CASE STUDY

Predicting Wi-Fi link quality through artificial neural networks

Stefano Scanzio | Francesco Xia | Gianluca Cena | Adriano Valenzano

National Research Council of Italy (CNR-IEIIT), Turin, Italy

Correspondence
Stefano Scanzio, National Research Council of Italy (CNR-IEIIT), Turin, Italy.
Email: stefano.scanzio@ieiit.cnr.it

Funding information
Consiglio Nazionale delle Ricerche

Artificial neural networks are one of the key technologies behind the Industry 4.0 (r)evolution. It can be profitably employed in a variety of different applications contexts and with different goals, most of which are characterized by the fact that reliable models for some parts of the involved systems either do not exist or are unavailable. In this paper we tried to exploit artificial neural networks to predict the quality of the transmission channel in Wi-Fi better than what techniques employed in conventional adaptive solutions permit. Applicability of such an approach is quite broad, but we believe that the main intended goal is to improve communication dependability and system resilience. Preliminary results highlighted that artificial neural networks show higher prediction accuracy, and there is still extensive room for improvements.

KEYWORDS
artificial neural networks, dependable wireless communication, IEEE 802.11, machine learning

1 | INTRODUCTION

Artificial intelligence (AI) and, in particular, machine learning (ML), are nowadays customarily adopted in many application fields, including consumer electronics (photography, virtual assistants, smart bands) and advanced driver assistance systems (ADAS) in the automotive domain. They also constitute one of the enabling technologies of Industry 4.0, along with the Industrial Internet of Things (IIoT), big data, cloud, robotics, and additive manufacturing. In particular, ML can be profitably applied in many different areas in automated industrial scenarios, for example, to drive preventive maintenance (by discovering failures before they occur), for inspection and optical sorting in industrial machine vision, and in collaborative robots, to cite a few.

On the other hand, automated systems are becoming more and more distributed in nature, as this increases their flexibility and resilience, besides leading to a tangible reduction of costs. Industrial networks are becoming increasingly heterogeneous, by integrating both a wired infrastructure and a number of wireless extensions with different characteristics. Among the most relevant technologies for the wireless part we find: 5G for ultra-reliable low-latency communication (URLLC), IEEE 802.15.4 (WirelessHART, ZigBee, 6TiSCH) for low-power mesh networks, Bluetooth Low Energy (BLE) for connecting field devices (IO-Link wireless), LoRaWAN for long range interconnections, and IEEE 802.11 (Wi-Fi) for high-performance communication in unlicensed bands. In this context, the use of ML to support communication is a relatively new research direction.

When wireless networks are used in industrial scenarios, the main requirements are dependability and timeliness (the ability to meet the deadlines imposed by applications). Unfortunately, the shared transmission support and the huge variability of the spectrum conditions may occasionally cause communication quality to worsen unacceptably. The former aspect can be counteracted by deterministic medium access control mechanisms, whereas the second is mostly due to unpredictable phenomena and its effects can be mitigated using proactive mechanisms. For example, time slotted channel
hopping (TSCH) keeps the transmission frequency changing in order to minimize the occurrence of packet losses,\textsuperscript{10} while seamless redundancy techniques operate by sending the same frame on distinct channels at the same time.\textsuperscript{11}

Adaptive mechanisms, instead, operate on a reactive basis, for example, by changing the transmission channel whenever communication quality falls below a given threshold. Among them we find frequency agility in ZigBee, rate adaptation in Wi-Fi,\textsuperscript{12,13} and rank computation in RPL (to select candidate parents during network construction). These approaches are meant to optimize communication, and typically require quite fast dynamics (milliseconds to some seconds). However, similar methods can be adopted also at the application level. For instance, if the spectrum is expected to worsen, the speed of a fleet of automated guided vehicles that coordinate their actions over the air can be decreased consequently. Another example is roaming, where handover between base stations can be optimized based on the expected channel quality. Since mechanical parts are involved, dynamics in these cases are noticeably slower (seconds to minutes). Above cases could benefit from ML, but their highly dynamic context may make it less effective. The use case we consider below is a Wi-Fi link used as cable replacement to interconnect two physically separate parts of a controlled system (e.g., on the two sides of a large road or a river). On this high-speed link many slower data streams can be conveyed, for example collected by a multitude of sensors. In the case the quality of communication on the Wi-Fi link is expected to worsen (the horizon here spans from minutes to tens of minutes), either the sampling periods of sensors could be proactively enlarged or some of them can be temporarily disabled, to decrease traffic and avoid congestion on the link.

