 50 videos, while MLP_1 already reaches its peak performance after the first video episode. The same trend can be seen in the *train* phase, where both schemes maintain a significant advantage on the simple heuristic and standard Q-learning.

One of the main advantages of smart QoE-aware adaptation schemes is

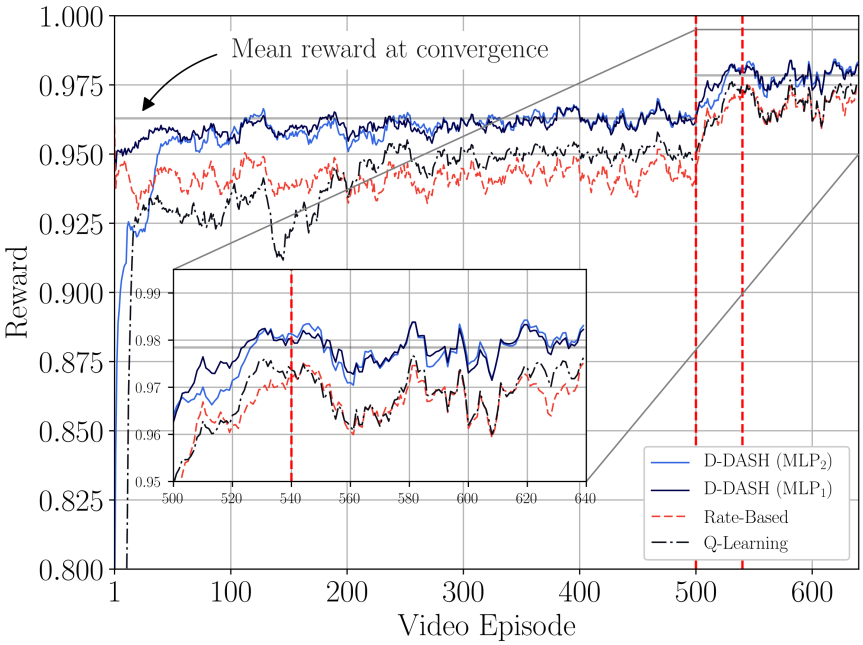


FIG. 5.6: Reward for the standard Q-learning, D-DASH and rate-based algorithm (using real traces as test set). The pretrain phase takes the first 500 video episodes, the training phase is enclosed within the two vertical lines and the test phase takes the remaining points, after the second line.

a better buffer management: while most classical algorithms try to keep the buffer as stable as possible, the D-DASH algorithms can use the buffered segments when the available capacity drops and build up the buffer when it is convenient to do so. Fig. 5.7 shows an example taken from the *test* phase: while FESTIVE keeps an extremely stable buffer (except for a sudden drop after about 200 segments, due to a sharp change in the capacity), the MLP_2 algorithm has a large dynamic range, building up the buffer when capacity is high and using it up to cover temporary outages. The other D-DASH algorithms make a similar use of the buffer, as well as standard Q-learning. MPC also maintains a high and stable quality throughout the video, but it incurs in several rebuffering events because of its optimistic throughput prediction, as Fig. 5.7 shows.

The benefits of the smarter buffer use are clearly visible in the quality graph of Fig. 5.8: MLP_2 can avoid sharp drops in the video quality without triggering rebuffering events.

In the next figures the performance of the different algorithms in the *test* phase are compared in terms of image quality (SSIM), rebuffering, and quality stability.

Fig. 5.9 shows the achieved SSIM: all the learning solutions, including

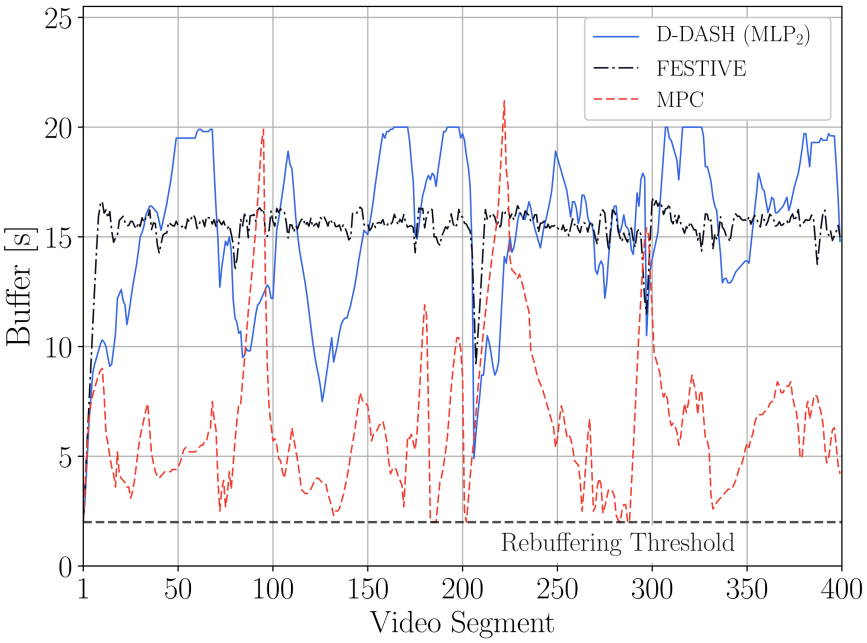


FIG. 5.7: Evolution of the buffer during a video episode in the *test* phase.

standard Q-learning, achieve a higher SSIM than FESTIVE and the rate-based heuristic; aside from the higher median, the bottom 5% of the videos still have an average SSIM of over 0.97, while FESTIVE and the rate-based heuristic go below that threshold for a significant fraction of the videos. It should be noted that FESTIVE has a lower average SSIM than the rate-based heuristic, since it privileges stability over instantaneous quality. MPC performs about as well as the learning-based algorithms, but with a slightly larger variance.

Fig. 5.10 shows the average difference between the SSIM of one segment and the next. As expected, FESTIVE keeps the quality more stable than the rate-based heuristic. Similar performance is obtained by standard Q-learning, which also keeps a higher average SSIM. The two Deep-Q algorithms outperform FESTIVE and Q-learning, but the best quality stability is achieved by MPC, with an average SSIM variation always lower than 0.006, while that of the other algorithms reaches 0.008 for at least one video.

Finally, Fig. 5.11 shows the frequency of rebuffering events for each algorithm: since FESTIVE is extremely conservative, there were no rebuffering events over the whole *test* phase. However, even the most aggressive learning algorithms (standard Q-learning and MLP_{do}) do not experience rebuffering events often. The LSTM and LSTM_{ph} algorithms experience at least one rebuffering in 25% of the videos, but they never have more than three,

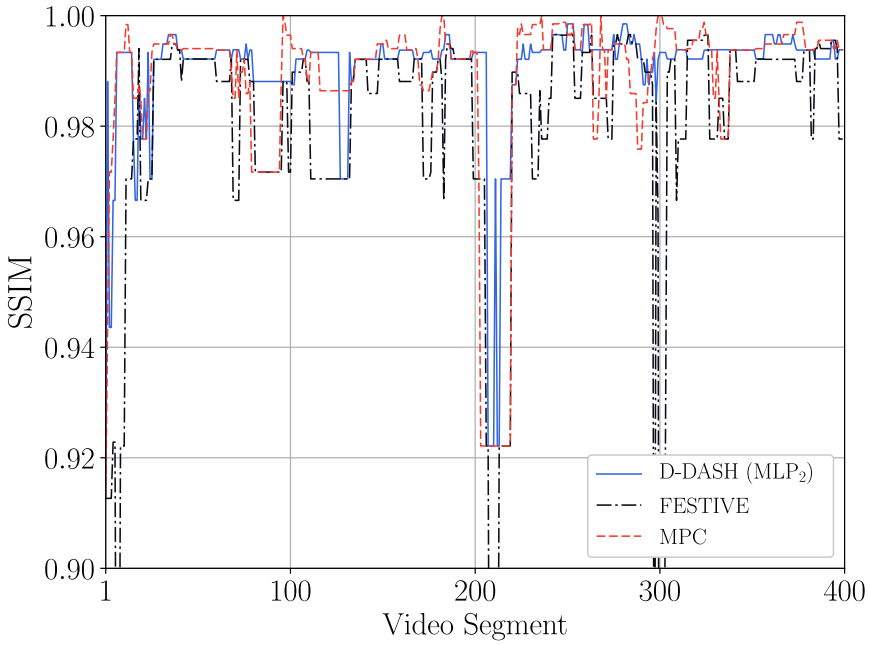


FIG. 5.8: Evolution of the SSIM during a video episode in the *test* phase.

and one rebuffering over a whole 400 segment video episode can be considered acceptable. MPC pays for its higher stability by having an average of 5 rebuffering events per video, far higher than any of the other algorithms'. MPC's reliance on an imperfect throughput predictor shows its limits when dealing with highly variable environments.

Another version of MLP_2has been included, called MLP_{lh}, which uses the last 5 throughput samples as input instead of the last 2. Its aim is to show the benefits of a longer memory in the presence of long-term correlation. It is observed that, with real channel capacity traces, the performance of MLP_{lh} are basically the same of LSTM, probably because the correlation of the empirical channel data considered in this study is low, so that the extra memory in the LSTM is not necessary. Finally, the use of dropout techniques (MLP_{do}) to avoid overfitting does not provide any improvement, as expected.

Its relative simplicity, low rebuffering rate and quick convergence arguably make MLP_1 the best adaptation algorithm in this scenario: aside from some very rare rebuffering events, it is an improvement over the state of the art in all the considered QoE metrics over real capacity traces. Nevertheless, as shown in the following section, any network with a limited memory shows its limitations when a more complicated scenario is analyzed, and the presence of long-term correlation in the channel capacity makes a more complex approach necessary.

