IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 September 2023, accepted 22 October 2023, date of publication 25 October 2023, date of current version 1 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3327523

RESEARCH ARTICLE

On the Use of Artificial Neural Networks to Predict the Quality of Wi-Fi Links

ALBERTO SALVATORE COLLETTO[1], STEFANO SCANZIO[2], (Senior Member, IEEE), GABRIELE FORMIS[1,2], (Student Member, IEEE), AND GIANLUCA CENA[2], (Senior Member, IEEE)
[1]Dipartimento di Automatica e Informatica, Politecnico di Torino, 10129 Turin, Italy
[2]National Research Council of Italy (CNR-IEIIT), 10129 Turin, Italy

Corresponding author: Stefano Scanzio (stefano.scanzio@cnr.it)

This work was supported in part by the European Union through the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART").

ABSTRACT One of the aspects that mainly characterize wireless networks is their apparent unpredictability. Although several attempts were made in the past years to define for them deterministic medium access techniques, for instance by having data exchanges scheduled by an access point, as a matter of fact they remain a partial solution and are unable to ensure the same behavior as wired infrastructures, since interference may also come from devices outside the network, which obey different rules.
A possible way to cope with disturbance on air, both internal and external to the network, is to obtain some knowledge about it by analyzing what happened in the recent past. This information, usually expressed in terms of suitable metrics, is then employed to optimize network operation, for example by prioritizing time-sensitive traffic when needed. In the simplest approaches such metrics coincide with statistical indices evaluated on transmission outcomes, like the failure rate.
In this paper we analyze a more sophisticated solution that relies on machine learning, and in particular on artificial neural networks, to predict the behavior of a Wi-Fi link in terms of its frame delivery ratio. Results confirm that more accurate predictions than simpler methods (e.g., moving average) are possible, even when training is partially independent from the specific conditions experienced on the different channels.

INDEX TERMS Channel quality prediction, wireless networks, dependable wireless communication, IEEE 802.11, artificial neural networks, machine learning.

I. INTRODUCTION

Wireless networks are progressively replacing cables in a variety of contexts. In application fields like, e.g., home and car entertainment [1], [2], home automation [3], building automation [4], and sensing [5], their adoption to support wire-free communication (and hence, interaction) among humans and machines, in any combination (H2H, M2M, and H2M), is a popular ongoing trend and one of the main enablers of the so-called Internet of Things (IoT) [6].

Besides throughput, also the requirements about latency and dependability may sometimes be quite demanding. This is particularly true in contexts like modern industrial environments (both factory and process automation), which are being re-shaped by recent paradigms like Industry 4.0 [7], Industry 5.0 [8], and the Industrial Internet of Things (IIoT) [9]. In particular, upper bounds on response times are customarily defined in soft/firm real-time systems, which in wireless networks are often specified in probabilistic terms. For example, a given fraction of time-sensitive messages (e.g., 99.99%) is required to not exceed the intended deadlines, otherwise the controlled system is not ensured to operate correctly.

To tackle such a wide range of requirements, wireless networking is currently characterized by a noticeable heterogeneity of protocols [10]. Among the most relevant technologies, which suit different scenarios, we find: 5G/6G [11] and IEEE 802.11 (Wi-Fi) [12] that apply to contexts

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Feng.

where high throughput is demanded on geographic and local areas, respectively, LoRaWAN for long-range low-rate data exchanges [13], IEEE 802.15.4 (e.g., ZigBee, WirelessHART, WIA-PA, 6TiSCH, ISA 100.11) for ultra low-power mesh networks [14], and Bluetooth Low Energy (BLE) to connect both personal and industrial devices [15].

In the past years, a fair amount of research activities were spent for improving the quality of communication on wireless networks by exploiting techniques like seamless redundancy [16], transmission scheduling [17], software-defined networking [18], automatic network configuration [19], and so on. Some solutions rely on enhanced medium access control (MAC) mechanisms that enable deterministic behavior. Besides legacy solutions, like the Point Coordination Function (PCF) and the Hybrid Coordination Function (HCF) Controlled Channel Access (HCCA) in IEEE 802.11 (which knew limited usage), the use of trigger frames in Wi-Fi 6 and multi-link operation (MLO) in Wi-Fi 7 [20] achieve tangible improvements. The same holds for Time Slotted Channel Hopping (TSCH) [21], [22] and the Deterministic and Synchronous Multi-channel Extension (DSME), which exploit mixed time-frequency diversity in IEEE 802.15.4.

It is important to point out that, although effective, deterministic MACs are unable to ensure for wireless networks the same dependability as the wired ones. In fact, besides intra-network interference, transmission on air may also suffer from extra-network disturbance, caused by nearby wireless networks (possibly based on different transmission technologies) and electromagnetic noise (generated by industrial/power equipment). In the latter cases, there are simply no means to prevent interference.

A different approach to cope with above phenomena, without necessarily changing the MAC mechanism, consists in applying machine learning (ML) to wireless communications. In the recent years, a number of research activities were started targeted at exploiting intelligence to improve communication quality. Some proposals rely on suitable algorithms to predict the future quality of a link in terms of metrics like, e.g., the frame delivery ratio (FDR). The ability to foresee variations of the communication quality can be then exploited by end nodes and intermediate equipment to proactively react to events that affect the wireless spectrum (either worsening or improving it), in an attempt to provide some probabilistic guarantees for, e.g., the typical key performance indicators (KPIs) relevant to industrial communication networks, namely, end-to-end latency, dependability, determinism, and even power consumption [23].

For example, if the prediction model foresees that channel conditions are likely to deteriorate in a while, applications could react in advance by reducing the amount of generated best-effort traffic (e.g., by lowering the sampling rate at the perception layer for non-critical information, like environmental monitoring), so as to privilege higher-priority traffic with soft/firm real-time requirements. As an alternative, communication could be switched to a different channel before the current one worsens too much. This solution may bring tangible advantages also for hand-over procedures of mobile nodes: ML predictions can be exploited to determine the instant when a roaming Client Station (STA) needs to reassociate to an different Access Point (AP) when it moves away from the AP it is currently associated. In this case, the quality of every available channel must be predicted, and this requires that all of them are probed, e.g., using reinforcement learning (RL) techniques based on exploration and exploitation. This resembles the operation of the Minstrel algorithm, customarily implemented in commercial Wi-Fi equipment, and rate adaptation algorithms [24].

Similar approaches can be adopted when redundancy is exploited to increase reliability and decrease communication latency, by allowing frames to be sent on different channels (MLO). In this case, ML can be exploited to drive channel selection. They also apply to seamless redundancy, when the same frame is sent on multiple channels at the same time. In particular, deferral techniques [25] could use ML to determine the primary channel on which the first copy of the frame is sent. Transmission of the second copy is deferred for a while to achieve the best trade-off between reliability and resource consumption. The last, more complex example, among many possible ones, concerns the combined adoption of seamless redundancy and ML to support node mobility, for preserving communication quality during hand-over [26].

In this work, artificial neural networks (ANNs) are exploited to predict the FDR in the near future starting from the outcomes of transmission attempts logged by a wireless node in the recent past. Our analysis is based on extensive experimental campaigns that involved real Wi-Fi devices. Results show that the quality of a wireless channel in the near future (a few minutes) can be predicted with satisfactory accuracy.

The structure of the paper is the following: Section II summarizes the state-of-the-art about the use of artificial intelligence (AI) to improve the behavior of wireless networks; Section III presents a simple model for a wireless link, while prediction models are introduced in Section IV; Section V describes the experimental testbed we used to acquire the databases that characterize the behavior of real wireless links and the software implementations related to this work; Section VI outlines the results obtained by applying the different prediction models to real data; finally, Section VIII draws relevant conclusions and outlines future work.

II. LITERATURE REVIEW

The two surveys [27] and [28] identify many challenges intrinsic to the use of ML and AI techniques to improve key performance indicators related to wireless networks. In [29] and [30] ML techniques are used for the selection of the best AP, whereas in [31] and [32] they are exploited to drive the handover decision between APs. Other works are based on traffic prediction [33], [34]. This information can be used to indirectly infer the quality of the wireless channel:

for example, in [35] it is used to select channel allocation. Although this is an interesting research direction, there is not a linear dependency between traffic and FDR. Finally, reinforcement learning was used to select the best channel for transmission in [36], where the reduction of packet losses was assessed through simulation.

Regarding the subset of works in the scientific literature related explicitly to Wi-Fi and the prediction of the future behavior of the channel, the use of ANNs was envisaged in [37] but only on artificial data. In [38], ML techniques were used for predicting the signal strength. Instead, in [39] and [40] models based on ANNs are described aimed at predicting the channel gain in specific application contexts. All of the above works refer to the prediction of aspects related to the physical layer. Generally speaking, they cannot be directly and easily employed to assess the KPIs that customarily characterize communication at the application level, such as the FDR.

In [41] and [42], ANNs were applied in a preliminary form to data acquired on a real Wi-Fi testbed, using the FDR as a metric. Compared to the research activities presented in [41] and [42], the analysis in this work was done: 1) with a noticeably more complex software, suitable for efficiently handling big data; 2) with much larger databases in terms of their size, especially for what concern testing (93.5 days for training and 53.8 days for test instead of 38 days for training and 2.5 days for test), hence providing more reliable results; 3) analyzing four Wi-Fi channels (1, 5, 9, and 13) instead of just channel 13, hence providing better significance; 4) including potentially interesting architectures like multi-output ANNs, which will be presented in Subsection VI-D and are aimed to lower training time and memory occupation in the end-user device; and, 5) considering the effects on FDR prediction accuracy achieved by extending the training database with data acquired on different channels (to this purpose, relevant experiments were carried out where training relied on channels other than the one used for testing).

