IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 29 July 2025, accepted 21 August 2025, date of publication 28 August 2025, date of current version 8 September 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3603920

RESEARCH ARTICLE

Air Quality Prediction via Embedded ML/DL and Quantized Models

ARMIN MAZINANI[1], DANIELE ANTONUCCI[1], DANILO PIETRO PAU[2], (Fellow, IEEE), LUCA DAVOLI[1], (Member, IEEE), AND GIANLUIGI FERRARI[1], (Senior Member, IEEE)

[1]Internet of Things (IoT) Laboratory, Department of Engineering and Architecture, University of Parma, 43124 Parma, Italy
[2]System Research and Applications, STMicroelectronics, 20864 Agrate Brianza, Italy

Corresponding author: Luca Davoli (luca.davoli@unipr.it)

The work of Armin Mazinani, Luca Davoli, and Gianluigi Ferrari was supported in part by European Union–NextGenerationEU through the Italian National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.3, partnership on "Telecommunications of the Future" (PE00000001-program "[Research and Innovation on Future Telecommunications Systems and Networks, to make Italy more Smart (RESTART)]"), and Spoke 5 "Industrial and Digital Transition Networks" (IN), "Technologies, Algorithms, and Protocols for Use Cases of Industrial Networks" (TAP-IN) Project, under Grant CUP J33C22002880001.

ABSTRACT In the people's everyday life, one of the most significant (and unfortunately well-known) environmental problems with substantial impact and effects is *air pollution*. To this end, extensive research (e.g., conducted by the World Health Organization, WHO) has uncovered a non-negligible correlation between hazardous environments and airborne Particulate Matters (PMs), emphasizing the importance of monitoring pollution to mitigate its adverse effects. In fact, since PMs are microscopic particles of solid or liquid materials suspended in the atmosphere and have negative effects on both weather and precipitations, it has been highlighted that the diffusion of diseases through the inhalation or absorption of contaminants is a concerning outcome of polluted environments. Hence, focusing on the critical need to monitor air quality in common (livable and work) environments (e.g., airports, public transports, or any closed environment) using affordable, compact IoT devices capable of collecting and processing environmental data, as well as forecasting future air quality trends, in this paper we present an experimental evaluation of several Machine Learning (ML), Deep Learning (DL), and quantized DL models designed to accurately predict air pollution (in terms of $PM_{2.5}$ concentration). Furthermore, we conduct a comparative analysis of state-of-the-art DL models, unveiling the *trade-offs* associated with each ML/DL model, and discussing their practical applications and potential deployment on tiny IoT devices for on-site computing. Our results show that Convolutional Neural Network-Bidirectional Gated Recurrent Unit (CNN-BiGRU) is the best performing model. According to the obtained results, using 8-bit quantization on CNN-BiGRU allows to achieve a 66% model size's reduction with only a 1% drop in average prediction accuracy, making it a balanced and efficient choice for several applications.

INDEX TERMS Air quality, deep learning, forecasting, Internet of Things, machine learning, tiny devices.

I. INTRODUCTION

Environmental contamination (and, in particular, air pollution) has been a commonly discussed issue in recent years, with Particulate Matters (PMs)—short particles or droplets suspended in the atmosphere having the potential to be breathed and penetrate the bloodstream—being a major concern due to their harmful impacts on human health [1], [2]. In fact, the World Health Organization (WHO) estimates that illnesses caused by air pollution are responsible for more than 7M premature deaths annually [3]. Furthermore, according to [4], a daily exposure to 10 $\mu g/m^3$ of fine PM increases mortality risk by up to 1% for respiratory diseases, especially with lung cancer (where mortality rate rises by 15 to 27%), and cardiovascular disease mortality by 10%. Additionally, hospital admissions for non-communicable respiratory conditions, namely asthma, pneumonia, flu, and bronchiolitis, increase by up to 4%, 3.9%, and 7%, respectively. air

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

pollution increases the risk of neurodegenerative diseases such as Alzheimer's and Parkinson's [5], [6]. Then, air pollution poses risks also to plants and animals: with the air mainly composed of nitrogen (N_2, about 78%), oxygen (O_2, about 21%, particularly important for plants and animals' growth and development), carbon dioxide (CO_2, in small percentage, useful for the photosynthesis of green plants), water vapor, and other compounds, an excessive increase of pollutants in the air could alter the correct development and lead to pathologies [7]. This further motivates the need to define proactive long-term strategies to mitigate pollution (as much as possible), especially for population well-being [8], [9]. Therefore, such strategies might be enabled through enhanced monitoring systems defined on the basis of Internet of Things (IoT)-oriented well-known architectures and widely adopted processing and communication patterns—e.g., based on Machine Learning (ML), Deep Learning (DL), and quantized models [10]. In particular, this allows to take properly into account non-linear relationships between input variables (i.e., time series [11]) and process large amounts of data, in the end obtaining a higher prediction accuracy than that with traditional approaches [12].

In this paper, a comparative performance evaluation (in terms of accuracy-complexity *trade-offs*) of different ML/DL models (namely: Recurrent Neural Network, RNN; Convolutional Neural Network, CNN; Long Short-Term Memory, LSTM; Gated Recurrent Unit, GRU; Legendre Memory Unit, LMU; Temporal Convolutional Network, TCN; Bidirectional GRU, BiGRU; Bidirectional LSTM, BiLSTM; CNN-GRU, CNN-LSTM, CNN-BiGRU, CNN-BiLSTM) used to predict $PM_{2.5}$-related Air Quality Index (AQI) [13], is discussed. Then, different evaluation metrics (namely: Root Mean Square Error, RMSE; Mean Absolute Error, MAE; Mean Absolute Percentage Error, MAPE; coefficient of determination, denoted as R^2) and various values of the time lag—corresponding to the number of past observations to be considered to predict the next time step observation value—are considered to maximize estimation accuracy. Finally, a Post-Training Quantization (PTQ) technique is applied to evaluate the feasibility of further optimizing (as much as possible) DL models on *tiny* IoT devices, resource-constrained in terms of memory and computational capabilities.

The main contributions of this paper can be summarized as follows: (i) a comparative analysis of different ML and DL models for predicting air quality (in terms of $PM_{2.5}$ concentration) will be presented; (ii) the impact of time lag on the prediction accuracy will be investigated; (iii) the complexity and the memory space required to deploy and run the considered models on *tiny* IoT devices will be considered, in particular evaluating the adoption of PTQ techniques.

The remainder of the paper is organized as follows. In Section II, an overview on related works is given. Section III introduces the considered ML and DL models, while Section IV discusses the proposed data analysis technique. Section V presents the reference air quality dataset, the experimental performance results, and the PTQ technique. Finally, in Section VI conclusions are drawn.

II. LITERATURE REVIEW

In the following, an overview on air quality prediction techniques proposed in the literature is presented, focusing on ML/DL approaches and on the use of IoT devices, respectively.

A. AIR QUALITY PREDICTION THROUGH ML AND DL APPROACHES

Several studies focus on RNN architectures [14], such as LSTM [15] and GRU [16], due to their ability to capture temporal dependencies in air quality samples. These models have shown to successfully predict pollutant concentrations with high accuracy.

In [17], a comprehensive comparison between ML algorithms—including Support Vector Machine (SVM), k-Nearest Neighbors (k-NN), and Random Forest (RF)—to predict atmospheric $PM_{2.5}$ levels in the city of Isfahan, Iran, on the basis of data from 7 stations in the time period 2011–2019, with a total amount of 9 features, is presented. According to the obtained results, Artificial Neural Networks (ANNs) hit a 91.1% accuracy, followed by RF, SVM, and k-NN.

Authors in [18] conduct a comparison between LSTM and Seasonal AutoRegressive Integrated Moving Average with eXogenous regressor (SARIMAX) models [19] for $PM_{2.5}$ prediction in Bangkok in the time period 2017–2018. Their results show that LSTM achieves lower (in terms of 24 hour average value) RMSE and MAE values equal to 6.04 units and 4.86 units, respectively, while SARIMAX returns values equal to 7.99 units and 5.74 units, respectively.

In [20], a model, denoted as Aggregated LSTM (ALSTM), allows to forecast the concentration of $PM_{2.5}$ in five distinct industrial air quality sites in Taiwan. In detail, the adopted dataset consists of 17 environmental features collected in the time period 2012–2017, while the proposed model consists of stacked LSTM layers followed by fully-connected dense layers. The ALSTM model demonstrates superior accuracy when compared to LSTM, Gradient Boosted Tree Regression (GBTR) [21], and Support Vector Regression (SVR) [22] models.

Authors in [23] target the prediction of $PM_{2.5}$ in eight Korean cities, taking into account a dimensionality problem caused by a dataset with numerous variables and limited observations, and reducing it using RNN, LSTM, and BiLSTM models with Principal Component Analysis (PCA) [24]. The obtained results show that using PCA with LSTM and BiLSTM reduces RMSE and MAE by 16.6% and 33.3%, respectively.

In [25], a CNN-LSTM model is proposed in order to predict the $PM_{2.5}$ concentration in Beijing considering spatio-temporal correlations, over a dataset composed of 20,757 pollutant concentration and meteorological

TABLE 1. Literature works categorized along their main features.

Study	Adopted technique	HPO	IoT deployability evaluation	Quantization
[17]	ANN	✗	✗	✗
[18]	LSTM	✗	✗	✗
[20]	Three combined LSTM	✗	✗	✗
[23]	PCA-BiLSTM	✗	✗	✗
[25]	CNN-LSTM	✗	✗	✗
[26]	CNN-BiGRU	✗	✗	✗
[28]	Stacked LSTM	✗	✗	✗
[29]	1D-CNN-BiLSTM	✗	✗	✗
[30]	XGBoost	✓	✓	✗
[31]	GRU	✗	✓	✗
Proposed model	CNN-BiGRU	✓	✓	✓

parameters' observations collected in the time period January 2015–March 2020. According to the obtained results, CNN-LSTM outperforms Multi-Layer Perceptron (MLP) and LSTM. Moreover, a combination of one-dimensional (1D) Convolutional and BiGRU (CBGRU) model for $PM_{2.5}$ short-time prediction is proposed in [26]. In detail, the chosen dataset is composed of 43,8008-feature hourly observations collected in the time period January 2010–December 2014, while the proposed model is composed of two 1D convolutional layers followed by two BiGRU layers. According to the obtained results, CBGRU provides a lower prediction error in comparison to DL models (including LSTM, GRU, and RNN). Similarly, the performances of LSTM, GRU, and BiLSTM are compared in [27], in order to predict $PM_{2.5}$ in three Korean cities (namely: Seoul, Daejeon, and Busan), using a 1-hour interval dataset collected in the time period May 2014–December 2021. The presented results show that BiLSTM is the best performing in terms of long-term predictions.

In [28], a stacked LSTM model is proposed to predict the next hour $PM_{2.5}$ concentration in the city of Istanbul, Turkey, training the model over a dataset collected in the time period January 2015–November 2019. According to the obtained results, the proposed model outperforms LSTM, RNN, and GRU, with an RMSE equal to 25.20 units.

Finally, a 1D-CNN-BiLSTM model with parallel input from the target and neighboring monitoring stations is presented in [29] for the hourly prediction of $PM_{2.5}$. As a result, the suggested method chooses neighboring stations based on distance and wind statistics, proving to be effective when a combined use of target and chosen monitoring stations is performed.