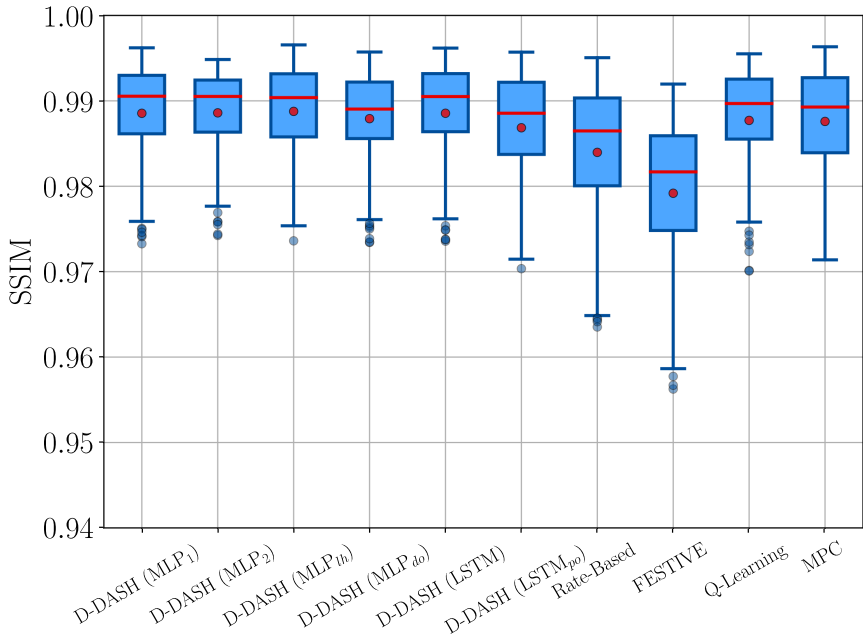


FIG. 5.9: Boxplot of the average SSIM for the 100 video episodes in the *test* phase (real traces).

5.6.3 Results: synthetic traces

After running the adaptation algorithms over real capacity traces, a set of traces with the *ns-3* simulator have been generated to gauge the effects of TCP cross-traffic on the algorithms' performance. The traces were generated by measuring the throughput of a saturated TCP flow sharing a bottleneck with a capacity of 10 Mb/s and a latency of 50 ms with 19 other TCP clients. Each of the cross-traffic clients generated TCP traffic as an on/off source: the off period was exponentially distributed with a mean of 2 seconds, while the on time was set to 4 seconds. This traffic model was designed to introduce long-term correlations in the available capacity [221], which are notoriously hard to handle for adaptation algorithms. For clarity, in this part of the study, only the MLP_2 (both with the short and long memory) and LSTM algorithms have been tested, since the performance of the other variants was similar.

Fig. 5.12 shows the average SSIM over the *test* phase using the synthetic traces. In this situation, the advantage of the D-DASH algorithms is more marked: the MLP_2 and LSTM are able to maintain an average SSIM above 0.98 for all the considered videos; standard Q-learning and MPC also perform better than the Rate-Based heuristic and FESTIVE.

Fig. 5.13 shows the net advantage of a long memory when dealing with long-term correlations: although the average SSIM of LSTM and MLP_{lh} is

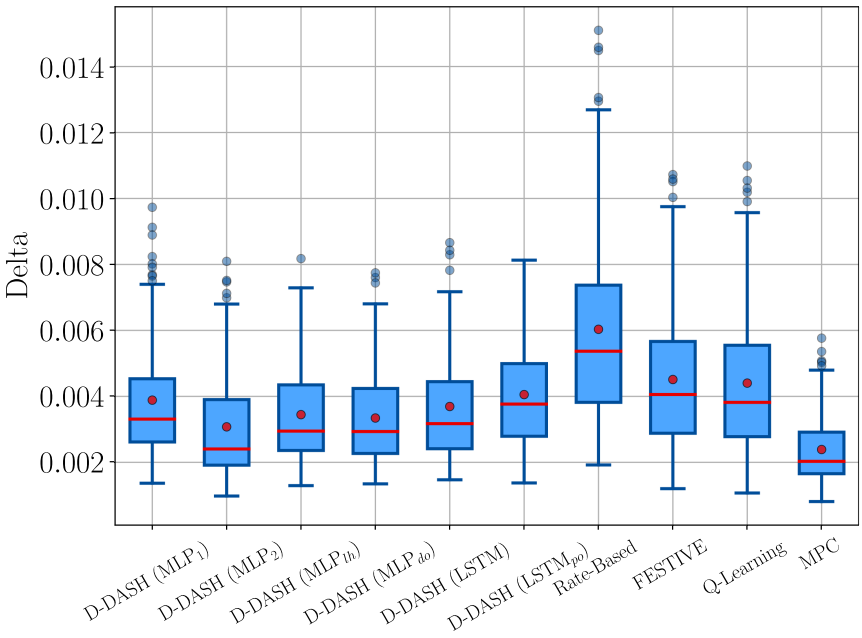


FIG. 5.10: Boxplot of the average SSIM variation for the 100 video episodes in the *test* phase (real traces).

the same as that of MLP_2 they can achieve it with half of the quality variation. Even if MLP_{lh} performs slightly better than LSTM, it considers a fixed amount of memory. The LSTM network has the additional capability to automatically adapt to different memory requirements, without increasing the state space dimension. The D-DASH algorithms and standard Q-learning all perform significantly better than FESTIVE and the Rate-Based heuristic on this metric. As with the real traces, MPC is the best at keeping a stable quality among the considered algorithms.

Fig. 5.14 shows the real advantage of the D-DASH framework in this scenario: while standard Q-learning has almost the same average SSIM as LSTM, MLP_{lh} and MLP_2, and actually has smaller quality fluctuations than the latter, it can only achieve this at the cost of relatively frequent rebufferings: a quarter of the video episodes have at least one rebuffering event, while this only happens for a few episodes with MLP_2 and never for LSTM. FESTIVE is also able to avoid rebuffering events, while the Rate-based heuristic is not. MPC still suffers from a high number of rebuffering events, totaling an average of 3 events per video episode.

Fig. 5.15 shows the higher stability of LSTM with respect to MLP_2: for the sake of a higher stability LSTM avoids some of the quality increases. For this complex channel, LSTM is able to predict more accurately when a certain quality increase is likely to be sustainable, i.e., without having to shortly

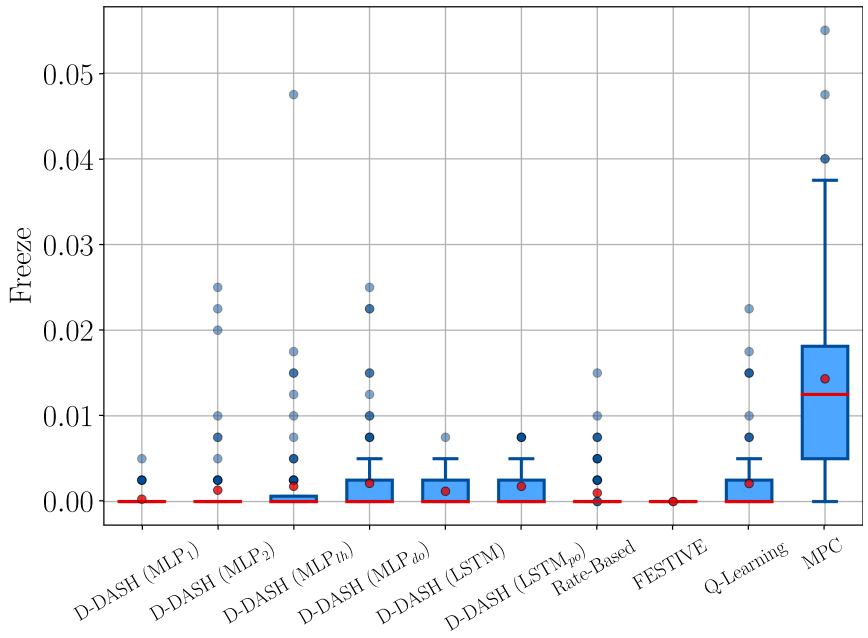


FIG. 5.11: Boxplot of the frequency of rebuffering events for the 100 video episodes in the *test* phase (real traces).

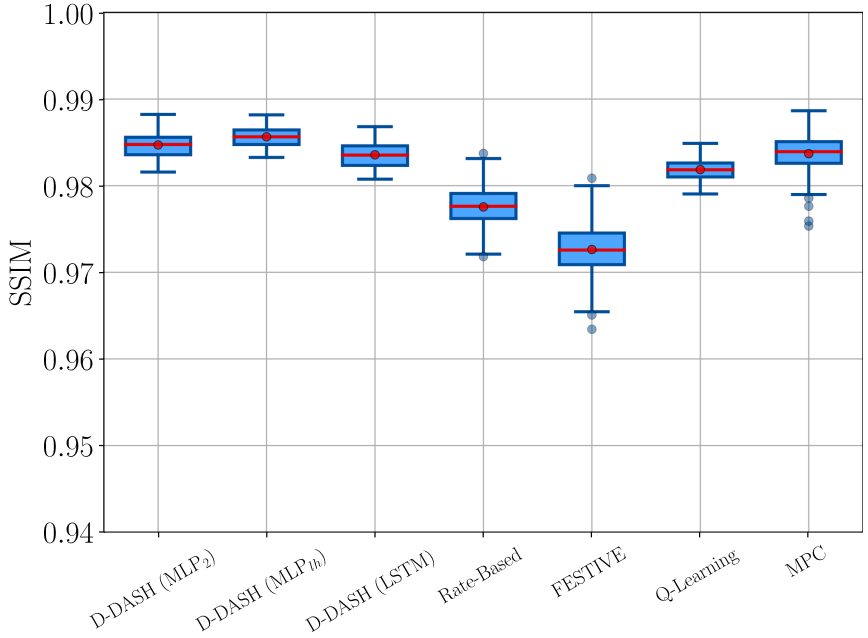


FIG. 5.12: Boxplot of the average SSIM for the 100 video episodes in the *test* phase (synthetic traces).

move back to the previous quality setting. Finally, a comparison of the convergence speed between standard Q-learning and D-DASH, performed during the pretrain phase, is provided in Fig. 5.16: D-DASH algorithms converge much faster, making a better use of the video examples that are supplied during the learning phase.

