POLITECNICO DI TORINO
Repository ISTITUZIONALE

Improving Wi-Fi Network Performance Prediction with Deep Learning Models

Original
Improving Wi-Fi Network Performance Prediction with Deep Learning Models / Formis, G., Ericson, A., Forsstrom, S., Thar, K., Cena, G., Scanzio, S.. - (2025), pp. 1-8. (34th IEEE International Symposium on Industrial Electronics, ISIE 2025 Toronto ON (CAN) 20-23 June 2025) [10.1109/isie62713.2025.11124605].

Availability:
This version is available at: 11583/3004398 since: 2025-10-23T13:24:03Z

Publisher:
Institute of Electrical and Electronics Engineers

Published
DOI:10.1109/isie62713.2025.11124605

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2025 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

26 November 2026

This is the author's version of an article that has been accepted for publication.
Changes were made to this version by the publisher prior to publication.

Improving Wi-Fi Network Performance Prediction with Deep Learning Models

Gabriele Formis*‡, Amanda Ericson†, Stefan Forsström†, Kyi Thar†, Gianluca Cena*, and Stefano Scanzio*
*National Research Council of Italy (CNR–IEIIT), Italy ‡Politecnico di Torino, Italy
†Department of Computer and Electrical Engineering, Mid Sweden University, Sundsvall, Sweden
gabrieleformis@cnr.it, {amanda.ericson, stefan.forsstrom, kyi.thar}@miun.se,{gianluca.cena, stefano.scanzio}@cnr.it

arXiv:2507.11168v1 [cs.NI] 15 Jul 2025

***Abstract*—The increasing need for robustness, reliability, and determinism in wireless networks for industrial and mission-critical applications is the driver for the growth of new innovative methods. The study presented in this work makes use of machine learning techniques to predict channel quality in a Wi-Fi network in terms of the frame delivery ratio. Predictions can be used proactively to adjust communication parameters at runtime and optimize network operations for industrial applications. Methods including convolutional neural networks and long short-term memory were analyzed on datasets acquired from a real Wi-Fi setup across multiple channels. The models were compared in terms of prediction accuracy and computational complexity. Results show that the frame delivery ratio can be reliably predicted, and convolutional neural networks, although slightly less effective than other models, are more efficient in terms of CPU usage and memory consumption. This enhances the model's usability on embedded and industrial systems.**

***Index Terms*—Wi-Fi, Channel quality prediction, Machine Learning, Convolutional Neural Networks (CNN), Recurrent Neural Networks, Long Short-Term Memory (LSTM), Bidirectional LSTM (Bi-LSTM).**

I. Introduction

Robustness and dependability are the main challenges in next-generation communication systems, especially in wireless networks for industrial applications like Wi-Fi [1], but also in the context of smart cities and buildings, transportation, and agriculture. Factory automation and real-time monitoring require communication technologies that can maintain predictable performance even under fluctuating conditions of the quality of the channel [2]. Due to the numerous sources of interference and electromagnetic noise, such as machinery, and the coexistence of multiple wireless systems [3], industrial environments are particularly prone to these fluctuations. Making a wireless connection able to fulfill the requirements of industrial applications, in terms of determinism, resilience, and bounded latencies, in these harsh conditions makes the automatic tuning of communication parameters to channel quality variations truly challenging.

Several methods are available in the scientific literature to improve the quality of a Wi-Fi link, some of which exploit the prediction of the future quality of the wireless channel (which is the target of this work) to proactively react to its variations, in order to maintain stable performance indicators. The dynamic and complex nature of the wireless spectrum makes the adoption of traditional probabilistic approaches based on statistics not so effective. The perceived quality of wireless communication depends on factors such as the number and type of interferers, the communication technologies used, e.g., Wi-Fi (IEEE 802.11), Bluetooth (IEEE 802.15.1), low-rate wireless personal area networks (LR-WPANs, IEEE 802.15.4), etc., the operating environment, the mobility of nodes, and so on. Taken together, all of these effects make reliable channel prediction a very complex task and a significant research area with many practical implications. Among the different key performance indicators to be predicted, this work focuses on the frame delivery ratio (FDR). This quantity can be used by real-time applications to directly infer the future quality of the wireless channel and to timely trigger suitable corrective actions. For instance, knowing what the FDR will be in the near future can be exploited to guarantee real-time constraints.

Generally speaking, machine learning (ML) algorithms can be used to capture temporal dependencies. To address the related challenges, advanced ML models such as convolutional neural networks (CNN), long short-term memory (LSTM), and bidirectional LSTM (Bi-LSTM) are investigated in this work for the first time in the context of Wi-Fi channel quality prediction in industrial scenarios. These models are particularly effective in analyzing sequential data, making them well-suited for predicting the FDR in such contexts. By comparing their performance, this work highlights the strengths and limitations of every single approach, focusing on its ability to handle varying network conditions and to balance computational efficiency with prediction accuracy.

The estimation of the FDR performed by the ML models proposed and compared here can be profitably used to change specific network and application parameters. When the predicted FDR worsens, the application could, e.g., decrease the sampling rate or the allowed maximum amount of concurrent best-effort traffic, to reduce the overall load on the wireless channel and consequently improve its real-time quality. Other important contexts where the prediction of the FDR can be used in practice include the selection at runtime of the best channel for redundant transmission schemes [4], the automatic configuration of network parameters [5], and enhanced handover procedures that may anticipate some operations in

This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART") as well as the Swedish Knowledge Foundation through the research profile NIIT.

Copyright (c) 2025 IEEE. Personal use is permitted.
For any other purposes, permission must be obtained from the IEEE by emailing pubs-permissions@ieee.org.

order to perform an early channel switching [6]. Finally, forecasts can also be exploited to improve the effectiveness of Wi-Fi network digital twins [7]. Suitable algorithms for implementing above mechanisms could be integrated into the wireless adapter driver.

