Machine Learning for 5G Mobile Networks: a Pragmatic Essay on *where*, *how* and *why*

Paolo Dini, Michele Rossi

Abstract In this chapter, we discuss selected applications of Machine Learning (ML) algorithms to 5G systems. We follow a pragmatic approach, leveraging our recent research work, which encompasses algorithms for energy efficiency, Quality of Service (QoS) enhancements and the use of inference tools for the analysis and prediction of context information (traffic, mobility, etc.). Our objective is to demonstrate *where* (which applications), *how* (in combination with which techniques) and *why* (the benefits) ML may be useful, with an emphasis on the *how*, i.e., what are the usage models, and what they imply in terms of *stability*, *convergence* and *optimality* guarantees of the resulting control algorithms. Within our discourse, the use of open *vs* closed control loop architectures and the role of supervised *vs* unsupervised learning are emphasized.

1 Introduction

The design for machines able to perform human tasks with high precision and reliability is a hot research topic across nearly all fields of knowledge. A large amount of work is being carried out towards building computer programs that are capable of processing complex and noisy data from the environment and that make decisions autonomously, based on some extracted features or on statistical traits of collected data samples/time series. The artificial intelligence (AI) field studies the use of computers in tasks that require cognitive abilities like understanding, reasoning and learning. Especially, machine learning (ML) is the branch of the AI that investigates algorithms that are capable of learning and improving themselves

Paolo Dini
Centre Tecnològic Telecomunicacions Catalunya (CTTC), Barcelona, Spain e-mail: paolo.dini@cttc.es

Michele Rossi
Department of Information Engineering, University of Padova, Italy, e-mail: rossi@dei.unipd.it

directly from data examples, without being explicitly pre-programmed through a set of hard coded rules. So far, ML techniques have been successfully applied to diverse applications like natural language processing, speech and handwritten recognition, image classification, and financial trading.

A great deal of work is being carried out within the information and communication technologies (ICT) field, where new wireless radios and network protocols are rapidly integrating new learning and adaptation capabilities. As for 5G, mobile networks that learn and adapt their behavior to user data are expected to offer better services to end users and are likewise appealing to network operators:

- **End users** are expected to benefit from the adaptation capabilities offered by ML, by enjoining a higher quality of service (QoS), e.g., experiencing a higher throughput, ultra-low latencies and seamless communications when on the move.
- **Network operators** can instead optimize the way in which they assign and use their network resources (e.g., bandwidth and computation), with a subsequent increment in the efficiency of the deployed communication (and computing) hardware. Such efficiency enhancement may be in terms of: 1) an increase in the energy efficiency of the whole network (e.g., energy spent per delivered information bit) or 2) an increase in the network capacity, intended as the amount of traffic or number of users that can be supported. Note that both 1) and 2) entail a *revenue* for the operators, as for 1) smaller energy amounts have to be purchased from the power grid and, for 2), more users can be supported, or the same load can be supported at a reduced installation (hardware) cost. A third major incentive towards integrating learning algorithms into 5G networks is the possibility of offering *new services*, which will be enabled by the so called "network intelligence", i.e., by algorithms residing within the 5G network that will be capable of monitoring user activities and processing end-user data to enable advanced applications such as *augmented reality* in work environments, amusement parks, libraries, museums, etc. Other applications that are expected to benefit from high bandwidth, low delay and intelligent processing are virtual reality, autonomous driving and industrial Internet of Things (IoT).

Many research papers are appearing on the use of machine intelligence in 5G systems, protocols, and on the new applications that 5G networks are expected to enable. The overall scenario is however still unclear, as many of such applications are not yet available and the way in which they will be eventually implemented depends on many factors, some of which are technological and some commercial. To keep things concrete, in this chapter we discuss the main technical benefits that learning may bring about by presenting some example applications of machine intelligence from our own recent research work. The technical aspects that we address are:

1. **Energy efficiency**: using machine learning techniques to improve the *energy efficiency* of 5G deployments. This is a benefit for the network operators and will be addressed in Section 3;
2. **Performance enhancements**: using reinforcement learning to provide *performance improvements*, either for 5G users or network protocols/procedures, see Section 4;

3. **Context awareness**: using machine learning to provide *context awareness* in 5G networks. This is a new feature, enabled by learning algorithms, that may be exploited to deliver new services to the end users and/or to enhance 5G network procedures and protocols, see Section 5.

Before delving into the description of the above technical aspects, in Section 2 we provide a primer on machine learning, through a brief taxonomy of the most popular learning approaches and of their goals.

2 A primer on machine learning techniques

To start with, let us consider the task of training a ML algorithm in a **supervised** fashion. In a supervised setting we own a (possibly large) number of training examples and, associated with each of them, we also own a *label*, which is either precomputed, measured or assessed by a human expert. Supervised learning is probably the most effective way in which we can teach an algorithm to perform a certain task. So far, it has been the most successful and widely utilized approach, due to the existence of powerful means for training (i.e., the *backpropagation* algorithm).