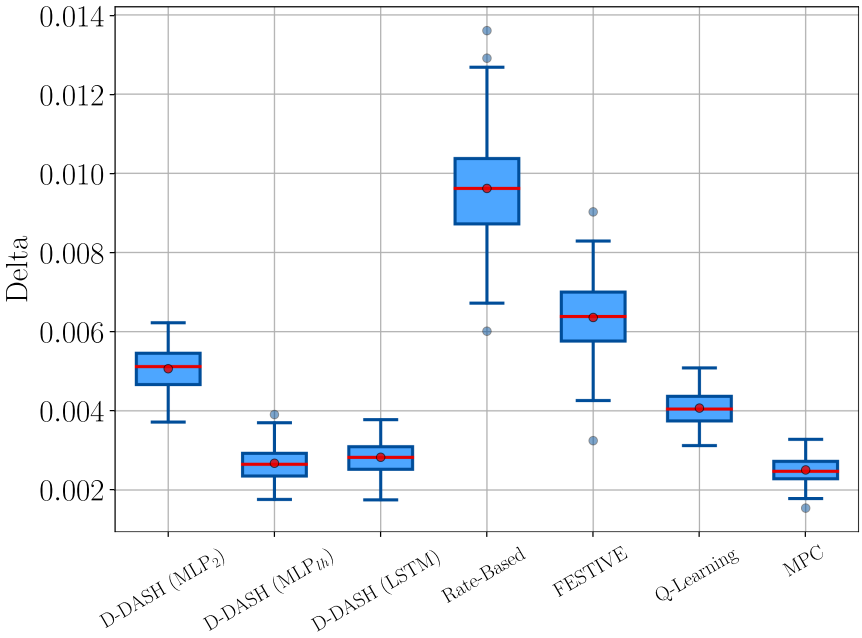


FIG. 5.13: Boxplot of the average SSIM variation for the 100 video episodes in the *test* phase (synthetic traces).

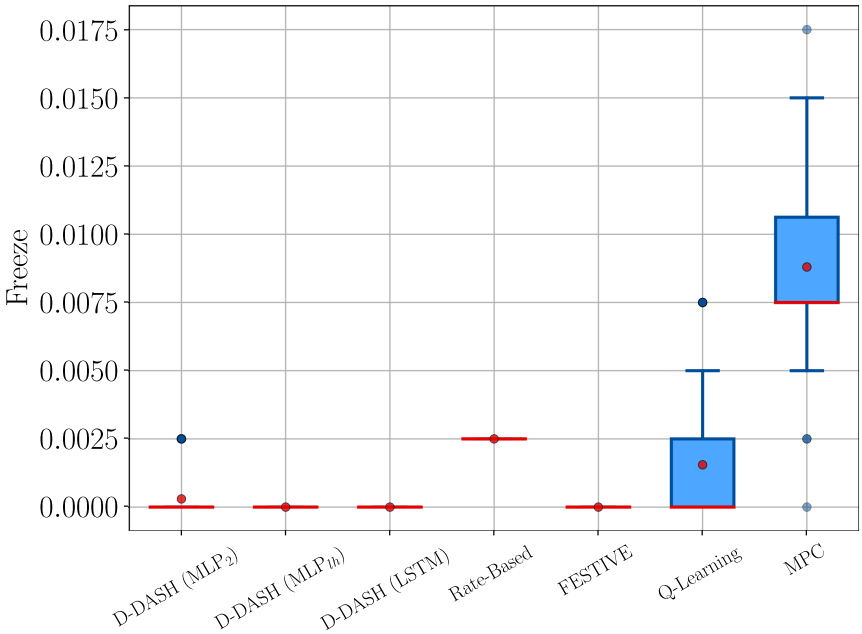


FIG. 5.14: Boxplot of the frequency of rebuffering events for the 100 video episodes in the *test* phase (synthetic traces).

5.6.4 Memory allocation

Since adaptation algorithms are client-side, their memory footprint is an important consideration. The number of variables required for D-DASH, and

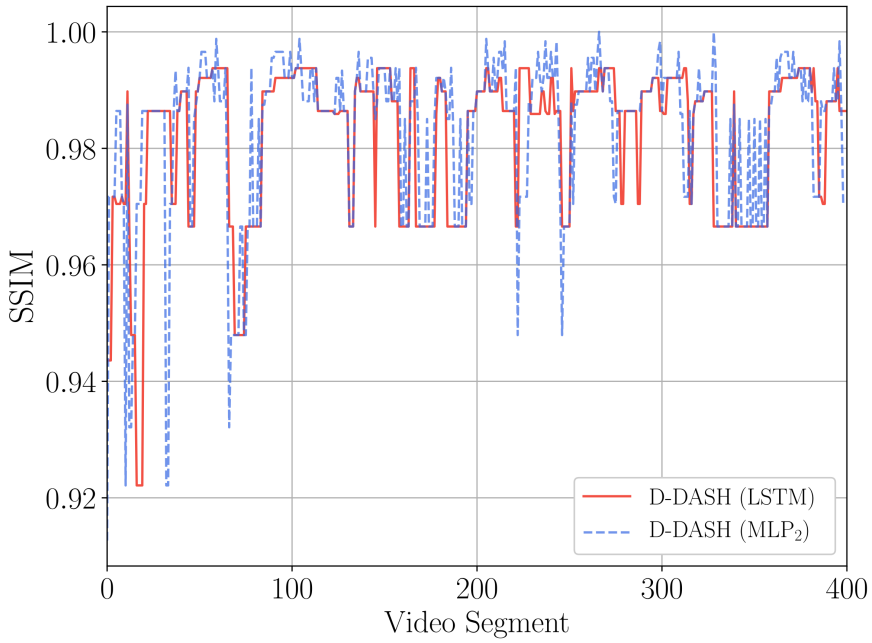


FIG. 5.15: Evolution of the SSIM during a video episode in the *test* phase, using the synthetic traces.

also for the classic Q-learning (N_Q), are as follows:

$$N_Q = N_s N_a \tag{5.16}$$

$$N_{MLP_1} = (V_s + 1)N_h + (N_h + 1)N_a \tag{5.17}$$

$$N_{MLP_2} = (V_s + 1)N_{h1} + (N_{h1} + 1)N_{h2} + (N_{h2} + 1)N_a \tag{5.18}$$

$$N_{LSTM} = 4(V_s + N_c + 1)N_c + (N_c + 1)N_a \tag{5.19}$$

where N_s is the cardinality of the state set, V_s is the number of state variables and N_a is the cardinality of the action set. Note that V_s and N_a also corresponds to the number of inputs and outputs of the neural networks, respectively (see also Fig. 5.1). N_h, N_{h1} and N_{h2} are the number of hidden layer's neurons in the MLP networks, and N_c are the number of units in the LSTM cell. According to Eq. (5.17), Eq. (5.18) and Eq. (5.19), considering the values in Table 5.2 and 32 bit floating-point representation for storing real variables, the memory space required for MLP_1, MLP_2 and LSTM is about 14.4 kB, 143.4 kB and 276.5 kB, respectively. It is remarked that these values are fixed given a specific network implementation, in the sense that they do not depend on the number of states. On the other hand, classic Q-learning space requirement directly depends on the cardinality of the state set, which is closely related to the quantization granularity of continuous state variables, as discussed in Section 5.3.3. The granularity that was used for the

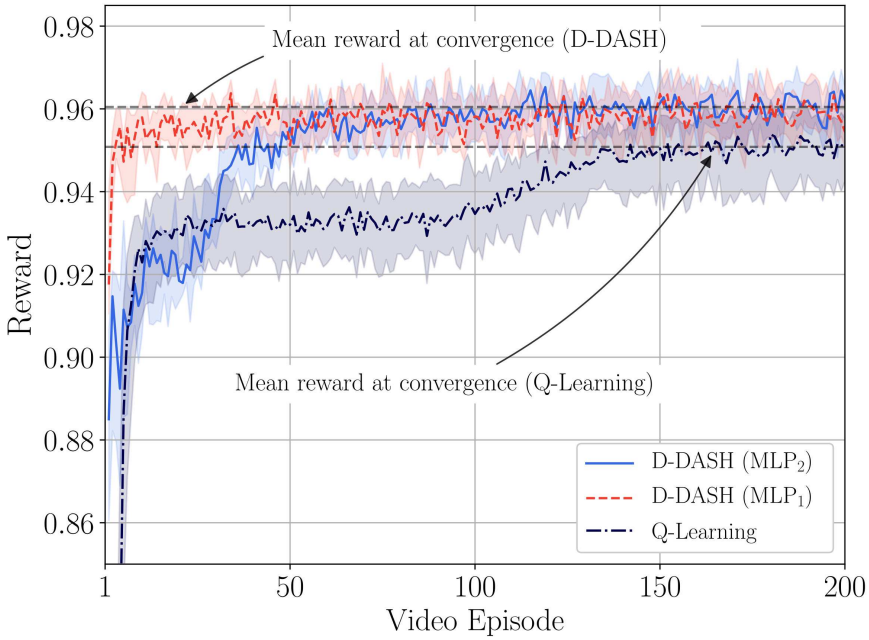


FIG. 5.16: Comparison of the convergence speed between standard Q-learning and D-DASH implementations (MLP_1 and MLP_2). The shaded area represents the interquartile range.

results in this study leads to a total memory space of 32 kB for Q-learning. The memory footprint needed by each of the presented methods appears reasonable considering the hardware of modern client devices. Note that, even if the number of variables for Q-learning is lower than those required by the D-DASH algorithms, their generalization capabilities and the concurrent update of the network's weights allow for a more efficient utilization of the experience acquired in the learning phase. This behavior can be seen in Fig. 5.16, where the convergence speed of D-DASH is considerably lower.