B. AIR QUALITY PREDICTION ON RESOURCE-CONSTRAINED DEVICES

A tiny ML model effective in predicting the AQI using trained Decision Tree (DT), RF, and XGBoost models is proposed in [30]. Although faced with computational limitations, the authors achieve an accuracy of 75.2% adopting XGBoost, that outperforms both RF and DT while complying with a 2 MB memory limitation. Similarly, in [32] a low-cost device—based on a Raspberry Pi Pico as the main micro controller—for air quality monitoring and prediction is proposed. In detail, two models (composed of 2D convolutional layers making an AutoEncoder, AE) are used: the first model forecasts CO_2, temperature, and humidity, by analyzing six hours of past observations; the second model (namely, a model imputer [33]) is used for missing values imputation. Then, with model predictor's and model imputer's sizes of 107,708 and 126,536 B, respectively, the light models' size decrease by 47.7% and 56.6%, respectively.

Finally, in [31] a comparative analysis and performance evaluation (in terms of computational complexity) of different ML and DL models on a tiny IoT device to forecast the $PM_{2.5}$ concentration is proposed. According to the obtained results, GRU provides the most accurate prediction with respect to the other considered models.

1) STATEMENT OF CONTRIBUTION

For the sake of clarity, a tabular comparison between the previously detailed literature works and the approach proposed in this work is shown in Table 1. In the following, the key contributions of our proposed air quality prediction mechanism are listed.

1) This paper provides a comprehensive evaluation of various DL models in terms of both prediction accuracy and computational complexity, addressing *trade-offs* required for their deployment on resource-constrained devices.
2) This paper discusses the implementation of 8-bit quantization to evaluate its influence on the prediction performance, providing insights into the effects of model compression on accuracy and efficiency.
3) This paper highlights the use of HyperParameters Optimization (HPO) to identify the optimal architecture and hyperparameters, thus improving both performance and resource efficiency.
4) This paper focuses on IoT deployability by evaluating models for real-world implementation, demonstrating their practical feasibility in edge computing environments.

III. BACKGROUND

Before investigating the proposed air quality estimation and prediction model (detailed in Section IV), as well as the experimental performance (discussed in Section V), in the following an overview on the considered ML and DL models is presented, to highlight their features.

A. RECURRENT NEURAL NETWORK (RNN)

Due to their simple internal architecture, RNN cells are fundamental for sequential data modeling, making them suitable for time series prediction tasks (such as air quality

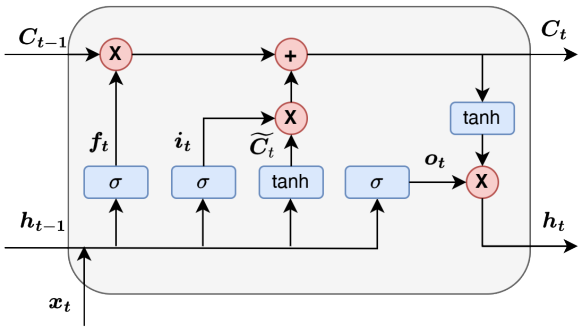

FIGURE 1. Simplified mathematical model of an LSTM.

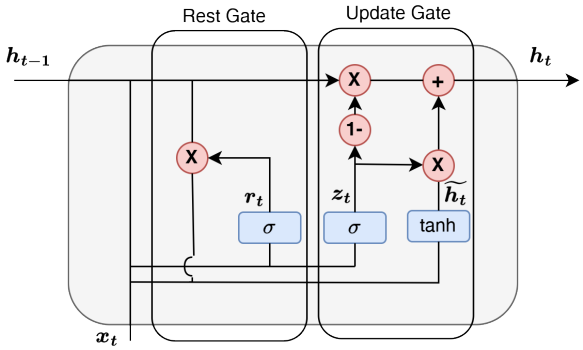

FIGURE 2. Simplified mathematical model of a GRU.

forecasting). In detail, even if traditional RNNs may suffer from the vanishing gradient problem, limiting their ability to capture long-term dependencies effectively, they can be represented as follows:

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{1}$$

$$\boldsymbol{h_t} = \sigma(\mathbb{W}_{hh} \cdot \boldsymbol{h}_{t-1} + \mathbb{W}_{xh} \cdot \boldsymbol{x}_t + \boldsymbol{b}_h) \tag{2}$$

$$\boldsymbol{y}_t = \text{softmax}(\mathbb{W}_{hy} \cdot \boldsymbol{h}_t + \boldsymbol{b}_y) \tag{3}$$

where: σ is the sigmoid activation function; $\boldsymbol{h_t} \in \mathbb{R}^{n_h \times 1}$ is the hidden state at time t, with n_h representing the dimension of the hidden state vector; $\boldsymbol{x_t} \in \mathbb{R}^{n_x \times 1}$ is the input at time t, with n_x corresponding to the dimension of the input vector; $\boldsymbol{y_t} \in \mathbb{R}^{n_y \times 1}$ is the output at time t, n_y representing the dimension of the output vector; $\mathbb{W}_{hh} \in \mathbb{R}^{n_h \times n_h}$, $\mathbb{W}_{xh} \in \mathbb{R}^{n_h \times n_x}$ and $\mathbb{W}_{hy} \in \mathbb{R}^{n_y \times n_h}$ are the weight matrices; $\boldsymbol{b}_h \in \mathbb{R}^{n_h \times 1}$ and $\boldsymbol{b}_y \in \mathbb{R}^{n_y \times 1}$ are bias vectors.

B. LONG SHORT-TERM MEMORY (LSTM)

LSTM networks integrate specialized memory cells and gating mechanisms, enabling to selectively retain or discard information across long sequences. For the sake of comprehension, an LSTM block (whose block representation at a certain time instant t is shown in Figure 1) can be defined as follows:

$$\boldsymbol{f}_t = \sigma(\mathbb{W}_{fh}\boldsymbol{h}_{t-1} + \mathbb{W}_{fx}\boldsymbol{x}_t + \boldsymbol{b}_f) \tag{4}$$

$$\boldsymbol{i}_t = \sigma(\mathbb{W}_{ih}\boldsymbol{h}_{t-1} + \mathbb{W}_{ix}\boldsymbol{x}_t + \boldsymbol{b}_i) \tag{5}$$

$$\boldsymbol{o}_t = \sigma(\mathbb{W}_{oh}\boldsymbol{h}_{t-1} + \mathbb{W}_{ox}\boldsymbol{x}_t + \boldsymbol{b}_o) \tag{6}$$

$$\widetilde{\boldsymbol{C}}_t = \tanh(\mathbb{W}_{ch}\boldsymbol{h}_{t-1} + \mathbb{W}_{cx}\boldsymbol{x}_t + \boldsymbol{b}_c) \tag{7}$$

$$\boldsymbol{C}_t = \boldsymbol{f}_t \cdot \boldsymbol{C}_{t-1} + \boldsymbol{i}_t \cdot \widetilde{\boldsymbol{C}}_t \tag{8}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \cdot \tanh(\boldsymbol{C}_t) \tag{9}$$

where: $\mathbb{W}_{fh}, \mathbb{W}_{ih}, \mathbb{W}_{oh}, \mathbb{W}_{ch} \in \mathbb{R}^{n_h \times n_h}$ (with n_h representing the dimension of the hidden state vector) and $\mathbb{W}_{fx}$, $\mathbb{W}_{ix}$, $\mathbb{W}_{ox} \in \mathbb{R}^{n_h \times n_x}$ are the weight matrices for forget gate, input gate, output gate, and candidate cell state, respectively (with n_x corresponding to the dimension of the input vector); $\boldsymbol{b}_f, \boldsymbol{b}_i, \boldsymbol{b}_o, \boldsymbol{b}_c \in \mathbb{R}^{n_h \times 1}$ are the bias vectors for forget gate, input gate, output gate, and candidate cell state, respectively; $\boldsymbol{h_{t-1}} \in \mathbb{R}^{n_h \times 1}$ is the previous hidden state (output) of the LSTM cell; $\boldsymbol{x_t} \in \mathbb{R}^{n_x \times 1}$ represents the current input at time t; $\boldsymbol{f_t}$ is the output of the forget gate; $\boldsymbol{i_t}$ corresponds to the output for the input gate; $\boldsymbol{o_t} \in \mathbb{R}^{n_h \times 1}$ is the output of the output gate; $\widetilde{\boldsymbol{C}}_t \in \mathbb{R}^{n_h \times 1}$ represents the state of the candidate cell; $\boldsymbol{C_t} \in \mathbb{R}^{n_h \times 1}$ corresponds to the state of the final cell; $\boldsymbol{h_t} \in \mathbb{R}^{n_h \times 1}$ represents the output of the hidden state, which is the output of the LSTM cell at time t; and

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}}. \tag{10}$$

C. GATED RECURRENT UNIT (GRU)

A GRU represents a particular RNN with a simpler structure with respect to an LSTM network, since GRU does not use the *memory cell* and the *forget gate* is removed to reduce complexity. Then, GRUs are effective in capturing sequential dependencies (even if they might not perform as well as LSTMs when dealing with complex long-term dependencies), while being computationally more efficient than LSTMs (using fewer parameters). However, a GRU's cell is composed of two gates, namely *update gate* and *reset gate*. In addition, only the *hidden state* is used in its structure, for the sake of simpler and faster model training [34]. For the sake of completeness, the architecture of a GRU (shown in Figure 2) can be defined as follows:

$$\boldsymbol{z}_t = \sigma(\mathbb{W}_{zh}\boldsymbol{h}_{t-1} + \mathbb{W}_{zx}\boldsymbol{x}_t + \boldsymbol{b}_z) \tag{11}$$

$$\boldsymbol{r}_t = \sigma(\mathbb{W}_{rh}\boldsymbol{h}_{t-1} + \mathbb{W}_{rx}\boldsymbol{x}_t + \boldsymbol{b}_r) \tag{12}$$

$$\widetilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W_{hh}}(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}) + \mathbb{W}_{hx}\boldsymbol{x}_t + \boldsymbol{b}_h) \tag{13}$$

$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \widetilde{\boldsymbol{h}}_t \tag{14}$$

where: $\boldsymbol{z}_t \in \mathbb{R}^{n_h \times 1}$ is the update gate and $\boldsymbol{r}_t \in \mathbb{R}^{n_h \times 1}$ is the reset gate; $\widetilde{\boldsymbol{h}}_t \in \mathbb{R}^{n_h \times 1}$ is the new candidate activation, with n_h corresponding to the dimension of the hidden state vector; $\boldsymbol{h}_t$ is the new hidden state; $\mathbb{W}_{zh}$, $\mathbb{W}_{rx}$, $\mathbb{W}_{hh} \in \mathbb{R}^{n_h \times n_h}$ and $\mathbb{W}_{zx}$, $\mathbb{W}_{rx}$, $\mathbb{W}_{hx} \in \mathbb{R}^{n_h \times n_x}$ are the weight matrices (with n_x representing the dimension of the input vector); $\boldsymbol{b}_z$, $\boldsymbol{b}_r \in \mathbb{R}^{n_h \times 1}$ are the bias vectors; $\odot$ corresponds to the element-wise multiplication.

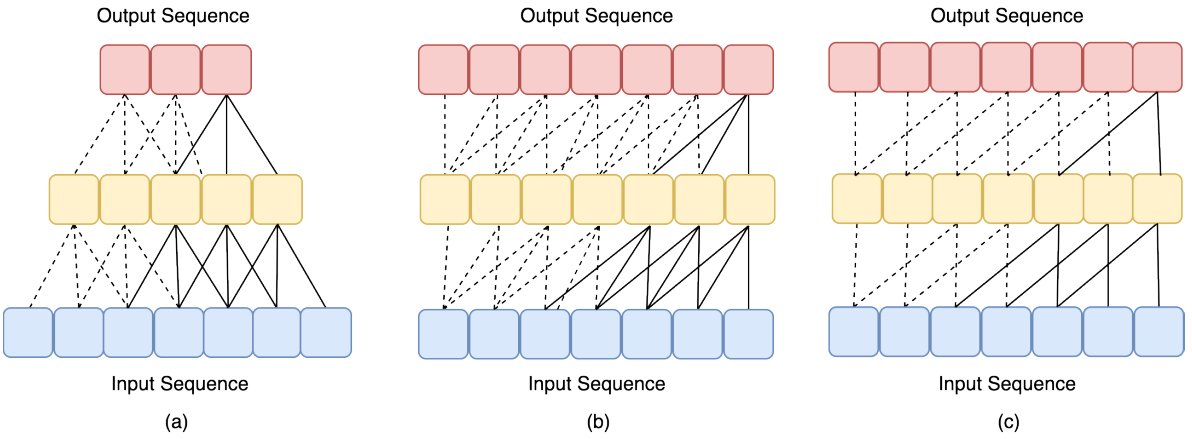

FIGURE 3. (a) Typical convolutional network with kernel size equal to 3; (b) causal convolutional network with kernel size equal to 3; (c) dilated causal convolutional network with kernel size equal to 3 and dilation rate equal to 2.