III. CHANNEL AND PROTOCOL MODEL

All the experimental campaigns reported in this work specifically concern Wi-Fi. Nevertheless, this kind of analysis can be easily extended to any wireless communication technologies that support confirmed transmission services. The simple network model we consider includes two nodes communicating over a wireless link. In particular, the sender node repeatedly sends data frames to a recipient node at a fixed rate. When a frame is correctly received, the latter replies by returning an acknowledgment (ACK) frame to the sender to notify that the transmission succeeded. How this mechanism was actually implemented on commercial Wi-Fi devices (e.g., by disabling retransmission, fixing the transmission speed, etc.) is described in detail in Section V.

Data frame transmission is performed cyclically with period $T_s = 0.5$ s, and the reception of every ACK frame is logged. Possible outcomes are *success* ($x_i = 1$) if the ACK frame associated with the i-th data frame is correctly

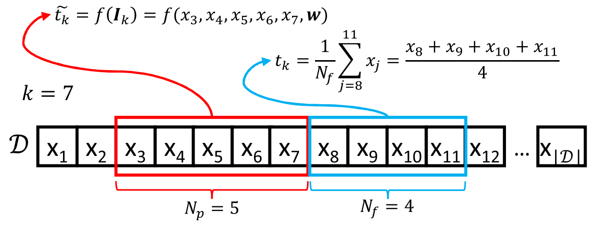
$\tilde{t}_k = f(\boldsymbol{I}_k) = f(x_3, x_4, x_5, x_6, x_7, \boldsymbol{w})$
$t_k = \frac{1}{N_f}\sum_{j=8}^{11} x_j = \frac{x_8 + x_9 + x_{10} + x_{11}}{4}$
$k = 7$
$\mathcal{D}$
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12} … $x_{|\mathcal{D}|}$
$N_p = 5$
$N_f = 4$

FIGURE 1. Example of past and future sliding windows, on which the prediction function $f(\cdot)$ is applied and the target t_k is computed, respectively.

received by the sender node; otherwise *failure* ($x_i = 0$) in those cases when either the data frame or the ACK frame are corrupted and the transmission timeout expires. This behavior is intentional, and mimics the point of view of the application executing in the sender node, which only indirectly has a way to know about transmission errors. Generally speaking, this provides an ordered sequence of outcomes $\mathcal{D} = (x_1, \ldots, x_i, \ldots, x_{|\mathcal{D}|})$ as output, we call *database*, whose size (in terms of the number of elements) is denoted $|\mathcal{D}|$. The database is depicted schematically in Fig. 1, along with some quantities whose definition is provided below.

The goal of this research work is to find a *prediction function* $f(\cdot)$ that, at any time, estimates the value of the FDR evaluated over a given future horizon (one to ten minutes) given the previous N_p transmission outcomes as input. Let x_k be the outcome of the most recent transmission attempt. Then, the input of the prediction function is the sequence $\boldsymbol{I}_k = (x_{k-N_p+1}, \ldots, x_{k-1}, x_k)$, and the predicted FDR can be expressed as

$$\tilde{t}_k = f(\boldsymbol{I}_k, \boldsymbol{w}) = f(x_{k-N_p+1}, \ldots, x_{k-1}, x_k, \boldsymbol{w}), \qquad (1)$$

where $\boldsymbol{w}$ (a vector, in the most general case) describes the model parameters determined in the training phase (it was explicitly added to the prediction function to stress the fact that it directly affects the related output $\tilde{t}_k$).

During the training phase a specific database $\mathcal{D}_{\mathrm{tr}}^{ch}$ is employed, where *ch* represents the channel on which it was acquired (e.g., $\mathcal{D}_{\mathrm{tr}}^{5}$ represents the training database obtained on Wi-Fi channel 5). The training phase consists in estimating the model parameters $\boldsymbol{w}^*$ that minimize the average error between the predicted FDR value $\tilde{t}_k$ and the target t_k, which characterizes the real failure rate in the future. The mean squared error is typically used for this purpose, which is the same as considering the sum $J(\boldsymbol{w})$ of squared errors:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} J(\boldsymbol{w}), \qquad (2)$$

where

$$J(\boldsymbol{w}) = \sum_{k=N_p}^{|\mathcal{D}_{\mathrm{tr}}|-N_f} [t_k - f(\boldsymbol{I}_k, \boldsymbol{w})]^2. \qquad (3)$$

The target coincides with the FDR observed in a future interval whose width is $T_f = N_f \cdot T_s$. At any time, it is computed as the simple moving average (SMA) of the sequence $(x_{k+1}, x_{k+2}, \ldots, x_{k+N_\mathrm{f}})$, which includes N_f future outcomes that belong to database $\mathcal{D}$,

$$t_k = \frac{1}{N_\mathrm{f}} \sum_{j=k+1}^{k+N_\mathrm{f}} x_j. \quad (4)$$

The value t_k given by (4) provides a statistical estimate of the probability that a frame transmission attempt on the testbed succeeds, which is what the prediction function should in theory seek. It unavoidably suffers from a certain error, as it is computed using a limited number N_f of outcomes. In the case of quasi-stationary spectrum conditions, transmission attempts can be approximately modeled as independent and identically distributed (iid) random variables, and the mean squared error affecting the estimate t_k is equal to σ_x^2/N_f, where σ_x^2 represents the variance of samples (x_k) in the database. This makes the problem considered here more complex than usual time series analysis, where the true target to be predicted is directly available.

To evaluate the accuracy of the prediction function $f(\boldsymbol{I}_k, \boldsymbol{w})$ a test database $\mathcal{D}_\mathrm{te}^{ch}$ is employed. The same procedure described above is applied to $\mathcal{D}_\mathrm{te}^{ch}$ to obtain the target values t_k. Then, the prediction function is applied to every input interval $\boldsymbol{I}_k$ for evaluating its ability to estimate the target. Two kinds of prediction errors were defined in this work: the *absolute error* $|e_k| = |t_k - \tilde{t}_k|$, where $|\cdot|$ is the modulus operator, and the *squared error* $e_k^2 = (t_k - \tilde{t}_k)^2$. Starting from them, the *mean absolute error* (MAE) $\mu_{|e|}$ and the *mean squared error* (MSE) μ_{e^2} can be computed as $\frac{1}{|\mathcal{D}_{te}|-N_\mathrm{f}-N_\mathrm{p}+1} \sum_{k=N_\mathrm{p}}^{|\mathcal{D}_{te}|-N_\mathrm{f}} e_k$, where e_k stands for $|e_k|$ and e_k^2, respectively. Besides averages, other statistical indices can be also evaluated for prediction errors, like percentiles. For example, $|e|_{\mathrm{p}_{90}}$ represents the 90-percentile of the absolute error.

IV. PREDICTION MODELS

The prediction models we took into account for estimating the FDR on a wireless link in the immediate future are the *simple moving average* (SMA), which is used as a reference for the other proposed techniques, and *artificial neural networks* (ANNs).

A. SIMPLE MOVING AVERAGE (SMA)

The main assumption behind SMA prediction is that, the FDR in the immediate future remains (almost) the same as in the immediate past. The prediction function mostly resembles (4), but applies to the past samples

$$f^{\mathrm{SMA}}(\boldsymbol{I}_k, N_\mathrm{p}) = \frac{1}{N_\mathrm{p}} \sum_{j=k-N_\mathrm{p}+1}^{k} x_j, \quad (5)$$

where N_p is the number of outcomes exploited for prediction. The N_p value is explicitly included in the prediction function to remark that model accuracy heavily depends on it.

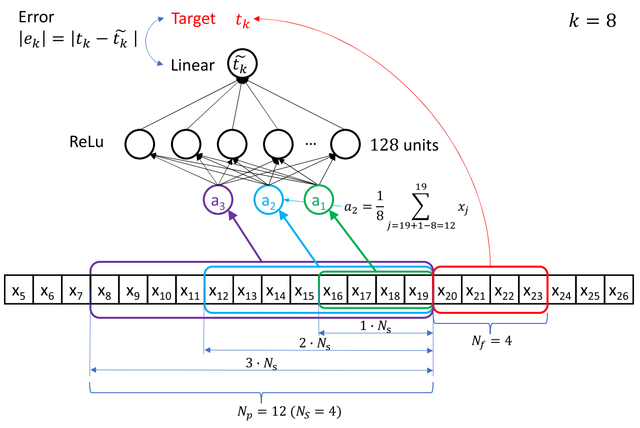
Error
$|e_k| = |t_k - \tilde{t}_k|$
Target t_k
Linear $\tilde{t}_k$
$k = 8$
ReLu
128 units
a_3
a_2
a_1
$a_2 = \frac{1}{8} \sum_{j=19+1-8=12}^{19} x_j$
x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12} x_{13} x_{14} x_{15} x_{16} x_{17} x_{18} x_{19} x_{20} x_{21} x_{22} x_{23} x_{24} x_{25} x_{26}
$1 \cdot N_s$
$2 \cdot N_s$
$3 \cdot N_s$
$N_f = 4$
$N_p = 12\ (N_s = 4)$

FIGURE 2. Front-end module operations and structure of the ANN.

Accuracy of SMA prediction can be optimized by exploiting the training database $\mathcal{D}_\mathrm{tr}$. In particular, the optimal value N_p^* can be found as

$$N_\mathrm{p}^* = \arg\min_{N_\mathrm{p}} \sum_{k=N_\mathrm{p}}^{|\mathcal{D}_\mathrm{tr}|-N_\mathrm{f}} \left(t_k - f^{\mathrm{SMA}}(\boldsymbol{I}_k, N_\mathrm{p})\right)^2. \quad (6)$$

Practically, N_p is varied over a range of values wide enough to find the minimum of the error function. The N_p^* value determined in the training phase is then used to parameterize the prediction function (5) when the test database $\mathcal{D}_\mathrm{te}$ is used.