Above mechanisms try to infer the future state of the wireless spectrum starting from its measured conditions in the recent past. Many metrics can be used to this purpose, the most popular of which is probably the fraction of successful attempts, obtained through simple mathematical tools like the simple and weighted mean. In this work we wish to assess if and to what extent ML-based tools, for example artificial neural networks (ANNs), may bring any advantages in terms of the ability to correctly predict spectrum conditions. In particular, we captured the state of a Wi-Fi channel operating in the 2.4GHz band by performing endurance experiments. Then, we defined a specific target (the frame delivery ratio evaluated over a given time window) and compared how well two approaches, the first based on a conventional simple average and the other on an ANN, are able to track this target starting from the same kind of features. Other approaches have appeared in the recent literature concerning the use of ML to predict the quality of a wireless channel\textsuperscript{14,15} but, as far as we know, none operates on experimental data captured at the MAC layer by performing a periodic probing on a real Wi-Fi link, as we did in this work. In particular, in\textsuperscript{16,17} an ANN is used to predict the channel gain and to improve performance of specific applications, such as transmit antenna selection in Wi-Fi MIMO systems. In these works, artificial data were used to evaluate performance, and the statistical quantity predicted by the model is different from the one analyzed here. Also approaches other than ML have been successfully exploited. For example, in\textsuperscript{18} a coalition formation game is used to improve coexistence of Wi-Fi and time division duplex long term evolution-unlicensed (LTE-U), while in\textsuperscript{19} optimizations on power consumption and latency are brought to green Wi-Fi thanks to Tabu search and reinforcement learning, respectively.

The paper is organized as follow: in Section 2 the methods used to perform link quality estimation are presented, then Section 3 describes the experimental setup, while results are reported and discussed in Section 4. Finally, Section 5 concludes the paper.

## 2 Prediction of the Link Quality

Generally speaking, the quantity one wishes to predict is the probability that a single transmission attempt succeeds. From it, statistics on most of the other metrics about communication quality, for example, the packet loss ratio or the transmission latency when automatic retransmission schemes are exploited, can be derived. The probability \( P_i \) for a frame to be correctly delivered to destination in one attempt over Wi-Fi can be measured by probing the channel periodically with period \( T_s \). Let \( \{ x_i \} \) describe the outcomes of such procedure. From the point of view of the sender, the event that frame \( i \) has arrived is implied by the reception of the ack frame (\( x_i = 1 \)). Conversely, the expiration of the ack timeout is related to losses (\( x_i = 0 \)). Our analyses are based on two disjoint ordered sets of samples: the \emph{training} set \( \overline{X}_{\text{train}} = \{ x_1', x_2', \ldots, x_{N_{\text{train}}}' \} \) and the \emph{test} set \( \overline{X}_{\text{test}} = \{ x_1, x_2, \ldots, x_{N_{\text{test}}} \} \).

The goal of our work is predicting the “channel conditions in the near future” given the recent past. In other words, we aim at estimating the success probability for single transmission attempts (i.e., not exploiting retransmissions), evaluated as the arithmetic mean of \( x_i \) computed over the following \( N_{\text{next}} \) frames. In this paper, we selected \( N_{\text{next}} = 3600 \), which corresponds to half an hour. Although this time is quite long, it is coherent with, for example, the need to reconfigure the sampling rate of motes in a wireless sensor network whose gateway is connected to the data sink through Wi-Fi. At any time, as specified by \( i \) (current sample), the target we want to predict is \( t_i = \frac{1}{N_{\text{next}}} \sum_{j=1}^{N_{\text{next}}} x_{i+j} \). Since targets were computed
FIGURE 1  Input features of the ANN model when the current time $i$ is 16 (example with $N_{\text{input}} = 3$ and $N_{\text{step}} = 4$) using sliding windows over sample sets $\overline{X}$, the number of available target values for the test database (consisting in $N_{\text{test}}$ samples) is $N_{\text{target}} = N_{\text{test}} - N_{\text{next}} + 1$.