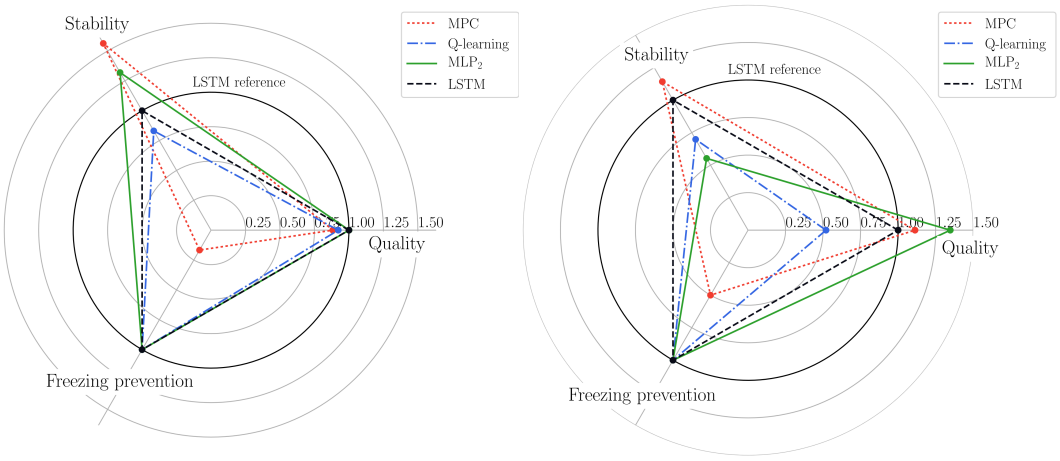
5.6.5 Summary of performance

A summary of the performance of video adaptation techniques is shown in Fig. 5.17. For a convenient visual comparison, the three metrics have been scaled and normalized with respect to the LSTM performance, according to the following criteria (the normalization term has been omitted for simplicity):

$$\text{Quality} = 0.98 - \text{SSIM} \tag{5.20}$$

$$\text{Stability} = 1 \,/\, \text{Quality Variation} \tag{5.21}$$

$$\text{Freezing Prevention} = 0.015 - \text{Frequency of rebuffering} \tag{5.22}$$



(A) real traces (B) synthetic traces

FIG. 5.17: Summary of performance of video adaptation algorithms: (a) real traces, (b) synthetic traces (exhibit long-term correlation).

The proposed deep-Q learning based schemes significantly outperform existing Q-learning and standard techniques from the literature. As Fig. 5.17a shows, with real traces MPC achieves a more stable, albeit slightly lower, quality than either MLP_2 or LSTM, but it fails to avoid rebuffering events and results in a far worse QoE for the user. Other state of the art algorithms, such as FESTIVE, which is not shown in the plot for readability, manage to avoid rebuffering events but perform much worse in the other two metrics. The D-DASH algorithms are the only ones reaching high scores on all the three considered metrics, i.e., video quality, stability and rebuffering avoidance. Furthermore, the D-DASH algorithms converge faster compared to standard Q-learning schemes, which require hundreds of video episodes to reach an acceptable performance. In fact, they approach optimal policies after just a few videos, or even just a couple in the case of the MLP_1 scheme. This also makes it possible to consider an online version that adapts to video and network conditions, learning how to deal with each new situation as it arises and memorizing it for future use. The longer memory of the LSTM algorithm proved to be very valuable on channel traces with long-term correlation: when the channel correlation stretches to over 10 seconds, LSTM shows significantly better performance than other schemes in all the three metrics, as Fig. 5.17b shows. Also in this case, MPC obtains a higher stability than LSTM, but it pays for it by having several rebuffering events per episode.

5.7 Discussion

In this work, several Reinforcement Learning-based DASH adaptation algorithms have been designed, exploiting Deep Q-networks to speed up the convergence and adaptability of the system, and improve its efficiency. The D-DASH framework uses different deep learning structures to approximate the Q-values, taking the raw system state as input and requiring no arbitrary design choices that might influence its performance. The deep learning algorithms also have low memory and computational requirements after an initial training phase (referred to as *pretraining*), and although the training may be computationally heavy for a mobile terminal, it can be performed offline (once for all) on a dedicated server and then passed on to the mobile device with minimal signaling overhead.

This work represents a significant improvement on the state of the art on Q-learning DASH adaptation designs, achieving good trade-offs between policy optimality and convergence speed. D-DASH performed better than several of the most popular adaptation approaches from the literature, maintaining a high video quality without paying a significant cost either in terms of rebuffering events or stability (i.e., avoiding sudden drops in the instantaneous quality). There are several ways in which this work could be extended and built upon: aside from changes and tweaks to include more complex video content models into the algorithm input, the main challenge we foresee for intelligent DASH adaptation is collaboration with network protocols to gain additional context information. Several works in the past few years [222–224] have tried to use cooperation between DASH clients and network elements to improve the fairness and efficiency of video streaming on a network scale.

Conclusion

This dissertation presents a novel effort to develop artificial intelligence techniques applied to real world scenarios, showing the potential benefits that they can bring over knowledge-based approaches. Among the same line of research, several application have been explored.

In Chapter 1, the experimental results show the superiority of a deep learning approach against state of the art techniques for human identification from inertial motion data acquired during a walking session. The proposed method leads to misclassification rates (either false negatives or positives) smaller than 0.15%.

Similar benefits can be also observed in Chapter 2, where several deep learning architectures have been explored to solve the problem of atrial fibrillation (AF) detection from short-length and noisy ECG traces. The architecture based on depthwise separable convolutions provides the best overall performance, performing either on par or considerably better than all the other DL-based solutions in terms of classification performance, memory/data efficiency and computational complexity (training time). The results indicate that the *automatic feature extraction* performed by deep learning has much better classification performance, both in terms of precision and recall, constituting a promising approach for the detection of AF and other heart rhythm disorders. It is not yet clear whether the automatic feature extraction implemented by deep networks is ready to replace the expert feature approach, which remains of primary importance to facilitate the human interpretation of the results. DL-based algorithms are nevertheless more effective in the presence of noisy signals from wireless sensors. The availability of large datasets, and the possibility of automatic labeling when the wearable is worn together with a clinical sensor, will allow the validation of these techniques for a future use inside and outside the clinic.

Moreover, the use of clinical sensors in everyday life is already spreading for the continuous monitoring of some clinical conditions, and the integration of advanced artificial intelligence techniques for improving their

diagnosis capabilities is fundamental. An important example is given by the glucose monitoring, analyzed in Chapter 3, where several machine learning algorithms have been compared for the prediction of future dangerous hypo/hyperglycemic events. The numerical results show that a static training approach exhibits better performance, in particular when regression methods are considered. However, classifiers show some improvement when trained for a specific event category, such as hyperglycemia, achieving performance comparable to the regressors, with the advantage of predicting the events sooner. Furthermore, a promising research avenue is the implementation of a combined approach, towards subject-adaptive and personalized algorithms. Having data covering longer time periods, we may define and train a static method, and then progressively tune it in a dynamic way, adapting it to a specific subject as new measurements become available.

This study also demonstrates the benefits of learning algorithms for vehicular traffic anomaly detection (Chapter 4) and for the optimization of video streaming user experience (Chapter 5). The proposed solution implements an intelligent adaptation engine for DASH video streaming clients, based on deep reinforcement learning strategies. The numerical results are obtained on real and simulated channel traces and show the superiority of this method against other state of the art approaches in nearly all the considered quality metrics. Besides yielding a considerably higher QoE, the framework exhibits faster convergence to the rate-selection strategy than the other learning algorithms considered in the study.

In conclusion, the representation learning capabilities of artificial intelligence techniques have been proved to be beneficial in several scenarios. A high impact on the society is expected in the near future, from every day user experiences such as transportation, security or entertainment, to more critical applications, like medicine. The widespread adoption of devices capable of acquiring large amount of data, among with the progress on promising data-driven learning techniques, allow for a deep integration of artificial intelligence with the potential to be transformational for the society.

Bibliography

[1] K. Amit, *Artificial intelligence and soft computing: behavioral and cognitive modeling of the human brain*. CRC-Press, 2000.

[2] S. Sprager and M. B. Juric, "Inertial sensor-based gait recognition: A review," *Sensors*, vol. 15, no. 9, p. 22089, 2015.

[3] D. Gafurov, "A survey of biometric gait recognition: Approaches, security and challenges," in *Annual Norwegian computer science conference*, Oslo, Norway, 2007.

[4] W. Zeng, C. Wang, and F. Yang, "Silhouette-based gait recognition via deterministic learning," *Pattern Recognition*, vol. 47, no. 11, pp. 3568–3584, 2014.

[5] J. Luo, J. Tang, T. Tjahjadi, and X. Xiao, "Robust arbitrary view gait recognition based on parametric 3D human body reconstruction and virtual posture synthesis," *Pattern Recognition*, vol. 60, pp. 361–377, 2016.

[6] X. Xing, K. Wang, T. Yan, and Z. Lv, "Complete canonical correlation analysis with application to multi-view gait recognition," *Pattern Recognition*, vol. 50, pp. 107–117, 2016.

[7] X. Chen and J. Xu, "Uncooperative gait recognition: Re-ranking based on sparse coding and multi-view hypergraph learning," *Pattern Recognition*, vol. 53, pp. 116–129, 2016.

[8] S. D. Choudhury and T. Tjahjadi, "Robust view-invariant multiscale gait recognition," *Pattern Recognition*, vol. 48, no. 3, pp. 798–811, 2015.

[9] M. Whittle, *Gait Analysis: An Introduction*. Edinburgh: Butterworth-Heinemann, 2007.

[10] H. Chan, H. Zheng, H. Wang, and R. Sterritt, "Evaluating and overcoming the challenges in utilizing smart mobile phones and standalone accelerometer for gait analysis," in *IET Irish Signals and Systems Conference*, Jun 2012, pp. 1–5.

[11] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Columbus, OH, US, Jun 2014.

[12] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2007.

[13] H. Thang, V. Viet, N. Thuc, and D. Choi, "Gait identification using accelerometer on mobile phone," in *International Conference on Control, Automation and Information Sciences*, Saigon, Vietnam, Nov 2012.

[14] C. Nickel, T. Wirtl, and C. Busch, "Authentication of smartphone users based on the way they walk using k-nn algorithm," in *International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, Piraeus-Athens, Greece, Jul 2012.

[15] Y. Watanabe, "Influence of holding smart phone for acceleration-based gait authentication," in *International Conference on Emerging Security Technologies*, Alcala de Henares, Spain, Sept 2014.

[16] S. Choi, I. Youn, R. LeMay, S. Burns, and J. Youn, "Biometric gait recognition based on wireless acceleration sensor using k-nearest neighbor classification," in *International Conference on Computing, Networking and Communications*, Honolulu, Hawaii, US, Feb 2014.

[17] Y. Ren, Y. Chen, M. Chuah, and J. Yang, "Smartphone based user verification leveraging gait recognition for mobile healthcare systems," in *IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, New Orleans, LA, US, Jun 2013.

[18] S. Sprager and M. B. Juric, "An Efficient HOS-Based Gait Authentication of Accelerometer Data," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 7, pp. 1486–1498, 2015.