In its simplest form, a supervised ML framework is composed of a set of input data examples $X = [x_1, x_2, \ldots, x_N]$, a ML algorithm, and a set of target data objects (i.e., the labels), $Y = [y_1, y_2, \ldots, y_K]$. Depending on the specific task that is to be solved, the x_i and the y_i can be scalars, vectors or matrices, and K can be greater than, equal to or smaller than N. The goal of a ML algorithm is to find a proper way to automatically *map* each input sample x_i into the corresponding target sample y_i. The process through which the algorithm learns the mapping is called *training* and is executed using a set of training data. After the training process, the algorithm can be applied to a new (never seen) dataset to perform the required task. The ML algorithm is expected (and in fact required) to generalize to unseen data, which usually occurs if the new samples that are inputted at runtime are sampled from the same statistical distribution that generated the samples in the training set. For this reason, the selection of an adequate training set is of vital importance, and it is equally important to train the ML algorithm so as to avoid it to **overfit** the training dataset, in which case it would fail to provide satisfactory answers on unseen data, even if sampled from the same distribution.

Broadly speaking, a ML algorithm can learn through one of the following techniques:

- **Supervised learning.** These algorithms require the knowledge of pairs (x_i, y_i), for $i = 1, 2, \ldots, N$. The learning is supervised because the output y_i produced by the algorithm when x_i is applied as an input, is compared with the target value y_i, and a (user-defined) *error* is used to adjust the algorithm itself.
- **Unsupervised learning.** In this case, the algorithm is not provided with the target data Y. The objective is therefore to learn the implicit structure of the data. Tech-

nically, this amounts to automatically learning the subspace (the "manifold") over which the input data samples reside. Often, such manifold has a smaller dimensionality than the data space itself and in this case the ML algorithm performs a dimensionality reduction task. Classification of input samples, anomaly detection or inference tasks are then performed using the acquired knowledge.

- **Reinforcement learning.** As with unsupervised algorithms set Y is not provided. However, with respect to unsupervised learning, the algorithm interacts with the environment through the following cycle: 1) making decisions based on the current knowledge about the environment, 2) implemented these decisions, acting on the environment, 3) measuring the environment's response, in the form of a *reward*, 3) updating the algorithm's internal knowledge about the environment/problem and repeating from 1). The goal of the learning process usually amounts to maximizing the cumulative reward over some time horizon.

The last classification that we make is about what a ML algorithm is usually good at solving. In fact, ML algorithms can be trained for regression or classification tasks:

- **Regression.** The objective of regression is to predict the value of one or more continuous target variables y_i, given the value of the input data samples x_i. Using the simplest regression model, linear regression, the algorithm learns a linear function to map each input to the corresponding output, e.g., $y_i' = f(x_i) = w_0 + x_i^T w$, where the function weights w_0, w_1 are tuned during the training process, so as to minimize the error between the output y_i' and the target y_i.
- **Classification.** The goal of classification is to assign to each input sample x_i one out of K possible classes C_k, with $k = 1, \ldots, K$. A simple method to classify input samples into two classes consists of assigning class C_1 to the x_i for which $w_0 + x_i^T w_1 > 0$ and class C_2 to the others. Once again, the weights are all we need to learn from the data to fully specify this simple classification algorithm.

Nowadays, the previous linear models are seldom used, as they are largely superseeded by non-linear functions. These non-linear functions are implemented by stacking layers of artificial neurons, which are the simplest block in any artificial neural network (ANN) architecture. The use of stacked non linear architectures has been found to be key to achieving unprecedented regression and classification performance, and can be realized through a variety of architectural designs including feed forward neural networks (FFNN), convolutional neural networks (CNN) or recurrent neural networks (RNN).

An excellent text on FFNN and backpropagation is [1], whereas the reader is referred to [2] for Deep neural networks designs. A primer on the use of machine learning in communications systems is provided in [3].

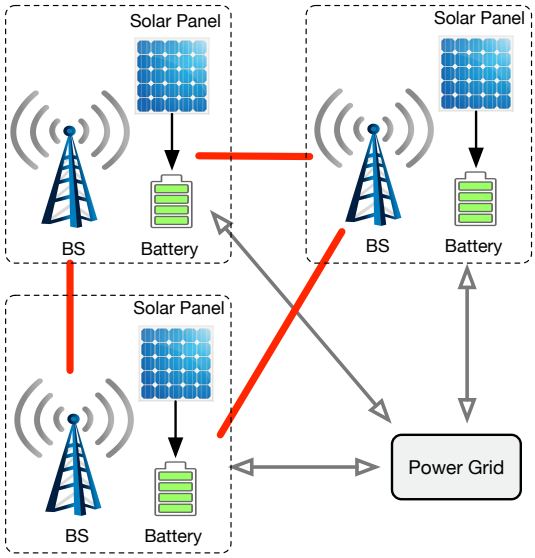

Fig. 1 Reference network scenario. Base stations are equipped with energy harvesting (solar panel) and accumulation (battery) means. Energy can be sold to and/or purchased from the power grid and can also be exchanged between base stations (red links represent electrical lines provided by an electrical micro-grid infrastructure connecting the base stations).