Two methods were exploited for prediction: averaging a selected number of past samples (AVG) and using an ANN. They can be modeled as two functions $f^{\text{AVG}}(\overline{X}_i)$ and $f^{\text{ANN}}(\overline{X}_i)$ that, given a subset of $N_{\text{past}}$ input samples $\overline{X}_i = \{x_{i}, x_{i-1}, \ldots, x_{i-N_{\text{past}}+1}\}$ and a set of parameters evaluated in the training phase, provide at runtime an estimation of the target $t_i$. Commonly used metrics to assess estimators’ reliability (that is, their ability to predict the future quality of the wireless channel) are the mean squared error $\text{MSE} = \mu_{\epsilon_i} = \frac{1}{N_{\text{target}}} \sum_{i=1}^{N_{\text{target}}} \epsilon_i^2$ and the mean absolute error $\text{MAE} = \mu_{\epsilon_{i,l}} = \frac{1}{N_{\text{target}}} \sum_{i=1}^{N_{\text{target}}} |\epsilon_i|$, where $\epsilon_i = t_i - f(\overline{X}_i)$ is the instantaneous estimation error and the estimate for AVG and ANN is computed through $f^{\text{AVG}}(\overline{X}_i)$ and $f^{\text{ANN}}(\overline{X}_i)$, respectively.

The AVG approach relies on the reasonable assumption that the future behavior of the channel can be derived by averaging the last $N_{\text{past}}$ samples, that is, $f^{\text{AVG}}(\overline{X}_i) = \frac{1}{N_{\text{past}}} \sum_{j=1}^{N_{\text{past}}} x_{i+j-1}$. The only configuration parameter of this model is $N_{\text{past}}$. Regarding the ANN-based method, the input vector $\overline{t} = \{t_1, t_2, \ldots, t_{N_{\text{input}}}\}$ of the ANN is derived from the training and test databases through a front-end module. As depicted in Figure 1, the $N_{\text{past}}$ samples are first split in blocks of size $N_{\text{step}}$. Then, the input feature $t_k$ is obtained from the $k$ most recent blocks by averaging the related $k \cdot N_{\text{step}}$ samples. The front-end module can be seen as a function $g$ that maps the space of the input samples onto the space of the actual inputs to the ANN, so that the latter can be obtained as $\overline{t} = g(\overline{X}_i)$. Therefore, by denoting the output of the ANN as $f^{\text{ANN}}(\overline{t})$, we have $f^{\text{ANN}}(\overline{X}_i) = f^{\text{ANN}}(g(\overline{X}_i))$.

3  EXPERIMENTAL SETUP

Databases were acquired through a commercial PC running Linux Ubuntu 18.04 (kernel v. 4.4.0) and equipped with a dual-band TP-Link TL-WDN4800 Wi-Fi adapter managed by the ath9k device driver (v. 4.4.2-1). The software-defined MAC (SDMAC)\textsuperscript{30} architecture and application programming interface (API) were used to disable automatic retries and return the notifications of the transmission outcomes, either $\text{ack}(x_i = 1)$ or $\text{ack timeout}(x_i = 0)$, back to the measurement process in user space. The transmission rate was set to 54 Mbps and frame aggregation was explicitly disabled to ensure that the channel state is probed evenly over time. As an alternative, both retransmission and rate adaptation can be left enabled, and a lightweight SDMAC version can be used to count the number of retries for each frame, calculating then the success ratio. Although this can be practically implemented in real setups with affordable effort, accuracy of measurements would not be as good as in our experimental procedure, since attempts are no longer evenly spaced.