D. TEMPORAL CONVOLUTIONAL NETWORK (TCN)

TCN is a particular type of 1D-CNN effective in analyzing sequential data [35], which incorporates three components: (i) *causal convolution*, discussed in Subsection III-D1; (ii) *dilated convolution*, discussed in Subsection III-D2; (iii) *residual connections*, discussed in Subsection III-D3. In detail, a TCN allows (using causal convolutions) to prevent data from leaking from the future to the past, allowing the architecture to generate an output sequence with length similar to any input sequence using zero padding [36]. In Figure 3, a comparison between (a) typical, (b) causal, and (c) dilated convolutional network structures is shown. In the following, these architectures will be shortly discussed.

1) CAUSAL CONVOLUTION

Causal convolutions limit predictions to past observed data, with the convolution operation at time epoch t only applying to data preceding time epoch t (namely: $x_0, x_1, \ldots, x_{t-1}$). This ensures that the convolution outcome at time t is unaffected by future information.

2) DILATED CONVOLUTION

The dilated convolution allows TCN to increase its receptive field without drastically increasing the number of parameters, thus allowing the model to capture long-range dependencies by increasing the dilation rate at each layer, but without losing resolution or computational efficiency [37].

3) RESIDUAL CONNECTION

It is well-known that the size of the convolution kernel, dilation factor, and network depth determine the receptive field of a TCN. Moreover, the accuracy of the training set will be saturated or worsened as the network deepens and the model's parameters increase, resulting in vanishing or exploding gradient. Therefore, in order to address these issues, the TCN incorporates residual connections in its output layer, which help the propagation of data across the layers [38].

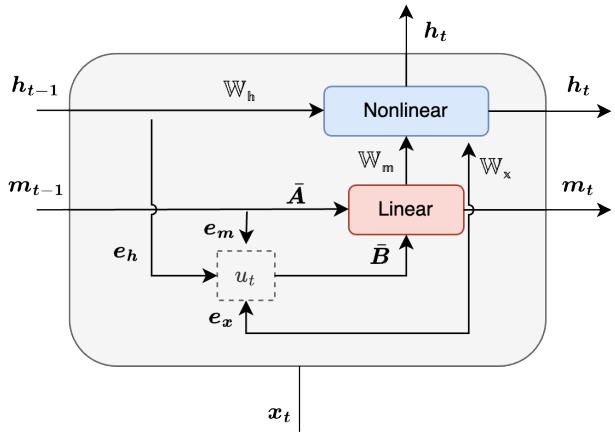

FIGURE 4. Simplified mathematical model of an LMU.

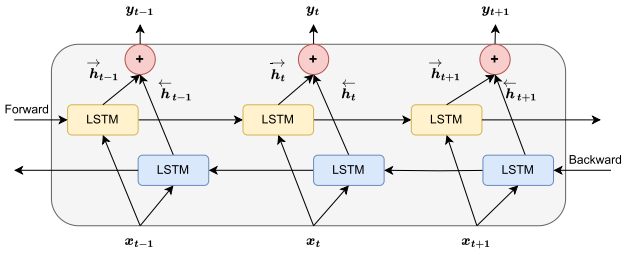

FIGURE 5. Simplified mathematical model of a BiRNN.

E. LEGENDRE MEMORY UNIT (LMU)

LMU is a recently introduced RNN using less computational resources to preserve long-range time dependencies. In detail, LMUs break down the time history into d Ordinary Differential Equation (ODEs) exploiting Legendre

polynomials and Fourier attributes to orthogonalize the continuous-time history of its encoded input signal (denoted as $u_t \in \mathbb{R}$) [39]. For the sake of completeness, a LMU (whose block architecture is shown in Figure 4) can be defined as follows:

$$\boldsymbol{u_t} = \boldsymbol{e_x^T x_t} + \boldsymbol{e_h^T h_{t-1}} + \boldsymbol{e_m^T m_{t-1}} \tag{15}$$

$$\boldsymbol{m_t} = \bar{\boldsymbol{A}} \boldsymbol{m_{t-1}} + \bar{\boldsymbol{B}} u_t \tag{16}$$

$$\boldsymbol{h_t} = f(\mathbb{W}_x \boldsymbol{x_t} + \mathbb{W}_m \boldsymbol{m_t} + \mathbb{W}_h \boldsymbol{h_{t-1}}) \tag{17}$$

where: $f(\cdot)$ is a non-linear function (such as tanh); $\mathbb{W}_x \in \mathbb{R}^{n_h \times n_x}$, $\mathbb{W}_h \in \mathbb{R}^{n_h \times n_h}$, $\mathbb{W}_m \in \mathbb{R}^{n_h \times d}$ represent the weight matrices; $\boldsymbol{e_x} \in \mathbb{R}^{n_x}$, $\boldsymbol{e_h} \in \mathbb{R}^{n_h}$, $\boldsymbol{e_m} \in \mathbb{R}^d$ are the encoding vectors; $\boldsymbol{x_t}$ represents the input at the current time step t; $\boldsymbol{m_t} \in \mathbb{R}^d$ denotes the unit's linear memory; $\boldsymbol{h_t}$ represents the non-linear hidden state; $\bar{\boldsymbol{A}} \in \mathbb{R}^{d \times d}$ and $\bar{\boldsymbol{B}} \in \mathbb{R}^{d \times 1}$ correspond to the discretized matrices provided by ODEs; n_h and n_x are the dimension of the hidden state and input vectors, respectively.

F. BIDIRECTIONAL RNN (BIRNN)

In a BiRNN, data sequences are processed in two directions, namely *forward* and *backward*. This motivates the reason why designing BiRNNs generally comprises two hidden layers at each time step (as shown in Figure 5): the *first* layer is used to process the sequence in the forward direction, while the *second* layer is expedient to process the sequence in the backward direction.

G. OVERALL DISCUSSION

Overall, the ML/DL models previously discussed and considered in the further experimental performance evaluation are particularly well-suited for correlated sequential data with multiple inputs and outputs. Moreover, TCN and LMU have been included as novel models capable of efficiently capturing long- and short-term temporal dependencies. In fact, traditional ML models (such as AdaBoost, DT and RF) are susceptible to overfitting, particularly when the models are complex. As the model complexity increases, both the training and prediction times tend to increase significantly [40], [41]. Furthermore, they often face challenges in complex scenarios involving long-term information [42]. Then, as will be better highlighted in Subsection V-D, including a quantization layer in the processing pipeline for DL models ensures an optimized deployment for constrained devices.

IV. PROPOSED MODEL

On the basis of the ML and DL models detailed in Section III, in the following the different steps defined in the proposed comparative performance analysis are discussed, namely: data preparation (detailed in Subsection IV-A); Bayesian Optimization (BO, detailed in Subsection IV-B); and a sliding window technique (detailed in Subsection IV-C).

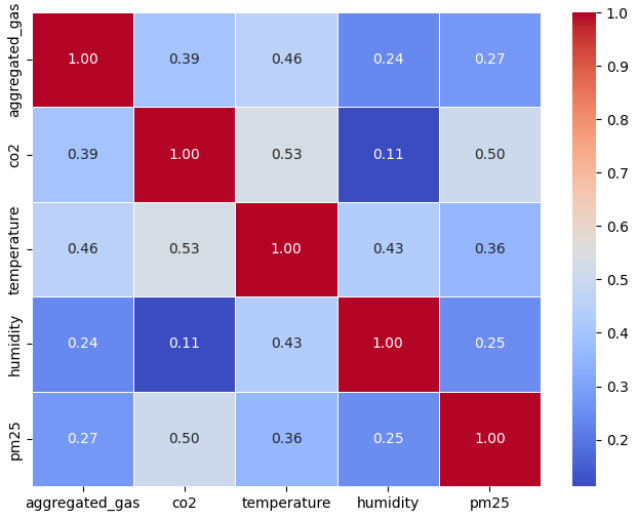

FIGURE 6. PCC of the features of the adopted dataset.

A. DATA PREPARATION

The data preparation step includes data correlation analysis (Subsection IV-A1) and data normalization (Subsection IV-A2), as follows.

1) DATA CORRELATION ANALYSIS

In order to perform a data correlation analysis, the Pearson Correlation Coefficient (PCC) [43] has been adopted. In detail, PCC (denoted as ρ) is a statistical metric accurately measuring importance and trend of the dependence between two variables, and defined as

$$\rho = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \tag{18}$$

where: n corresponds to the number of observations; X_i and Y_i are individual data points; $\bar{X}$ and $\bar{Y}$ represent the average values of the random variables X and Y, respectively. To this end, PCC can range from -1 to 1: two variables with the same direction of variation lead to the highest positive correlation ($\rho = 1$); two variables with opposite directions of variation lead to the highest (in norm) negative correlation ($\rho = -1$); finally, the absence of a linear dependency between the considered variables leads to $\rho = 0$. For the sake of completeness, the PCC obtained on the features still present in the adopted dataset—further described in Subsection V-A—is shown in Figure 6.

Looking at the PCC values shown in Figure 6, it can be observed that each feature exhibits a weak or moderate positive correlation with the others. This indicates that all features contribute complementary information to predict $PM_{2.5}$ levels. Moreover, with this limited information set, a feature augmentation may result in noise, potentially leading to overfitting and increasing model complexity. Consequently, only the declared features are retained for models training.

2) DATA NORMALIZATION

Data normalization eases the training process of NNs by ensuring that the different features representing the candidate dataset are mapped to a comparable scale, thus accelerating the convergence of different algorithms while enhancing the stability and efficiency of the proposed model. For this reason, the MinMaxScaler function [44], adopted in the proposed performance analysis, allows to map input data within a specific range (typically [0, 1] or [−1, 1]) and can be defined as

$$x_{scaled} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{19}$$

where: x and x_{scaled} represent original and scaled values of a specific feature, respectively; $x_{\min}$ and $x_{\max}$ correspond to the minimum and maximum values of that feature, respectively. In fact, the MinMaxScaler function is applied in order to prevent data distortion from changes in standard deviation, as it reshapes the data to fit within a specified range and preserves data sparsity.

B. BAYESIAN OPTIMIZATION (BO)

BO represents a popular method for hyperparameter tweaking in DL models, as it allows performance optimization, reducing overfitting and underfitting, and improving generalization, resource efficiency, and durability [45]. These benefits become more and more evident when compared to traditional techniques (e.g., grid search and random search), which show significant limitations. In particular, grid search becomes computationally inefficient because of an exponential growth of the number of evaluations, as the number of hyperparameters increases, while random search may waste resources by exploring poorly performing regions and failing to focus on optimal configurations [46]. In contrast, BO performs a global optimization of black-box functions by employing a probabilistic model, typically a Gaussian Process (GP), to more effectively explore the search space [47]. To this end, the HPO task can be formally defined as:

$$\boldsymbol{x}^{\star} = \arg\min_{\boldsymbol{x} \in \boldsymbol{X}} f(\boldsymbol{x}) \tag{20}$$

where $f(\boldsymbol{x})$ represents the objective function to be minimized, and $\boldsymbol{x}$ denotes the vector of hyperparameter configurations within the search space $\boldsymbol{X}$. Consequently, the set of hyperparameters $\boldsymbol{x}^{\star}$ will yield the minimal objective function value observed thus far.