[19] H. Chan, H. Zheng, H. Wang, R. Sterritt, and D. Newell, "Smart mobile phone based gait assessment of patients with low back pain," in *International Conference on Natural Computation*, Shenyang, China, Jul 2013.

[20] G. Huang, C. Wu, and J. Lin, "Gait analysis by using tri-axial accelerometer of smart phones," in *International Conference on Computerized Healthcare*, Dec 2012, pp. 29–34.

[21] C. Nickel, M. Derawi, P. Bours, and C. Busch, "Scenario test of accelerometer-based biometric gait recognition," in *International Workshop on Security and Communication Networks*, Gjøvik, Norway, May 2011.

[22] C. Nickel, C. Busch, S. Rangarajan, and M. Mobius, "Using hidden markov models for accelerometer-based biometric gait recognition," in *IEEE International Colloquium on Signal Processing and its Applications*, Penang, Malaysia, Mar 2011.

[23] T. Kobayashi, K. Hasida, and N. Otsu, "Rotation invariant feature extraction from 3-d acceleration signals," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2011, pp. 3684–3687.

[24] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[25] M. Gadaleta and M. Rossi, "Idnet: Smartphone-based gait recognition with convolutional neural networks," *Pattern Recognition*, vol. 74, pp. 25–37, 2018.

[26] M. Murray, A. Drought, and R. Kory, "Walking patterns of normal men," *The Journal of Bone & Joint Surgery*, vol. 46, no. 2, pp. 335–360, 1964.

[27] M. Murray, "Gait as a total pattern of movement: Including a bibliography on gait," *American Journal of Physical Medicine & Rehabilitation*, vol. 46, no. 1, pp. 290–333, 1967.

[28] T. Nixon, M.S. ans Tieniu and C. Rama, *Human identification based on gait*. Springer, 2006.

[29] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Cell phone-based biometric identification," in *Fourth IEEE International Conference on Biometrics: Theory Applications and Systems (BTAS)*, 2010.

[30] J. Mantyjarvi, M. Lindholm, E. Vildjiounaite, S. Makela, and H. Ailisto, "Identifying users of portable devices from gait pattern with accelerometers," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Oulu, Finland, Mar 2005.

[31] M. Derawi, C. Nickel, P. Bours, and C. Busch, "Unobtrusive user-authentication on mobile phones using biometric gait recognition," in *Intelligent Information Hiding and Multimedia Signal Processing*, Darmstadt, Germany, Oct 2010.

[32] E. Keogh and C. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and Information Systems*, vol. 7, no. 3, pp. 358–386, 2005.

[33] F. Juefei-Xu, C. Bhagavatula, A. Jaech, U. Prasad, and M. Savvides, "Gait-id on the move: Pace independent human identification using cell phone accelerometer dynamics," in *International Conference on Biometrics: Theory, Applications and Systems*, Washington DC, US, Sept 2012.

[34] S. Jiang, B. Zhang, G. Zou, and D. Wei, "The possibility of normal gait analysis based on a smart phone for healthcare," in *IEEE International Conference on Green Computing and Communications*, Aug 2013, pp. 2235–2240.

[35] Y. Zhong and Y. Deng, "Sensor orientation invariant mobile gait biometrics," in *IEEE International Joint Conference on Biometrics (IJCB)*, Clearwater, FL, USA, 2014.

[36] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale Video Classification with Convolutional Neural Networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Columbus, Ohio, US, Jun 2014.

[37] T. T. Ngo, Y. Makihara, H. Nagahara, Y. Mukaigawa, and Y. Yagi, "The largest inertial sensor-based gait database and performance evaluation of gait-based personal authentication," *Pattern Recognition*, vol. 47, no. 1, pp. 228–237, 2014.

[38] P. Casale, O. Pujol, and P. Radeva, "Personalization and user verification in wearable systems using biometric walking patterns," *Personal and Ubiquitous Computing*, vol. 16, no. 5, pp. 563–580, 2012.

[39] J. Tilmanne, R. Sebbe, and T. Dutoit, "A database for stylistic human gait modeling and synthesis," 2008.

[40] J. Frank, S. Mannor, and D. Precup, "Data sets: Mobile phone gait recognition data," 2010. [Online]. Available: http://www.cs.mcgill.ca/~jfrank8/data/gait-dataset.html

[41] P. D. Welch, "The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms," *IEEE Transactions on Audio and Electroacoustics*, vol. 15, no. 2, pp. 70–73, 1967.

[42] T. Teixeira, D. Jung, G. Dublon, and A. Savvides, "Pem-id: Identifying people by gait-matching using cameras and wearable accelerometers," in *ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)*, Como, Italy, Aug 2009.

[43] K. Kunze, P. Lukowicz, K. Partridge, and B. Begole, "Which Way Am I Facing: Inferring Horizontal Device Orientation from an Accelerometer Signal," in *IEEE International Symposium on Wearable Computers*, Linz, Austria, Sept 2009.

[44] Z.-A. Deng, G. Wang, Y. Hu, and D. Wu, "Heading Estimation for Indoor Pedestrian Navigation Using a Smartphone in the Pocket," *MDPI Sensors*, vol. 15, no. 9, pp. 21 518–21 536, 2015.

[45] 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.

[46] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Networks*. MIT Press, 1998, pp. 255–258.

[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[48] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *20th International Conference on Artificial Neural Networks (ICANN)*, Thessaloniki, Greece, 2010.

[49] R. Hanka and T. P. Harte, *Computer Intensive Methods in Control and Signal Processing: The Curse of Dimensionality*. Birkhäuser Boston, 1997, ch. Curse of Dimensionality: Classifying Large Multi-Dimensional Images with Neural Networks, pp. 249–260.

[50] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., 1993.

[51] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine Learning*, vol. 29, no. 2, pp. 131–163, Feb 1997.

[52] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, Jan 1967.

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

[54] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, J. C. Platt *et al.*, "Support vector method for novelty detection," *Neural Information Processing Systems (NIPS)*, vol. 12, pp. 582–588, 1999.

[55] D. R. Musicant, V. Kumar, and A. Ozgur, "Optimizing F-Measure with Support Vector Machines," in *16-th International FLAIRS Conference*. St. Augustine, Florida, US: FLAIRS, May 2003.

[56] D. Tax and K. Müller, *Artificial Neural Networks and Neural Information Processing*. Berlin, Heidelberg: Springer, 2003, ch. Feature Extraction for One-Class Classification, pp. 342–349. [Online]. Available: http://dx.doi.org/10.1007/3-540-44989-2_41

[57] A. Wald, *Sequential analysis*. New York, NY, US: Dover, 1947.

[58] A. Tartakovsky, I. Nikiforov, and M. Basseville, *Sequential Analysis Hypothesis Testing and Changepoint Detection*. CRC Press, 2015.

[59] G. Quer, E. D. Muse, N. Nikzad, E. J. Topol, and S. R. Steinhubl, "Augmenting diagnostic vision with AI," *Lancet*, vol. 390, no. 10091, p. 221, 2017.

[60] V. Gulshan, L. Peng, M. Coram, M. C. Stumpe, D. Wu, A. Narayanaswamy, S. Venugopalan, K. Widner, T. Madams, J. Cuadros, R. Kim, R. Raman, P. C. Nelson, J. L. Mega, and D. R. Webster, "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs," *JAMA*, vol. 316, no. 22, pp. 2402–2410, 2016.

[61] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, pp. 115–118, 2017.

[62] S. S. Chugh, R. Havmoeller, K. Narayanan, D. Singh, M. Rienstra, E. J. Benjamin, R. F. Gillum, Y.-H. Kim, J. H. McAnulty, Z.-J. Zheng, M. H. Forouzanfar, M. Naghavi, G. A. Mensah, M. Ezzati, and C. J. L. Murray, "Worldwide epidemiology of atrial fibrillation," *Circulation*, vol. 129, no. 8, pp. 837–847, 2014.

[63] A. B. Freedman, "Worldwide epidemiology of atrial fibrillation," *Circulation*, vol. 135, no. 19, pp. 1851–1867, 2017.

[64] G. D. Clifford, C. Liu, B. Moody, L.-w. H. Lehman, I. Silva, Q. Li, E. Johnson, and R. G. Mark, "AF classification from a short single lead ECG recording: the PhysioNet/Computing in cardiology challenge 2017," *Computing in Cardiology*, vol. 44, pp. 65–69, 2017.

[65] T. Teijeiro, C. A. García, D. Castro, and P. Félix, "Arrhythmia classification from the abductive interpretation of short single-lead ECG records," in *IEEE Computing in Cardiology (CinC), 2017*, vol. 44, 2017, pp. 1–4.

[66] S. Hong, M. Wu, Y. Zhou, Q. Wang, J. Shang, H. Li, and J. Xie, "Encase: An ensemble classifier for ECG classification using expert features and deep neural networks," in *IEEE Computing in Cardiology (CinC), 2017*, 2017, pp. 1–4.

[67] S. Datta, C. Puri, A. Mukherjee, R. Banerjee, A. D. Choudhury, R. Singh, A. Ukil, S. Bandyopadhyay, A. Pal, and S. Khandelwal, "Identifying normal, af and other abnormal ECG rhythms using a cascaded binary classifier," in *IEEE Computing in Cardiology (CinC), 2017*, vol. 44, 2017, pp. 1–4.

[68] M. Zabihi, A. B. Rad, A. K. Katsaggelos, S. Kiranyaz, S. Narkilahti, and M. Gabbouj, "Detection of atrial fibrillation in ECG hand-held devices using a random forest classifier," in *IEEE Computing in Cardiology (CinC),*

2017, vol. 44, 2017, pp. 1–4.

[69] M. Zihlmann, D. Perekrestenko, and M. Tschannen, "Convolutional recurrent neural networks for electrocardiogram classification," in *IEEE Computing in Cardiology (CinC), 2017*, vol. 44, 2017, pp. 1–4.

[70] Z. Xiong, M. K. Stiles, and J. Zhao, "Robust ECG signal classification for detection of atrial fibrillation using a novel neural network," in *IEEE Computing in Cardiology (CinC), 2017*, vol. 44, 2017, pp. 1–4.