The remainder of this paper is organized as follows. Section II provides an overview of selected related work on wireless channel prediction, with a focus on deep learning models such as CNN, LSTM, and other time series prediction techniques. Section III describes the methodology we used for dataset acquisition, data preprocessing, and model selection. Section IV presents the implementation details of the analyzed ML models, including their architectural design, the selection of hyperparameters, and how hyperparameters are tuned, while Section V reports comparative results concerning the accuracy of CNNs, LSTM, and Bi-LSTM networks, as well as their computational complexity. Section VI discusses the implications of adopting the proposed models in real-world applications that rely on industrial wireless networks, and finally Section VII concludes the paper by summarizing the key achievements and suggesting future research directions.

II. Related Work

The increasing demand for dependability in wireless communications has led to the proposal of various channel quality prediction models. In particular, the application of ML to Wi-Fi has been a constantly growing research trend [8], [9], and the use of ML models has been suggested in several contexts concerning industrial communication, a concrete example being handover management [10].

Due to their simplicity and computational efficiency, traditional statistical approaches [11], [12] are still widely used for providing satisfactory forecasts about the quality of a wireless link. Studies have shown that models based on exponential moving averages (EMA) can effectively predict the FDR based on historical data, offering a cost-effective alternative to complex machine learning models. The linear combination of EMA models (ELC) [13] has also been proposed to improve prediction accuracy over traditional EMA models, with a slight increase in computational complexity.

More recently, common ML models such as artificial neural networks (ANN) of multilayer perceptron (MLP) type [14] have gained attention for channel quality prediction. Interestingly, models like the deep learning encoder-decoder architecture have shown significantly better performance than statistical methods [15], but not for FDR estimation.

LSTM networks [16] have been recognized as an effective tool for predicting the behavior of wireless channels in the case of sequential data [17]. Comparative studies indicate that LSTM-based models outperform conventional methods, such as auto-regression and linear regression, in the prediction of the signal strength for different communication standards, including Wi-Fi and 4G networks [15].

Recent research has also focused on comparing the performance of different ANN architectures, including MLP, CNN, and transformers, for channel prediction [18]. These comparisons demonstrate that while models like LSTM and gated recurrent unit (GRU) work well to capture sequential dependencies, other architectures such as CNNs can be optimized to recognize spatio-temporal patterns in multi-antenna configurations [19]. Although ANN-based models with a large number of parameters provide higher accuracy, they require more computational resources, which can limit their applicability in embedded systems. Hybrid approaches that combine traditional statistical models with machine learning offer a trade-off between accuracy and computational efficiency [20].

III. Method

An overview of the proposed methodologies behind the prediction of the FDR is outlined in Fig. 1. It covers some key stages, including data collection, data processing, and model selection. These stages are aimed at finding a trained model that is usable to predict the FDR without requiring excessive resources in terms of computational power and memory footprint.

A. Data collection

The collected dataset contains more than 186 million patterns, which are acquired on four non-overlapping Wi-Fi channels (1, 5, 9, and 13) in the 2.4 GHz band. This band was selected because it is characterized by intense interfering traffic due to several networks located near the one used for dataset acquisition. The overall duration of the dataset corresponds to more than 1000 days, i.e., more than 250 days for every channel. The evolution of the FDR over time differs significantly for the four considered channels.

For dataset acquisition, many features of the Wi-Fi protocol were disabled, including frame retransmissions, rate adaptation (speed was set to 54 Mbps), backoff, and frame aggregation. This is because our aim was to periodically sample the quality of the channel by performing a single acknowledged (ACK) transmission attempt every 0.5 seconds. Disabling some protocol features and using the selected transmission period had the goal of impacting channel quality as few as possible, by injecting a minimal amount of additional traffic and sampling the channel at a constant rate. The software and hardware used for dataset collection were previously described in [21].

Every pattern in the dataset includes features such as the transmission outcome (success/failure), receive signal strength indicator (RSSI) of the ACK frame, and transmission latency. This work specifically focuses on transmission outcomes. The analysis of the other acquired features is left as future work. In particular, upon the reception at discrete time i of the ACK frame that is sent back by the receiver—which in our case is the access point (AP)—to notify the correct reception of the data frame, the transmission outcome $x_i = 1$ (corresponding to a successful transmission) is logged. Conversely, in the case the ACK frame does not arrive (which could be due to the loss of either the data or the ACK frame), the outcome $x_i = 0$ is logged. We point out that all the features that permit the prediction of the FDR are intentionally acquired on the sending node, so that relevant decisions about countermeasures can be

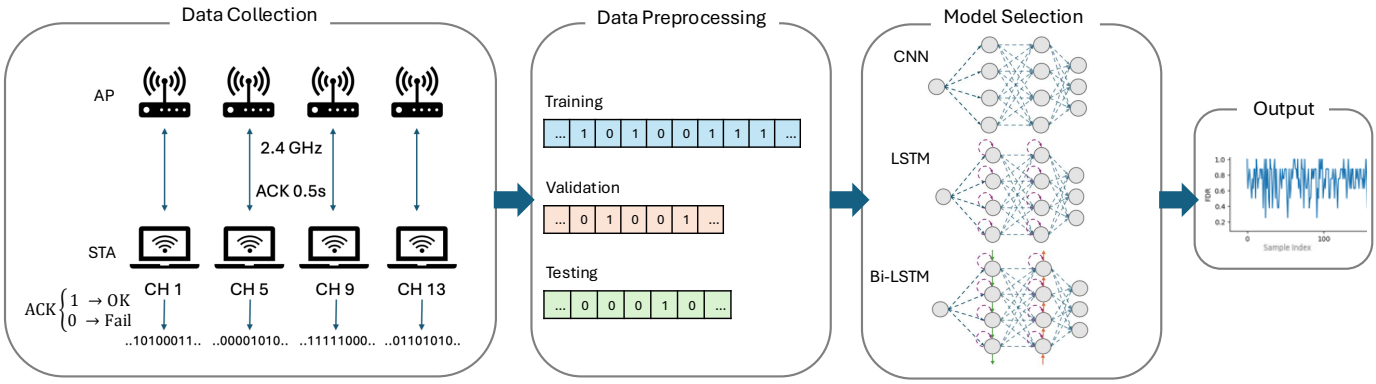

Fig. 1: Main steps of the proposed methodology for predicting Wi-Fi network performance.