For completeness, the following hyperparameters have been optimized: learning rate, units per layer, batch size, and activation function. More in detail, they have been chosen due to their significant impact on models' performance, efficiency, and generalization capabilities. Thus, in order to guide the search, we employed the Expected Improvement (EI) acquisition function—calculating the potential for improvement at each new point in the search space and focusing the optimization on promising areas—defined as follows:

$$\mathrm{EI}(\boldsymbol{x}) = \mathbb{E}\big[\max\Big(0, f(\boldsymbol{x}^{\text{best}}) - f(\boldsymbol{x})\Big)\big] \tag{21}$$

where: $\boldsymbol{x}^{\text{best}}$ corresponds to the set of hyperparameters providing the best objective function up to the current optimization step. To this end, the hyperparameter search space was carefully defined to ensure a comprehensive exploration. With regard to the learning rate, we set a continuous range from 10^{-5} to 10^{-3}, while the number of units per layer was optimized with integers ranging from 32 to 128, the batch size was optimized with integers ranging from 16 to 32, and the activation function was chosen between ReLU and tanh.

In the following a performance comparison of the BO technique using different surrogate models, namely GPs and RFs [48], [49], is detailed. Then, since in BO a surrogate model offers an approximation of the actual objective function under optimization, correlating input data to output data, when the actual relationship is either unknown or computationally costly, in order to compare original and surrogate models, the accuracy is evaluated through the RMSE, chosen as optimization metric and described in Subsection V-B. In particular:

- RMSE_{GP} represents the RMSE obtained using a GP surrogate model;
- RMSE_{RF} represents the RMSE obtained using a RF surrogate model.

Hence, a ratio (denoted as r) between GPs and RFs, with BO serving as the baseline model, is calculated as follows:

$$r = 1 - \left(\frac{\mathrm{RMSE}_{GP}}{\mathrm{RMSE}_{RF}}\right) \tag{22}$$

where: $r > 0$ indicates that GPs outperform RFs ($\mathrm{RMSE}_{GP} < \mathrm{RMSE}_{RF}$); while $r < 0$ shows an improved performance of RFs compared to GPs ($\mathrm{RMSE}_{GP} > \mathrm{RMSE}_{RF}$).

In Table 2, the ratio r obtained by all the considered ML/DL models is shown, considering various values of $\mathcal{W}$, which corresponds to the time lag. From the obtained results, it can be observed how GPs yield an average improvement (in terms of model accuracy) equal to 0.37%, at the same

TABLE 2. Accuracy of the considered ML and DL models in terms of the ratio r between GPs and RFs considering different time lags $\mathcal{W}$.

Model	RMSE Ratio r [%]			
	$\mathcal{W}=5$	$\mathcal{W}=10$	$\mathcal{W}=15$	$\mathcal{W}=20$
BiGRU	0,43%	0,07%	1,05%	-1,18%
BiLSTM	-1,93%	-0,47%	3,58%	-0,32%
CNN	1,85%	2,37%	1,20%	-0,09%
CNN-BiGRU	-0,91%	2,06%	1,13%	-7,35%
CNN-BiLSTM	-1,89%	-1,96%	-0,57%	-17,28%
CNN-GRU	-0,07%	0,22%	-0,17%	-1,09%
CNN-LSTM	3,29%	5,77%	2,31%	8,49%
GRU	-1,29%	-1,44%	0,49%	1,20%
LSTM	-1,62%	6,95%	3,23%	5,84%
RNN	3,45%	0,83%	0,58%	-0,21%

TABLE 3. Bayesian hyperparameters for the different time lags $\mathcal{W}$.

Model	$\mathcal{W} = 5$	$\mathcal{W} = 10$	$\mathcal{W} = 15$	$\mathcal{W} = 20$
BiGRU	neurons = 113 activation = relu	neurons = 128 activation = tanh	neurons = 32 activation = relu	neurons = 128 activation = relu
BiLSTM	neurons = 92 activation = tanh	neurons = 128 activation = relu	neurons = 32 activation = relu	neurons = 128 activation = relu
CNN	filters = 118 kernel_size = 3 activation = relu	filters = 128 kernel_size = 3 activation = relu	filters = 128 kernel_size = 3 activation = relu	filters = 128 kernel_size = 3 activation = tanh
CNN-BiGRU	neurons/filters = 68 kernel_size = 3 activation = relu	neurons/filters = 46 kernel_size = 3 activation = relu	neurons/filters = 128 kernel_size = 3 activation = relu	neurons/filters = 32 kernel_size = 3 activation = relu
CNN-BiLSTM	neurons/filters = 73 kernel_size = 3 activation = relu	neurons/filters = 128 kernel_size = 3 activation = tanh	neurons/filters = 128 kernel_size = 3 activation = tanh	neurons/filters = 128 kernel_size = 3 activation = relu
CNN-GRU	neurons/filters = 114 kernel_size = 3 activation = relu	neurons/filters = 83 kernel_size = 3 activation = relu	neurons/filters = 100 kernel_size = 3 activation = relu	neurons/filters = 128 kernel_size = 3 activation = relu
CNN-LSTM	neurons/filters = 98 kernel_size = 3 activation = relu	neurons/filters = 66 kernel_size = 3 activation = tanh	neurons/filters = 110 kernel_size = 3 activation = tanh	neurons/filters = 116 kernel_size = 3 activation = tanh
GRU	neurons = 32 activation = relu	neurons = 128 activation = relu	neurons = 84 activation = relu	neurons = 89 activation = relu
LMU	neurons = 96 activation = relu	neurons = 64 activation = relu	neurons = 128 activation = tanh	neurons = 128 activation = tanh
LSTM	neurons = 94 activation = relu	neurons = 128 activation = tanh	neurons = 128 activation = tanh	neurons = 56 activation = relu
RNN	neurons = 110 activation = relu	neurons = 128 activation = relu	neurons = 95 activation = tanh	neurons = 126 activation = relu
TCN	neurons/filters = 70 kernel_size = 3 activation = relu	neurons/filters = 119 kernel_size = 3 activation = tanh	neurons/filters = 128 kernel_size = 3 activation = relu	neurons/filters = 115 kernel_size = 3 activation = relu

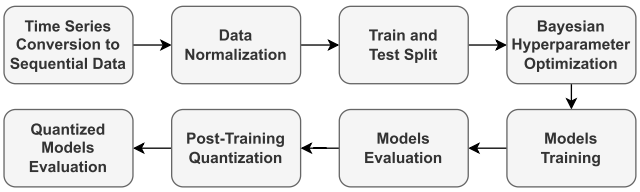

FIGURE 7. Flowchart of the proposed prediction process.

TABLE 4. Combinations of sampling time $t_{sampling}$, prediction horizon s, and number of predictions n_{pred}, applied on the chosen air quality dataset.

t_{sampling}	Prediction horizon s [hr]	n_{pred} [num]
1 hr	8	8
30 min	3	6
2 min	1	30

time not increasing the model's complexity. GP surrogates are particularly effective where the hyperparameter space is low-dimensional and composed mainly of continuous parameters [50]. In contrast, RF surrogates naturally manage high-dimensional spaces that include a mix of continuous, discrete, categorical, and conditional variables. The inherent structure of ensemble trees allows RF models to scale better as the hyperparameter space increases while also providing a lower computational cost [51].

Ultimately, the decision to use GP or RF surrogate models in BO depends primarily on the characteristics of the hyperparameter space. In our study, by focusing on optimizing four hyperparameters, GP models can explore more efficiently and at a lower computational cost, thereby narrowing the gap between GP and RF approaches.

C. SLIDING WINDOW TECHNIQUE

Finally, a sliding window technique has been adopted in the proposed model to convert time series data into sequential data, in order to forecast $PM_{2.5}$ levels for both short-term and long-term time horizons. In particular, this approach involves the evaluation of several time lags $\mathcal{W}$ to identify the setting which enables the model to adequately represent temporal dependencies. The selected time lag allows to identify patterns and relationships between past $PM_{2.5}$ concentrations and future values by examining historical data through the sliding window. By integrating the relevant information from several time intervals, the sliding window approach improves the model's precision in predicting $PM_{2.5}$ levels across different time horizons.

For the sake of completeness and clarification, a graphical representation of the proposed prediction process (returning the experimental results further detailed in Section V) is shown in Figure 7, while the detailed architectures of the proposed models are listed, considering various values of the time lag $\mathcal{W}$, in Table 3.

V. RESULTS

A. DATASET

The experimental dataset (publicly available at [52]) is composed of air quality measurements collected in an

interior travelers transit area at the Brindisi airport, in the south of Italy [53] in the time period October 2022–October 2023. In detail, three inexpensive commercial sensors—namely, Adafruit MiCS5524, Sensirion SCD30, and Sensirion SPS30—have been used to collect air quality data, such as CO_2, temperature, humidity, aggregated gas levels, and $PM_{2.5}$, with a 2 sec sampling period. By employing the sliding window mechanism (detailed in Subsection IV-C) there has been the flexibility to re-sample the dataset at different sampling periods. The considered combinations of values of the sampling period (denoted as $t_{sampling}$, dimension: [s]), the prediction horizon (denoted as s, dimension: [h]), and the number of predictions (denoted as n_{pred}, adimensional) are listed in Table 4. Moreover, we remark that an overlapping time lag mechanism was used during the preparation of both training and test sets: this increases the number of input samples available in the experimental evaluation stage.

In the following, the performance with the dataset re-sampled as a 1-hour dataset ($t_{sampling} = 1$ hour) are investigated. The chosen combinations guarantee a sufficient level of prediction granularity. As expected, re-sampling the dataset with $t_{sampling} = 2$ min allows to predict $PM_{2.5}$ variations in the subsequent hour with a finer detail than in the case with $t_{sampling} = 1$ hour. Considering $t_{sampling} = 30$ min allows to improve the trend estimation accuracy. Nevertheless, even if values lower than 1 hour (e.g., 2 min or 30 min) can be employed for trend analysis, they are not suitable for long-term prediction. On the other end, considering values of $t_{sampling}$ longer than 1 hour will degrade the performance, leading to a smaller dataset.

B. EVALUATION METRICS

The performance of the considered algorithms (detailed in Section III) is evaluated on the basis of RMSE, MAE, MAPE, and R^2, defined as follows:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{23}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{24}$$

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{25}$$

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \tag{26}$$

where: n corresponds to the number of data points in the test set; y_i represents the actual value; $\hat{y}_i$ represents the corresponding predicted value; and $\bar{y} = 1/n \sum_{i=1}^{n} y_i$ is the mean value of data points.

Overall, the chosen evaluation metrics highlight different behaviors of the considered models. More precisely, two models may have nearly identical MAE/RMSE, yet greatly differ in terms of R^2, thus indicating distinct residual-variance patterns. Conversely, a model having a higher R^2 but a larger MAE can return lower performance because of short-term volatility. Together, RMSE and MAE are sensitive to outliers, while MAPE adds a scale-free, relative-error perspective. More in detail, RMSE and MAE are measured on the basis of the predicted feature, i.e., $PM_{2.5}$, while MAPE is expressed as a percentage of the error of $PM_{2.5}$, and R^2 is adimensional.