[71] P. De Chazal, M. O'Dwyer, and R. B. Reilly, "Automatic classification of heartbeats using ECG morphology and heartbeat interval features," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 7, pp. 1196–1206, 2004.

[72] M. Korürek and B. Doğan, "ECG beat classification using particle swarm optimization and radial basis function neural network," *Expert Systems with Applications*, vol. 37, no. 12, pp. 7563–7569, 2010.

[73] İ. Güler and E. D. Übeylı, "ECG beat classifier designed by combined neural network model," *Pattern Recognition*, vol. 38, no. 2, pp. 199–208, 2005.

[74] Y. Kutlu and D. Kuntalp, "Feature extraction for ECG heartbeats using higher order statistics of WPD coefficients," *Computer Methods and Programs in Biomedicine*, vol. 105, no. 3, pp. 257–267, 2012.

[75] C.-H. Lin, Y.-C. Du, and T. Chen, "Adaptive wavelet network for multiple cardiac arrhythmias recognition," *Expert Systems with Applications*, vol. 34, no. 4, pp. 2601–2611, 2008.

[76] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[77] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[78] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, Inception-Resnet and the impact of residual connections on learning." in *Proceedings of AAAI Conference on Artificial Intelligence*, vol. 4, 2017, p. 12.

[79] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[80] M. S. Chavan, R. Agarwala, and M. Uplane, "Suppression of baseline wander and power line interference in ECG using digital IIR filter," *International Journal of Circuits, Systems and Signal Processing*, vol. 2, no. 2, pp. 356–365, 2008.

[81] C. Meyer and H. Keiser, "Electrocardiogram baseline noise estimation and removal using cubic splines and state-space computation techniques," *Computers and Biomedical Research*, vol. 10, no. 5, pp. 459–470, 1977.

[82] P. De Chazal, C. Heneghan, E. Sheridan, R. Reilly, P. Nolan, and M. O'Malley, "Automated processing of the single-lead electrocardiogram for the detection of obstructive sleep apnoea," *IEEE Trans. Biomed. Eng.*, vol. 50, no. 6, pp. 686–696, 2003.

[83] A. Khawaja, *Automatic ECG analysis using principal component analysis and wavelet transformation*. Univ.-Verlag Karlsruhe, 2007.

[84] G. Lenis, N. Pilia, A. Loewe, W. H. Schulze, and O. Dössel, "Comparison of baseline wander removal techniques considering the preservation of st changes in the ischemic ECG: A simulation study," *Computational and Mathematical Methods in Medicine*, 2017.

[85] M. Elgendi, "Fast QRS detection with an optimized knowledge-based method: Evaluation on 11 standard ECG databases," *PloS one*, vol. 8, no. 9, p. e73557, 2013.

[86] M. Gadaleta and A. Giorgio, "A method for ventricular late potentials detection using time-frequency representation and wavelet denoising," *ISRN Cardiology*, vol. 2012, 2012.

[87] J. P. Martínez, R. Almeida, S. Olmos, A. P. Rocha, and P. Laguna, "A wavelet-based ECG delineator: evaluation on standard databases," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 4, pp. 570–581, 2004.

[88] F. Censi, I. Corazza, E. Reggiani, G. Calcagnini, E. Mattei, M. Triventi, and G. Boriani, "P-wave variability and atrial fibrillation," *Scientific Reports*, vol. 6, p. 26799, 2016.

[89] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2010, pp. 807–814.

[90] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[91] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[92] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[93] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[94] P. Rajpurkar, A. Y. Hannun, M. Haghpanahi, C. Bourn, and A. Y. Ng, "Cardiologist-level arrhythmia detection with convolutional neural networks," *arXiv preprint arXiv:1707.01836*, 2017.

[95] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[96] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$," in *Doklady AN USSR*, vol. 269, 1983, pp. 543–547.

[97] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of International Conference on Machine Learning*, 2015, pp. 448–456.

[98] N. V. Thakor and Y.-S. Zhu, "Applications of adaptive filtering to ECG analysis: noise cancellation and arrhythmia detection," *IEEE Trans. Biomed. Eng.*, vol. 38, no. 8, pp. 785–794, 1991.

[99] S. G. Guillén, M. T. Arredondo, G. Martin, and J. M. F. Corral, "Ventricular fibrillation detection by autocorrelation function peak analysis," *Journal of Electrocardiology*, vol. 22, pp. 253–262, 1990.

[100] N. V. Thakor, A. Natarajan, and G. F. Tomaselli, "Multiway sequential hypothesis testing for tachyarrhythmia discrimination," *IEEE Trans. Biomed. Eng.*, vol. 41, no. 5, pp. 480–487, 1994.

[101] B. S. Raghavendra, D. Bera, A. S. Bopardikar, and R. Narayanan, "Cardiac arrhythmia detection using dynamic time warping of ECG beats in e-healthcare systems," in *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2011, pp. 1–6.

[102] P. Laguna, R. Jané, and P. Caminal, "Automatic detection of wave boundaries in multilead ECG signals: Validation with the CSE database," *Computers and Biomedical Research*, vol. 27, no. 1, pp. 45–60, 1994.

[103] B. Celler and P. de Chazal, "Selection of parameters from power spectral density, wavelet transforms and other methods for the automated interpretation of the ECG," in *Proceedings of IEEE Digital Signal Processing Conference (DSP)*, vol. 1. IEEE, 1997, pp. 71–74.

[104] C. Ye, M. T. Coimbra, and B. V. Kumar, "Arrhythmia detection and classification using morphological and dynamic features of ECG signals," in *Proceedings of IEEE Engineering in Medicine and Biology Society Conference (EMBC)*. IEEE, 2010, pp. 1918–1921.

[105] G. H. Tison, J. M. Sanchez, B. Ballinger, A. Singh, J. E. Olgin, M. J. Pletcher, E. Vittinghoff, E. S. Lee, S. M. Fan, R. A. Gladstone, C. Mikell, N. Sohoni, J. Hsieh, and G. M. Marcus, "Passive detection of atrial fibrillation using a commercially available smartwatch," *JAMA Cardiology*, vol. 3, no. 5, pp. 409–416, 2018.

[106] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[107] R. Taylor, "Type 2 diabetes: etiology and reversibility," *Diabetes care*, vol. 36, no. 4, pp. 1047–1055, Apr 2013.

[108] C. Hayes and A. Kriska, "Role of physical activity in diabetes management and prevention," *Journal of the American Dietetic Association*, vol. 108, no. 4, pp. S19–S23, Apr 2008.

[109] S. H. Ley, O. Hamdy, V. Mohan, and F. B. Hu, "Prevention and management of type 2 diabetes: dietary components and nutritional strategies," *The Lancet*, vol. 383, no. 9933, pp. 1999–2007, Jun 2014.

[110] D. Rodbard, "Continuous glucose monitoring: a review of recent studies demonstrating improved glycemic outcomes," *Diabetes Technology & Therapeutics*, vol. 19, no. 3, pp. 25–37, Jun 2017.

[111] A. Facchinetti, "Continuous glucose monitoring sensors: past, present and future algorithmic challenges," *Sensors*, vol. 16, no. 12, p. 2093, Dec 2016.

[112] C. Zecchin, A. Facchinetti, G. Sparacino, and C. Cobelli, "Reduction of number and duration of hypoglycemic events by glucose prediction methods: a proof-of-concept in silico study," *Diabetes technology & therapeutics*, vol. 15, no. 1, pp. 66–77, Jan 2013.

[113] B. Buckingham, H. P. Chase, E. Dassau, E. Cobry, P. Clinton, V. Gage, K. Caswell, J. Wilkinson, F. Cameron, H. Lee, B. W. Bequette, and F. J. Doyle, "Prevention of nocturnal hypoglycemia using predictive alarm algorithms and insulin pump suspension," *Diabetes Care*, vol. 33, no. 5, pp. 1013–1017, May 2010.

[114] A. Zhong, P. Choudhary, C. McMahon, P. Agrawal, J. B. Welsh, T. L. Cordero, and F. R. Kaufman, "Effectiveness of automated insulin management features of the MiniMed® 640G sensor-augmented insulin pump," *Diabetes technology & therapeutics*, vol. 18, no. 10, pp. 657–663, Oct 2016.

[115] G. Sparacino, F. Zanderigo, S. Corazza, A. Maran, A. Facchinetti, and C. Cobelli, "Glucose concentration can be predicted ahead in time from continuous glucose monitoring sensor time-series," *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 5, pp. 931–937, May 2007.

[116] A. Gani, A. V. Gribok, S. Rajaraman, W. K. Ward, and J. Reifman, "Predicting subcutaneous glucose concentration in humans: data-driven glucose modeling," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 2,

pp. 246–254, Feb 2009.

[117] M. Eren-Oruklu, A. Cinar, L. Quinn, and D. Smith, "Estimation of future glucose concentrations with subject-specific recursive linear models," *Diabetes Technology and Therapeutics*, vol. 11, no. 4, pp. 243–253, Apr 2009.

[118] C. Perez-Gandia, A. Facchinetti, G. Sparacino, C. Cobelli, E. J. Gomez, M. Rigla, A. de Leiva, and M. E. Hernando, "Artificial neural network algorithm for online glucose prediction from continuous glucose monitoring," *Diabetes Technology and Therapeutics*, vol. 12, no. 1, pp. 81–88, Jan 2010.

[119] V. Naumova, S. V. Pereverzyev, and S. Sivananthan, "A meta-learning approach to the regularized learning-case study: blood glucose prediction," *Neural Networks*, vol. 33, pp. 181–193, Sep 2012.

[120] D. A. Finan, F. J. Doyle, C. C. Palerm, W. C. Bevier, H. C. Zisser, L. Jovanovic, and D. E. Seborg, "Experimental evaluation of a recursive model identification technique for type 1 diabetes," *Journal of Diabetes Science and Technology*, vol. 3, no. 5, pp. 1192–1202, Sep 2009.

[121] M. Eren-Oruklu, A. Cinar, D. K. Rollins, and L. Quinn, "Adaptive system identification for estimating future glucose concentrations and hypoglycemia alarms," *Automatica*, vol. 48, no. 8, pp. 1892–1897, Aug 2012.