made locally, depending on its estimated value. The collected dataset consists of the ordered sequence of all the outcomes $(x_1, ..., x_i, ..., x_N)$ and includes N patterns.

B. Data Preprocessing

The acquired set of data was subdivided into three datasets, disjoint in time. For each one of the four channels, 20% of patterns were selected for testing, 20% for validation, and 60% for training. The two classes of the dataset related to outcomes are quite unbalanced: $x_i = 0$ corresponds to 15% of the patterns, while $x_i = 1$ corresponds to 85% of the patterns. In this context, a simple oversampling technique was first tested, which consists of repeating the patterns related to the underbalanced class to make the cardinality of the two classes equal. However, it was not used, as it did not provide tangible improvements while greatly increasing the training time of the model.

Let i be the current time instant. The corresponding target

$$t_i = \frac{1}{N_\mathrm{f}} \sum_{j=i+1}^{i+N_\mathrm{f}} x_j \tag{1}$$

(i.e., the FDR) is computed by averaging the following $N_\mathrm{f} = 3600$ outcomes, which correspond to 30 minutes.

C. Model Selection

The process of selecting ML models focused firstly on the accuracy of prediction in terms of FDR in non-stationary network conditions, which are typical for real environments. Models were chosen to ensure a balance between simplicity, computational efficiency, and the ability to model temporal dependencies, satisfying the requirements of industrial applications. LSTM models were chosen for their ability to identify patterns in sequential data. Instead, Bi-LSTM models were considered for their ability to capture both past and future dependencies, providing a more comprehensive understanding of temporal patterns. Finally, CNN models were evaluated as a simpler and more efficient alternative for predicting the FDR in devices with low computational power. Performance comparisons among these models are a main goal of this work.

IV. Implementation

This section focuses on the technical decisions and adaptations made while implementing the proposed technique, including architectural design and hyperparameter tuning. All the ML models were trained with the `Keras` module of `TensorFlow`. To be more adherent to the implementation details and to enhance reproducibility, in the next sections the `Keras` nomenclature is directly used to describe the implemented software.

A. Models

The input of each model is a sequence of outcomes of length l, which is shuffled at each training epoch in the case of CNN and is provided to the ML model sequentially in the case of LSTM and Bi-LSTM.

The CNN model consists of one `Conv1D` convolutional layer, which was parameterized by the number of `filters` and the `kernel_size`, and uses *ReLu* as activation function. As usual, a `Flatten` layer is used to transform the output of the convolutional layer in a single dimension. The following layers are three `Dense` layers (four in the case of one experimental configuration). The first two (or three) are based on *ReLu*, while the latter is *linear* because the model was used in a regressive fashion to predict the FDR. A max-pooling layer was not used because it causes information loss when the sequence is time-dependent. A *dropout* layer, which is typically used to reduce overfitting, was tested but not used because it did not improve the prediction accuracy.

The LSTM model consists of one `LSTM` layer with *hyperbolic tangent* activating function, followed by a `Dense` layer with a *linear* output. The Bi-LSTM Model is similar to the LSTM model but includes bidirectional connections to capture dependencies in both the forward and backward directions. This architecture improves the ability of the model, compared to traditional LSTM, to handle complex temporal patterns.

The selected loss function for all models was the *mean squared error* with the *Adam* optimizer, which was chosen due to its robustness in managing varying gradient magnitudes.

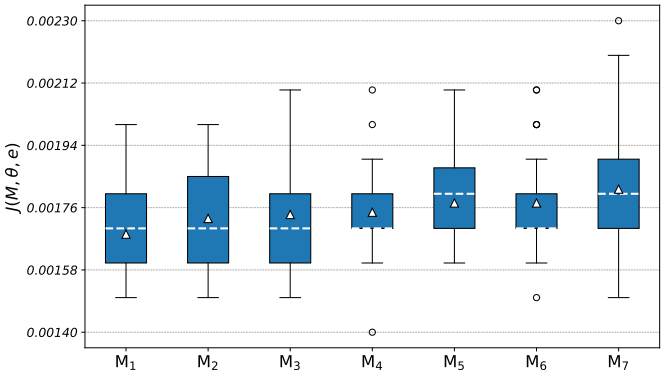

Fig. 2: Box plot showing the loss on the validation dataset $\mathcal{V}$ for channel 1 concerning the seven best configurations related to the CNN model.

B. Hyperparameter tuning

Hyperparameter optimization played a critical role in achieving optimal performance for each model. Different combinations of learning rate, batch size, training epochs, sequence lengths l, number of units, number of layers, and learning rate decay policy were explored to find the best model parameterization. The learning rate decay was implemented to improve convergence during training, by halving the learning rate each epoch. Early stopping was used in all the experimental campaigns to monitor validation loss and prevent overfitting.

