IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 13 September 2022, accepted 12 November 2022, date of publication 17 November 2022,
date of current version 22 November 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3222986

RESEARCH ARTICLE

Comparison of Machine Learning Algorithms for Performance Evaluation of Photovoltaic Energy Forecasting and Management in the TinyML Framework

GIAMBATTISTA GRUOSSO, (Senior Member, IEEE), AND GIANCARLO STORTI GAJANI, (Senior Member, IEEE)
DEIB, Politecnico di Milano, 20133 Milan, Italy
Corresponding author: Giambattista Gruosso (giambattista.gruosso@polimi.it)

ABSTRACT The availability of reliable photovoltaic (PV) power forecasting tools is an important factor for the dissemination of this technology. This is true not only for the integration of these difficult to predict sources in large power grids but also for small grids or standalone applications. The concept of *edge computing*, through the use of small, low power and inexpensive devices can help to make predictions more localized and feasible also in small size applications. In this article prediction methods based on Artificial Neural Networks (ANNs) models are considered and compared, along with the possibility of reducing their cost in terms of memory and computational power requirements possibly without increasing prediction error. It is shown that quantization and pruning methods, implemented in the AI libraries of a common platform for Microcontroller programming, is a viable solution of this problem. Solar panel aging effects are also considered, and it is shown how the same system used for the prediction can be an indicator of reduced plant efficiency.

INDEX TERMS Neural networks, forecasting, photovoltaic, microcontroller, estimation, data-driven, TinyML, machine learning, edge computing.

I. INTRODUCTION

Photovoltaic (PV) power generation is one of the most important renewable and sustainable energy sources. Although this has an indisputable positive effect on the environment, the introduction of a relatively unpredictable source, such as PV, poses a number of challenges to the entities responsible for distributing and controlling energy use. This is true not only for the integration of these sources into a large grid, but also for microgrids or, as in the case of the main target application of this article, single panel autonomous systems.

As a general remark, the availability of tools capable of reliable PV performance forecasting can help to create reconfigurable systems, such as in the case of microinverters for partial shadowing of panels [1], or to determine aging of panels or other forms of inefficiency. In most cases forecasting is centralized, but it would make much more sense to do it locally, through edge computing systems [2], [3] based on low power microcontrollers.

In these cases, the implementation of algorithms requiring a large amount of memory or computationally intensive could be too expensive or even unfeasible. An example of this situation is given by small, off-grid, PV based power generators with storage capabilities for rural areas in developing countries. The energy produced by these standalone units could be used to operate different tasks with different properties in terms of power, importance, and continuity requirements; tasks that range from low power non critical tasks such as phone charging to more energy-intensive and non interruptible tasks such as water purification. Prediction of power availability is mandatory to efficiently program which tasks should be activated and which should be postponed.

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

In any case, tools capable of predicting, even in the short term, the power output expected from a PV plant are needed, and this need has spurred a significant number of contributions from the scientific community.

Several solutions based on different classic Machine Learning (ML) and Deep Learning (DL) can be found in literature, see e.g. [4] for a non-commented list of about 40 contributions up to 2016. An interesting machine learning method for PV fault detection is presented in [5]. In [6] a least-squares support vector machine (SVM) method is used for short-term power prediction; the authors of this article show that, for the application considered, their approach is superior to autoregressive (AR) and autoregressive moving average models (ARMA) such as those proposed in [7]. SVM models are also proposed in [8], using as input data satellite images. In addition, in this case, the model is compared with conventional linear models and Artificial Neural Networks (ANNs). Fuzzy models have also been considered; the Takagi-Sugeno fuzzy model is proposed in [9] and a different fuzzy approach is found in [10], where it is combined with ANNs models. Wavelet-based methods, also used in combination with ANNs are proposed in [11] with application to both wind and solar power prediction.

A nice comparison of ANNs, SVM models, Multivariate Linear Regression, and k-nearest neighbor methods is found in [12], in this paper it is shown that ANNs outperform all other algorithms. Among the relatively early uses of ANNs for solar prediction, we have [13] and [14], where the Radial Basis Function (RBF) networks are used to predict daily global solar radiation. Among more recent ANN based contributions an interesting model, based on Convolutional Neural Networks (CNNs) and using regional data maps as input, is found in [15] and is used to forecast potential power output for Photovoltaic (PV) plant installation. A problem very similar to the one presented here and also based on the use of TinyML methods can be found in [16]. Another interesting model, in which three different ANN architectures are compared, is found in [17]. A recent comparison of the performance of the Supervised Learning Algorithm for solar power prediction is found in [18]; this paper considers and compares both classical Machine Learning (ML) algorithms, such as k-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Linear Regression, and a very basic ANN model. Finally, in [19] a review and comparison of several recent results is presented with respect to the ML and ANN models applied to solar, wind and PV power sources.