3 Pattern learning and predictive control for energy efficiency

In this section, we discuss the use of learning (and control) techniques to increase the energy efficiency of 5G networks. The reference scenario is shown in Fig. 1. Base stations (BSs) have energy harvesting (e.g., a solar panel) and accumulation (e.g., a battery) capabilities. A micro-grid connects BSs so that they can exchange excess energy among them (see the red links in the figure). Some of the BSs, referred to as *ongrid*, are connected to the power grid and, in turn, can send excess energy to the power grid or purchase energy from it, while *offgrid* BSs are not connected to the power grid and can solely use the energy that they harvest locally or that they receive from other BSs. With this setup in mind, we are concerned with the following facts, informally stated in the following:

- To maintain the buffer levels $B_n(t)$ at all the BSs n in the network as close as possible to a given reference level B_{ref}, at all times t.
- To minimize the frequency of *energy outage* events for the offgrid BSs, during which they are unable to serve the local traffic due to energy scarcity.
- To prioritize the use of the energy harvested over that purchased.

In [4], these objectives are achieved through the combined use of *machine learning* and *model predictive control (MPC)* theory [5]. In the considered scenario, there are two time varying *exogenous processes*: **P1)** the incoming energy from harvesting

devices, **P2)** the traffic load from the connected users. Remarkably, ML tools are not used to make decisions, but rather to solve an *inference problem*. Specifically, ML algorithms are trained to act as predictors for the exogenous time series **P1** and **P2**, whereas standard control theory uses such predictions to make suboptimal, but generally very good (close to optimal) decisions, through MPC based optimization. We found this approach to be very pragmatic and effective, and as well applicable to diverse settings. Also, it allows plugging machine learning algorithms into a standard control architecture with tracking and stability guarantees.

ML algorithms are utilized as follows.

- **P1** The amount of energy harvested can be independently measured by each BS and stored into time series with a time granularity of, e.g., 30 minutes / one hour. Such time series can be provided as input to machine learning tools such as Gaussian processes [6] for time series analysis or RNNs [2]. These ML models can be trained for each BS, based on its own historical energy harvesting data. The *learning objective* amounts to predicting the harvested energy time series within a certain prediction window into the future (W time slots). Usually, this job is carried out excellently by the considered ML tools, especially because solar energy time series have bell-shaped patterns (with a peak around midday and zero energy income at night) with a high amount of temporal correlation, both within a day and across days.
- **P2** The second exogenous process that is to be estimated by the BSs is the aggregated local traffic load. The same ML algorithms of above can be trained to act as predictors, forecasting the (aggregated) load intensity over a prediction horizon of W time slots. The forecasting performance for the load process is usually lower than that achieved for energy harvesting time series, as the load generally shows a higher temporal variability. However, we found the considered ML predictors to be accurate enough to correctly track the general behavior of the load process, i.e., whether it will be increasing, about constant or decreasing within the prediction horizon. Remarkably, we found that when MPC algorithms are used in combination with ML predictors, as we did in [4], even inaccurate predictions may lead to nearly optimal results, as long as the general "direction" of the forecast processes is guessed correctly (e.g., whether they will be increasing, decreasing or about constant within the prediction horizon).

To quantify the benefits of the discussed predictive control approach, in Fig. 2 we show the outage probability for two cases. "Optimal – myopic" implements an optimal energy allocation strategy, but without using pattern learning and predictions. "Optimal – MPC" refers to the predictive control approach that was discussed above, where pattern learning and predictions are utilized by an MPC-based controller. The energy threshold η in the abscissa represents the ratio between the amount of energy that each ongrid BS is allowed to purchase daily and the total amount of energy it would require to serve a fully loaded scenario during a full day. Setting η amounts to putting a daily limit on the total amount of energy that can be purchased by the network of base stations. As can be seen from this plot, the use of predictions allows

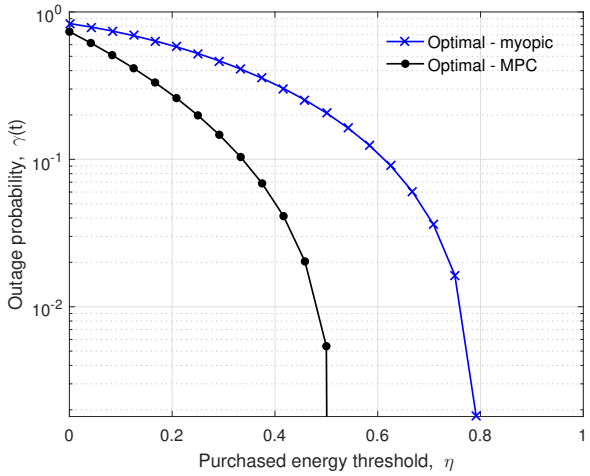

Fig. 2 Outage probability for a network with 18 base stations. "Optimal – myopic" refers to an optimal allocation of energy resources in each time slot, with no ML-based forecasting. "Optimal – MPC" refers to the optimal predictive based control strategy exploiting ML-based forecasting of harvested energy and load.

a substantial reduction of outage events (the two curves are computed accounting for the same traffic load and energy harvesting processes). A similar comparison, performed removing the limit on the energy that can be purchased reveals that, for the same value of outage probability, the total energy amount that is acquired by the power grid is reduced by at least 35% in the worst case and up to 70% in the best one, depending on the distribution of load across the BSs, see [4] for details.