To better select the optimal hyperparameters, the `fmin` function of the `hyperopt` Python library was employed, with the goal of minimizing the following objective function:

$$J(M,\theta) = \sum_{p_i,t_i \in \mathcal{V}} \Big(t_i - f_M(p_i,\theta)\Big)^2, \tag{2}$$

where $J(M,\theta)$ is the loss function based on the mean squared error provided by model M (i.e., CNN, LSTM, or Bi-LSTM) with hyperparameters θ, p_i and t_i are the input patterns and targets computed on the validation dataset $\mathcal{V}$, and $f_M(p_i,\theta)$ is the prediction of model M evaluated with the input pattern p_i and the model hyperparameters θ.

All the hyperparameters θ examined by the `fmin` function and the quality of the models obtained for each training epoch were analyzed. Let $J(M,\theta,\tau)$ be the loss computed for model M, hyperparameter θ, and training epoch τ. All these $J(M,\theta,\tau)$ values are summarized by the box plot in Fig. 2 in the case of the CNN model trained and tested on the dataset acquired on channel 1. Among all the analyzed configurations, the best seven were reported in order of *average loss* $\overline{J}(M,\theta)$ evaluated on the validation dataset $\mathcal{V}$.

For any model M characterized by hyperparameters θ, the average loss $\overline{J}(M,\theta)$ is computed by averaging the loss $J(M,\theta,\tau)$ over all the training epochs (excluding the first five ones):

$$\overline{J}(M,\theta) = \frac{1}{N_\tau - 5} \sum_{\tau=6}^{N_\tau} J(M,\theta,\tau), \tag{3}$$

where N_τ is the number of training epochs. The first five epochs are excluded because, for them, the model has not reached a stable behavior yet in terms of the loss. From the box plot, it can be observed that model quality and stability highly depend on the choice of hyperparameters θ. Since the model's losses $J(M,\theta,\tau)$ fluctuate significantly for the different training epochs, the selection of the best hyperparameters is based on $\overline{J}(M,\theta)$, which is marked with white triangles in each box of the plot. The dashed white line in the same plot represents the median.

C. Selected hyperparameters

The selected best hyperparameters θ^*, given a model M, are based on the hyperparametes θ that minimize $\overline{J}(M,\theta)$:

$$\theta^* = \arg\min_{\theta} \overline{J}(M,\theta). \tag{4}$$

The experimental evaluation discussed in the next Section V reports the results for each individual channel (i.e., "ch1", "ch5", "ch9", and "ch13" conditions for channels 1, 5, 9, 13, respectively) and those where the individual channel datasets were concatenated (i.e., "all" condition). The latter "all" condition helps understanding the generalization capabilities of the proposed ML algorithm using only a single model for all the channels. In the case of individual channels, a single parameterization in terms of the selected hyperparameters was chosen for all of them. This led to a slight worsening of the prediction accuracy but, at the same time, to a choice of the hyperparameters that does not overfit (i.e., it is not specific to) the characteristics of the given channel.

For the case of the single channel analysis, hyperparameters θ^* are selected in order to minimize the average loss on the four channels as follows:

$$\begin{aligned} \theta^* &= \arg\min_{\theta} \frac{1}{4} \cdot \Big(\overline{J}(M,\theta)_{\text{ch1}} + \overline{J}(M,\theta)_{\text{ch5}} + \\ &+ \overline{J}(M,\theta)_{\text{ch9}} + \overline{J}(M,\theta)_{\text{ch13}}\Big), \end{aligned} \tag{5}$$

where $\overline{J}(M,\theta)_{\text{ch1}}$ is the average loss evaluated on the validation dataset related to "ch1", $\overline{J}(M,\theta)_{\text{ch5}}$ is the average loss evaluated on the validation dataset related to "ch5", etc.

Instead, for the "all" condition, the outcome provided by (4) can be directly used to select the best hyperparameters θ^*_{all}. In this case, the validation dataset $\mathcal{V}$ is the concatenation of the validation datasets of the four channels. All the selected hyperparameters, which were used in the experimental campaign, are reported in Table I. As can be noticed from the table, the selected hyperparameters for every single "ch" condition and the "all" condition do not differ significantly. An interesting point is the use of a more complex model structure in the case of "all" (i.e., four dense layers in the case of CNN, and two dense layers in the case of LSTM and Bi-LSTM), to have a model with enough parameters for learning the larger amount of information contained in a database that is four times larger.

TABLE I: Model parameters for CNN, LSTM, and Bi-LSTM ("ch" and "all" conditions)

Parameter	**CNN**		**LSTM / Bi-LSTM**	
	ch	**all**	**ch**	**all**
Input Sequence Length (l)	3600	3600	1200	1200
Batch Size	64	128	32	64
Epochs (N_τ)	30	40	15	25
Optimizer	Adam	Adam	Adam	Adam
Loss Function	Mean Squared Error	Mean Squared Error	Mean Squared Error	Mean Squared Error
Learning Rate	0.01 (decay)	0.005 (decay)	0.01 (decay)	0.005 (decay)
Number of Filters/Units	128 (filters)	256 (filters)	25 (LSTM units)	25,50 (LSTM units)
Kernel Size	3	5	N/A	N/A
Pooling	MaxPooling (size 2)	MaxPooling (size 2)	N/A	N/A
Dense Layers	3 (128, 64, 1 unit)	4 (256, 128, 64, 1 unit)	1 (1 unit)	2 (8, 1 unit)
Activation Functions	ReLU (Conv1D/Dense)	ReLU (Conv1D/Dense)	Tanh	Tanh