As shown in Figure 8, RMSE has been designated as reference metric for both training and testing phases. Then, each metric is assessed for various values of the time

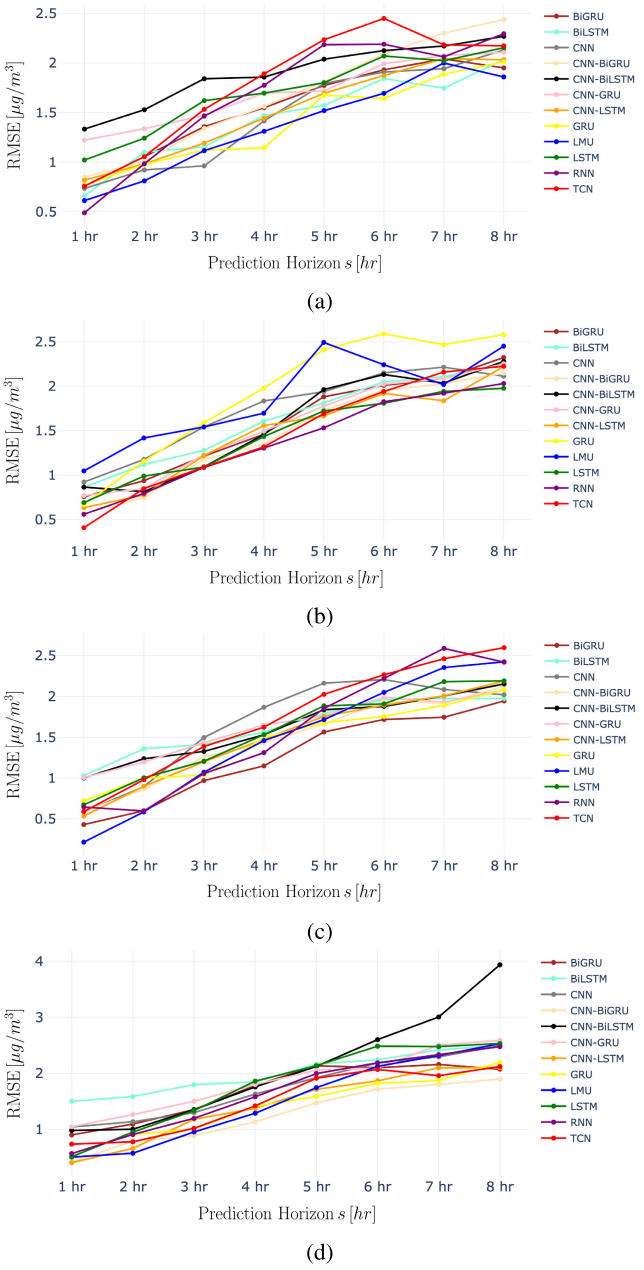

FIGURE 8. Performance of the considered ML and DL models on the basis of RMSE (with lower values of RMSE being the better) for different values of the time lag $\mathcal{W}$ in the test set: (a) $\mathcal{W} = 5$, (b) $\mathcal{W} = 10$, (c) $\mathcal{W} = 15$, (d) $\mathcal{W} = 20$.

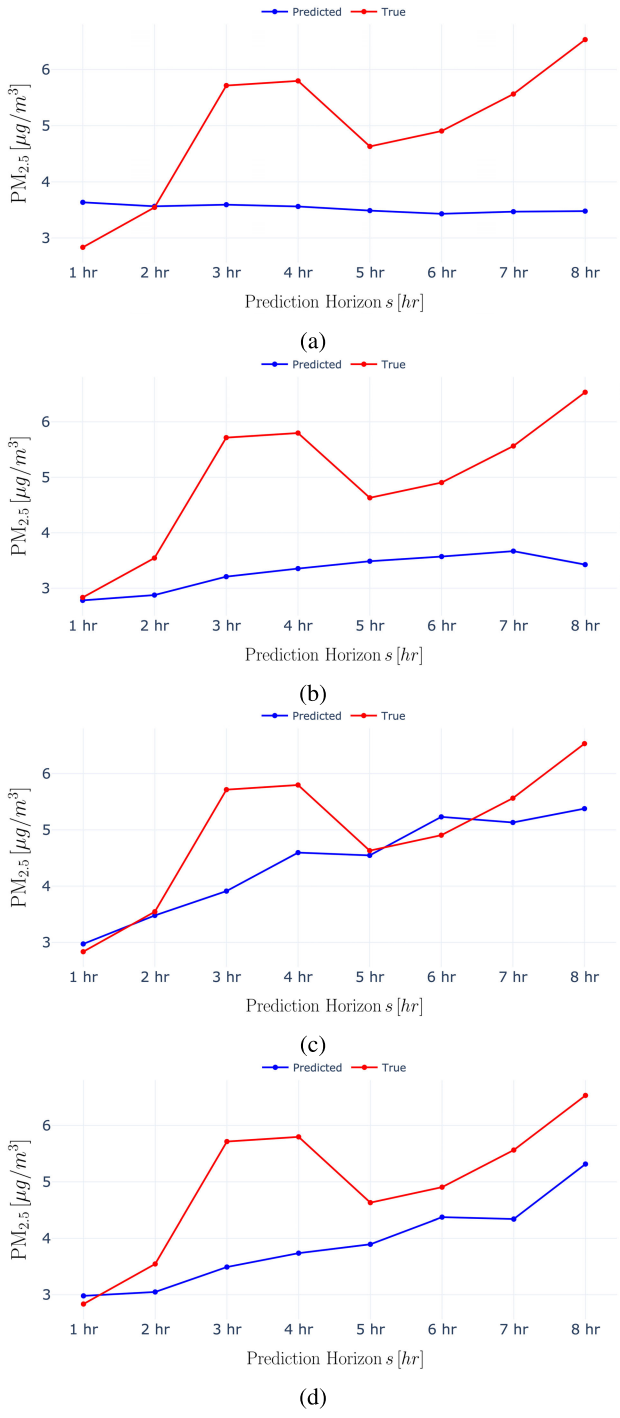

FIGURE 9. Performance of the CNN-BiGRU model for different values of the time lag $\mathcal{W}$: (a) $\mathcal{W} = 5$, (b) $\mathcal{W} = 10$, (c) $\mathcal{W} = 15$, (d) $\mathcal{W} = 20$.

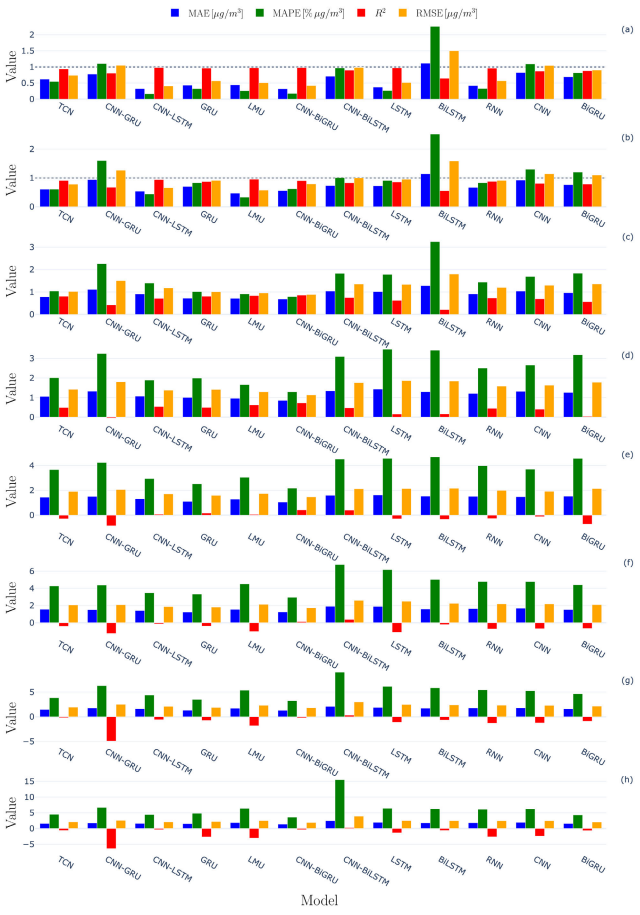

FIGURE 10. Evaluated metrics for each analyzed ML and DL model considering a time lag $\mathcal{W} = 20$ across each prediction hour: (a) 1-hour ($s = 1$), (b) 2-hour ($s = 2$), (c) 3-hour ($s = 3$), (d) 4-hour ($s = 4$), (e) 5-hour ($s = 5$), (f) 6-hour ($s = 6$), (g) 7-hour ($s = 7$), (h) 8-hour ($s = 8$).

lag $\mathcal{W}$, leading to a fair performance comparison of the considered ML and DL models in the test set. In fact, less architectural complex models may experience good overall performance with small values of $\mathcal{W}$, but may face challenges in identifying correlations for long-term predictions, particularly with larger input dimensions, compensating with higher complexity layers. In most cases, increasing the historical knowledge leads to improved accuracy across various models, as shown in Figure 9, with CNN-BiGRU returning the best accuracy among the considered models.

It might also be reasonable for model predictions to become less accurate for increasing values of the prediction horizon s: if s is extended, then the model needs to identify additional correlations within the data and this may become increasingly challenging and computationally heavy. As a reference, in Figure 10 all the metrics for a given dataset and $\mathcal{W} = 20$ are shown, while Figure 11 shows the increasing prediction error as a function of the time horizon. It can be observed that, by increasing s, ML and DL models lose their ability to predict variations, while they still capture the trend in long-term predictions.

Finally, Figure 12 shows the performance of various models across three key metrics—namely, average MAE, MAPE, and RMSE—evaluated over all the considered time lags $\mathcal{W} = \{5, 10, 15, 20\}$ and all the considered prediction horizons $s \in \{1, \ldots, 8\}$. The obtained results demonstrate that CNN-BiGRU achieves, on average, the lowest error rate (in terms of MAE, MAPE, and RMSE), thus reducing

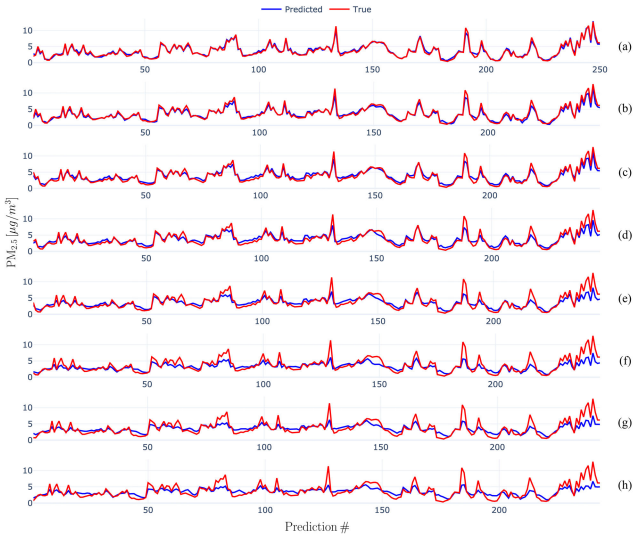

FIGURE 11. Prediction performance of the CNN-BiGRU model with a time lag $\mathcal{W} = 20$ across each prediction hour: (a) 1-hour ($s = 1$), (b) 2-hour ($s = 2$), (c) 3-hour ($s = 3$), (d) 4-hour ($s = 4$), (e) 5-hour ($s = 5$), (f) 6-hour ($s = 6$), (g) 7-hour ($s = 7$), (h) 8-hour ($s = 8$).