[122] K. Turksoy, E. S. Bayrak, L. Quinn, E. Littlejohn, D. Rollins, and A. Cinar, "Hypoglycemia early alarm systems based on multivariable models," *Industrial & engineering chemistry research*, vol. 52, no. 35, pp. 12 329–12 336, Sep 2013.

[123] E. I. Georga, V. C. Protopappas, D. Polyzos, and D. I. Fotiadis, "A predictive model of subcutaneous glucose concentration in type 1 diabetes based on random forests," in *34th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, San Diego, CA, USA, Aug 2012.

[124] E. I. Georga, V. C. Protopappas, D. Ardigo, M. Marina, I. Zavaroni, D. Polyzos, and D. I. Fotiadis, "Multivariate prediction of subcutaneous glucose concentration in type 1 diabetes patients based on support vector regression," *IEEE Journal of Biomedical and Health Informatics*, vol. 17, no. 1, pp. 71–81, Jan 2013.

[125] E. I. Georga, V. C. Protopappas, D. Polyzos, and D. I. Fotiadis, "Evaluation of short-term predictors of glucose concentration in type 1 diabetes combining feature ranking with regression models," *Medical and Biological Engineering and Computing*, vol. 53, no. 12, pp. 1305–1318, Dec 2015.

[126] M. Cescon, R. Johansson, and E. Renard, "Subspace-based linear multi-step predictors in type 1 diabetes mellitus," *Biomedical Signal Processing and Control*, vol. 22, pp. 99–110, Sep 2015.

[127] C. Zecchin, A. Facchinetti, G. Sparacino, G. De Nicolao, and C. Cobelli, "Neural network incorporating meal information improves accuracy of short-time prediction of glucose concentration," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 6, pp. 1550–1560, Jun 2012.

[128] C. Zecchin, A. Facchinetti, G. Sparacino, and C. Cobelli, "Jump neural network for online short-time prediction of blood glucose from continuous monitoring sensors and meal information," *Computer Methods and Programs in Biomedicine*, vol. 113, no. 1, pp. 144–152, Jan 2014.

[129] C. Zecchin, A. Facchinetti, G. Sparacino, and C. Cobelli, "How much is short-term glucose prediction in type 1 diabetes improved by adding insulin delivery and meal content information to CGM data? A proof-of-concept study," *Journal of diabetes science and technology*, vol. 10, no. 5, pp. 1149–1160, Sep 2016.

[130] K. Zarkogianni, K. Mitsis, E. Litsa, M.-T. Arredondo, G. Fico, A. Fioravanti, and K. S. Nikita, "Comparative assessment of glucose prediction models for patients with type 1 diabetes mellitus applying sensors for glucose and physical activity monitoring," *Medical & biological engineering & computing*, vol. 53, no. 12, pp. 1333–1343, Dec 2015.

[131] E. Daskalaki, A. Prountzou, P. Diem, and S. G. Mougiakakou, "Real-time adaptive models for the personalized prediction of glycemic profile in type 1 diabetes patients," *Diabetes technology & therapeutics*, vol. 14, no. 2, pp. 168–174, Feb 2012.

[132] P. Tkachenko, G. Kriukova, M. Aleksandrova, O. Chertov, E. Renard, and S. V. Pereverzyev, "Prediction of nocturnal hypoglycemia by an aggregation of previously known prediction approaches: proof of concept for clinical application," *Computer methods and programs in biomedicine*, vol. 134, pp. 179–186, Oct 2016.

[133] I. Contreras, S. Oviedo, M. Vettoretti, R. Visentin, and J. Vehí, "Personalized blood glucose prediction: A hybrid approach using grammatical evolution and physiological models," *PloS one*, vol. 12, no. 11, pp. 1–16, Nov 2017.

[134] R. M. Bergenstal, S. Garg, S. A. Weinzimer, B. A. Buckingham, B. W. Bode, W. V. Tamborlane, and F. R. Kaufman, "Safety of a hybrid closed-loop insulin delivery system in patients with type 1 diabetes," *Jama*, vol. 316, no. 13, pp. 1407–1408, 2016.

[135] T. Koutny, "Modelling of glucose dynamics for diabetes," in *International Conference on Bioinformatics and Biomedical Engineering*, Seoul, South Korea, Nov 2017.

[136] M. Gadaleta, A. Facchinetti, E. Grisan, and M. Rossi, "Prediction of adverse glycemic events from continuous glucose monitoring signal," *IEEE Journal of Biomedical and Health Informatics*, 2018.

[137] G. E. Box and G. C. Tiao, *Bayesian inference in statistical analysis*. John Wiley & Sons, 2011, vol. 40.

[138] D. Basak, S. Pal, and D. C. Patranabis, "Support vector regression," *Neural Information Processing-Letters and Reviews*, vol. 11, no. 10, pp. 203–224, Oct 2007.

[139] M. Vettoretti, A. Facchinetti, G. Sparacino, and C. Cobelli, "Type 1 diabetes patient decision simulator for in silico testing safety and effectiveness of insulin treatments," *IEEE Transactions on Biomedical Engineering*, vol. PP, no. 99, pp. 1–1, Aug 2017.

[140] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, Jan 2001.

[141] H. Drucker, "Improving regressors using boosting techniques," in *14th International Conference on Machine Learning (ICML)*, San Francisco, CA, USA, Jul 1997.

[142] A. Facchinetti, S. Del Favero, G. Sparacino, J. R. Castle, W. K. Ward, and C. Cobelli, "Modeling the glucose sensor error," *IEEE Transactions on Biomedical Engineering*, vol. 61, no. 3, pp. 620–629, Mar 2014.

[143] A. Facchinetti, S. Del Favero, G. Sparacino, and C. Cobelli, "Model of glucose sensor error components: identification and assessment for new dexcom g4 generation devices," *Medical & biological engineering & computing*, vol. 53, no. 12, pp. 1259–1269, Dec 2015.

[144] C. J. V. Rijsbergen, *Information Retrieval*, 2nd ed. Butterworth-Heinemann, 1979.

[145] M. Christiansen, T. Bailey, E. Watkins, D. Liljenquist, D. Price, K. Nakamura, R. Boock, and T. Peyser, "A new-generation continuous glucose monitoring system: improved accuracy and reliability compared with a previous-generation system," *Diabetes Technology and Therapeutics*, vol. 15, no. 10, pp. 881–888, Oct 2013.

[146] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, Feb 2009.

[147] L. Breiman, J. Friedman, R. Olshen, and C. J. Stone, "Classification and regression trees," *Wadsworth*, 1984.

[148] C. Dance, "Lean Smart Parking," *The Parking Professional*, vol. 30, no. 6, pp. 26–29, 2014.

[149] G. Pierce and D. Shoup, "Getting the Prices Right," *Journal of the American Planning Association*, vol. 79, no. 1, pp. 67–81, 2013.

[150] Worldsensing, "Smartprk – Making Smart Cities Happen," http://www.fastprk.com/.

[151] N. Piovesan, L. Turi, E. Toigo, B. Martinez, and M. Rossi, "Data Analytics for Smart Parking Applications," *Sensors*, vol. 16, no. 10, 2016.

[152] E. I. Vlahogiannia, K. Kepaptsogloua, V. Tsetsosa, and M. G. Karlaftisa, "A Real-Time Parking Prediction System for Smart Cities," *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, vol. 20, no. 2, pp. 192–204, 2016.

[153] A. Kanungo, A. Sharma, and C. Singla, "Smart traffic lights switching and traffic density calculation using video processing," in *IEEE Recent Advances in Engineering and Computational Sciences (RAECS)*, 2014.

[154] B. Ghazal, K. ElKhatib, K. Chahine, and M. Kherfan, "Smart traffic light control system," in *International Conference on Electrical, Electronics, Computer Engineering and their Applications (EECEA)*, 2016.

[155] J. Rzeszótko and S. H. Nguyen, "Machine Learning for Traffic Prediction," *Fundamenta Informaticae - Concurrency Specification and Programming*, vol. 119, no. 3-4, pp. 407–420, 2012.

[156] M. Scalabrin, M. Gadaleta, R. Bonetto, and M. Rossi, "A bayesian forecasting and anomaly detection framework for vehicular monitoring networks," in *Machine Learning for Signal Processing (MLSP), 2017 IEEE 27th International Workshop on*. IEEE, 2017, pp. 1–6.

[157] R. J. Beckman and R. D. Cook, "Outlier..... s," *Technometrics*, vol. 25, no. 2, pp. 119–149, 1983.

[158] Z. A. Bakar, R. Mohemad, A. Ahmad, and M. M. Deris, "A Comparative Study for Outlier Detection Techniques in Data Mining," in *Cybernetics and Intelligent Systems, 2006 IEEE Conference on*, 2006.

[159] V. J. Hodge and J. Austin, "A Survey of Outlier Detection Methodologies," *Artificial intelligence review*, vol. 22, no. 2, pp. 85–126, 2004.

[160] M. Agyemang, K. Barker, and R. Alhajj, "A Comprehensive Survey of Numeric and Symbolic Outlier Mining Techniques," *Intelligent Data Analysis*, vol. 10, no. 6, pp. 521–538, 2006.

[161] J. Lan, C. Long, R. C.-W. Wong, Y. Chen, Y. Fu, D. Guo, S. Liu, Y. Ge, Y. Zhou, and J. Li, "A New Framework for Traffic Anomaly Detection," in *Proceedings of the 2014 SIAM International Conference on Data Mining*, 2014.

[162] W. Liu, Y. Zheng, S. Chawla, J. Yuan, and X. Xing, "Discovering Spatio-temporal Causal Interactions in Traffic Data Streams," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2011.

[163] L. X. Pang, S. Chawla, W. Liu, and Y. Zheng, "On Mining Anomalous Patterns in Road Traffic Streams," in *International Conference on Advanced Data Mining and Applications*, 2011.

[164] X. Li, Z. Li, J. Han, and J. G. Lee, "Temporal Outlier Detection in Vehicle Traffic Data," in *2009 IEEE 25th International Conference on Data Engineering*, 2009.

[165] S. Sun, C. Zhang, and G. Yu, "A Bayesian Network Approach to Traffic Flow Forecasting," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 1, pp. 124–132, 2006.

[166] H. Yin, S. Wong, J. Xu, and C. Wong, "Urban Traffic Flow Prediction Using a Fuzzy-Neural Approach," *Transportation Research Part C: Emerging Technologies*, vol. 10, no. 2, pp. 85–98, 2002.