Training and test databases were acquired on a fixed channel (13, corresponding to 2.472GHz), which is compatible with a Wi-Fi link used as cable replacement. In this case, ANN training can be carried out once and for all, since endpoints of the link are deployed in fixed positions. The sending period for probing frames (UDP packets with payload 50bytes) was set equal to $T_i = 0.5$ s (Table 1). Two training databases were acquired, DB1 and DB2, where the former is a proper subset of the samples contained in the second. Capturing samples required more than 16 and 38 days, respectively ($N_{\text{train}}^{\text{DB1}} = 2807524$ and $N_{\text{train}}^{\text{DB2}} = 6592482$), while measurement for the test set lasted more than two and a half days ($N_{\text{test}} = 460927$ samples). Training databases were acquired immediately after the test dataset. Variability over time of targets, which are used to train the ANN model and, in the test phase, to assess both AVG and ANN performance, provides a direct indication of how much the frame delivery probability changes over time. Plots related to the targets for the training and test databases are included in Figure 2. As can be seen, channel behavior varied frequently and in a seemingly unpredictable way.
### Table 1

| Quantity | Description                                      | Value |
|----------|--------------------------------------------------|-------|
| $T_s$    | Sending period of Wi-Fi frames                   | 0.5s  |
| $N_{DB1\ train}$ | Size of the training set DB1                  | 2807524 |
| $N_{DB2\ train}$ | Size of the training set DB2               | 6592482 |
| $N_{test}$ | Size of the test set                             | 460927 |
| $N_{next}$ | Next samples on which target $t_i$ is computed  | 3600  |
| $N_{target}$ | Number of patterns/targets in test set          | 457327 |
| $N_{AVG\ past}$ | Optimal $N_{past}$ value for AVG estimation on training set | 4080 |
| $N_{step}$ | Step for computing input features of the ANN    | 120   |
| $N_{input}$ | Number of inputs of the ANN model              | 34/120 |
| $N_{past}$ | Past samples used by AVG and ANN methods        | 4080/14400 |

Concerning the ANN, many configurations, input features, and databases were analyzed as part of our work. Due to space reasons, only the most representative ones have been reported here. The model used to predict the frame delivery probability is sequential and composed of three fully connected layers (also known as dense layers). It has $N_{input}$ inputs, 32 neurons on the two hidden layers (32 × 32), and one single neuron on the output layer. The activation function is ReLu for the first two layers and linear for the latter. The Adam optimizer\(^\text{21}\) with mean squared error as loss function was selected to train the model in 15 epochs. Weights are updated every 64 patterns, that is, the batch size was set to 64, and the learning rate was decreased at each epoch, starting from 0.01, by halving it on every new epoch. The ANN model was trained and tested with the Keras module included in TensorFlow. All initial weights were set using a Glorot normal initializer, which is available in the tensorflow.keras.initializers module. Functions `model.fit()` and `model.predict()` were used to train and test the model, respectively. The training time with the Google Colaboratory (Colab) platform and using CPUs (instead of GPUs or TPUs) was about 15 minutes for DB1, that is, about 1 minute for every epoch.

### RESULTS

To achieve a fair comparison, AVG performance was evaluated on the training set to find the specific value of $N_{past}$, denoted $N_{AVG\ past}$, that minimizes the MSE. The plot in Figure 3 shows the MSE obtained in AVG by varying the value of $N_{past}$. As can be seen, $N_{AVG\ past} = 4080$ for DB1, and practically the same value was obtained for DB2. This means that, for our target, spectrum dynamics are actually not so fast. $N_{AVG\ past}$ constitutes the optimum for AVG with the training set, and was used for evaluating $f_{AVG}^\left(\bar{X}_t\right)$ with the test set. For the ANN model, tests were performed with both $N_{past} = 4080$ and $N_{past} = 14400$, setting $N_{step} = 120$. ANN performance evaluated on the test set for values of $N_{past}$ greater than 14400 remained quite stable.