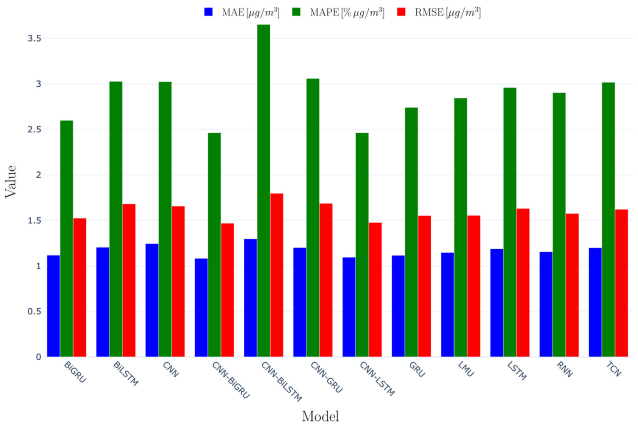

FIGURE 12. Average MAE, MAPE, and RMSE of the evaluated DL models considering all the time lags $\mathcal{W} = \{5, 10, 15, 20\}$ and all the prediction horizons $s \in \{1, \ldots, 8\}$.

the prediction error compared to other DL models. This superior performance highlights its effectiveness in capturing both spatial and temporal dependencies for $PM_{2.5}$ prediction. In fact, the CNN layers excel in the spatial patterns extraction, while BiGRU captures temporal dependencies in both forward and backward directions.

C. MODEL COMPLEXITY

In order to better focus on the utilisation of ML and DL models by IoT nodes (often characterized by limited, if not constrained, resources), in addition to the performance analyzed in Subsection V-B, the computational complexity of the considered models has been further investigated considering the number of Multiply and ACCumulate (MACC) operations (denoted as n_{MACC}, adimensional), the RAM usage (dimension: [KiB]), and the execution time (denoted

TABLE 5. Evaluation cycle per MACC for various values of the time lags $\mathcal{W}$.

Model	$\mathcal{W} = 5$	$\mathcal{W} = 10$	$\mathcal{W} = 15$	$\mathcal{W} = 20$
BiGRU	10.045	10.261	14.571	9.924
BiLSTM	12.065	10.462	16.487	10.381
CNN	10.426	11.779	11.073	11.985
CNN-BiGRU	10.475	11.608	9.197	12.416
CNN-BiLSTM	10.630	9.847	9.513	9.515
CNN-GRU	9.421	10.032	9.962	9.354
CNN-LSTM	9.963	11.619	10.310	10.272
GRU	14.231	9.879	10.820	11.181
LSTM	11.145	11.013	11.062	14.723
RNN	8.836	8.688	10.608	8.763

FIGURE 13. Comparison, in terms of number of cycles per MACC, between the considered ML and DL models for different values of the time lag $\mathcal{W}$.

as t_{exec}, dimension: [ms]), defined as follows:

$$t_{exec} = \frac{n_{MACC} \cdot 11}{f_{board}} \tag{27}$$

where: f_{board} corresponds the frequency of the STM32F469I-DISCO board [54], equal to 180 MHz; n_{MACC} is multiplied by 11 because of the requirement of 11 cycles per MACC (on average, with minimum and maximum values equal to 8.688 and 16.487, respectively, as further detailed in Table 5 and shown in Figure 13) on that board—the value of 11 has been achieved considering the average n_{MACC} for all of the evaluated models, except for LMU and TCN.

In order to compute the aforementioned metrics, we first uploaded each model to ST boards via the STM Edge AI Developer Cloud tool [55], which allows a seamless import of AI models to different tiny-IoT platforms—such as the STM32F469I-DISCO board [54]. Then, through the STM32CubeMX tool [56], a real board may be used by generating the firmware associated with the selected model and flashing it into the selected board. Finally, the performance can be validated through the cloud service by using testing and validation dataset [52]. This has been kept as reference board for the evaluation of the models' complexities, which are shown in Table 6 and, for the sake of visual representation, in Figure 14 (in terms of n_{MACC}), Figure 15 (in terms of RAM), and Figure 16 (in terms of t_{exec}), respectively. As can be seen from Table 6, among all the models deployable on the chosen IoT board, CNN is the

TABLE 6. MACC operations n_{MACC}, RAM usage, and execution time t_{exec} returned by the analyzed ML and DL models, as a function of different time lags $\mathcal{W}$.

Model	n_{MACC}				RAM (KiB)				t_{exec} (ms)			
	$\mathcal{W}=5$	$\mathcal{W}=10$	$\mathcal{W}=15$	$\mathcal{W}=20$	$\mathcal{W}=5$	$\mathcal{W}=10$	$\mathcal{W}=15$	$\mathcal{W}=20$	$\mathcal{W}=5$	$\mathcal{W}=10$	$\mathcal{W}=15$	$\mathcal{W}=20$
BiGRU	442,047	1,079,656	112,040	2,103,912	6.76	4.2	4.43	7.54	24.67	61.55	9.07	116.00
BiLSTM	389,816	1,425,512	150,440	2,800,232	6.59	7.94	4.66	8.14	26.13	82.86	13.78	161.5
CNN	49,546	112,616	161,256	227,816	4.57	6.14	7.64	8.64	2.87	7.37	9.92	15.17
CNN-BiGRU	185,404	217,546	2,640,104	236,296	6.73	6.69	14.31	7.08	10.79	14.03	134.90	16.30
CNN-BiLSTM	278,526	2,186,216	3,502,056	4,829,416	7.35	12.44	14.93	17.43	16.45	119.6	185.1	255.3
CNN-GRU	267,662	357,585	824,804	1,845,480	7.58	7.99	1094	15.64	14.01	19.93	45.65	95.91
CNN-LSTM	257,256	305,486	1,329,014	2,030,916	7.43	7.36	12.26	15.02	14.24	19.72	76.13	115.9
GRU	20,616	547,688	354,164	522,534	3.45	6.45	5.07	5.27	1.63	30.06	21.29	32.46
LMU	97,892	87,652	112,228	112,228	//	//	//	//	6.11	5.35	6.85	6.85
LSTM	207,374	724,972	1,067,752	286,936	5.85	7.05	7.05	4.70	12.84	44.36	65.62	23.47
TCN	568,400	1,642,676	1,900,544	1,534,100	//	//	//	//	34.73	100.3	116	93.75
RNN	89,424	205,928	164,929	366,008	3.81	4.12	3.92	4.39	4.388	9.94	9.72	17.82

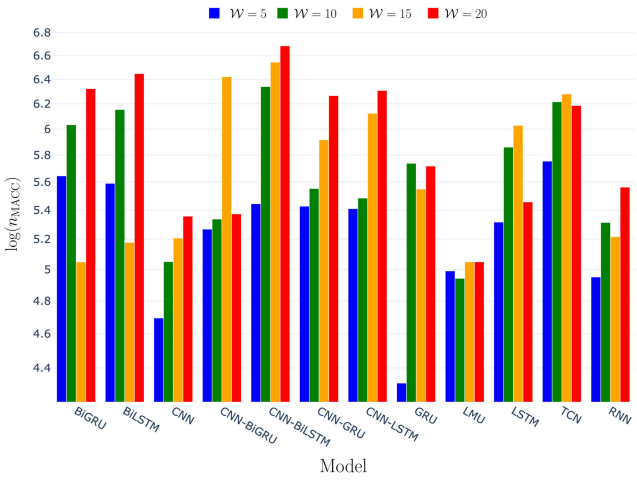

FIGURE 14. Comparison, in terms of n_{MACC} (in logarithmic scale), between the considered ML and DL models for different values of the time lag $\mathcal{W}$.

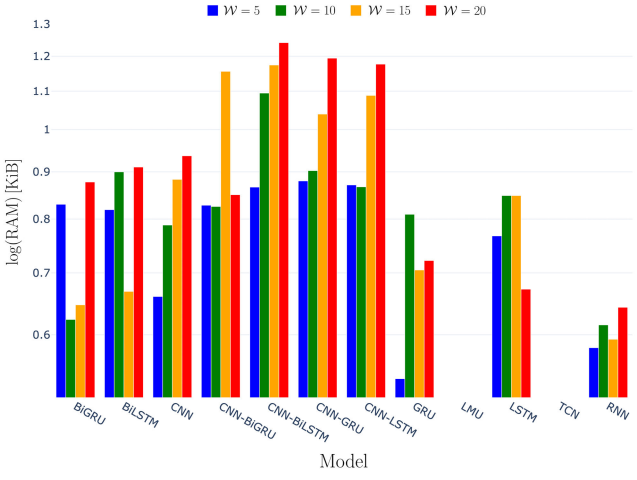

FIGURE 15. Comparison, in terms of RAM (in logarithmic scale), between the considered ML and DL models for different values of the time lag $\mathcal{W}$.

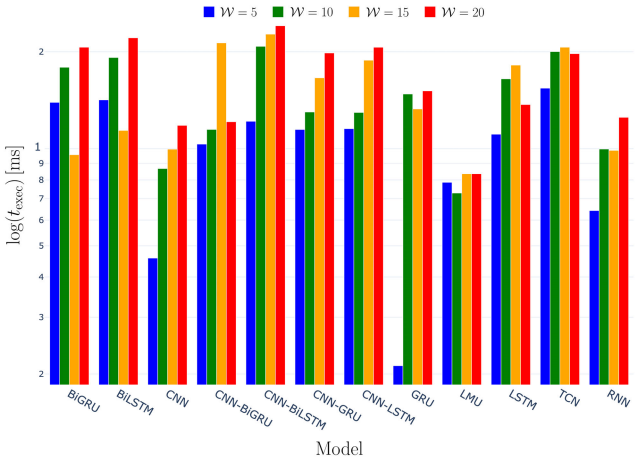

FIGURE 16. Comparison, in terms of t_{exec} (in logarithmic scale), between the considered ML and DL models for different values of the time lag $\mathcal{W}$.

most computationally-efficient (in terms of n_{MACC}) for all time lags, with GRU and RNN ranking as second and third best performing models. At the opposite, CNN-BiLSTM has the higher computational complexity, closely followed by BiLSTM. Then, models with balanced computational efficiency across various time lags include CNN-BiGRU, BiGRU, GRU, and LSTM. Finally, even if LMU and TCN represent novel architectures for time-series prediction (with LMU achieving a high accuracy with a low complexity), for completeness it should be highlighted that they are currently not natively supported by ST Edge AI Developer Cloud [55], a remote tool exploited to run the considered ML and DL models on a remote tiny IoT board directly in the cloud (before being deployed on real devices). Nevertheless, knowing the functional parameters of the chosen IoT board, it has been possible to estimate n_{MACC} and RAM usage also for TCN and LMU.

D. POST-TRAINING QUANTIZATION (PTQ)

Finally, in order to further optimize DL models predicting $PM_{2.5}$ values on tiny IoT devices, an additional investigation has been performed applying a Post-Training Quantization (PTQ) technique exploiting the TensorFlow Lite (TFLite)

TABLE 7. Comparison (in terms of mean size) between original DL models and their corresponding TFLite-compressed versions.

Model	Mean original model size [KiB]	Mean TFLite model size [KiB]	Model size reduction [%]
BiGRU	1347.28	455.86	66.16
BiLSTM	1484.96	499.86	66.34
CNN	1308.85	431.29	67.04
CNN-BiGRU	1157.80	390.12	66.30
CNN-BiLSTM	3143.70	1052.67	66.51
CNN-GRU	1178.98	392.40	66.71
CNN-LSTM	1246.75	414.85	66.72
GRU	551.09	183.82	66.64
LSTM	873.60	291.91	66.58
RNN	537.26	177.76	66.91

FIGURE 17. Comparison (in terms of mean size) between original DL models and their corresponding TFLite-compressed versions.

TABLE 8. Comparison (in terms of inference time) between original DL models and their corresponding TFLite-compressed versions.