Further developments. This research line is further explored in [7], where a similar problem is tackled using RNN for pattern forecasting and additionally accounting for hourly energy prices into the optimization problem. Another example is provided by [8], where the authors propose a layered learning mechanism for massive deployments of 5G small base stations powered by solar energy: in the first layer, BSs locally decide upon ON/OFF policies accounting for the local energy income and traffic demand through an accelerated reinforcement learning scheme (see Section 4). In a second layer, an artificial neural network estimates the network load at a global level, acting as a centralized controller to steer the decisions of the local BS agents.

Observation. Notably, in [4, 7] ML tools were used within a *closed loop* MPC decision making architecture. This makes it possible to retain the convergence and stability qualities of MPC control, while exploiting the excellent pattern learning and forecasting capabilities of ML algorithms. Further control architectures that exploit this approach are presented in [5].

4 Reinforcement learning for 5G performance enhancement

Reinforcement learning (RL) [9] is an optimization technique for *sequential decision making*, i.e., when actions have to be taken as time evolves to steer the behavior of a certain stochastic system that we would like to control. RL has been developed to overcome the limitations of Markov decision processes (MDP), where the system model (the full statistical description of the system evolution) is to be known in advance. The main advance of MDPs is that they compute optimal policies, but this comes at the costs of

- **C1** – knowing exactly the Markov chain describing the system evolution;
- **C2** – running optimization algorithms that iteratively compute Bellman equations across *all* system states and *all* actions.

In some cases, **C1** is a major problem as the system is not entirely known to the designer. This especially applies to complex 5G systems, for which an exact derivation of the involved transition probabilities is difficult to carry out, e.g., due to the lack of an appropriate statistical description of the exogenous processes that interact with the decision making algorithm (the controller). Moreover, MDPs are usually formulated considering discrete state spaces by quantizing the involved variables into a number of levels. Due to **C2**, the complexity entailed in searching for an optimal solution usually grows exponentially with the number of states. This fact is known as the *curse of dimensionality*, and often prevents the applicability of MDP to real world problems where the number of states is considerable.

RL was developed as a solution to **C1** and **C2**, providing the following benefits

- RL is said to be *model free*, as the probability distribution of the underlying Markov model that describes the problem at hand is not explicitly used. Instead, the system behavior is captured by an action-reward mapping, which is iteratively learned through a *trial-and-error* approach, i.e., interacting with the environment and measuring its response;
- the full exploration of the state space is not needed. Only relevant states (and actions) will be visited and used to update the internal knowledge of the decision maker (the action-reward maps): this considerably reduces the complexity of the search for optimal policies.

On the other hand, RL has the main disadvantages that convergence to (nearly) optimal solutions is only achieved asymptotically, upon completing a sufficient number of trial-and-error cycles, and that such convergence may be problematic if the system behavior is non-stationary.

Optimization example – RL for video streaming performance enhancement. In [10], we proposed an RL algorithm for the control of the quality of experience (QoE) of mobile users connected to a Dynamic Adaptive Streaming over HTTP (DASH) video service.

The controller (decision maker) runs at the mobile device with the objective of controlling the video quality of a DASH streaming service. This amounts to adapting the video quality that is requested from the DASH server (within the 5G or Internet network) at runtime, according to channel and video dynamics.

The learning cycle used to compute optimal policies is shown in Fig. 3. At each time t a new action a_t is picked using the Q-table, $Q(s_t, a_t)$, which is implemented by the neural network (labeled "Network") in the diagram. The Q-table is a function mapping the current state s_t into an output vector of long-term rewards, one for each action $a \in A(s_t)$, where $A(s_t)$ is the feasible action set for state s_t. The next action a_t is picked either using a `softmax` or an ε-greedy strategy. With `softmax`, the next action $a^\star$ is the one that maximizes the long-term reward, i.e., $a^\star = \max_{a \in A(s_t)} Q(s_t, a)$. With ε-greedy, the next action is $a^\star$ with probability ε, and it is instead picked at random with probability $1 - \varepsilon$. This allows controlling the exploration versus exploitation tradeoff: with probability ε we stick with what has been learned so far, steering the search towards the best local direction ("exploiting" the local gradient), whereas with probability $1 - \varepsilon$ we allow the solver to explore new regions incurring a risk, but allowing the algorithm to escape local minima ("exploration"). The action a_t is implemented and, in return, the environment will return a reward r_t and the state will advance to s_{t+1}.

Reward function. The reward function r_t encodes our preferences: 1) the instantaneous picture quality, 2) the playout buffer state, which should ideally be above a certain threshold, at all times, 3) the video quality stability, i.e., we would like to avoid that the video quality changes sharply or oscillates noticeably.