As can be seen, the scientific interest in this topic is significant and, in most cases, it is shown that ANNs models perform better than the classic ML methods; however, the limitations posed by the application considered here require further study. In fact, Deep Neural Network (DNN) models are generally based on a very large number of parameters and, correspondingly, requirements in terms of storage memory. In general, a large number of parameters will also require the same order of magnitude of computationally intensive operations, meaning that powerful hardware platforms are generally required; and this is true not only for the initial learning phase, but also when previously trained models are used to perform *inference*. The application envisioned in this article requires a low-power, portable, and light hardware implementation to be effective, and the hardware of choice is a platform based on a Microcontroller Unit (MCU). For this reason, after accuracy, the main requirements of any type of control or prediction algorithm useful for this application are in terms of memory and computational power.

This is the perspective of *Edge Computing* [20] where the computation is driven as much as possible toward the edge of the network where the data are generated. The main idea is thus to train the ANN on a separate platform and then export the model on a MCU finding a way to simplify the model without losing much in terms of accuracy [21], [22]. To do so, the tool that has been used is TensorFlow Lite (TFLite), a framework developed by Google for embedded devices. TFLite has been tested with good results on several different classes of devices, ranging from Android and iOS devices to embedded Linux; there is also a MCU version of TFLite, the one used in this paper, that is capable of running on small platforms with only kBs of memory. The MCU version of TFLite has a runtime of only 18 kB and has been tested with success on platforms such as Arduino and other devices based on Arm Cortex-M Series or the ESP2 architectures.

The main original aspects of this work can be summarised as follows:

- Implementation of different models and evaluation of their performance in solving the photovoltaic energy production prediction problem.
- The performance of neural networks is also compared in terms of accuracy, in terms of memory and computing power required, especially with a view to implementation on a microcontroller.
- Reduction and simplification, where possible, through quantization and relative performance evaluation. Indeed, this operation significantly reduces memory requirements but, in some cases, may introduce a certain decay in accuracy.
- Implementation of photovoltaic panel aging prediction models.
- The paper give an original framework to test tinyML algorithms on MCUs

In section II, the methodology used will first be described, along with the structure of the dataset and some *feature engineering* required to introduce time periodicity effects in the data. In Section III the architecture of the neural network models considered in this paper is presented along with a general description of the TinyML framework. In section IV results, including complexity reduction and ANN model accuracy will be presented. Finally, in section V concluding remarks are given.

II. METHODOLOGY AND AVAILABLE DATA

A. PROBLEM DESCRIPTION

The main goal of this study is to assess the feasibility and performance of different PV power prediction models when implemented on low power MCUs using the TinyML paradigm. Among all possible applications, one is given by small stand-alone micro PV plants for regions that have no power grid connection, this is the case of remote areas in developing countries. In this case, PV power prediction is needed to decide which load can be accepted or denied depending on priority and load interruptibility *versus* predicted power availability.

For this type of problem, a short-term prediction model is needed; the focus is on models that can predict $1h$, $6h$ and $12h$ in the future given the past knowledge of time windows ranging from $1h$ to $24h$.

Another important issue to analyze is the power decay due to aging, which, as shown below, implies having data ranging over a significant number of years.

Since ML models often require extensive resources in terms of memory and computing power, the main challenge of this study is the implementation of algorithms that can be easily reduced and made compatible to the limitations of low power MCUs. In our case model training will be performed offline, the trained model will then be reduced using pruning and quantization methods and, at this point, uploaded to the MCU for testing.

B. THE DATASET

The dataset used to evaluate the performance of the Neural Network models that have been considered in this study is extracted from the public data provided by DKASC (Desert Knowledge Australia Solar Centre) [23]. The complete dataset includes data from several different PV technologies, such as Monocrystalline Silicon, Polycrystalline Silicon, Cadmium Telluride thin film, from about 40 micro plants from different manufacturers. For each plant electrical data is sampled every 5 minutes and, at the same time, a central weather station samples environmental data such as solar radiation, temperature, wind direction and speed, and relative humidity.