B. ARTIFICIAL NEURAL NETWORK (ANN)

The ANN model of this work is the *multi layer perceptron* (MLP). A second model was also briefly investigated, namely *long short-term memory* (LSTM), which is customarily employed for time series forecasting. Results about LSTM were not reported because, for the specific task of predicting channel quality, a conventional MLP fed with a sufficiently large number of past samples, as the one presented in this work, provides a better prediction accuracy.

A front-end module was implemented to carry out some preliminary elaboration on the input features. Many types of transformation were analyzed and tested, and the one that provided the best results consists of a front-end function $g(\cdot)$ that, given the past samples $\boldsymbol{I}_k$, produces a sequence

$$g(\boldsymbol{I}_k) = \left(a_i = \frac{1}{i \cdot N_\mathrm{s}} \sum_{j=k-i\cdot N_\mathrm{s}+1}^{k} x_j\right)_{i\in 1,\ldots,\frac{N_\mathrm{p}}{N_\mathrm{s}}}, \quad (7)$$

where $i \cdot N_\mathrm{s}$ is the number of outcomes used for computing ANN input a_i and $N_\mathrm{p}/N_\mathrm{s}$ represents the number of such inputs. This sequence is then fed to the ANN.

As highlighted in the lower part of Fig. 2, every ANN feature corresponds to the arithmetic mean of the most recent outcomes of $\boldsymbol{I}_k$, evaluated on sequences (intervals) of increasing width. In particular, each such interval includes a number of samples that is a multiple of the constant N_s. As a consequence, the function $f^{\mathrm{ANN}}(\boldsymbol{I}_k, \boldsymbol{w})$ can be split into two components

$$f^{\mathrm{ANN}}(\boldsymbol{I}_k, \boldsymbol{w}) = h^{\mathrm{ANN}}\left(g(\boldsymbol{I}_k), \boldsymbol{w}\right), \tag{8}$$

where $g(\boldsymbol{I}_k)$ is the front-end module given by (7) and $h^{\mathrm{ANN}}(\cdot, \boldsymbol{w})$ is a function that implements only the operations related to ANN (upper part of Fig. 2).

To determine the optimal values for the parameters of the ANN prediction model, the following optimization process is used

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \sum_{k=N_\mathrm{p}}^{|\mathcal{D}_\mathrm{tr}|-N_\mathrm{f}} \left(t_k - f^{\mathrm{ANN}}(\boldsymbol{I}_k, \boldsymbol{w})\right)^2, \tag{9}$$

where $\boldsymbol{w}$ is the sets of weights and other quantities that characterize the model (e.g., biases). Again, model parameter estimation is assessed on the training database $\mathcal{D}_\mathrm{tr}$.

Using an ANN rather than existing statistical approaches like SMA has the potential to improve prediction accuracy, since it tries to infer a trend from the past and not just a single value. Linear and polynomial interpolation techniques were also preliminarily analyzed, but their performance was found to be quite poor, often worse than SMA.

By applying a linear transformation (similar to the one carried out by the ANN input layer) to the input features computed by the front-end module as per (7), evenly spaced FDR estimates can be easily obtained that are evaluated on adjacent intervals whose width is $N_\mathrm{s} \cdot T_\mathrm{s}$ (one minute in our case). These values accurately describe the trend of the channel quality in the recent past. Therefore, the ANN can be seen as sort of a nonlinear finite input response (FIR) filter applied to the FDR, which is potentially able to cope with dynamically changing spectrum conditions.

For the above reasons, an ANN suitably trained on real data could capture some properties of Wi-Fi transmission that known theoretical models fail to describe properly. This is exactly what our work is meant to determine.

V. TESTBED

The first part of our work, which lasted many months, consisted in the experimental characterization of the wireless spectrum as seen by real Wi-Fi equipment. Starting from the outcomes of this part, several prediction models were then created and their accuracy tested.

Investigation relied on several components. In the following subsection, the hardware and software we developed for the acquisition of the relevant databases are described. Then, the architecture of the ANN and all details about its training are discussed in Subsection V-B, while Subsection V-C illustrates the software we implemented to support training starting from the huge amount of experimental data.

A. DATABASE ACQUISITION

The acquisition of databases relied on an experimental setup made up of two Linux PCs, each one equipped with two Wi-Fi adapters of type TP-Link TL-WDN4800 that comply with IEEE 802.11n. Overall, there were four Wi-Fi STAs, each of which associated with a distinct AP located about 3÷4 meters apart. In this experimental analysis we focused on the four "canonical" channels 1, 5, 9, and 13 in the 2.4 GHz band. They are spaced wide enough to prevent any adjacent channel interference (ACI) [43] effects, and every pair STA/AP was configured to operate on one of these channels.

TABLE 1. Length of training ($\mathcal{D}_\mathrm{tr}$) and test ($\mathcal{D}_\mathrm{te}$) databases (days).

Channel	Training ($\mathcal{D}_\mathrm{tr}$)	Test ($\mathcal{D}_\mathrm{te}$)	Total
ch1	21.2	12.8	34.0
ch5	21.9	10.6	32.5
ch9	24.9	15.2	40.1
ch13	25.5	15.2	40.7
All	93.5	53.8	**147.3**

Frame transmissions (frame size was set to 50 B) were performed almost simultaneously by the four STAs, and were triggered by the two Linux operating systems, which were time-synchronized through the network time protocol (NTP) and installed with the RT-Preempt Linux patch [44], [45] to improve soft real-time capabilities.

The main goal of the testbed is to sample the channels' conditions periodically, causing a negligible perturbation of the related spectrum. To this purpose the device driver of the STAs was modified in order to: 1) set transmissions to a fixed bit rate of 54 Mb/s (consequently disabling the operations of the Minstrel algorithm, which automatically optimizes the transmission speed); 2) disable automatic frame retransmissions (every frame is sent only once); 3) disable the backoff procedure (frames are sent immediately when the channel is idle); 4) disable the request to send/clear to send (RTS/CTS) mechanism; 5) disable some specific features of the IEEE 802.11n version of the standard like frame aggregation, by downgrading adapters' operation to IEEE 802.11g.

Enforcing this behavior is possible thanks to the use of the `ath9k` device driver along with the SDMAC framework [18], [46], which allows transferring some relevant information related to the transmission of a frame from kernel space, where the driver executes, to the application devoted to database acquisition, which runs in user space. Every time the ACK frame associated with a data frame arrives at the sender or the relevant timeout expires, the outcome x_i is conveyed by the driver to the application, which logs it.

The testbed was employed to characterize the wireless spectrum in the 2.4 GHz band in our lab, which is shared by a few tens of Wi-Fi networks (also including some wireless sensor networks) and the related nodes. The substantial number of active nearby mobiles and notebooks exchanging data over the air, which varies over time as researchers and students keep entering and leaving facilities during the day, makes the spectrum conditions not stationary at all.

From the overall logs, the training and test databases were extracted with no specific criteria (we are seeking for results of general validity). Fig. 3 includes 8 timing diagrams that

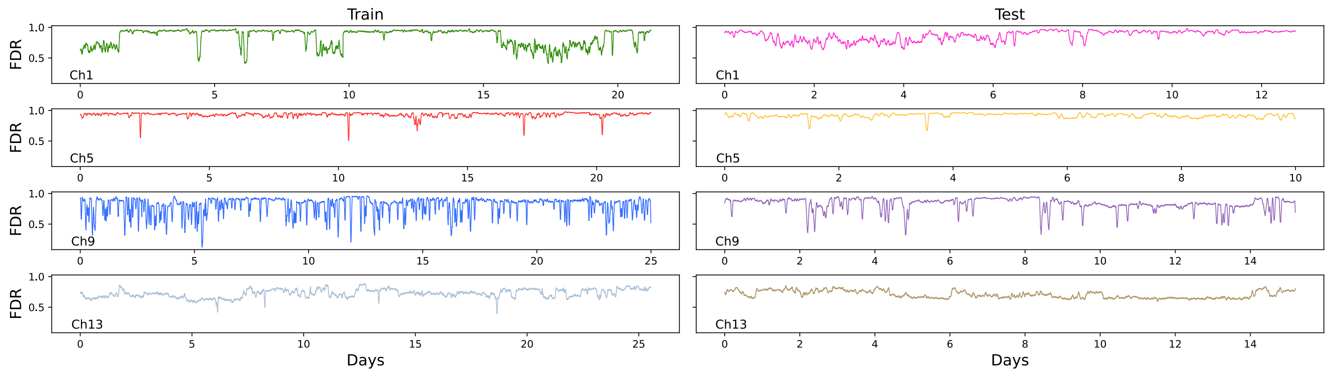

FIGURE 3. Timing diagrams about the measured FDR for the training (on the left) and test (on the right) databases on channels 1, 5, 9, and 13 (top to bottom).

report the FDR (that is, the target evaluated on 30 minutes) for these databases. In particular, the column on the left shows training databases, while the column on the right refers to test databases. Rows identify channels 1, 5, 9, and 13, respectively. As can be seen, the overall amount of interference differs tangibly among the considered channels, and the same holds for the related interference patterns.

Table 1 summarizes the length (in days) of databases. The training database for any channel spanned over at least three weeks, while test databases lasted ten days or more. The total amount of data used for training and test embraced 93.5 and 53.8 days, respectively. Overall, all channels included, databases covered 147.3 days, corresponding to about 5 months.

B. ANN ARCHITECTURE

The topology of the ANN model is reported in Fig. 2. Regarding the front-end module, we set $N_{\mathrm{p}} = 14400$ and $N_{\mathrm{s}} = 120$, and hence the input layer of the ANN consisted of $N_{\mathrm{p}}/N_{\mathrm{s}} = 120$ inputs. A number of additional experiments were performed, like those for finding N_{p}^* in SMA, using larger N_{p} values, but we did not observe any improvements concerning accuracy. Conversely, using smaller values led to a slight worsening. An important property of ANNs is that, a specific optimization like (6) is not required, because the ANN automatically trains the model parameters to weight less those inputs that have a lower influence on the prediction.