Results about this comparison, obtained by applying AVG and ANN methods to the test database, are reported in the first two lines of Table 2 (condition C\(_1\)), which refer to the case where the ANN is trained on DB1. As can be seen, the ANN method outperformed the AVG method concerning all the key performance indicators (KPIs) analyzed in this

![Figure 2](image1.png) **Figure 2** Variation of the target $t_i$ (frame delivery ratio) over time in the training and test databases

![Figure 3](image2.png) **Figure 3** MSE for $f_{AVG}^\left(\bar{X}_t\right)$ evaluated on the training set vs $N_{past}$ and optimal window width $N_{AVG\ past}$
work. Besides $MSE$, which is the objective function the ANN is intended to minimize, tangible improvements were also achieved concerning $MAE$. In particular, in the case of $N_{\text{past}} = 14400$ the value of $MAE$ is $2.32\%$ for AVG and $2.04\%$ for ANN. This means that, exploiting the ANN, the mean frame delivery probability over the next 30 minutes can be predicted with a mean error of $\pm 2.04\%$.

More interesting KPIs are the percentiles of the absolute errors $|\epsilon_i|$. In particular, referring to the results reported in Table 2, the 95-percentile for the ANN with $N_{\text{past}} = 14400$ is $|\epsilon_{I}|_{p95} = 5.16\%$, which means that the absolute error of its predictions is less than $5.16\%$ in $95\%$ of the cases, vs the $6.47\%$ achieved by AVG. Another relevant KPI is the win ratio $w$, defined as the number $N_{\text{win}}$ of times ANN won over AVG (i.e., it showed a lower absolute error $|\epsilon_i|$) divided by the number of samples in the test set for which predictions can be performed and verified, that is, $w = \frac{N_{\text{win}}}{N_{\text{past}}}$. The $w$ value was equal to $53.5\%$ for $N_{\text{past}} = N_{\text{AVG}} = 4080$, and increased to $55.6\%$ when $N_{\text{past}} = 14400$. This means that, even by using a streamlined set of features (which makes training fast and inexpensive), $55.6\%$ of the times ANN prediction outdid the AVG method.

Above results show that, for the use cases we considered here, exploiting a simple ANN to predict the quality of a Wi-Fi link, so as to optimize the behavior of a distributed application by adaptively re-configuring operating and communication parameters (e.g., sampling times), is a viable solution. Experimental results with the larger DB2 database are reported in Table 2 as conditions $C_2$ and $C_3$. As highlighted by condition $C_2$, increasing the size of the training database is beneficial and improved all the KPIs. Instead, an increase in the number of parameters of the ANN as done for $C_3$, that is, 64 neurons in the two hidden layers ($64 \times 64$), did not lead to any substantial improvement.

Computation complexity for AVG and ANN was obtained by profiling the implementation in the C programming language of the test procedure. The program was compiled with the -O2 option of the gcc compiler and run on both a personal computer equipped with an Intel Core i7-3700 processor (PC) and a Raspberry Pi 3 Model B based on an ARMv7 processor (PI). Execution times to obtain one prediction on PC and PI are reported in Table 2 as $T_{\text{PC}}$ and $T_{\text{PI}}$, respectively. As expected, AVG is much faster than the ANN, which however only takes as low as $5.687 \mu s$ on the PC and $84.051 \mu s$ on the PI with the most complex configuration (last row of the table). While the execution time of the ANN scales linearly with the number $N_{\text{input}}$ of inputs, the AVG execution time is fixed. The rightmost column of the table reports the memory footprint for data of each implementation. Even if AVG occupies a relatively small amount of memory, the most complex ANN configuration analyzed in this work has a memory occupation lower than 50KB, which is compatible with many devices.