System state. The state s_t encodes the channel capacity for the last $n \geq 1$ segments, the buffer playout time for segment t (the lapse of time from the start of the segment download and the time when its playout is due at the client) and the re-buffering time for video segment t (greater than zero if the download time for the current segment exceeds the buffer playout time).

Deep Q-learning. The adopted RL algorithm uses a Q-learning strategy. The Q-table is represented by the neural network that, for each state s_t returns the list of estimated long-term rewards $Q(s_t, a)$ for all the actions $a \in A(s_t)$. Traditional Q-learning uses standard memory tables to store the $Q(s_t, a)$ long-term reward estimates. However, when s_t is very large Q-learning suffers from the curse of dimensionality problem, as the size of the tables is proportional to the cardinality of the state space. Common approaches encode s_t into quantized values, by quantizing the system variables, but this may imply a considerable loss in the quality of the solution. The approach proposed in [10] solves this as a deep neural network (hence the name "deep Q-learning") is utilized in place of a memory table, with the following advantages:

1. all variables in s_t can be kept continuous within a bounded set,
2. only a fine set of neural network parameters (the "weights") are to be learned to fully specify the mapping $Q(s_t, a)$. If learning is carried out correctly, the neural

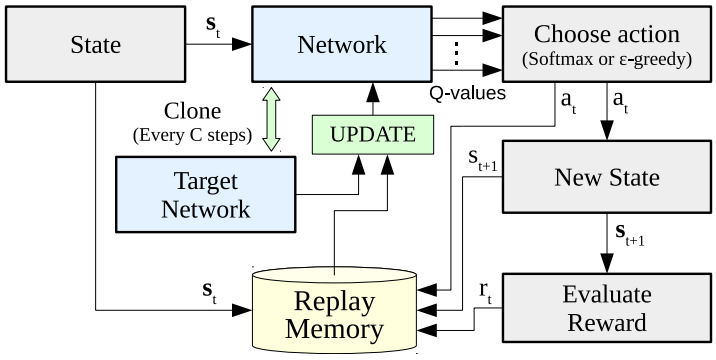

Fig. 3 RL learning cycle. At each time t, s_t indicates the system state, the next action a_t is picked using the Q-table (the block called "network" on the top), as a response the environment returns a new state s_{t+1} and a reward r_t.

network table remains accurate (it "generalizes") even for state-action pairs that were not encountered during the training phase.

The trick of using a neural network as a mapping engine empowers RL algorithms, allowing them to effectively cope with the curse of dimensionality problem. Other subtleties, such as using a replay buffer and a clone neural network, are needed to ensure a proper learning of the Q-tables (i.e., the eventual convergence to nearly optimal policies). The reader is referred to [10] for additional implementation details.

As shown in [10], the results achieved with this RL approach greatly outperform those obtained with standard Q-learning and with MPC-based algorithms both in terms of convergence speed and quality of the solution (e.g., freezing prevention and playout stability).

Distributed RL for functional split control for 5G small cells. A distributed application of RL is presented in [11], where the authors consider dense deployment of 5G small cells powered by renewable energy sources. In this setup, they propose a novel algorithm to handle the split of baseband processing functions between the BSs and the central cloud, that can be hosted by a server facility within the *network edge*. As radio cells become smaller and deployed in a densified manner, the energy consumption associated with baseband processing takes a large share of the total BS energy expenditure, so it becomes important to assess which baseband functions (or part of) are to be executed locally, and which ones shall be offloaded to network servers to strike the right balance between processing delay and overall energy consumption. In [11], an energy minimization problem for the 5G small cell deployment is posed and solved through two multi-agent RL techniques: distributed fuzzy Q-Learning and Q-Learning online algorithms. Coordination among the distributed agents is achieved by broadcasting system level information. The proposed algorithms bring the following main benefits: 1) a *continuous* state-action representation is adopted,

thus favoring convergence speed and generality of the approach, 2) the broadcast information allows coordinating the distributed RL agents, leading to higher stability and better convergence properties.

Concluding remarks about RL. Similarly to what presented in Section 3, neural networks are used in a *closed control loop*, empowering RL algorithms with advanced mapping and generalization capabilities. Their use ameliorates the curse of dimensionality problem, whereas the closed control loop allows retaining the convergence and optimality properties of standard control algorithms. As for distributed RL, it is an open research field on its own, which needs further theoretical developments to establish solid *convergence* and *stability* results, as well as viable approaches to achieve *coordination* among the distributed agents. Most of the papers in the literature that apply distributed RL to real world problems, and in particular to networking problems, exploit heuristic approaches providing little theoretical guarantees in terms of optimality and convergence properties. Recent research moving the first steps toward a mathematical theory of distributed RL is [12, 13].

5 Machine learning for context awareness

In this section, we elaborate on the use of machine learning to capture communications and mobility patterns in 5G systems. In the traffic case, load patterns represent transmission flows whose behavior depends on the user applications. In the mobility case, the patterns depend on the physical mobility of the users, as well as on the physical topology of road links and on their morphology. Machine learning tools appear to be good candidates to learn these user-generated patterns, possibly in an unsupervised fashion, and then perform inference and classification to be used, e.g., to enhance network procedures or within new services. Below, selected applications of machine learning to traffic and mobility traces are discussed, underlining the ML tools that were used, the targeted applications/services and the obtained results.