The hidden layer of the neural networks is composed of 128 neurons of type *ReLu*, and there is a single *linear* output neuron that provides the prediction $\tilde{t}_k$. Training consists of 15 epochs, the *momentum* was set to 0, and the *learning rate* was initialized to 0.01, halving at each epoch (i.e., 0.01, 0.005, 0.0025, ...). In every epoch the patterns used to train the model are reordered randomly. The *batch size* was set to 64, which means that model weights are updated every 64 input patterns.

The progress of the training procedure is sketched in Fig. 4, where the loss (i.e., the MSE) was reported with respect to the training epoch for channel 1 and with different values

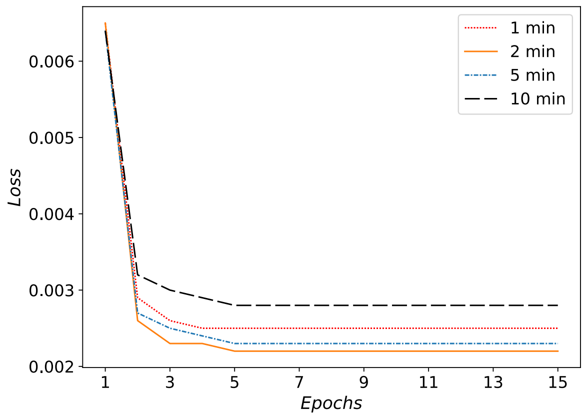

FIGURE 4. Loss vs. training epochs for channel 1 and T_{f} equal to 1, 2, 5, and 10 min.

of T_{f}. As can be seen, the most part of the loss decrease takes place within the first 5 epochs; then, losses converge asymptotically to fixed values. We decided to extend the training up to 15 epochs because, although the improvements are minimal, the obtained model is slightly better. Moreover, by halving the learning rate at each epoch more stable results are achieved, since in the final epoch very small adjustments are brought to weights (and biases). The behavior for the other three channels is quite similar. Finally, the loss computed on the test database (by saving the ANN model at the end of each epoch) showed a rather similar evolution, except that it converges asymptotically to different loss values.

To obtain more reliable values of the ANN prediction accuracy, any given configuration in the experiments below (corresponding, e.g., to single rows in Table 3) was evaluated five times. Every time, the ANN was trained from scratch and then tested, leaving databases unchanged. Reported results have been obtained by evaluating the relevant statistics on the concatenation of all the prediction errors obtained from tests, for the five separate repetitions of every specific configuration.

C. TRAINING SOFTWARE

The training and test software was written in `python` and makes use of the `Keras` module, which is included in `tensorflow`. The ANN model weights are randomly initialized at the beginning of the training with the `Glorot normal` initializer, and the `SGD` optimizer was used for weights update.

Due to the huge dimension of the training database, a specific software was developed to manage such "big data". The main problem stems from the fact that every outcome in the input database involves the generation of a pattern composed of $N_\mathrm{p}/N_\mathrm{s} = 120$ inputs and one target. In addition, at each epoch such patterns must be supplied randomly to the ANN for training.

Loading all patterns in memory simultaneously is not feasible due to memory limits, and randomly loading them from the hard disk is too time-consuming. Therefore, we specifically developed a suitable software that splits the training database into groups that include N_m patterns at most. At the beginning of every epoch, a random number is assigned to each group, which permits to identify the order in which they are selected. Following this order, the program loads N_g groups for a total of $N_\mathrm{g} \cdot N_\mathrm{m}$ patterns into memory, shuffles them, and executes the training process. Then, the same process is repeated using another set of N_g groups. When all groups are trained the epoch finishes, and the process restarts by assigning new random numbers to each group. The size N_m is a compromise between speedup and randomness of the patterns provided to the ANN (a higher value of N_m makes operations faster). Instead, N_g depends on the available memory of the PC used for training, and must be maximized. Increasing the number N_g of groups loaded contextually into memory increases the randomness of the patterns provided to the ANN, because groups are randomly selected over the entire training database. The values used in this paper are $N_\mathrm{m} = 100000$ and $N_\mathrm{g} = 10$.

VI. RESULTS

Starting from the training and test databases ($\mathcal{D}_\mathrm{tr}^{ch}$ and $\mathcal{D}_\mathrm{te}^{ch}$) we obtained from the above experimental testbed on every Wi-Fi channel $ch \in \{1, 5, 9, 13\}$, a number of campaigns were carried out to analyze how much aspects like the prediction model (plain SMA heuristic vs. ANN), the ANN architecture (single output vs. multiple output), and the kind of training (specific channel-dependent vs. generalized channel-independent) impact on accuracy. Doing so makes results comparable.

A. SPECIFICALLY PARAMETERIZED SMA

In the initial campaign a simple moving average was used for prediction. This is a very basic approach which assumes that, from a probabilistic viewpoint, behavior in the near future is the same as the recent past. Since we wished to make a fair comparison with ANNs, the width of the moving window on which the average is evaluated was not defined once and for all, but was instead specifically set for every given channel and for every given horizon based on a preliminary training phase according to (6).

The four plots in Fig. 5 show the SMA prediction error (MSE) on channels 1, 5, 9, and 13 for two different N_f values (5 and 10 min) when N_p is varied. As can be seen, a minimum can be always singled out clearly, whose position provides the optimal value N_p^* (for the considered conditions, it lies in the range from 120 to 1000). This can be explained by considering the two main phenomena that contribute to the prediction error:

1) The spectrum is non-stationary, hence knowing the past is not enough to characterize the future precisely. The farther the horizon, and the deeper the past on which the SMA is computed, the more the behavior of the wireless link may change in the meanwhile, also depending on the specific spectrum dynamics. This contribution *increases* as T_p and T_f grow.
2) The target we used for training (FDR) is obtained by averaging the outcomes of N_f transmissions occurring in the future. The lower the number of samples, the higher the intrinsic variability affecting the estimation of the related success probability. The very same holds for the past interval, which includes N_p outcomes. These contributions *decrease* as T_p and T_f grow.

It is worth noting that the above reasoning mostly applies also to the case where ANNs are used for prediction.

Results are reported in Table 2, which consists of four parts, each one referring to a specific channel. Every such part is split in four rows, which refer to different horizons T_f equal to 1, 2, 5, and 10 min, respectively. For every row several metrics about prediction accuracy are reported, evaluated on the related test database $\mathcal{D}_\mathrm{te}^{ch}$. Besides the mean squared error (MSE), the absolute error is also considered, and in particular its mean value (MAE), as well as its 90 and 95 percentiles. On the rightmost column, parameter N_p^* is included.

In the following, prediction accuracy refers to the MAE, unless otherwise stated. For the reasons described above, it depends on both the considered channel and the horizon T_f on which the target FDR is evaluated. By looking at the table, one can see that the best accuracy ($\mu_{|e|} = 1.91\%$) is achieved for channel 5 when the future horizon spans over 5 minutes. Behavior of channel 9 is the hardest to predict: in this case the best estimates are obtained for a future interval of 2 minutes, and when the horizon is enlarged to 10 minutes the absolute error grows and exceeds 4.4%.

This can be explained by looking at the time diagrams in Fig. 3. As can be seen, the failure rate on channel 5 is quite low, whereas it is sensibly worse on channel 9. Non-negligible interference also affects channels 1 and 13, but in these cases the failure patterns are more "regular" and hence they can be predicted to a better extent. Conversely, channel 9 is characterized by faster spectrum dynamics, with rapidly changing interference.

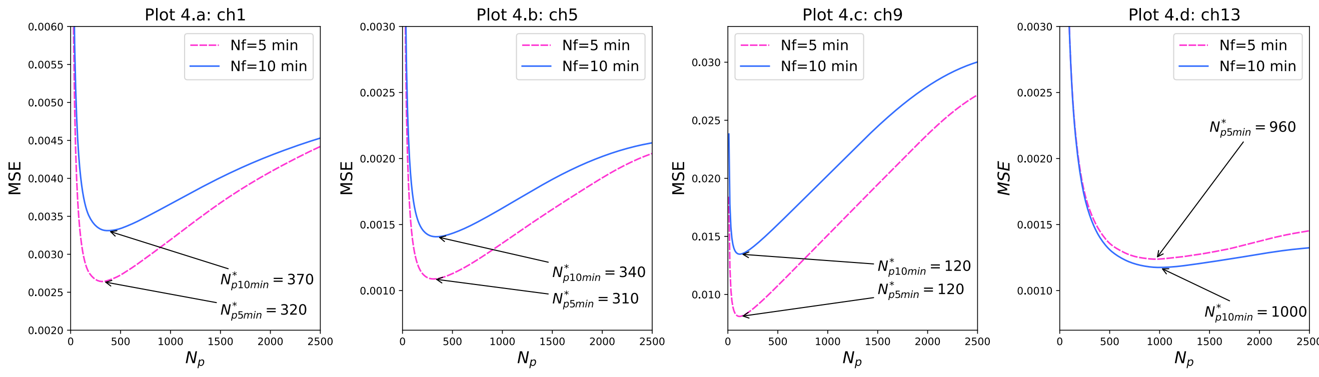

FIGURE 5. SMA training: prediction error (MSE) vs. past interval width N_p for channels 1, 5, 9, and 13 (optimal N^*_p values are highlighted).

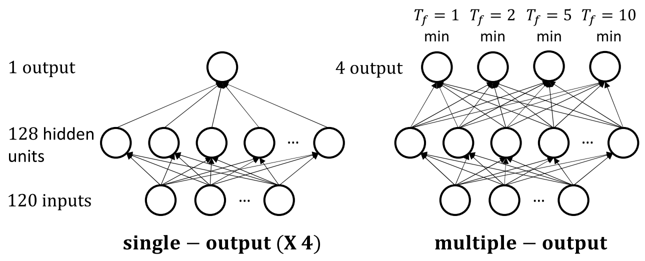

FIGURE 6. Single-output and multi-output ANN architectures: predictions are made for a single vs. multiple (four) time horizons.