Model	Mean original inference time [ms]	Mean TFLite inference time [ms]	Inference time reduction [%]
BiGRU	24.12	0.335	98.61%
BiLSTM	23.10	0.340	98.53%
CNN	24.64	0.015	99.94%
CNN-BiGRU	22.67	0.275	98.79%
CNN-BiLSTM	24.74	0.473	98.09%
CNN-GRU	24.29	0.182	99.25%
CNN-LSTM	22.78	0.192	99.16%
GRU	23.27	0.158	99.32%
LSTM	22.78	0.170	99.25%
RNN	22.66	0.090	99.60%

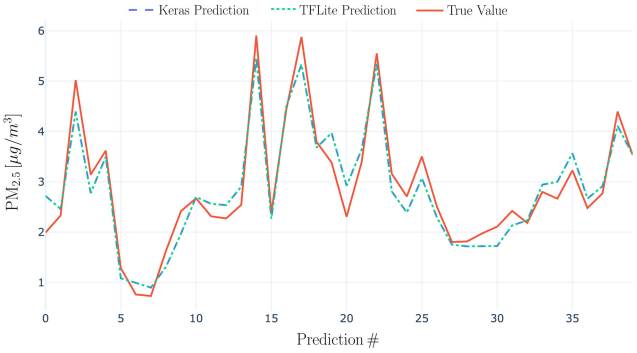

FIGURE 18. Comparison (in terms of prediction accuracy) between true $PM_{2.5}$ values and predicted values obtained through original and quantized CNN-BiGRU models.

library [57]. In detail, by implementing PTQ, a significant decrease (in terms of models' sizes) has been accomplished at the cost of a minimal prediction accuracy decrease. Nevertheless, despite this slight performance reduction, the benefits of reduced model size and improved computational efficiency outweigh the accuracy *trade-off*. In fact, as highlighted in Table 7 and Figure 17 (where the sizes of original and TFLite-compressed DL models are directly compared), PTQ provides significant benefits in practical situations, especially with resource-constrained systems when computational effectiveness is crucial.

Moreover, Table 8 provides a quantitative comparison in terms of average inference time (dimension: [ms]), between the Keras versions of the DL models and their TFLite-compressed counterparts. These results return over 98% reduction in inference time across all evaluated models, confirming the effectiveness of TFLite compression for deployment in resource-constrained environments not only to reduce model size but also to reduce inference time.

As a matter of fact, PTQ with TFLite shows a great potential if applied to efficient and real-time $PM_{2.5}$ monitoring systems, by reducing the computational complexity during model deployment. Finally, it can be noted that reducing the size of the models by means of quantization does not affect significantly the accuracy, as shown in Figure 18, where only a (negligible) 1% accuracy reduction is experienced by CNN-BiGRU despite a 66% average model's size's reduction.

E. STUDY LIMITATIONS

For the sake of clarity and completeness, it is useful to briefly discuss some limitations of the proposed performance analysis.

- A comprehensive evaluation of an optimal sliding window length (as a hyperparameter) would be useful to thoroughly assess the performance of the considered DL models.
- The analysis returned that no single model consistently outperforms the others across all the scenarios considered in our research study.
- A dataset with long-term observations and diverse features would be necessary for training ML and DL models to accurately predict $PM_{2.5}$ levels.

VI. CONCLUSION AND FUTURE WORK

This paper presents a comparative study of air quality estimation, specifically targeting resource-constrained IoT devices. To this end, several ML and DL models (namely: RNN, CNN, LSTM, GRU, LMU, TCN, BiGRU, BiLSTM, CNN-GRU, CNN-LSTM, CNN-BiGRU, CNN-BiLSTM) have been considered for both short-term and long-term prediction of the $PM_{2.5}$ value, exploiting Bayesian optimization to select the best hyperparameters for model training. Among the analyzed algorithms, CNN-BiGRU shows the highest predictive accuracy and provides an

effective trade-off between computational complexity (in terms of number of MACCs), memory usage (in terms of RAM), and execution time. In general, while some DL models may excel in one of the considered performance metrics, they often perform poorly in others. Then, in order to investigate the impact of 8-bit quantization on prediction performance, PTQ has been applied to the considered models: 8-bit quantized CNN-BiGRU achieves approximately a 66% model size reduction with a prediction accuracy reduction equal to only 1% (on average) and a 98.79% inference time reduction. Future research activities might explore the adoption of communication paradigms ensuring data integrity and privacy. Finally, CNN-BiGRU, which is proposed as the best model, can be considered for a real-time $PM_{2.5}$ prediction deployment in particular (e.g., indoor) areas of interest, such as an airport transit area. Additionally, expanding the dataset to include multiple seasons and locations, as well as incorporating heterogeneous datasets, will represent important and interesting future research directions and extensions to further enhance generalizability and external validity of the proposed approach.

REFERENCES

[1] F. H. Dominski, J. H. L. Branco, G. Buonanno, L. Stabile, M. G. da Silva, and A. Andrade, "Effects of air pollution on health: A mapping review of systematic reviews and meta-analyses," *Environ. Res.*, vol. 201, Oct. 2021, Art. no. 111487, doi: 10.1016/j.envres.2021.111487.

[2] R. B. Hamanaka and G. M. Mutlu, "Particulate matter air pollution: Effects on the cardiovascular system," *Frontiers Endocrinol.*, vol. 9, pp. 1–15, Nov. 2018, doi: 10.3389/fendo.2018.00680.

[3] S. Bae and Y.-C. Hong, "Health effects of particulate matter," *J. Korean Med. Assoc.*, vol. 61, no. 12, pp. 749–755, 2018, doi: 10.5124/jkma.2018.61.12.749.

[4] R. D. Arias-Pérez, N. A. Taborda, D. M. Gómez, J. F. Narvaez, J. Porras, and J. C. Hernandez, "Inflammatory effects of particulate matter air pollution," *Environ. Sci. Pollut. Res.*, vol. 27, no. 34, pp. 42390–42404, Dec. 2020, doi: 10.1007/s11356-020-10574-w.

[5] P. Fu and K. K. L. Yung, "Air pollution and Alzheimer's disease: A systematic review and meta-analysis," *J. Alzheimer's Disease*, vol. 77, no. 2, pp. 701–714, 2020.

[6] H. Murata, L. M. Barnhill, and J. M. Bronstein, "Air pollution and the risk of Parkinson's disease: A review," *Movement Disorders*, vol. 37, no. 5, pp. 894–904, May 2022, doi: 10.1002/mds.28922.

[7] C. J. Stevens, J. N. B. Bell, P. Brimblecombe, C. M. Clark, N. B. Dise, D. Fowler, G. M. Lovett, and P. A. Wolseley, "The impact of air pollution on terrestrial managed and natural vegetation," *Phil. Trans. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 378, no. 2183, Sep. 2020, Art. no. 20190317, doi: 10.1098/rsta.2019.0317.

[8] D. Sofia, F. Gioiella, N. Lotrecchiano, and A. Giuliano, "Mitigation strategies for reducing air pollution," *Environ. Sci. Pollut. Res.*, vol. 27, no. 16, pp. 19226–19235, Jun. 2020, doi: 10.1007/s11356-020-08647-x.

[9] A. Kwilinski, O. Lyulyov, and T. Pimonenko, "Reducing transport sector CO_2 emissions patterns: Environmental technologies and renewable energy," *J. Open Innovation: Technol., Market, Complex.*, vol. 10, no. 1, Mar. 2024, Art. no. 100217, doi: 10.1016/j.joitmc.2024.100217.

[10] S. S. Somvanshi, A. Vashisht, U. Chandra, and G. Kaushik, *Delhi Air Pollution Modeling Using Remote Sensing Technique*. Cham, Switzerland: Springer, 2019, pp. 1–27, doi: 10.1007/978-3-319-58538-3_174-1.

[11] Y. Chen, S. Zhang, W. Zhang, J. Peng, and Y. Cai, "Multifactor spatio-temporal correlation model based on a combination of convolutional neural network and long short-term memory neural network for wind speed forecasting," *Energy Convers. Manage.*, vol. 185, pp. 783–799, Apr. 2019, doi: 10.1016/j.enconman.2019.02.018.

[12] J. Ma, J. C. P. Cheng, C. Lin, Y. Tan, and J. Zhang, "Improving air quality prediction accuracy at larger temporal resolutions using deep learning and transfer learning techniques," *Atmos. Environ.*, vol. 214, Oct. 2019, Art. no. 116885, doi: 10.1016/j.atmosenv.2019.116885.

[13] S. Al-Eidi, F. Amsaad, O. Darwish, Y. Tashtoush, A. Alqahtani, and N. Niveshitha, "Comparative analysis study for air quality prediction in smart cities using regression techniques," *IEEE Access*, vol. 11, pp. 115140–115149, 2023, doi: 10.1109/ACCESS.2023.3323447.

[14] M. Kaur, D. Singh, M. Y. Jabarulla, V. Kumar, J. Kang, and H.-N. Lee, "Computational deep air quality prediction techniques: A systematic review," *Artif. Intell. Rev.*, vol. 56, no. S2, pp. 2053–2098, Nov. 2023, doi: 10.1007/s10462-023-10570-9.

[15] Y. Hua, Z. Zhao, R. Li, X. Chen, Z. Liu, and H. Zhang, "Deep learning with long short-term memory for time series prediction," *IEEE Commun. Mag.*, vol. 57, no. 6, pp. 114–119, Jun. 2019, doi: 10.1109/MCOM.2019.1800155.

[16] N. Zhai, P. Yao, and X. Zhou, "Multivariate time series forecast in industrial process based on XGBoost and GRU," in *Proc. IEEE 9th Joint Int. Inf. Technol. Artif. Intell. Conf. (ITAIC)*, vol. 9, Chongqing, China, Dec. 2020, pp. 1397–1400, doi: 10.1109/ITAIC49862.2020.9338878.

[17] F. Mohammadi, H. Teiri, Y. Hajizadeh, A. Abdolahnejad, and A. Ebrahimi, "Prediction of atmospheric $PM_{2.5}$ level by machine learning techniques in Isfahan, Iran," *Sci. Rep.*, vol. 14, no. 1, p. 2109, Jan. 2024, doi: 10.1038/s41598-024-52617-z.

[18] K. Thaweephol and N. Wiwatwattana, "Long short-term memory deep neural network model for $PM_{2.5}$ forecasting in the Bangkok urban area," in *Proc. 17th Int. Conf. ICT Knowl. Eng. (ICT KE)*, Bangkok, Thailand, Nov. 2019, pp. 1–6, doi: 10.1109/ICTKE47035.2019.8966854.

[19] F. R. Alharbi and D. Csala, "A seasonal autoregressive integrated moving average with exogenous factors (SARIMAX) forecasting model-based time series approach," *Inventions*, vol. 7, no. 4, p. 94, Oct. 2022, doi: 10.3390/inventions7040094.

[20] Y.-S. Chang, H.-T. Chiao, S. Abimannan, Y.-P. Huang, Y.-T. Tsai, and K.-M. Lin, "An LSTM-based aggregated model for air pollution forecasting," *Atmos. Pollut. Res.*, vol. 11, no. 8, pp. 1451–1463, Aug. 2020, doi: 10.1016/j.apr.2020.05.015.

[21] A. U. Ruby, J. G. C. Chandran, P. Theerthagiri, R. Patil, B. N. Chaithanya, and T. J. S. Jain, "Forecasting $PM_{2.5}$ concentration using gradient-boosted regression tree with CNN learning model," *Opt. Memory Neural Netw.*, vol. 33, no. 1, pp. 86–96, Mar. 2024, doi: 10.3103/s1060992x24010107.

[22] A. Luo, X. Li, Y. Li, and J. Li, "Application of accurate online support vector regression in atmospheric SO_2 concentration prediction," in *Proc. Chin. Control Decis. Conf. (CCDC)*, Shenyang, China, Jun. 2018, pp. 6274–6279, doi: 10.1109/CCDC.2018.8408231.

[23] S. Choi and B. Kim, "Applying PCA to deep learning forecasting models for predicting $PM_{2.5}$," *Sustainability*, vol. 13, no. 7, p. 3726, Mar. 2021, doi: 10.3390/su13073726.