TABLE II: Metrics for the accuracy of the CNN model across different channels

Test Ch.	Train Ch.	Model	μ_{e^2}	e^2_{p90}	e^2_{p95}	e^2_{p99}	$e^2_{\max}$	$\mu_{\lvert e\rvert}$	$\sigma_{\lvert e\rvert}$	$\lvert e\rvert_{\mathrm{p90}}$	$\lvert e\rvert_{\mathrm{p95}}$	$\lvert e\rvert_{\mathrm{p99}}$	$\lvert e\rvert_{\max}$	$e_{\min}$	e_{p5}	e_{p95}	$e_{\max}$
			$[\cdot 10^{-3}]$					[%]						[%]			
1	1	CNN	2.15	5.64	11.77	30.26	100.24	2.80	3.69	7.51	10.85	17.40	31.66	-31.63	-6.41	4.69	31.66
1	All	CNN	2.03	5.53	11.62	29.72	99.83	2.73	3.63	7.47	10.79	17.28	31.48	-31.54	-6.34	4.63	31.44
5	5	CNN	1.17	1.58	2.70	17.56	176.41	2.18	2.64	3.97	5.20	13.25	42.00	-42.00	-5.20	3.61	42.00
5	All	CNN	1.13	1.54	2.67	17.43	175.88	2.15	2.62	3.93	5.16	13.19	41.83	-41.89	-5.17	3.57	41.73
9	9	CNN	4.23	5.92	7.81	60.43	248.19	4.89	4.29	7.69	8.84	24.58	49.82	-49.82	-2.10	6.01	49.82
9	All	CNN	4.11	5.86	7.74	60.21	247.47	4.83	4.25	7.63	8.79	24.55	49.78	-49.85	-2.09	5.97	49.77
13	13	CNN	4.89	9.88	14.61	22.94	28.45	6.10	3.42	9.94	12.09	15.14	16.87	-13.27	-9.52	11.98	16.87
13	All	CNN	4.73	9.73	14.49	22.79	28.33	6.04	3.37	9.87	12.06	15.08	16.74	-13.22	-9.47	11.87	16.72

V. Results

This section presents the performance evaluation of the CNN, LSTM, and Bi-LSTM FDR prediction models.

A. CNN accuracy

The first experimental campaign was targeted at analyzing the prediction accuracy of the CNN model. The outcomes of this evaluation are reported in Table II, which provides several statistical indicators computed when the CNN model is tested separately on every one of the four channels included in the dataset (i.e., "ch1", "ch5", "ch9", "ch13" conditions). In the "all" condition, the model was trained using the concatenation of all the four channels. The average FDR was computed on the test datasets to have an indication of the channels' quality. For the four individual channels and for the "all" condition, the average FDR is 88.4%, 92.4%, 86.3%, 72.3%, and 85.3%, respectively. For the same sequence of test conditions, their standard deviations are 8.9%, 3.3%, 6.4%, 6.9%, and 10.1%, respectively.

Analyzed statistical indicators include metrics such as the mean squared error (μ_{e^2}), mean absolute error ($\mu_{|e|}$), signed error (e), some relevant percentiles on errors ($e^2_{p90}, e^2_{p95}, e^2_{p99}$, $|e|_{p90}, |e|_{p95}, |e|_{p99}, e_{p5}, e_{p95}$), and maximum values ($e^2_{\max}$, $|e|_{\max}$, $e_{\max}$). For the signed error, also the minimum is reported ($e_{\min}$). Percentiles play an important role in soft real-time systems, such as for latency in wireless communication, where deadlines can only be met probabilistically.

Results show that the FDR prediction related to channels 9 and 13 is less accurate than the other channels (the average absolute error $\mu_{|e|}$ for them is $\sim 4.9\%$ and 6.1%, respectively). This is due to the less stable conditions of these two channels.

One interesting outcome is that, for all methods, results related to the "all" condition are better than those regarding single channels. On the one hand, as expected, this improvement shows that CNNs (but also other tested models) scale well as the size of the training dataset increases. On the other hand, even more importantly, this demonstrates that a general-purpose channel-independent model not only retains the performance of any single channel model (when compared with a channel-dependent specific training, such as the "ch1" condition) but provides a better approach by exploiting the knowledge acquired on other channels. In practice, taking the

TABLE III: Metrics for the accuracy of the LSTM model across different channels

Test Ch.	Train Ch.	Model	μ_{e^2}	e^2_{p90}	e^2_{p95}	e^2_{p99}	$e^2_{\max}$	$\mu_{\lvert e\rvert}$	$\sigma_{\lvert e\rvert}$	$\lvert e\rvert_{\mathrm{p90}}$	$\lvert e\rvert_{\mathrm{p95}}$	$\lvert e\rvert_{\mathrm{p99}}$	$\lvert e\rvert_{\max}$	$e_{\min}$	e_{p5}	e_{p95}	$e_{\max}$
			$[\cdot 10^{-3}]$					[%]						[%]			
1	1	LSTM	2.08	5.59	11.68	30.00	99.90	2.77	3.67	7.48	10.82	17.35	31.50	-31.60	-6.39	4.67	31.50
1	All	LSTM	1.98	5.45	11.50	29.60	99.50	2.68	3.60	7.42	10.70	17.20	31.40	-31.50	-6.30	4.60	31.30
5	5	LSTM	1.15	1.57	2.69	17.50	176.30	2.16	2.63	3.95	5.18	13.23	41.95	-42.10	-5.21	3.60	42.10
5	All	LSTM	1.10	1.51	2.63	17.35	175.75	2.10	2.61	3.90	5.12	13.12	41.80	-41.85	-5.15	3.55	41.70
9	9	LSTM	4.21	5.91	7.80	60.40	247.90	4.88	4.27	7.68	8.85	24.60	49.90	-49.95	-2.12	6.00	49.95
9	All	LSTM	4.12	5.85	7.72	60.18	247.40	4.82	4.24	7.62	8.80	24.50	49.80	-49.88	-2.08	5.96	49.85
13	13	LSTM	4.87	9.87	14.60	22.90	28.42	6.11	3.42	9.92	12.08	15.12	16.84	-13.25	-9.53	11.95	16.84
13	All	LSTM	4.75	9.75	14.50	22.82	28.30	6.05	3.38	9.88	12.05	15.05	16.75	-13.20	-9.48	11.85	16.78