B. SPECIFICALLY TRAINED ANN

In this campaign a distinct ANN was considered for every channel and for every horizon. As for the SMA in the previous section, the databases used for training and test refer to the same channel (*channel-dependent* prediction model), which implies that they are strongly correlated. In fact, although the interference pattern observed by the testbed on any channel is likely to vary over time, this implying that it differs in the training and test phases (displaced by several weeks), it depends on the same set of nearby APs (deployed in fixed positions) and similar sets of STAs. Intuitively, doing so should provide the best prediction accuracy, since every ANN is optimally trained using data acquired in conditions that mostly resemble those it will encounter when used for prediction. For this reason the results reported here constitute sort of a best case for ANNs, against which the outcomes in the following sections have to be checked.

Results are shown in Table 3, whose structure resembles the one used for SMA. The rightmost column reports the winning ratio W, that specifies how many times the ANN provided a better accuracy than SMA. However, this quantity is not particularly interesting, because it does not consider how much the prediction of the loser is actually worse. From the point of view of the MAE, the best accuracy is obtained for $T_\mathrm{f} = 2\,\mathrm{min}$ on channels 1 and 9, whereas on channel 5 it is seemingly found when T_f approaches 10 min ($\mu_{|e|} = 1.70\%$). Finally, on channel 13 error is minimal when T_f exceeds 10 min. Since the variance of the target depends on the number of samples on which FDR is evaluated (and hence, on T_f), this behavior can be explained by the faster dynamics of channels 1 and 9, which make predictions over larger horizons worse.

TABLE 2. Prediction accuracy by using 16 SMAs where the window width is specifically optimized for channels 1, 5, 9, and 13 (channel-dependent prediction model) and future horizons 1, 2, 5, and 10 minutes.

Channel	T_f [min]	μ_{e^2} $[\cdot 10^{-3}]$	$\mu_{\lvert e\rvert}$	$\lvert e\rvert_{\mathrm{p}90}$ [%]	$\lvert e\rvert_{\mathrm{p}95}$	N^*_p
ch1	1	2.064	3.31	7.08	9.16	240
	2	**1.789**	**2.95**	**6.45**	**8.83**	290
	5	2.059	3.03	7.04	10.00	320
	10	2.794	3.49	8.85	12.06	370
ch5	1	1.115	2.51	5.21	6.45	310
	2	0.842	2.11	4.36	5.43	310
	5	**0.793**	**1.91**	**3.95**	**5.02**	310
	10	0.976	1.97	4.09	5.46	340
ch9	1	2.993	3.83	7.82	9.67	130
	2	**2.931**	**3.52**	**6.98**	**8.81**	130
	5	4.072	3.76	7.16	9.66	120
	10	6.291	4.43	8.16	14.99	120
ch13	1	2.379	3.86	7.98	9.60	970
	2	1.555	3.09	6.40	7.76	950
	5	1.107	2.56	5.33	6.60	960
	10	**1.029**	**2.43**	**5.09**	**6.43**	1000

Note: boldface denotes best cases.

By comparing prediction accuracy on the different channels, we can observe that the error on channel 5 is generally small, whereas it is quite large on channel 9. ANN always managed to outperform SMA. This is a relevant result, and implies that ML, even in its simplest form, goes beyond the simple assumption that the future resembles the past. In particular, it proves to be able to model non-trivial hidden aspects of the spectrum in the presence of Wi-Fi traffic, which leads to better predictions.

C. MULTI-TARGET ANN

In this campaign, we used a single ANN with four outputs for every channel (*channel-dependent* prediction model) to perform contextual FDR predictions for all time horizons. Changes with respect to the previous campaign are purely

TABLE 3. **Prediction accuracy by using 16 single-output ANNs specifically trained on channels 1, 5, 9, and 13 (channel-dependent prediction model) and future horizons 1, 2, 5, and 10 minutes.**

Channel	T_f [min]	μ_{e^2} [$\cdot 10^{-3}$]	$\mu_{\lvert e \rvert}$	$\lvert e \rvert_{\mathrm{p}90}$ [%]	$\lvert e \rvert_{\mathrm{p}95}$	W [%]
ch1	1	1.973	3.22	6.96	9.23	54.2
	2	**1.670**	**2.86**	**6.32**	**8.65**	54.4
	5	1.880	2.92	6.95	9.64	55.4
	10	2.254	3.19	8.00	10.71	57.7
ch5	1	1.041	2.40	4.94	6.14	55.7
	2	0.768	1.97	4.04	5.10	56.8
	5	**0.664**	1.71	3.51	**4.53**	57.3
	10	0.759	**1.70**	**3.43**	4.62	59.1
ch9	1	2.662	3.51	7.03	8.86	58.6
	2	**2.514**	**3.15**	6.12	**8.01**	59.3
	5	3.184	3.27	**6.03**	9.25	58.2
	10	4.434	3.95	7.18	12.23	53.3
ch13	1	2.243	3.76	7.77	9.31	53.7
	2	1.416	2.96	6.12	7.40	54.3
	5	0.941	2.38	4.93	6.04	55.6
	10	**0.813**	**2.15**	**4.54**	**5.68**	59.4

Note: boldface denotes best cases.

TABLE 4. **Prediction accuracy by using 4 multiple-output ANNs specifically trained on channels 1, 5, 9, and 13 (channel-dependent prediction model) providing contextual outputs for future horizons 1, 2, 5, and 10 minutes.**

Channel	T_f [min]	μ_{e^2} [$\cdot 10^{-3}$]	$\mu_{\lvert e \rvert}$	$\lvert e \rvert_{\mathrm{p}90}$ [%]	$\lvert e \rvert_{\mathrm{p}95}$	W [%]
ch1	1	2.171	3.35	7.36	9.80	51.9
	2	**1.833**	**2.99**	**6.75**	**9.23**	52.0
	5	1.963	**2.99**	7.17	9.88	53.8
	10	2.372	3.27	8.28	11.03	55.8
ch5	1	1.090	2.43	5.01	6.23	54.6
	2	0.854	2.03	4.17	5.30	55.0
	5	**0.654**	**1.69**	**3.45**	**4.47**	58.5
	10	0.767	1.70	**3.45**	4.65	59.0
ch9	1	2.735	3.55	7.09	8.98	57.6
	2	**2.587**	**3.18**	**6.15**	**8.13**	58.5
	5	3.312	3.37	6.32	9.59	56.7
	10	4.726	4.18	7.80	13.22	50.1
ch13	1	2.261	3.77	7.80	9.35	52.9
	2	1.437	2.98	6.17	7.46	53.8
	5	0.942	2.37	4.92	6.07	56.4
	10	**0.847**	**2.20**	**4.63**	**5.86**	56.3

Note: boldface denotes best cases.

architectural. In fact, the same databases as before were used for training and test. The single-output architecture of the previous campaign and the multi-output architecture of the current one are depicted side by side in Fig. 6.

Results are reported in Table 4. As can be seen, accuracy of a multi-output ANN (MAE is taken again into account as the metric for comparison) somehow resembles what provided by a plurality of single-output ANNs, but the latter typically show a lower error. There are some exceptions, e.g., predictions over the next 5 min for channels 5 and 13, but they are mostly inessential. Sometimes, the multi-output ANN behaved worse than SMA. Summing up, a simpler (and cheaper, in terms of both memory occupation and training time) ANN implementation is possible at the price of a diminished accuracy.

D. GENERALIZED ANN TRAINING

A criticism about the use of ANNs to predict the quality of wireless channels concerns their training. In the above campaigns, ANNs were trained on a channel-by-channel basis, to reflect differences between the related spectrum conditions. To face spectrum non-stationarity in the long term, training could be reiterated periodically by the involved devices, e.g., by automatically invoking a suitable procedure. Clearly, doing so is not trivial at all, and demands for the permanent availability of computational resources on APs and STAs.

It could be interesting to determine whether or not a generic training can be performed for ANNs, in order to characterize Wi-Fi interference independently of the channel, and hence of the traffic actually found on it (*channel-independent* prediction model). To this purpose, we trained a single ANN for every horizon with a database obtained by merging all the training data used in the previous campaigns

$$\mathcal{D}_{\mathrm{tr}}^{\mathrm{all}} = \bigcup_{c \in \{1,5,9,13\}} \mathcal{D}_{\mathrm{tr}}^{c}, \tag{10}$$

where the union symbol denotes the ordered concatenation of sequences. Then, we evaluated the prediction accuracy for every single test database $\mathcal{D}_{\mathrm{te}}^{ch}$.

This campaign permitted to appreciate how much a specific, channel-dependent training, improves accuracy. Channel-independent training demands for a noticeably lower effort, and also provides a more generic solution. In fact, the interference patterns observed by the testbed on the different channels are mostly uncorrelated, since their frequencies do not overlap. This is not completely true because of the presence of nearby equipment (APs and STAs, whose placement and traffic was not under our control) tuned on non-canonical channels (other than 1, 5, 9, and 13). Luckily, there were only a few of them. For example, an AP operating on channel 3 interferes with both channels 1 and 5 of the testbed, which implies some correlation between the related databases. Likewise, the contextual presence of traffic on channel 3 and a 40 MHz link obtained by bonding channels 6 and 10 may create some dependency between channels 1 and 13. In theory, the farther the channels, the lower correlation.