In particular, the weather data in the DKASC database that will be used include Global Horizontal Radiation (GHR), Diffuse Horizontal Radiation (DHR), Daily Rainfall (DR) and temperature, and for each PV plant, Active Energyy Delivered or Received (AEDR), Current Phase Average (CPA), Active Power (AP) and Performance Ratio (PR). In the following, only the AP data have been considered for each plant. New PV elements have been gradually added to the DKASC field since 2008; nevertheless, for most types of PV panels, the available data span several years.

Statistical data summarizing the properties of some of the data available in the DKASK subset chosen for this study are shown in Table 1, where GHR and DHR are expressed in W, DR is represented by cumulative mm of water updated each 5' and reset each 24h, AP is in kW, while T is in K.

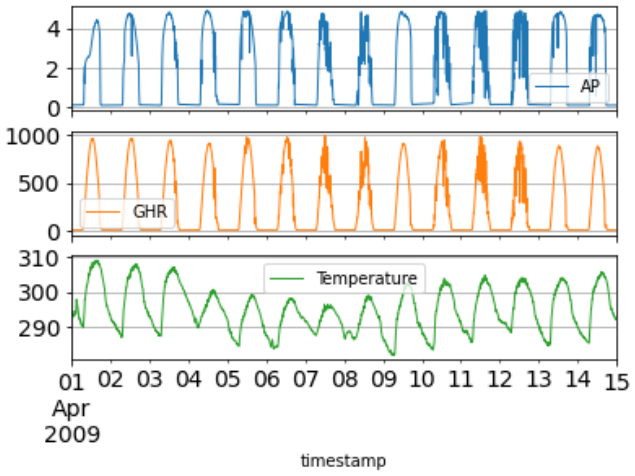


FIGURE 1. Two weeks of sample data from April 2009 from the dataset used.

TABLE 1. Some statistical data that describe the properties of GHR, DHR, DR, AP and Temperature in the dataset used.

	GHR	DHR	DR	AP	T
count	411264	411264	411264	411264	411264
mean	246.597	49.482	0.049	1.373	294.186
std	335.429	67.989	0.180	1.784	8.775
min	1.825	0.561	0.000	0.081	278.005
max	1000.1	253.675	1.200	4.889	308.915

A graphical view of the distribution of solar radiation and temperature data for the time period considered is shown in the violin plots in Fig. 2 and Fig. 3 respectively. In these plots only daytime was considered, since it is pointless to consider also the zero radiation periods at night.

Obviously, all the data in these datasets display a strong daily periodicity and a weaker, but still significant, annual periodicity. For this reason, it has been decided to add, as additional features, periodic functions of time of day and of time of year to take into account these phenomena. These functions are simple sine and cosine functions whose period is one day or one year. The resulting new features are F^s_{dp}, F^c_{dp} in (1) for the daily periodicity and F^s_{yp}, F^c_{yp} in (2) for annual effects.

$$F^s_{dp} = \sin\left(m * \frac{2\pi}{24 * 60}\right) \quad F^c_{dp} = \cos\left(m * \frac{2\pi}{24 * 60}\right) \tag{1}$$

$$F^s_{yp} = \sin\left(d * \frac{2\pi}{365.25}\right) \quad F^c_{yp} = \cos\left(d * \frac{2\pi}{365.25}\right) \tag{2}$$

DKASC has been continuously collecting data since 2008, allowing for PV power decay analysis. Since the data in DKASC contains datasets from a number of different technologies, a dataset showing a significant difference from 2008 to 2021 has been chosen for the power decay analysis. To evaluate which PV dataset was best suited, active power output was compared in pairs of days showing similar

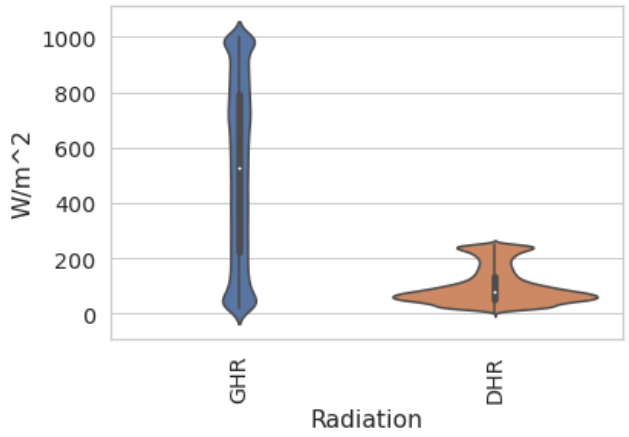


FIGURE 2. Violin graph for GHR and DHR in the dataset used.