TABLE IV: Metrics for the accuracy of the Bi-LSTM model across different channels

Test Ch.	Train Ch.	Model	μ_{e^2}	e^2_{p90}	e^2_{p95}	e^2_{p99}	$e^2_{\max}$	$\mu_{\lvert e\rvert}$	$\sigma_{\lvert e\rvert}$	$\lvert e\rvert_{\mathrm{p90}}$	$\lvert e\rvert_{\mathrm{p95}}$	$\lvert e\rvert_{\mathrm{p99}}$	$\lvert e\rvert_{\max}$	$e_{\min}$	e_{p5}	e_{p95}	$e_{\max}$
			$[\cdot 10^{-3}]$					[%]						[%]			
1	1	Bi-LSTM	2.07	5.57	11.65	29.95	99.85	2.76	3.66	7.46	10.80	17.30	31.45	-31.55	-6.37	4.65	31.45
1	All	Bi-LSTM	1.97	5.50	11.48	29.50	99.45	2.70	3.60	7.42	10.70	17.18	31.35	-31.50	-6.32	4.62	31.32
5	5	Bi-LSTM	1.14	1.56	2.68	17.48	176.00	2.14	2.62	3.94	5.15	13.20	41.80	-41.85	-5.18	3.58	41.85
5	All	Bi-LSTM	1.11	1.53	2.65	17.32	175.70	2.12	2.60	3.92	5.12	13.15	41.75	-41.80	-5.15	3.55	41.70
9	9	Bi-LSTM	4.19	5.90	7.78	60.38	247.50	4.87	4.26	7.67	8.83	24.58	49.88	-49.90	-2.11	5.98	49.88
9	All	Bi-LSTM	4.08	5.84	7.70	60.12	247.20	4.80	4.23	7.63	8.78	24.50	49.75	-49.85	-2.08	5.95	49.82
13	13	Bi-LSTM	4.85	9.85	14.58	22.88	28.40	6.09	3.41	9.91	12.06	15.10	16.82	-13.20	-9.51	11.92	16.82
13	All	Bi-LSTM	4.72	9.72	14.45	22.78	28.22	6.02	3.37	9.87	12.02	15.02	16.70	-13.15	-9.45	11.80	16.75

test performed on channel 1 as the reference, μ_{e^2} decreases from 2.15×10^{-3} in the "ch1" condition to 2.03×10^{-3} in the "all" condition. Similar improvements have been observed for the other channels and statistical indicators. For instance, channel 13 is characterized by a rather high average error (e.g., $\mu_{e^2} = 4.89 \times 10^{-3}$ and $\mu_{|e|} = 6.10\%$ for the "ch13" condition) in the prediction of the FDR. In this case, the 99 percentile $|e|_{\mathrm{p99}}$ slightly decreases from 15.14% for the "ch13" condition to 15.08% for the "all" condition, which is comparable to the test performed on other channels. In the "all" condition, having $|e|_{\mathrm{p99}} = 15.08\%$ means that the FDR prediction obtained by the model is affected, in 99% of the cases, from an absolute error $|e|$ less than or equal to 15.08%. In terms of 99-percentile of the error, the worst channel is channel 9.

B. LSTM / Bi-LSTM accuracy

Under the same experimental conditions used for CNN, the LSTM and Bi-LSTM models were analyzed. Results are reported in Tables III and IV for the first and the latter model, respectively. The LSTM model, but especially the Bi-LSTM, outperformed CNN in all metrics, particularly in dynamic environments like those encountered on channels 9 and 13. The ability of LSTM and Bi-LSTM to model temporal dependencies allowed slightly more accurate predictions in these conditions characterized by higher variability.

Going slightly deeper into the analysis of the results, it is clear that the enhancements in terms of prediction quality achieved through LSTM are marginal. The biggest improvement regarding the mean squared error (i.e., the quality minimized during the training of the model) was obtained on the experimental results related to channel 1 and the "ch1" condition, where the prediction mean squared error decreases from $\mu_{e^2} = 2.15 \times 10^{-3}$ in the case of CNN to $\mu_{e^2} = 2.08 \times 10^{-3}$ in the case of LSTM. The improvement, in terms of the mean absolute error, corresponds to $\mu^{\mathrm{LSTM}}_{|e|} - \mu^{\mathrm{CNN}}_{|e|} = 0.03\%$.

The use of Bi-LSTM provides further (slight) improvements. In this case, for the test performed on channel 9 and the "all" condition, the mean squared error μ_{e^2} decreases from 4.12×10^{-3} to 4.08×10^{-3}, with a reduction in terms of the absolute error of just $\mu^{\mathrm{Bi-LSTM}}_{|e|} - \mu^{\mathrm{CNN}}_{|e|} = 0.02\%$.

In order to obtain more comprehensive indications about the best model for practical applications in real-world scenarios, the next section will compare the three models in terms of computational complexity and memory footprint.

TABLE V: Computational complexity of the CNN, LSTM, and Bi-LSTM models

Model (Condition)	**Mean response time** (ms)	**Memory footprint** (MB)	**Peak memory** (MB)
CNN (ch)	2.7	0.02	0.03
LSTM (ch)	20.3	0.16	0.67
Bi-LSTM (ch)	42.2	0.17	0.69
CNN (all)	5.2	0.02	0.03
LSTM (all)	36.3	0.17	0.68
Bi-LSTM (all)	78.0	0.17	0.69

C. Computational Complexity

The technique presented in this work is meant to be implemented in embedded devices with low computational power, small amounts of dynamic memory, and sometimes limited battery capacity. For this reason, the efficiency of the three presented ML algorithms in terms of both computational demand and memory footprint was analyzed.