Protocol level analysis of mm-wave energy traces [14]. The authors of [14] propose a ML-based processing pipeline to automatically infer the type of protocol packets that are sent over the air by millimeter-ware radios, from narrowband (undersampled) energy traces, and without requiring one to decode the transmitted data flow. The simplest setup entails a transmitter-receiver pair, which communicate over a millimeter-wave ("mm-wave") wireless channel. A sniffer, equipped with an omni-directional antenna, intercepts this communication channel and samples it at a bandwidth that is much lower than that at which the signal was transmitted. In Fig. 4, we show a portion of a sub-sampled mm-wave signal, referred to as channel energy trace. Although decoding is not possible, an expert observer with knowledge of mm-wave transmission standards can infer that this energy trace contains two initial beacon packets, identifiable by their typical shape (peak at the beginning) and duration (between that of DATA and ACKs), followed by a sequence of DATA pack-

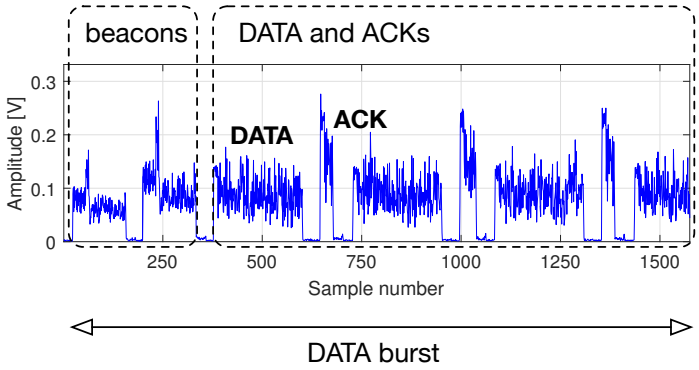

Fig. 4 Zoom of a portion of a mm-wave energy trace from a real measurement setup. Control packets (beacons), data packets and acknowledgments can still be identified, although the trace is subsampled and cannot be decoded to retrieve the original sequence of bits that was transmitted.

ets and acknowledgments (ACKs), discriminated by their duration (usually ACKs are much shorter than DATA packets). Control (beacons, ACKs) and DATA packets are separated by silence and noise is superimposed to all channel readings.

In [14], a ML-based algorithm is put forward to automatically extract and analyze DATA bursts, i.e., sequences of DATA/ACKs. As a first step, correlation-based template matching algorithms are used to mark the beginning of DATA bursts, through the identification of the pair of beacons at their beginning. After that, a dedicated algorithm exploiting Hidden Markov Models (HMM) [15] is *trained* on experimental data to *automatically* classify DATA packets and ACKs, returning the following pieces of information

1. identification of each protocol element, namely, Beacon, DATA packet, ACK;
2. start, end time and average energy level of each protocol element.

This amounts to performing a *semantic protocol decoding* of the mm-wave channel, which may be useful to perform an automatic assessment on the status of the wireless medium, e.g., presence of jammers, missing ACKs (which may indicate reception problems such as antenna misalignment), analysis of per-packet energy levels (to identify blockages/excessive channel interference, etc.).

To provide adaptability to varying channel conditions, the considered ML techniques (HMM and its extended version, i.e., the Extended HMM) are combined with K-means clustering. In this way, it is possible to adapt, *on the fly*, the proposed HMM-based classification algorithm to the time varying energy levels of Beacons, DATA packets and ACKs within each data burst. Finally, a technique is also presented to jointly process time synchronized traces collected by multiple sniffers.

Observation: HMM training is *unsupervised* as HMM parameters are tuned using the Expectation Maximization (Baum-Welch) algorithm [15], without the need

to perform an a priori labeling of DATA packets and ACKs.