[167] G. Yu, J. Hu, C. Zhang, L. Zhuang, and J. Song, "Short-Term Traffic Flow Forecasting Based on Markov Chain Model," in *IEEE IV2003 Intelligent Vehicles Symposium. Proceedings (Cat. No.03TH8683)*, 2003.

[168] C. M. Bishop, *Pattern recognition and machine learning*. Springer, New York, 2006.

[169] *Bitcarrier technology*, http://www.worldsensing.com/product/ bitcarrier/.

[170] C. R. Rao, *Linear statistical inference and its applications*. John Wiley & Sons, 2009.

[171] "Cisco Visual Networking Index: Global mobile data traffic forecast update, 2015–2020 white paper," Tech. Rep., 2015.

[172] "ISO/IEC 23009-1:2014: Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats," International Organization for Standardization, Standard, May 2014.

[173] R. K. Mok, E. W. Chan, and R. K. Chang, "Measuring the quality of experience of HTTP video streaming," in *IFIP/IEEE 12th International Symposium on Integrated Network Management (IM 2011)*, Dublin, Ireland, May 2011, pp. 485–492.

[174] T. Hoßfeld, M. Seufert, C. Sieber, and T. Zinner, "Assessing effect sizes of influence factors towards a QoE model for HTTP adaptive streaming," in *IEEE 6th International Workshop on Quality of Multimedia Experience (QoMEX)*, Singapore, Singapore, Sep 2014, pp. 111–116.

[175] T. Hoßfeld, S. Egger, R. Schatz, M. Fiedler, K. Masuch, and C. Lorentzen, "Initial delay vs. interruptions: between the devil and the deep blue sea," in *IEEE 4th International Workshop on Quality of Multimedia Experience (QoMEX)*, Melbourne, Australia, Jul 2012, pp. 1–6.

[176] Z. Li, A. C. Begen, J. Gahm, Y. Shan, B. Osler, and D. Oran, "Streaming video over HTTP with consistent quality," in *ACM 5th Multimedia Systems Conference (MMSys)*, Singapore, Singapore, Mar 2014, pp. 248–258.

[177] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[178] M. Gadaleta, F. Chiariotti, M. Rossi, and A. Zanella, "D-dash: A deep q-learning framework for dash video streaming," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 703–718, 2017.

[179] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "Developing a predictive model of quality of experience for internet video," *ACM SIGCOMM Computer Communication Review (CCR)*, vol. 43, no. 4, pp. 339–350, 2013.

[180] P. Juluri, V. Tamarapalli, and D. Medhi, "Measurement of quality of experience of video-on-demand services: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 401–418, 2016.

[181] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A survey on quality of experience of HTTP Adaptive Streaming," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 469–492, 2015.

[182] M. Shahid, A. Rossholm, B. Lövström, and H.-J. Zepernick, "No-reference image and video quality assessment: a classification and review of recent approaches," *EURASIP Journal on Image and Video Processing*, vol. 2014, no. 1, pp. 1–32, 2014.

[183] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[184] C. Kreuzberger, B. Rainer, H. Hellwagner, L. Toni, and P. Frossard, "A comparative study of DASH representation sets using real user characteristics," in *ACM 26th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2016, p. 4.

[185] S. Cicalo, N. Changuel, R. Miller, B. Sayadi, and V. Tralli, "Quality-fair HTTP adaptive streaming over LTE network," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 714–718.

[186] J. De Vriendt, D. De Vleeschauwer, and D. Robinson, "Model for estimating QoE of video delivered using HTTP adaptive streaming," in *IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, Ghent, Belgium, May 2013, pp. 1288–1293.

[187] J. Kua, G. Armitage, and P. Branch, "A survey of rate adaptation techniques for Dynamic Adaptive Streaming over HTTP," *IEEE Communications Surveys & Tutorials*, 2017.

[188] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE," in *ACM 8th International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, Nice, France, Dec 2012, pp. 97–108.

[189] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and adapt: Rate adaptation for HTTP video streaming at scale," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 719–733, 2014.

[190] S. Petrangeli, J. Famaey, M. Claeys, and F. De Turck, "A QoE-driven rate adaptation heuristic for enhanced adaptive video streaming," Ghent University-iMinds, Department of Information Technology, Tech. Rep., 2014.

[191] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 325–338, 2015.

[192] A. Bokani, M. Hassan, and S. Kanhere, "HTTP-based adaptive streaming for mobile clients using Markov Decision Process," in *IEEE 20th International Packet Video Workshop*, San Jose, CA, USA, Dec 2013, pp. 1–8.

[193] C. Zhou, C.-W. Lin, and Z. Guo, "mDASH: A Markov Decision-based rate adaptation approach for dynamic HTTP streaming," *IEEE Transactions on Multimedia*, vol. 18, no. 4, pp. 738–751, 2016.

[194] J. Lee and S. Bahk, "On the MDP-based cost minimization for video-on-demand services in a heterogeneous wireless network with multihomed terminals," *IEEE Transactions on Mobile Computing*, vol. 12, no. 9, pp. 1737–1749, 2013.

[195] S. Colonnese, F. Cuomo, T. Melodia, and R. Guida, "Cloud-assisted buffer management for HTTP-based mobile video streaming," in *ACM 10th Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks*, Barcelona, Spain, Nov 2013, pp. 1–8.

[196] M. Claeys, S. Latré, J. Famaey, T. Wu, W. Van Leekwijck, and F. De Turck, "Design of a Q-learning-based client quality selection algorithm for HTTP adaptive video streaming," in *Adaptive and Learning Agents Workshop (ALA-2013)*, Saint Paul, Minnesota, USA, May 2013, pp. 30–37.

[197] M. Claeys, S. Latré, J. Famaey, T. Wu, W. Van Leekwijck, and F. De Turck, "Design and optimisation of a (FA) Q-learning-based HTTP adaptive streaming client," *Connection Science*, vol. 26, no. 1, pp. 25–43, 2014.

[198] V. Martín, J. Cabrera, and N. García, "Q-learning based control algorithm for HTTP adaptive streaming," in *IEEE International Conference on Visual Communications and Image Processing (VCIP)*, Singapore, Singapore, Dec 2015, pp. 1–4.

[199] J. van der Hooft, S. Petrangeli, M. Claeys, J. Famaey, and F. De Turck, "A learning-based algorithm for improved bandwidth-awareness of adaptive streaming clients," in *IFIP/IEEE International Symposium on Integrated Network Management (IM 2015)*, May 2015, pp. 131–138.

[200] F. Chiariotti, S. D'Aronco, L. Toni, and P. Frossard, "Online learning adaptation strategy for dash clients," in *ACM 7th International Conference on Multimedia Systems (MMSys)*, Klagenfurt am Wörthersee, Austria, May 2016, pp. 8:1–8:12.

[201] S. Chikkerur, V. Sundaram, M. Reisslein, and L. J. Karam, "Objective video quality assessment methods: A classification, review, and performance comparison," *IEEE transactions on broadcasting*, vol. 57, no. 2, pp. 165–182, 2011.

[202] H. R. Sheikh, M. F. Sabir, and A. C. Bovik, "A statistical evaluation of recent full reference image quality assessment algorithms," *IEEE Transactions on image processing*, vol. 15, no. 11, pp. 3440–3451, 2006.

[203] M. Zorzi, A. Zanella, A. Testolin, M. D. F. D. Grazia, and M. Zorzi, "Cognition-based networks: A new perspective on network optimization using learning and distributed intelligence," *IEEE Access*, vol. 3, pp. 1512–1530, 2015.

[204] O. Rose, "Statistical properties of MPEG video traffic and their impact on traffic modeling in ATM systems," in *IEEE 20th Conference on Local Computer Networks*, Minneapolis, Minnesota, Oct 1995, pp. 397–406.

[205] R. Bellman, "A Markovian decision process," *Indiana University Mathematics Journal*, vol. 6, no. 4, pp. 679–684, 1957.

[206] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[207] M. Kearns and S. Singh, "Finite-sample convergence rates for Q-learning and indirect algorithms," *Advances in Neural Information Processing Systems*, vol. 11, pp. 996–1002, 1999.

[208] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[209] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[210] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with LSTM recurrent networks," *Journal of Machine Learning Research*, vol. 3, pp. 115–143, Aug 2002.

[211] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[212] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[213] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[214] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[215] L.-J. Lin, "Reinforcement learning for robots using neural networks," DTIC Document, Tech. Rep., 1993.

[216] X. Li and X. Wu, "Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, South Brisbane, Queensland, Australia, Apr 2015, pp. 4520–4524.

[217] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[218] P. Juluri, V. Tamarapalli, and D. Medhi, "Qoe management in dash systems using the segment aware rate adaptation algorithm," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2016, pp. 129–136.

[219] A. Testolin, M. Zanforlin, M. D. F. D. Grazia, D. Munaretto, A. Zanella, M. Zorzi, and M. Zorzi, "A machine learning approach to QoE-based video admission control and resource allocation in wireless systems," in *13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, Los Angeles, CA, USA, June 2014, pp. 31–38.

[220] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck, "HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks," *IEEE Communications Letters*, vol. 20, no. 11, pp. 2177–2180, 2016.

[221] H. X. Nguyen, P. Thiran, and C. Barakat, "On the correlation of TCP traffic in backbone networks," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 5, Vancouver, Canada, May 2004, pp. V–481.

[222] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race, "Towards network-wide QoE fairness using OpenFlow-assisted adaptive video streaming," in *ACM/SIGCOMM Workshop on Future human-centric multimedia networking (FhMN)*, Hong Kong, China, Aug 2013, pp. 15–20.

[223] A. Mansy, M. Fayed, and M. Ammar, "Network-layer fairness for adaptive video streams," in *IEEE/IFIP Networking Conference*, Toulouse, France, May 2015, pp. 1–9.

[224] J. W. Kleinrouweler, S. Cabrero, R. van der Mei, and P. Cesar, "Modeling stability and bitrate of network-assisted HTTP adaptive streaming players," in *IEEE 27th International Teletraffic Congress (ITC 27)*, Ghent, Belgium, Sep 2015, pp. 177–184.