All experiments regarding computational complexity were performed on a PC equipped with $8\,\mathrm{GB}$ of DRAM and an Intel Core i3-10105 CPU running with a base frequency of $3.7\,\mathrm{GHz}$ and a maximum frequency of $4.4\,\mathrm{GHz}$. Instead, the models were trained on the BEN cluster, a high-performance computing system equipped with 73 IBM POWER9 nodes, 292 NVIDIA Tesla V100 GPUs, $62\,\mathrm{TB}$ DRAM, and $1.6\,\mathrm{PB}$ storage. As a reference, the training of the model on a single CPU core of the BEN cluster took about 3 days for any "ch" condition and 7 days for the "all" condition for CNN, about 7 days for any "ch" condition and 15 days for the "all" condition for LSTM, and about 8 days for any "ch" condition and 24 days for the "all" condition for Bi-LSTM.

For testing, we purposely used a PC with low computational power, because our main interest is evaluating performance on commercial low-end devices. The results of the experimental campaign are shown in Table V. Regarding the memory footprint of the test phase, the average and peak memory footprints are small enough to be handled on inexpensive embedded devices with limited computational resources. The maximum footprint was always less than $0.7\,\mathrm{MB}$ for all methods. In any condition, the occupancy of CNN is almost one order of magnitude smaller than the occupancy of LSTM and Bi-LSTM.

The results regarding the execution time are very interesting. The CNN model, in its most performant version (i.e., the "all" condition), had an execution time of $5.2\,\mathrm{ms}$, while that of LSTM rose to $36.3\,\mathrm{ms}$ and that of Bi-LSTM doubled to $78.0\,\mathrm{ms}$. This leads us to the conclusion that the improvements provided by LSTM/Bi-LSTM are probably not enough to justify the adoption of these models, as they lead to an unjustified increase in response time during the online usage of the model (i.e., during the test phase). On the contrary, the results about execution time show how the CNN model can be implemented on machines with low computational power.

As reported in [22], throughput and latency of low-power ARM architectures are mostly comparable to conventional low-end PCs, suggesting that the implementation of our solutions on embedded devices is technically feasible without significantly compromising performance. In particular, [21] shows that the performance of commercial x86 CPUs (like those from the Intel Core i3 family) are just one order of magnitude higher than popular ARMv7 architectures (e.g., a Raspberry Pi 2) in the considered neural network inference task. This means that the execution times on ARMv7 are expected to be compatible with several real-time applications, which makes this platform a suitable choice for embedded devices like commercial APs.

VI. Discussion

This study highlights the strengths of CNN, LSTM, and Bi-LSTM models in predicting the performance of a Wi-Fi link in industrial settings. The CNN model demonstrated high computational efficiency, with shorter response time and lower resource requirements, making it a viable solution for real-time applications. On the contrary, LSTM and Bi-LSTM models required more computational resources and, when compared with CNN, provided only small improvements in terms of the prediction accuracy for the FDR. Bi-LSTM achieved slightly better accuracy than LSTM in some scenarios, at the cost of additional complexity. The bidirectional architecture of the Bi-LSTM model allowed it to capture both past and future dependencies in time series data, which was especially beneficial in the most challenging and highly dynamic environments (i.e., channels 9 and 13). However, the increased computational overhead suggests using the CNN model.

Future work will explore hybrid models combining LSTM with other architectures, such as CNN or transformer, to improve spatial-temporal accuracy. For example, a CNN-LSTM model could capture spatial correlations from FDR variations while maintaining the LSTM's ability to model sequential dependencies over time. Transformer-based models, which are deemed more advanced for handling long-term dependencies, could be an alternative to recurrent networks.

While this study focuses on predicting the FDR, additional communication network metrics, such as interference levels and latency, could be analyzed and predicted. Moreover, further features could be included as input to the model with the goal of improving prediction accuracy. Future research activities should also assess model performance under different mobility conditions, as many modern industrial environments involve moving devices (e.g., autonomous mobile robots), that must be interconnected over the air to coordinate their actions.

Testing the models on additional datasets, acquired with varying network conditions, would help validate their robustness, as well as to determine to what extent they can be generalized for broader applications. Any improvements concerning the computational and memory requirements of the models will permit to increase the number and type of edge devices on

which the proposed techniques can be implemented. Finally, addressing class imbalance and optimizing hyperparameters were important steps, and further exploring automated optimization techniques could enhance performance.

VII. Conclusion

This study analyzes the effectiveness of CNN, LSTM, and Bi-LSTM in predicting the FDR in Wi-Fi networks. Despite their superior ability to address challenges such as class imbalance and variability, LSTM-type models outperform CNN only in some specific conditions, which are characterized by high FDR variability. All the proposed models are compatible with embedded edge devices, which makes them suitable for real-time industrial applications. However, both LSTM and Bi-LSTM consistently require more computational resources than CNN. This makes CNNs the solution this work recommends for FDR prediction.

One of the key findings is that training the models on a dataset acquired on multiple Wi-Fi channels ("all" condition) leads to better performance than training them on individual channels (generalization). This suggests that knowledge transfer between different channel conditions can help improve robustness and adaptability in real-world deployments.

As future work, experiments should explore the performance of the model in more diverse network environments, including 5 GHz and 6 GHz bands, Wi-Fi 6 and 7, multihop wireless networks, and emerging 6G communication systems. Testing in real-world deployments, such as factory automation and smart city infrastructures, will be essential to validate the robustness and practicality of these models in real operational conditions.