Mobile traffic prediction and urban anomaly detection [16, 17, 18]. The traffic of a Long Term Evolution (LTE) mobile system is tracked by decoding the LTE Downlink Control Channel (DCI), which is transmitted in clear text and can thus be read without breaking any security/encryption layer. This allows the collection, for each connected mobile user, of physical layer configuration parameters such as the number of allocated resource blocks, the modulation order and the code rate (modulation and coding profile). From these pieces of information, the amount of data transmitted in uplink and downlink directions can be retrieved, obtaining aggregate traffic profiles for the monitored LTE radio cell(s). In [16], an *urban anomaly detection system* based on stacked Long Short-Term Memory (LSTM) neural networks is designed and evaluated. The intuition behind this work is that the amount of traffic within a certain city area is directly associated with the number of people within the same space, and that mobile traffic can be used as a proxy to detect anomalous events within an urban environment. To this aim, a stacked LSTM network is used due to its capabilities of extracting the long-term temporal dependencies of complex time series, embedding them into a suitable feature space of *fixed dimension*, where classification of traffic behavior should be much easier to carry out than working with the original sequential data. The training of the LSTM network is *supervised*, with labels specifying whether a particular event has occurred in the monitored area during a certain time of the day, e.g., a soccer match, a fair, an exhibition, a concert, etc. The authors have trained their LSTM-based algorithm using real data collected within the city of Barcelona. Their results confirm that the approach, although preliminary, shows promising results and that unusual events implying the gathering of people may be detected reliably. This work has been extended in [18], where the authors train LSTM-based algorithms in a semi-supervised fashion. This overcomes the imbalanced data problem of a purely supervised learning approach, where anomalous classes are often poorly represented with respect to the others, as anomalous events are usually *rare* events [19]. As a result, the problem is no longer tackled as a classification task, but rather, the algorithms are designed to detect traffic anomalies by learning from non-anomalous patterns. This training method makes the anomaly detection framework applicable to a broader set of urban anomalies, including those that are not known a priori. The prediction of mobile traffic is considered in [17] where, based on the same DCI channel monitoring technique of [16], the traffic load of LTE base stations is forecast from one to multiple Transmission Time Intervals (TTIs) into the future. Several forecasting architectures are trained in a *supervised* manner, including LSTM networks, Auto Regressive Integrated Moving Average (ARIMA) filters and Feed Forward Neural Networks (FFNNs). Results obtained with experimental traffic data show the sharp superiority of LSTM predictors.

Mobility-aware optimizations. A number of papers have appeared on the exploitation of mobility patterns within 5G networks, which seems to be a promising research direction. Reference [20] considers ultra-dense 5G deployments and proposes a Semi-Markov model based algorithm to predict the next serving BS. The algorithm

is trained on the sequence of past handover events and mobility predictions are used to balance the load among the BSs. In this work, it is shown that such load balancing leads to significant energy savings. The authors of [21] propose a very timely idea. The rationale is that mobility predictions can be effectively exploited in 5G systems to cache content to BSs in advance (referred to as "proactive caching"). Along the same lines, the recent paper [22] proposes a resource management framework for Multi-access Edge Computing (MEC), where mobility predictions are utilized to decide about the assignment of computing tasks to distributed edge servers.

While the previously discussed research demonstrate the usefulness of mobility predictions within several networking scenarios, they do not directly use ML, which is instead exploited by the following (recent) papers. The authors of [23] consider mobility prediction and ML based channel quality estimation jointly, to improve the resource-efficiency of car-to-cloud data transfer by scheduling the transmission time of the car data with respect to the anticipated behavior of the communication context within a 5G network setup. This mobility/context-predictive approach is assessed in a public mobile network scenario showing an increase of the average data rate by up to 194% and a reduction of the average uplink power consumption by about 50%. Reference [24] considers mobility prediction as one of the key enablers of (5G) Self Organizing Networks (SON). There, a comparative analysis is carried out among four prediction algorithms: Deep Neural Network (DNN), Extreme Gradient Boosting Trees (XGBoost), Semi-Markov based, and Support Vector Machine based (SVM). XGBoost is found as the clear winner among these techniques, with a prediction accuracy as high as 90%.

The above are representative examples, which were selected to demonstrate the broad range of cases where ML can be effectively used. ML is becoming a key technology for 5G systems, encompassing all components/sub-systems from physical layer to network/mobility/energy management, resource allocation (e.g., function placement), edge computing (e.g., scheduling of computing resources), etc. We expect many more applications of ML algorithms to emerge in the near future.

6 Concluding Remarks

Machine learning techniques are being massively utilized within many new designs at nearly all layers of 5G networks. In this book chapter, we have discussed some possible applications of such technology to selected networking problems, trying to show *where* ML may be best exploited, *how* and *why*.

Our first take home messages are that, in our opinion,

1. ML can be effectively exploited within **online learning** and control mechanisms for 5G protocols and networks.

2. **ML inference** (mapping, forecasting) and **classification** (clustering) techniques offer excellent performance, and we especially recommend their use within a *closed control loop* architecture, for example as means to perform forecasting (Section 3) or cost-to-go or action-reward mapping (Section 4). Well known MPC or RL techniques can then be used in conjunction with ML to control the evolution of 5G algorithms and procedures, providing *stability* and *convergence* guarantees, as well as *close-to-optimal* solutions. Their use within open loop architectures is not recommended as, to date, ML tools do not yet offer guarantees about whether they are going to fail for certain input patterns/data.

A further take home message is that

3. there is still ample room for **theoretical developments** of distributed RL and MPC (used in conjunction with ML techniques). 5G and wireless networking offer nice setups for the design and validation of these new algorithms, whose objectives include enabling fully distributed and stable control with guarantees in terms of optimality and convergence speed.