In the above campaign, the training and test databases are still correlated, since the former contain data acquired on every channel. To make them completely uncorrelated, at least in theory (*truly-channel-independent* prediction model) a further campaign was carried out where the training database for every channel includes all databases with the exception of the one related to the channel itself

$$\mathcal{D}_{\mathrm{tr}}^{\neg ch} = \bigcup_{c \in \{1,5,9,13\} \setminus ch} \mathcal{D}_{\mathrm{tr}}^{c}. \tag{11}$$

TABLE 5. Prediction accuracy on channels 1, 5, 9, and 13 and for future horizons 1, 2, 5, and 10 minutes by using ANNs trained with the (specific) related databases $\mathcal{D}_{\text{tr}}^{ch}$ (channel-dependent prediction model), a single (generic) combined database $\mathcal{D}_{\text{tr}}^{\text{all}}$ (channel-independent prediction model), and (specific) combined databases $\mathcal{D}_{\text{tr}}^{\neg ch}$ purposely created to be as independent as possible from the channel under test (truly-channel-independent prediction model).

Channel (test)	Training database	T_{f} [min]	μ_{e^2} [$\cdot 10^{-3}$]	$\mu_{\lvert e\rvert}$	$\lvert e\rvert_{\text{p90}}$ [%]	$\lvert e\rvert_{\text{p95}}$	W [%]	T_{f} [min]	μ_{e^2} [$\cdot 10^{-3}$]	$\mu_{\lvert e\rvert}$	$\lvert e\rvert_{\text{p90}}$ [%]	$\lvert e\rvert_{\text{p95}}$	W [%]
ch1	$\mathcal{D}_{\text{tr}}^{ch}$	1	$\underline{1.973}$	$\underline{3.22}$	6.96	9.23	54.2	5	$\underline{1.880}$	$\underline{2.92}$	6.95	9.64	(55.4)
	$\mathcal{D}_{\text{tr}}^{\text{all}}$		**1.882**	**3.19**	**6.81**	**8.87**	53.8		**1.735**	**2.87**	**6.52**	**8.97**	53.6
	$\mathcal{D}_{\text{tr}}^{\neg ch}$		1.887	3.20	**6.81**	8.88	53.4		1.760	2.91	6.56	9.01	52.0
	$\mathcal{D}_{\text{tr}}^{ch}$	2	$\underline{1.670}$	$\underline{2.86}$	6.32	8.65	54.4	10	$\underline{2.254}$	**3.19**	8.00	10.71	57.7
	$\mathcal{D}_{\text{tr}}^{\text{all}}$		**1.589**	**2.83**	**6.14**	**8.28**	53.5		**2.194**	3.25	7.71	10.37	52.9
	$\mathcal{D}_{\text{tr}}^{\neg ch}$		1.608	2.85	6.16	8.31	52.3		2.213	$\underline{3.31}$	**7.70**	**10.31**	50.4
ch5	$\mathcal{D}_{\text{tr}}^{ch}$	1	**1.041**	**2.40**	**4.94**	**6.14**	55.7	5	0.664	**1.71**	**3.51**	**4.53**	57.3
	$\mathcal{D}_{\text{tr}}^{\text{all}}$		1.045	2.47	5.11	6.28	51.4		**0.652**	1.82	3.70	4.65	52.4
	$\mathcal{D}_{\text{tr}}^{\neg ch}$		$\underline{1.060}$	$\underline{2.49}$	5.15	6.30	50.2		$\underline{0.703}$	$\underline{1.93}$	3.86	4.80	47.7
	$\mathcal{D}_{\text{tr}}^{ch}$	2	0.768	**1.97**	**4.04**	**5.10**	56.8	10	**0.759**	**1.70**	**3.43**	**4.62**	(59.1)
	$\mathcal{D}_{\text{tr}}^{\text{all}}$		**0.751**	2.04	4.20	5.19	52.6		0.777	1.92	3.89	4.97	49.2
	$\mathcal{D}_{\text{tr}}^{\neg ch}$		$\underline{0.773}$	$\underline{2.07}$	4.28	5.26	50.9		$\underline{0.895}$	$\underline{2.18}$	4.25	5.29	40.5
ch9	$\mathcal{D}_{\text{tr}}^{ch}$	1	2.662	3.51	7.03	8.86	58.6	5	**3.184**	3.27	6.03	**9.25**	58.2
	$\mathcal{D}_{\text{tr}}^{\text{all}}$		**2.649**	**3.46**	**6.92**	**8.77**	60.5		3.191	**3.11**	**5.80**	9.41	64.2
	$\mathcal{D}_{\text{tr}}^{\neg ch}$		$\underline{3.214}$	$\underline{3.62}$	7.11	9.51	58.5		$\underline{4.132}$	$\underline{3.40}$	6.95	12.59	62.6
	$\mathcal{D}_{\text{tr}}^{ch}$	2	2.514	3.15	6.12	8.01	59.3	10	**4.434**	$\underline{3.95}$	7.18	**12.23**	53.3
	$\mathcal{D}_{\text{tr}}^{\text{all}}$		**2.511**	**3.06**	**5.91**	**7.90**	62.3		4.485	**3.63**	**7.09**	13.46	62.9
	$\mathcal{D}_{\text{tr}}^{\neg ch}$		$\underline{3.140}$	$\underline{3.26}$	6.26	9.05	60.0		$\underline{5.684}$	3.92	9.65	17.1	63.7
ch13	$\mathcal{D}_{\text{tr}}^{ch}$	1	**2.243**	**3.76**	**7.77**	**9.31**	53.7	5	**0.941**	**2.38**	**4.93**	**6.04**	55.6
	$\mathcal{D}_{\text{tr}}^{\text{all}}$		2.499	3.97	8.21	9.84	47.6		1.156	2.66	5.53	6.68	47.6
	$\mathcal{D}_{\text{tr}}^{\neg ch}$		$\underline{2.628}$	$\underline{4.06}$	8.43	10.1	46.2		$\underline{1.271}$	$\underline{2.80}$	5.81	7.03	45.0
	$\mathcal{D}_{\text{tr}}^{ch}$	2	**1.416**	**2.96**	**6.12**	**7.40**	54.3	10	**0.813**	**2.15**	**4.54**	**5.68**	59.4
	$\mathcal{D}_{\text{tr}}^{\text{all}}$		1.646	3.21	6.62	7.95	47.1		1.053	2.52	5.28	6.44	47.5
	$\mathcal{D}_{\text{tr}}^{\neg ch}$		$\underline{1.768}$	$\underline{3.33}$	6.87	8.24	45.3		$\underline{1.179}$	$\underline{2.69}$	5.62	6.78	44.7

Note: boldface denotes best cases, underline denotes worst cases.

This is expected to represent the worst case for what concerns ANN training.

Results are reported in Table 5, which is split in 16 parts, one for every channel ch and future horizon T_{f}. Every such part includes in turn three rows. The first row refers to the channel-dependent prediction model, and shows the same results as Table 3 (included here to ease comparison), whereas the second is related to the channel-independent prediction model. Finally, the third row concerns the above case of bad training using databases created according to (11).

As can be seen, not necessarily specific training always bests generalized training. Curiously, excluding the channel under test from training did not always led to the worst results. Channels 5 and, especially, 13 are those which benefit more from a specialized training. By looking at time diagrams in Fig. 3, this likely depends on the fact that the interference patterns used for test and training are similar. Conversely, predictions that rely on a generalized training are typically more accurate on channel 9. By referring again to the time diagrams about the FDR, this is probably due to the fact that this channel suffers from a severe and irregularly-shaped interference. Therefore, widening the variety of cases on which training is performed is beneficial. The same holds for channel 1, where the patterns used for training and test look dissimilar (seemingly, something happened that made the spectrum conditions change tangibly in the course of the experiment). In this case, performing training on all channels with the exclusion of the one under test occasionally provided slightly better accuracy.

From above results one can see that generalized training often constitutes a valid alternative to channel-specific training. This is another relevant outcome of this work, since the former is way simpler to implement than the latter. Likely, this depends on the fact that ANN behavior is quite complex and cannot be described easily. Therefore, experimental campaigns like those described in this paper are the only way to assess their performance in the wild.

VII. PRACTICAL FEASIBILITY

A relevant point when ANNs are exploited to predict channel behavior in real equipment is the trade-off between prediction accuracy and computational complexity. Generally speaking, increasing the overall number N_{p} of past outcomes achieves better accuracy, because the ANN is provided a larger amount of information about channel's conditions. Accuracy is expected to improve also when N_{s} is shrunk, as doing so makes time resolution of FDR statistics more fine-grained. However, reducing N_{s} too much might increase variability when computing averages a_i on short time intervals (when i is low).

A. COMPLEXITY

The number of input features fed to the ANN, as specified by (7), is N_p/N_s. The asymptotic complexity of the ANN input layer (and hence, its contribution to the time taken to perform a test) is linear versus this quantity.

A first experiment was run to evaluate the average execution time $\bar{T}_p$ for a single test operation. Three different architectures were considered: the first and the second were based on Intel® Core™ CPUs, and in particular a desktop based on an i3-10105 @3.70GHz and a notebook based on an i7-11800H @2.30GHz, respectively, whereas the third was a Raspberry PI 2 based on an ARMv7 Processor rev 4 (v7l). The ANN was tested by letting $N_p = 14400$ and varying N_s from 30 to 480 in steps of 5. For every N_s value the test was repeated one million times, and the arithmetic mean of the related execution times was computed. Results for Intel Core and ARM are shown in plots a and b of Fig. 7, respectively. As expected, $\bar{T}_p$ is inversely proportional to N_s.

Concerning differences among the different architectures, when $N_s = 120$ (as in the experiments described in the previous section) $\bar{T}_p$ is 11.4 μs for the Intel i7, 14.6 μs for the Intel i3, and 142.6 μs for the ARMv7. These architectures (particularly the latter) are mostly compatible with high-end commercial APs software implementations, and execution times are low enough to enable adoption in real devices.