References

[1] S. Scanzio, H.-P. Bernhard, D. Cavalcanti, G. Cena, L. Shu, I. Val, and L. Wisniewski, "IEEE Access Special Section: Advances on High Performance Wireless Networks for Automation and IIoT," *IEEE Access*, vol. 11, pp. 142 748–142 753, 2023.

[2] I. Behnke and H. Austad, "Real-Time Performance of Industrial IoT Communication Technologies: A Review," *IEEE Internet of Things Journal*, vol. 11, no. 5, pp. 7399–7410, 2024.

[3] S. Scanzio, L. Wisniewski, and P. Gaj, "Heterogeneous and dependable networks in industry – A survey," *Computers in Industry*, vol. 125, p. 103388, 2021.

[4] G. Cena, S. Scanzio, and A. Valenzano, "A Prototype Implementation of Wi-Fi Seamless Redundancy with Reactive Duplication Avoidance," in *IEEE 23rd Int. Conf. on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2018, pp. 179–186.

[5] M. Friesen, L. Wisniewski, and J. Jasperneite, "Machine Learning for Zero- Touch Management in Heterogeneous Industrial Networks - A Review," in *IEEE 18th Int. Conf. on Factory Communication Systems (WFCS)*, 2022, pp. 1–8.

[6] M. A. Khan, R. Hamila, A. Gastli, S. Kiranyaz, and N. A. Al-Emadi, "ML-Based Handover Prediction and AP Selection in Cognitive Wi-Fi Networks," *Journal of Network and Systems Management*, vol. 30, no. 4, p. 72, Aug 2022.

[7] S. Sudhakaran, J. Perez-Ramirez, D. Cavalcanti, C. Cazan, N. Olson, R. Rosales, and V. Frascolla, "Wireless Network Digital Twin Calibrated by Real Time Telemetry and XR Feedback Interface," in *IEEE 20th Int. Conf. on Factory Communication Systems (WFCS)*, 2024, pp. 1–8.

[8] F. Wilhelmi, S. Szott, K. Kosek-Szott, and B. Bellalta, "Machine Learning and Wi-Fi: Unveiling the Path Toward AI/ML-Native IEEE 802.11 Networks," *IEEE Communications Magazine*, pp. 1–7, 2024.

[9] S. Szott, K. Kosek-Szott, P. Gawłowicz, 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 Communications Surveys & Tutorials*, vol. 24, no. 3, pp. 1843–1893, 2022.

[10] P. Avila-Campos, J. Haxhibeqiri, X. Jiao, I. Moerman, and J. Hoebeke, "Optimizing Handover in Time-Sensitive Wi-Fi Networks through Machine Learning," in *IEEE 29th Int. Conf. on Emerging Technologies and Factory Automation (ETFA)*, 2024, pp. 1–8.

[11] G. Formis, S. Scanzio, G. Cena, and A. Valenzano, "Predicting Wireless Channel Quality by Means of Moving Averages and Regression Models," in *IEEE 19th Int. Conf. on Factory Communication Systems (WFCS)*, 2023, pp. 1–8.

[12] I. Tinnirello and G. Bianchi, "Rethinking the IEEE 802.11e EDCA Performance Modeling Methodology," *IEEE/ACM Transactions on Networking*, vol. 18, no. 2, pp. 540–553, 2010.

[13] G. Formis, S. Scanzio, G. Cena, and A. Valenzano, "Linear Combination of Exponential Moving Averages for Wireless Channel Prediction," in *IEEE 21st Int. Conf. on Ind. Informat. (INDIN)*. IEEE, 2023, pp. 1–6.

[14] S. Scanzio, F. Xia, G. Cena, and A. Valenzano, "Predicting Wi-Fi link quality through artificial neural networks," *Internet Technology Letters*, vol. 5, no. 2, p. e326, 2022.

[15] J. D. Herath, A. Seetharam, and A. Ramesh, "A deep learning model for wireless channel quality prediction," in *IEEE Int. Conf. on Communications (ICC)*, 2019, pp. 1–6.

[16] Y. Kanto and K. Watabe, "Wireless Link Quality Estimation Using LSTM Model," in *NOMS 2024-2024 IEEE Network Operations and Management Symposium*. IEEE, 2024, pp. 1–5.

[17] O. Stenhammar, G. Fodor, and C. Fischione, "A comparison of neural networks for wireless channel prediction," *IEEE Wireless Commun.*, 2024.

[18] H. Jiang, M. Cui, D. W. K. Ng, and L. Dai, "Accurate channel prediction based on transformer: Making mobility negligible," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 9, pp. 2717–2732, 2022.

[19] T. Liu and A. E. Cerpa, "Temporal adaptive link quality prediction with online learning," *ACM Transactions on Sensor Networks (TOSN)*, vol. 10, no. 3, pp. 1–41, 2014.

[20] A. Ericson, S. Forsström, and K. Thar, "IIoT Intrusion Detection using Lightweight Deep Learning Models on Edge Devices," in *2024 IEEE 20th Int. Conf. on Factory Communication Systems (WFCS)*. IEEE, 2024, pp. 1–8.

[21] A. S. Colletto, S. Scanzio, G. Formis, and G. Cena, "On the Use of Artificial Neural Networks to Predict the Quality of Wi-Fi Links," *IEEE Access*, vol. 11, pp. 120 082–120 094, 2023.

[22] E. Gamess and S. Hernandez, "Performance Evaluation of Different Raspberry Pi Models for a Broad Spectrum of Interests," *International Journal of Advanced Computer Science and Applications*, vol. 13, 01 2022.