Our last take home messages are

4. ML technology offers very effective means to **learn patterns** in complex 5G systems (Section 5). These patterns, associated with user traffic, harvested energy, mobility, etc., may be exploited to improve the performance of existing algorithms and, most importantly, to spur the design of new applications and services.
5. Last but not least, learning so far has been carried out in a mostly **supervised** fashion as, to date, this is the most effective approach to training ML algorithms, with plenty of optimized software libraries available, e.g., Keras (https://keras.io/), TensorFlow (https://www.tensorflow.org/), PyTorch (https://pytorch.org/), etc. Nevertheless, obtaining labeled datasets is *costly* and, sometimes, impractical. Unsupervised and semi-supervised approaches are expected to be very useful and, in the near future, to become more important than their supervised counterpart. As the authors of [25] state in their seminal paper: *"we expect unsupervised learning to become far more important in the longer term. Human and animal learning is largely unsupervised: we discover the structure of the world by observing it, not by being told the name of every object"*.

Acknowledgment

This work has received funding from the European Union Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 675891 (SCAVENGE) and has been supported, in part, by MIUR (Italian Ministry of Education, University and Research) through the initiative "Departments of Excellence" (Law 232/2016).

References

1. R. Reed and R. J. Marks II, *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. The Mit Press, 1999.
2. I. Goodfellow and Y. Bengio, *Deep Learning*. The Mit Press, 2017.
3. O. Simeone, "A very brief introduction to machine learning with applications to communication systems," *IEEE Transactions on Cognitive Communications and Networking*, vol. 4, no. 4, pp. 648–664, Dec. 2018.
4. A. F. Gambin, M. Scalabrin, and M. Rossi, "Online Power Management Strategies for Energy Harvesting Mobile Networks," *IEEE Transactions on Green Communications and Networking*, vol. 3, no. 3, pp. 721–738, Sep. 2019.
5. M. Ławryńczuk, *Computationally Efficient Predictive Control Algorithms: a Neural Network Approach*. Springer, 2014.
6. C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.
7. Á. F. Gambín and M. Rossi, "Smart Energy Policies for Sustainable Mobile Networks via Forecasting and Adaptive Control," in *IEEE GLOBECOM: Workshop on "Wireless Energy Harvesting Communication Networks"*, Dec. 2019.
8. M. Miozzo, N. Piovesan, and P. Dini, "Coordinated Load Control of Renewable Powered Small Base Stations through Layered Learning," *IEEE Transactions on Green Communications and Networking*, Sep. 2019.
9. R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Bradford Books, 2018.
10. 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, Dec. 2019.
11. D. A. Temesgene, M. Miozzo, and P. Dini, "Dynamic control of functional splits for energy harvesting virtual small cells: A distributed reinforcement learning approach," *Computer Communications*, vol. 148, no. 4, pp. 48–61, Dec. 2019.
12. R. Tutunov, D. Kim, and H. B. Ammar, "Distributed Multitask Reinforcement Learning with Quadratic Convergence," in *Advances in Neural Information Processing Systems (NIPS)*, Dec. 2018.
13. D. Lee, H. Yoon, and N. Hovakimyan, "Primal-Dual Algorithm for Distributed Reinforcement Learning: Distributed GTD," in *IEEE Conference on Decision and Control (CDC)*, Dec. 2018.
14. M. Scalabrin, G. Bielsa, A. Loch, M. Rossi, and J. Widmer, "Machine Learning Based Network Analysis using Millimeter-Wave Narrow-Band Energy Traces," *IEEE Transactions on Mobile Computing*, Mar. 2019.
15. C. Bishop, *Pattern Recognition and Machine Learning*, 1st ed. Springer, 2006.
16. H. D. Trinh, L. Giupponi, and P. Dini, "Urban Anomaly Detection by processing Mobile Traffic Traces with LSTM Neural Networks," in *IEEE SECON: "Edge Computing for Cyber Physical Systems (CyberEdge) Workshop"*, Jun. 2019.
17. ——, "Mobile Traffic Prediction from Raw Data Using LSTM Networks," in *IEEE PIMRC*, Sep. 2018.
18. H. D. Trinh, E. Zeydan, L. Giupponi, and P. Dini, "Detecting Mobile Traffic Anomalies Through Physical Control Channel Fingerprinting: A Deep Semi-Supervised Approach," *IEEE Access*, vol. 7, pp. 152 187–152 201, 2019.
19. V. Ganganwar, "An overview of classification algorithms for imbalanced datasets," *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, no. 4, pp. 42–47, 2012.
20. H. Farooq, A. Asghar, and A. Imran, "Mobility Prediction Empowered Proactive Energy Saving Framework for 5G Ultra-Dense HetNets," in *IEEE GLOBECOM*, Dec. 2018.
21. E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 82–89, Aug. 2014.

22. T. Ojima and T. Fujii, "Resource management for mobile edge computing using user mobility prediction," in *IEEE ICOIN*, Jan. 2018.
23. B. Sliwa, R. Falkenberg, T. Liebig, J. Pillmann, and C. Wietfeld, "Machine Learning Based Context-Predictive Car-to-Cloud Communication Using Multi-Layer Connectivity Maps for Upcoming 5G Networks," in *IEEE VTC-Fall*, Aug. 2018.
24. H. Gebrie, H. Farooq, and A. Imran, "What Machine Learning Predictor Performs Best for Mobility Prediction in Cellular Networks?" in *IEEE International Conference on Communications Workshops (ICC Workshops)*, May 2018.
25. Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.