B. ACCURACY

As said above, the value selected for N_s also impacts on prediction accuracy. For this reason, a second experiment was carried out where N_s was set equal to 30, 60, 120, 240, and 480 (corresponding to basic interval widths from 15 s to 4 minutes). We considered, as an example, the specific case of a single-output ANN specifically trained (and tested) on channel 1 for a future horizon equal to 2 minutes (cf. Table 3). Again, $N_p = 14400$, i.e., the past interval was kept fixed to two hours.

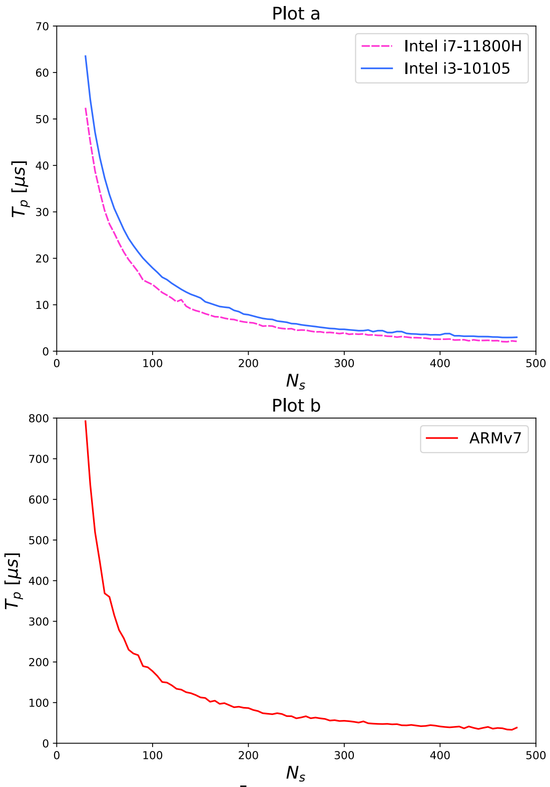

FIGURE 7. Mean execution time $\bar{T}_p$ vs. N_s (Intel Core and ARMv7).

TABLE 6. Relative improvement of the prediction accuracy by using a single-output ANNs trained and tested on channels 1 (channel-dependent prediction model) with T_f = 2 min and variable value of N_s = 30, 60, 120, 240, 480.

Channel	N_s [min]	μ_{e^2} [%]	$\mu_{\|e\|}$	$\|e\|_{p90}$ [%]	$\|e\|_{p95}$
ch1	30	-0.56	+0.21	-3.11	-2.73
	60	-0.13	+0.20	-3.11	-2.50
	120	0	0	0	0
	240	+3.09	+0.31	-3.18	-1.04
	480	+12.71	+3.11	+0.15	+5.48

Results, reported in Table 6, describe the relative variation (as a percentage) of accuracy metrics with respect to the reference case when $N_s = 120$. They clearly highlight that, when N_s is lowered down to 60 (or even 30), improvements on μ_{e^2} (the performance indicator minimized by the ANN) are negligible compared to the increase of computation times (see plots of Fig. 7). Increasing N_s to 240 (or 480) leads, on the one side, to a sensible decrease of computation times, but on the other it causes a substantial performance degradation, especially for what concerns μ_{e^2}. In conclusion, the value $N_s = 120$ we selected appears to be a good compromise between prediction accuracy and computational complexity.

VIII. CONCLUSION

Despite wireless networks are more and more used in many different context, due to their ability to provide wire-free connections, they are not able yet to offer applications the same quality of service as conventional wired solutions like Ethernet. While throughput has increased steadily over the past decades, to the point that, in theory, the performance of currently available technologies like Wi-Fi 6 and 5G is comparable (and, sometimes, higher) than Gigabit Ethernet, transmission on air lags behind cables for what concerns dependability and timeliness.

Deterministic MAC mechanisms, like trigger frames in Wi-Fi 6, effectively counteract intra-network interference. Unfortunately, they can do little or nothing against disturbance due to external sources. In this case, knowledge about the spectrum conditions in the recent past could be exploited by ML to statistically improve network behavior beyond what deterministic MACs can reasonably do.

This paper is aimed to present our most recent findings in this field. In particular, we predicted the mean failure rate on a wireless link over specific future time horizons by means of ANNs trained on the outcomes of the past transmission attempts. Results show that prediction accuracy achieved by

ML is better than conventional methods that rely on moving averages.

A second question we tried to answer is how much training impacts on accuracy. In particular, we compared the performance of ANNs specifically trained on the channel on which they will be used and those where training exploits a generic database that covers all channels. Results show that accuracy is mostly comparable, and channel-independent training could be the best option in those cases where spectrum conditions are likely to change (as happens over long time spans). Advantages of channel-independent prediction are undeniable: in fact, the same ANN could be employed in a variety of scenarios, e.g., by integrating it in the network equipment directly.

Among the activities we plan for our future work, the most important are the analysis of the proposed methodology under different conditions, including varying traffic loads and dynamically changing network configurations, as well as the study of ANN model scalability, its extendability to protocols other than Wi-Fi, and the suitability of alternative methods (for instance, probabilistic ones based on Markov chains).

ACKNOWLEDGMENT

(Stefano Scanzio and Alberto Salvatore Colletto are co-first authors.)

Computational resources were provided by HPC@POLITO, a project of Academic Computing within the Department of Control and Computer Engineering, Politecnico di Torino (http://www.hpc.polito.it).

REFERENCES

[1] Q. Huang, H. Chen, and Q. Zhang, "Joint design of sensing and communication systems for smart homes," *IEEE Netw.*, vol. 34, no. 6, pp. 191–197, Nov. 2020.

[2] W. Na, N.-N. Dao, and S. Cho, "Mitigating WiFi interference to improve throughput for in-vehicle infotainment networks," *IEEE Wireless Commun.*, vol. 23, no. 1, pp. 22–28, Feb. 2016.

[3] M. B. Attia, K. K. Nguyen, and M. Cheriet, "Dynamic QoS-aware scheduling for concurrent traffic in smart home," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5412–5425, Jun. 2020.

[4] S. Kwankajornkeat and C. Aswakul, "Differential private motion sensor and wasted energy in building energy management system," *IEEE Access*, vol. 10, pp. 486–501, 2022.

[5] S. Scanzio, G. Cena, and A. Valenzano, "Enhanced energy-saving mechanisms in TSCH networks for the IIoT: The PRIL approach," *IEEE Trans. Ind. Informat.*, vol. 19, no. 6, pp. 7445–7455, Jun. 2023.

[6] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 4th Quart., 2015.

[7] H. Cañas, J. Mula, M. Díaz-Madroñero, and F. Campuzano-Bolarín, "Implementing Industry 4.0 principles," *Comput. Ind. Eng.*, vol. 158, Aug. 2021, Art. no. 107379.

[8] P. K. R. Maddikunta, Q.-V. Pham, P. B, N. Deepa, K. Dev, T. R. Gadekallu, R. Ruby, and M. Liyanage, "Industry 5.0: A survey on enabling technologies and potential applications," *J. Ind. Inf. Integr.*, vol. 26, Mar. 2022, Art. no. 100257. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2452414X21000558

[9] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.

[10] S. Scanzio, L. Wisniewski, and P. Gaj, "Heterogeneous and dependable networks in industry—A survey," *Comput. Ind.*, vol. 125, Feb. 2021, Art. no. 103388. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166361520306229

[11] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, D. Niyato, O. Dobre, and H. V. Poor, "6G Internet of Things: A comprehensive survey," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 359–383, Jan. 2022.

[12] A. Garcia-Rodriguez, D. López-Pérez, L. Galati-Giordano, and G. Geraci, "IEEE 802.11be: Wi-Fi 7 strikes back," *IEEE Commun. Mag.*, vol. 59, no. 4, pp. 102–108, Apr. 2021.

[13] L. Leonardi, L. Lo Bello, and G. Patti, "LoRa support for long-range real-time inter-cluster communications over Bluetooth low energy industrial networks," *Comput. Commun.*, vol. 192, pp. 57–65, Aug. 2022.

[14] G. Cena, C. G. Demartini, M. G. Vakili, S. Scanzio, A. Valenzano, and C. Zunino, "Evaluating and modeling IEEE 802.15.4 TSCH resilience against Wi-Fi interference in new-generation highly-dependable wireless sensor networks," *Ad Hoc Netw.*, vol. 106, Sep. 2020, Art. no. 102199. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570870519310509

[15] K. E. Jeon, J. She, P. Soonsawad, and P. C. Ng, "BLE beacons for Internet of Things applications: Survey, challenges, and opportunities," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 811–828, Apr. 2018.

[16] G. Cena, S. Scanzio, and A. Valenzano, "Seamless link-level redundancy to improve reliability of industrial Wi-Fi networks," *IEEE Trans. Ind. Informat.*, vol. 12, no. 2, pp. 608–620, Apr. 2016.

[17] J. Li, S. K. Bose, and G. Shen, "Cooperative resource scheduling for time-sensitive services in an integrated XGS-PON and Wi-Fi 6 network," *IEEE Commun. Lett.*, vol. 26, no. 6, pp. 1338–1342, Jun. 2022.

[18] G. Cena, S. Scanzio, and A. Valenzano, "SDMAC: A software-defined MAC for Wi-Fi to ease implementation of soft real-time applications," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3143–3154, Jun. 2019.

[19] M. Friesen, L. Wisniewski, and J. Jasperneite, "Machine learning for zero-touch management in heterogeneous industrial networks–A review," in *Proc. IEEE 18th Int. Conf. Factory Commun. Syst. (WFCS)*, Apr. 2022, pp. 1–8.

[20] C. Deng, X. Fang, X. Han, X. Wang, L. Yan, R. He, Y. Long, and Y. Guo, "IEEE 802.11be Wi-Fi 7: New challenges and opportunities," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2136–2166, 4th Quart., 2020.