| Config. Method | $N_{\text{past}}$ | $MSE \times 10^{-3}$ | $\mu_{\epsilon_i}(\text{MAE})$ [%] | $|\epsilon_{I}|_{p95}$ [%] | $w$ [%] | $T_{\text{PC}}$ [CPU] $\mu s$ | $T_{\text{PI}}$ [CPU] $\mu s$ | Memory [B] |
|---------------|------------------|----------------------|-------------------------------|-------------------------|---------|------------------|------------------|-------------|
| AVG           | 4080             | 0.975                | 2.32                          | 5.16                    | 6.47    | —                | 0.008           | 0.025       | 522        |
| $32 \times 32$ ANN | 4080             | 0.789                | 2.14                          | 4.43                    | 5.63    | 53.5             | 0.988           | 15.719      | 9232       |
| $C_1$ DB1 ANN | 14400            | 0.691                | 2.04                          | 4.11                    | 5.16    | 55.6             | 2.116           | 35.787      | 20584      |
| $32 \times 32$ ANN | 4080             | 0.778                | 2.12                          | 4.39                    | 5.62    | 54.5             | 0.988           | 15.719      | 9232       |
| $C_2$ DB2 ANN | 14400            | 0.687                | 2.02                          | 4.06                    | 5.16    | 56.3             | 2.116           | 35.787      | 20584      |
| $64 \times 64$ ANN | 4080             | 0.778                | 2.12                          | 4.39                    | 5.62    | 54.5             | 2.861           | 45.164      | 26512      |
| $C_3$ DB2 ANN | 14400            | 0.687                | 2.02                          | 4.06                    | 5.16    | 56.2             | 5.687           | 84.051      | 48872      |

5 | CONCLUSIONS

In this work we investigated the use of machine learning, and in particular of ANNs, to predict the quality of a wireless channel. This ability can be exploited in many different ways, by both the communication stack and, as in our specific case, processes in distributed applications with slow dynamics, to improve the effectiveness of adaptive mechanisms aimed at, for example, preventing a Wi-Fi link from being overloaded. Although we are at the beginning of a much wider investigation work, results are encouraging, with the ANN outdoing conventional methods based on averages. So far, we evaluated the target over a quite large window, which provided stable results. As future work we will try to reduce it consistently, by applying suitable techniques to effectively narrow the horizon of predictions. Clearly, increasing the number of distinct input features, for example, by including the received signal strength indicator (RSSI) or the transmission latency (which accounts for backoff due to interference), is expected to increase substantially the ability of the ANN to perform reliable predictions, hence increasing overall system dependability and resilience.

ACKNOWLEDGMENTS

Open Access Funding provided by Consiglio Nazionale delle Ricerche within the CRUI-CARE Agreement.
DATA AVAILABILITY STATEMENT
The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID
Stefano Scanzio https://orcid.org/0000-0001-7643-2342
Francesco Xia https://orcid.org/0000-0003-3586-5031
Gianluca Cena https://orcid.org/0000-0003-0084-5321
Adriano Valenzano https://orcid.org/0000-0003-1238-0808