[24] M. Greenacre, P. J. F. Groenen, T. Hastie, A. I. D'Enza, A. Markos, and E. Tuzhilina, "Principal component analysis," *Nature Rev. Methods Primers*, vol. 2, no. 1, p. 100, Dec. 2022, doi: 10.1038/s43586-022-00184-w.

[25] C. Ding, G. Wang, X. Zhang, Q. Liu, and X. Liu, "A hybrid CNN-LSTM model for predicting $PM_{2.5}$ in Beijing based on spatiotemporal correlation," *Environ. Ecological Statist.*, vol. 28, no. 2, pp. 503–522, Apr. 2021, doi: 10.1007/s10651-021-00501-8.

[26] Q. Tao, F. Liu, Y. Li, and D. Sidorov, "Air pollution forecasting using a deep learning model based on 1D ConvNets and bidirectional GRU," *IEEE Access*, vol. 7, pp. 76690–76698, 2019, doi: 10.1109/ACCESS.2019.2921578.

[27] Y.-B. Kim, S.-B. Park, S. Lee, and Y.-K. Park, "Comparison of $PM_{2.5}$ prediction performance of the three deep learning models: A case study of seoul, daejeon, and busan," *J. Ind. Eng. Chem.*, vol. 120, pp. 159–169, Apr. 2023, doi: 10.1016/j.jiec.2022.12.022.

[28] B. Eren, İ. Aksangür, and C. Erden, "Predicting next hour fine particulate matter ($PM_{2.5}$) in the Istanbul metropolitan city using deep learning algorithms with time windowing strategy," *Urban Climate*, vol. 48, Mar. 2023, Art. no. 101418, doi: 10.1016/j.uclim.2023.101418.

[29] M. Zhu and J. Xie, "Investigation of nearby monitoring station for hourly $PM_{2.5}$ forecasting using parallel multi-input 1D-CNN-biLSTM," *Expert Syst. Appl.*, vol. 211, Jan. 2023, Art. no. 118707, doi: 10.1016/j.eswa.2022.118707.

[30] A. Datta, A. Pal, R. Marandi, N. Chattaraj, S. Nandi, and S. Saha, "Efficient air quality index prediction on resource-constrained devices using TinyML: Design, implementation, and evaluation," in *Proc. 25th Int. Conf. Distrib. Comput. Netw.*, Chennai, India, Jan. 2024, pp. 304–309, doi: 10.1145/3631461.3631956.

[31] A. Mazinani, L. Davoli, D. P. Pau, and G. Ferrari, "Air quality estimation with embedded AI-based prediction algorithms," in *Proc. Int. Conf. Inf. Technol. Res. Innov. (ICITRI)*, Jakarta, Indonesia, Aug. 2023, pp. 87–92, doi: 10.1109/icitri59340.2023.10249864.

[32] I. N. K. Wardana, S. A. Fahmy, and J. W. Gardner, "TinyML models for a low-cost air quality monitoring device," *IEEE Sensors Lett.*, vol. 7, no. 11, pp. 1–4, Nov. 2023, doi: 10.1109/LSENS.2023.3315249.

[33] Scikit Learn. *Imputation of Missing Values*. Accessed: Aug. 26, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/impute.html

[34] L. Wu, C. Kong, X. Hao, and W. Chen, "A short-term load forecasting method based on GRU-CNN hybrid neural network model," *Math. Problems Eng.*, vol. 2020, pp. 1–10, Mar. 2020, doi: 10.1155/2020/1428104.

[35] P. Lara-Benítez, M. Carranza-García, J. M. Luna-Romera, and J. C. Riquelme, "Temporal convolutional networks applied to energy-related time series forecasting," *Appl. Sci.*, vol. 10, no. 7, p. 2322, Mar. 2020, doi: 10.3390/app10072322.

[36] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018, *arXiv:1803.01271*.

[37] X. Luo, W. Gan, L. Wang, Y. Chen, and E. Ma, "A deep learning prediction model for structural deformation based on temporal convolutional networks," *Comput. Intell. Neurosci.*, vol. 2021, no. 1, pp. 1–12, Jan. 2021, doi: 10.1155/2021/8829639.

[38] J. Zhu, L. Su, and Y. Li, "Wind power forecasting based on new hybrid model with TCN residual modification," *Energy AI*, vol. 10, Nov. 2022, Art. no. 100199, doi: 10.1016/j.egyai.2022.100199.

[39] A. R. Voelker, I. Kajić, and C. Eliasmith, "Legendre memory units: Continuous-time representation in recurrent neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 15544–15553. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/952285b9b7e7a1be5aa7849f32ffff05-Paper.pdf

[40] M. Bansal, A. Goyal, and A. Choudhary, "A comparative analysis of K-nearest neighbor, genetic, support vector machine, decision tree, and long short term memory algorithms in machine learning," *Decis. Analytics J.*, vol. 3, Jun. 2022, Art. no. 100071.

[41] I. D. Mienye and Y. Sun, "A survey of ensemble learning: Concepts, algorithms, applications, and prospects," *IEEE Access*, vol. 10, pp. 99129–99149, 2022.

[42] A. Mazinani, D. Pau, L. Davoli, and G. Ferrari, "Deep neural quantization for speech detection of Parkinson disease," in *Proc. IEEE 8th Forum Res. Technol. Soc. Ind. Innov. (RTSI)*, Sep. 2024, pp. 178–183, doi: 10.1109/RTSI61910.2024.10761283.

[43] G. Li, A. Zhang, Q. Zhang, D. Wu, and C. Zhan, "Pearson correlation coefficient-based performance enhancement of broad learning system for stock price prediction," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, no. 5, pp. 2413–2417, May 2022, doi: 10.1109/TCSII.2022.3160266.

[44] Scikit Learn. *MinMaxScaler*. Accessed: Aug. 26, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

[45] J. Wu, X. Y. Chen, H. Zhang, L. D. Xiong, H. Lei, and S. Deng, "Hyperparameter optimization for machine learning models based on Bayesian optimization," *J. Electron. Sci. Technol.*, vol. 17, no. 1, pp. 26–40, 2019, doi: 10.11989/jest.1674-862x.80904120.

[46] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020, doi: 10.1016/j.neucom.2020.07.061.

[47] R. B. Gramacy, *Surrogates: Gaussian Process Modeling, Design, and Optimization for the Applied Sciences*. Boca Raton, FL, USA: CRC Press, Mar. 2020, doi: 10.1201/9780367815493.

[48] D. Pau, A. Pisani, and A. Candelieri, "Towards full forward on-tiny-device learning: A guided search for a randomly initialized neural network," *Algorithms*, vol. 17, no. 1, p. 22, Jan. 2024, doi: 10.3390/a17010022.

[49] J. van Hoof and J. Vanschoren, "Hyperboost: Hyperparameter optimization by gradient boosting surrogate models," 2021, *arXiv:2101.02289*.

[50] B. Bischl, M. Binder, M. Lang, T. Pielok, J. Richter, S. Coors, J. Thomas, T. Ullmann, M. Becker, A.-L. Boulesteix, D. Deng, and M. Lindauer, "Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges," *WIREs Data Mining Knowl. Discovery*, vol. 13, no. 2, Mar. 2023, Art. no. e1484, doi: 10.1002/widm.1484.

[51] M. Lindauer, K. Eggensperger, M. Feurer, A. Biedenkapp, D. Deng, C. Benjamins, T. Ruhopf, R. Sass, and F. Hutter, "SMAC3: A versatile Bayesian optimization package for hyperparameter optimization," *J. Mach. Learn. Res.*, vol. 23, no. 54, pp. 1–9, Oct. 2021.

[52] L. Davoli, L. Belli, and G. Ferrari, "Indoor air quality monitoring @ brindisi airport," Mendeley Data, 2023, doi: 10.17632/BV2HVM4PMZ.

[53] L. Davoli, L. Belli, and G. Ferrari, "Air quality dataset from an indoor airport travelers transit area," *Data Brief*, vol. 52, Feb. 2024, Art. no. 109821, doi: 10.1016/j.dib.2023.109821.

[54] STMicroelectronics. *Discovery Kit with STM32F469NI MCU*. Accessed: Aug. 26, 2025. [Online]. Available: https://www.st.com/en/evaluation-tools/32f469idiscovery.html

[55] STMicroelectronics. *ST Edge AI Developer Cloud*. Accessed: Aug. 26, 2025. [Online]. Available: https://stm32ai.st.com/st-edge-ai-developer-cloud/

[56] STMicroelectronics. (2024). *STM32CubeMX*. Accessed: Aug. 26, 2025. [Online]. Available: https://www.st.com/en/development-tools/stm32cubemx.html

[57] Google AI Edge. *LiteRT—Post-training Quantization*. Accessed: Aug. 26, 2025. [Online]. Available: https://ai.google.dev/edge/litert/models/post_training_quantization

ARMIN MAZINANI received the M.Sc. degree in computer engineering, with a focus on AI from Khayyam University, Mashhad, Iran, in 2018, and the Ph.D. degree in information technologies from the University of Parma, Italy, in 2025. He is a Postdoctoral Research Associate with the Internet of Things (IoT) Laboratory, Department of Engineering and Architecture, University of Parma, Italy. His research interests include cutting-edge advancements in AI, integration of DL techniques with resource-constrained devices, performance optimization in edge computing and IoT applications, and networking.

DANIELE ANTONUCCI received the master's degree in computer engineering from the University of Parma, Italy, in 2022, where he is currently pursuing the Ph.D. degree with the Internet of Things (IoT) Laboratory, Department of Engineering and Architecture. His Ph.D. topic is focused on predictive maintenance, fault, and anomaly detection for manufacturing process. He thrives related topics, such as AI, soft sensing, the IoT, and process optimization.

DANILO PIETRO PAU (Fellow, IEEE) received the degree from the Politecnico di Milano. His H-index is 30 and an i10-index is 86. He has co-authored more than 210 papers; produced 108 invention's requests, 78 EU and 69 U.S. application patents, 113 ISO/IEC/MPEG documents; and has 113 invited talks including, key notes, seminars, and tutorials at universities/conferences. He worked on memory-reduced HDMAC HW design, MPEG2 video memory reduction, video coding, transcoding, embedded (Khronos) 2-D/3-D graphics, and ISO/IEC/MPEG visual search MPEG7 standards on computer vision. Currently, his work focuses on the ST Unified AI Core Technology. He has supervised many students. He is the Technical Director of IEEE and AAIA, a ST Fellow, an APSIPA Life Member, and a Signa-Xi Member of STMicroelectronics.

LUCA DAVOLI (Member, IEEE) received the Dr.-Ing. degree in computer engineering and the Ph.D. degree in information technologies from the Department of Information Engineering, University of Parma, Italy, in 2013 and 2017, respectively. He is a non-tenured Assistant Professor with the Internet of Things (IoT) Laboratory, Department of Engineering and Architecture, University of Parma. He is a Research Scientist with things2i ltd., a spin-off of the University of Parma dedicated to the IoT and smart systems. His research interests include the IoT, pervasive computing, big stream, and software-defined networking.

GIANLUIGI FERRARI (Senior Member, IEEE) received the Laurea (summa cum laude) and Ph.D. degrees in electrical engineering from the University of Parma, Parma, Italy, in 1998 and 2002, respectively. Since 2002, he has been with the University of Parma, where he is currently a Full Professor of telecommunications and also the Coordinator of the Internet of Things (IoT) Laboratory, Department of Engineering and Architecture. He is the Co-Founder and the President of things2i Ltd., a spin-off of the University of Parma dedicated to the IoT and smart systems. His current research interests include signal processing, advanced communication and networking, the IoT, and smart systems.