[21] G. Cena, S. Scanzio, and A. Valenzano, "Ultra-low power wireless sensor networks based on time slotted channel hopping with probabilistic blacklisting," *Electronics*, vol. 11, no. 3, p. 304, Jan. 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/3/304

[22] X. Vilajosana, T. Watteyne, T. Chang, M. Vucinic, S. Duquennoy, and P. Thubert, "IETF 6TiSCH: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 595–615, 1st Quart., 2020.

[23] S. Scanzio, M. G. Vakili, G. Cena, C. G. Demartini, B. Montrucchio, A. Valenzano, and C. Zunino, "Wireless sensor networks and TSCH: A compromise between reliability, power consumption, and latency," *IEEE Access*, vol. 8, pp. 167042–167058, 2020.

[24] C.-Y. Li, S.-C. Chen, C.-T. Kuo, and C.-H. Chiu, "Practical machine learning-based rate adaptation solution for Wi-Fi NICs: IEEE 802.11ac as a case study," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 10264–10277, Sep. 2020.

[25] G. Cena, S. Scanzio, and A. Valenzano, "Experimental evaluation of techniques to lower spectrum consumption in Wi-Red," *IEEE Trans. Wireless Commun.*, vol. 18, no. 2, pp. 824–837, Feb. 2019.

[26] G. Cena, S. Scanzio, D. Cavalcanti, and V. Frascolla, "Seamless redundancy for high reliability Wi-Fi," in *Proc. IEEE 19th Int. Conf. Factory Commun. Syst. (WFCS)*, Apr. 2023, pp. 1–4.

[27] S. Szott, K. Kosek-Szott, P. Gawlowicz, J. T. Gómez, B. Bellalta, A. Zubow, and F. Dressler, "Wi-Fi meets ML: A survey on improving IEEE 802.11 performance with machine learning," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 3, pp. 1843–1893, 3rd Quart., 2022.

[28] I. Ahmad, S. Shahabuddin, T. Sauter, E. Harjula, T. Kumar, M. Meisel, M. Juntti, and M. Ylianttila, "The challenges of artificial intelligence in wireless networks for the Internet of Things: Exploring opportunities for growth," *IEEE Ind. Electron. Mag.*, vol. 15, no. 1, pp. 16–29, Mar. 2021.

[29] L. Song and A. Striegel, "Leveraging frame aggregation to improve access point selection," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, May 2017, pp. 325–330.

[30] C. Pei, Z. Wang, Y. Zhao, Z. Wang, Y. Meng, D. Pei, Y. Peng, W. Tang, and X. Qu, "Why it takes so long to connect to a WiFi access point," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2017, pp. 1–9.

[31] Z. Han, T. Lei, Z. Lu, X. Wen, W. Zheng, and L. Guo, ''Artificial intelligence-based handoff management for dense WLANs: A deep reinforcement learning approach,'' *IEEE Access*, vol. 7, pp. 31688–31701, 2019.

[32] E. Zeljkovic, N. Slamnik-Kriještorac, S. Latré, and J. M. Marquez-Barja, ''ABRAHAM: Machine learning backed proactive handover algorithm using SDN,'' *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 4, pp. 1522–1536, Dec. 2019.

[33] H. Feng, Y. Shu, S. Wang, and M. Ma, ''SVM-based models for predicting WLAN traffic,'' in *Proc. IEEE Int. Conf. Commun.*, vol. 2, Jun. 2006, pp. 597–602.

[34] A. Thapaliya, J. Schnebly, and S. Sengupta, ''Predicting congestion level in wireless networks using an integrated approach of supervised and unsupervised learning,'' in *Proc. 9th IEEE Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UEMCON)*, Nov. 2018, pp. 977–982.

[35] Y. Liu, B. R. Tamma, B. S. Manoj, and R. Rao, ''On cognitive network channel selection and the impact on transport layer performance,'' in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, pp. 1–5, Dec. 2010.

[36] Q. Cui, Z. Zhang, Y. Shi, W. Ni, M. Zeng, and M. Zhou, ''Dynamic multichannel access based on deep reinforcement learning in distributed wireless networks,'' *IEEE Syst. J.*, vol. 16, no. 4, pp. 5831–5834, Dec. 2022.

[37] A. K. Gizzini, M. Chafii, A. Nimr, and G. Fettweis, ''Deep learning based channel estimation schemes for IEEE 802.11p standard,'' *IEEE Access*, vol. 8, pp. 113751–113765, 2020.

[38] A. Kulkarni, A. Seetharam, A. Ramesh, and J. D. Herath, ''DeepChannel: Wireless channel quality prediction using deep learning,'' *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 443–456, Jan. 2020.

[39] W. Jiang and H. D. Schotten, ''Neural network-based fading channel prediction: A comprehensive overview,'' *IEEE Access*, vol. 7, pp. 118112–118124, 2019.

[40] W. Jiang, H. Dieter Schotten, and J.-y. Xiang, *Neural Network–Based Wireless Channel Prediction*. Hoboken, NJ, USA: Wiley, 2020, ch. 16, pp. 303–325, doi: 10.1002/9781119562306.ch16.

[41] S. Scanzio, F. Xia, G. Cena, and A. Valenzano, ''Predicting Wi-Fi link quality through artificial neural networks,'' *Internet Technol. Lett.*, vol. 5, no. 2, Mar. 2022, Art. no. e326, doi: 10.1002/itl2.326.

[42] S. Scanzio, G. Cena, C. Zunino, and A. Valenzano, ''Machine learning to support self-configuration of industrial systems interconnected over Wi-Fi,'' in *Proc. IEEE 27th Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2022, pp. 1–8.

[43] G. Cena, S. Scanzio, and A. Valenzano, ''Improving effectiveness of seamless redundancy in real industrial Wi-Fi networks,'' *IEEE Trans. Ind. Informat.*, vol. 14, no. 5, pp. 2095–2107, May 2018.

[44] M. Cereia, I. C. Bertolotti, and S. Scanzio, ''Performance of a real-time EtherCAT master under Linux,'' *IEEE Trans. Ind. Informat.*, vol. 7, no. 4, pp. 679–687, Nov. 2011.

[45] F. Gosewehr, M. Wermann, and A. W. Colombo, ''From RTAI to RT-preempt a quantative approach in replacing Linux based dual kernel real-time operating systems with Linux RT-preempt in distributed real-time networks for educational ICT systems,'' in *Proc. IECON 42nd Annu. Conf. IEEE Ind. Electron. Soc.*, Oct. 2016, pp. 6596–6601.

[46] G. Cena, S. Scanzio, and A. Valenzano, ''A software-defined MAC architecture for Wi-Fi operating in user space on conventional PCs,'' in *Proc. IEEE 13th Int. Workshop Factory Commun. Syst. (WFCS)*, May 2017, pp. 1–10.

ALBERTO SALVATORE COLLETTO received the B.Sc. degree in computer engineering and the M.Sc. degree in cybersecurity-oriented computer engineering from Politecnico di Torino, Italy, in 2019 and 2023, respectively. He is currently collaborating with the Institute of Electronics, Computer and Telecommunication Engineering, National Research Council of Italy (CNR-IEIIT), in the research field. His research interest includes artificial intelligence, with particular reference to artificial neural networks applied to wireless networks.

STEFANO SCANZIO (Senior Member, IEEE) received the Laurea and Ph.D. degrees in computer science from Politecnico di Torino, Turin, Italy, in 2004 and 2008, respectively. From 2004 to 2009, he was with the Department of Computer Engineering, Politecnico di Torino, where he was involved in research on speech recognition and classification methods and algorithms. Since 2009, he has been with the National Research Council of Italy, where he is currently a Senior Researcher with the Institute of Electronics, Computer and Telecommunication Engineering. He teaches several courses on computer science with Politecnico di Torino. He has authored and coauthored more than 90 papers in international journals and conferences, in the areas of industrial communication systems, real-time networks, wireless networks, and artificial intelligence. He took part in the program and organizing committees of many international conferences of primary importance in the research areas. He was a recipient of the 2017 Best Paper Award from the IEEE Transactions on Industrial Informatics and the Best Paper Awards for three papers presented from the IEEE Workshops on Factory Communication Systems, in 2010, 2017, and 2019, and for a paper presented by the IEEE International Conference on Factory Communication Systems, in 2020. He is an Associate Editor of *Ad Hoc Networks* (Elsevier), IEEE Access, and *Electronics* (MDPI) journals.

GABRIELE FORMIS (Student Member, IEEE) received the B.Sc. degree in mechanical engineering and the M.Sc. degree in automation and control engineering from Politecnico di Milano, Italy, in 2018 and 2020, respectively, where he is currently pursuing the National Ph.D. degree in artificial intelligence. He is a Research Associate with the Institute of Electronics, Computer and Telecommunication Engineering, National Research Council of Italy (CNR-IEIIT). His research interests include artificial intelligence, wireless networks, and autonomous driving.

GIANLUCA CENA (Senior Member, IEEE) received the M.S. degree in electronic engineering and the Ph.D. degree in information and system engineering from Politecnico di Torino, Turin, Italy, in 1991 and 1996, respectively. Since 2005, he has been a Director of Research with the Institute of Electronics, Information Engineering and Telecommunications, National Research Council of Italy (CNR-IEIIT), Turin. His research interests include wired and wireless industrial communication systems, real-time protocols, and automotive networks. In these areas, he has coauthored about 170 technical papers and one international patent. He was a recipient of the Best Paper Award from the IEEE Transactions on Industrial Informatics, in 2017, and the IEEE Workshop on Factory Communication Systems, in 2004, 2010, 2017, 2019, and 2020. He has served as the Program Co-Chairperson for the IEEE Workshop on Factory Communication Systems, in 2006 and 2008. Since 2009, he has been serving as an Associate Editor for the IEEE Transactions on Industrial Informatics.

Open Access funding provided by 'Consiglio Nazionale delle Ricerche-CARI-CARE-ITALY' within the CRUI CARE Agreement