REFERENCES
1. Borrego-Carazo J, Castells-Rufas D, Biempica E, Carrabina J. Resource-constrained machine learning for ADAS: a systematic review. IEEE Access. 2020;8:40573-40598. doi:10.1109/ACCESS.2020.2976513
2. Essien A, Giannetti C. A deep learning model for smart manufacturing using convolutional LSTM neural network autoencoders. IEEE Trans. Ind. Informat. 2020;16(9):6069-6078. doi:10.1109/TII.2020.2967556
3. Aydemir G, Paynabar K. Image-based prognostics using deep learning approach. IEEE Trans Ind Informat. 2020;16(9):5956-5964. doi:10.1109/TII.2019.2956220
4. Rathore MM, Shah SA, Shukla D, Bentafat E, Bakiras S. The role of AI, machine learning, and big data in digital twinning: a systematic literature review, challenges, and opportunities. IEEE Access. 2021;9:32030-32052. doi:10.1109/ACCESS.2021.3060863
5. Wang J, Wang D, Wang S, Li W, Song K. Fault diagnosis of bearings based on multi-sensor information fusion and 2D convolutional neural network. IEEE Access. 2021;9:23717-23725. doi:10.1109/ACCESS.2021.3056767
6. Scanzio S, Wisniewski L, Gaj P. Heterogeneous and dependable networks in industry—a survey. Comput Ind. 2021;125:103388. doi:10.1016/j.compind.2020.103388
7. Sun Y, Peng M, Zhou Y, Huang Y, Mao S. Application of machine learning in wireless networks: key techniques and open issues. IEEE Commun Surv Tutor. 2019;21(4):3072-3108. doi:10.1109/COMST.2019.2924243
8. Mengelli M, Scanzio S. A neural approach to synchronization in wireless networks with heterogeneous sources of noise. Ad Hoc Netw. 2016;49:1-16. doi:10.1016/j.adhoc.2016.06.002
9. Ahmad I, Shahabuddin S, Malik H, et al. Machine learning meets communication networks: current trends and future challenges. IEEE Access. 2020;8:223418-223460. doi:10.1109/ACCESS.2020.3041765
10. Scanzio S, Vakili MG, Cena G, et al. Wireless sensor networks and TSCH: a compromise between reliability, power consumption, and latency. IEEE Access. 2020;8:167042-167058. doi:10.1109/ACCESS.2020.3022434
11. Cena G, Scanzio S, Valenzano A. Improving effectiveness of seamless redundancy in real industrial Wi-fi networks. IEEE Trans. Ind. Informat. 2018;14(5):2095-2107. doi:10.1109/TII.2017.2759788
12. Krotov A, Kiryanov A, Khorov E. Rate control with spatial reuse for Wi-fi 6 dense deployments. IEEE Access. 2020;8:168898-168909. doi:10.1109/ACCESS.2020.3023552
13. Li CY, Chen SC, Kuo CT, Chiu CH. Practical machine learning-based rate adaptation solution for Wi-fi NICs: IEEE 802.11ac as a case study. IEEE Trans Veh Dent Technol. 2020;69(9):10264-10277. doi:10.1109/TVT.2020.3004471
14. Kulkarni A, Seetharam A, Ramesh A, Herath JD. DeepChannel: wireless channel quality prediction using deep learning. IEEE Trans Veh Dent Technol. 2020;69(1):443-456. doi:10.1109/TVT.2019.2949954
15. Gizzini AK, Chafii M, Nimr A, Fettweis G. Deep learning based channel estimation schemes for IEEE 802.11p standard. IEEE Access. 2020;8:113751-113765. doi:10.1109/ACCESS.2020.3003286
16. Jiang W, Schotten HD. Neural network-based Fading Channel prediction: a comprehensive overview. IEEE Access. 2019;7:118112-118124. doi:10.1109/ACCESS.2019.2937588
17. Jiang W, Dieter Schotten H, Xiang JY. Neural Network-Based Wireless Channel Prediction. Machine Learning for Future Wireless Communications. Vol 16. Hoboken, NJ: John Wiley & Sons, Ltd; 2020:303-325.
18. Zhang Y, Jiang C, Wang J, Han Z, Yuan J, Cao J. Coalition formation game based access point selection for LTE-U and Wi-fi coexistence. IEEE Trans. Ind. Informat. 2018;14(6):2653-2665. doi:10.1109/TII.2018.2797100
19. Zhang Y, Jiang C, Wang J, Han Z, Yuan J, Cao J. Green Wi-fi implementation and Management in Dense Autonomous Environments for smart cities. IEEE Trans. Ind. Informat. 2018;14(4):1552-1563. doi:10.1109/TII.2017.2785820
20. Cena G, Scanzio S, Valenzano A. SDMAC: a software-defined MAC for Wi-fi to ease implementation of soft real-time applications. IEEE Trans. Ind. Informat. 2018;15(3):3143-3154. doi:10.1109/TII.2018.2873205
21. Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. In: 3rd International Conference on Learning Representations (ICLR 2015). 2015:1-15.

How to cite this article: Scanzio S, Xia F, Cena G, Valenzano A. Predicting Wi-Fi link quality through artificial neural networks. Internet Technology Letters. 2022;5(2):e326. doi: 10.1002/itl2.326