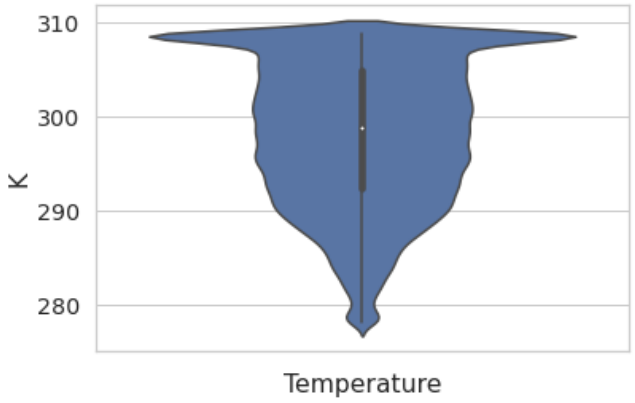


FIGURE 3. Violin graph for temperature in the dataset used.

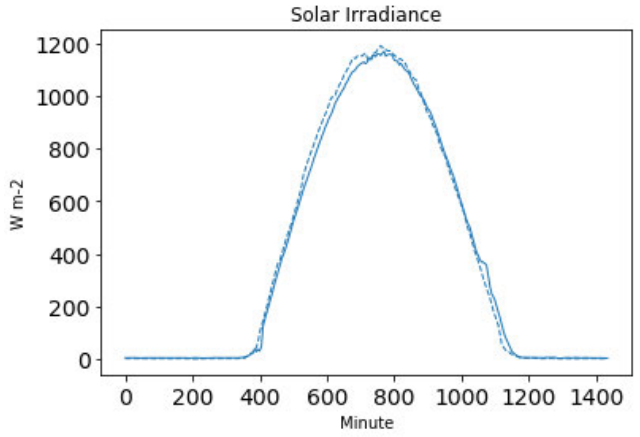


FIGURE 4. GHR in Jan. 12, 2009 (solid line) and in Jan. 1, 2021 (dashed line).

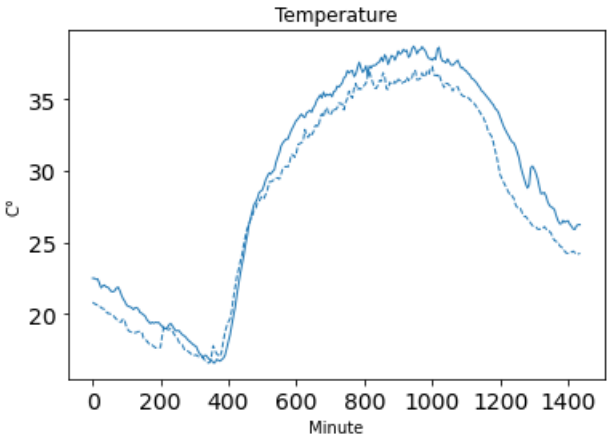


FIGURE 5. Temperature (Celsius) in Jan. 12, 2009 (solid line) and in Jan. 1, 2021 (dashed line).

weather conditions in 2009 and 2020; specifically, by comparing, for each day and using the modified cosine similarity function in (3), the values of GHR and T (Temperature) in vectors representing 24h data windows.

$$S(x, y) = \frac{xy}{\|x\|\|y\|}\left(1 - \frac{\text{abs}(\|x\| - \|y\|)}{\|x\| + \|y\|}\right) \tag{3}$$

A sample result in terms of GHR and T for a pair of days with a high similarity value is shown in Fig. 4 and Fig. 5.

Using this metric the dataset corresponding to Array 6, Kyocera, 5.4kW, poly-Si, with dual axis tracking [23], has been chosen for our analysis.

C. DATA PREPROCESSING

Some preprocessing of the data in the DKASK dataset was mandatory:

1) Power data and environmental data are in separate sets, these had to be aligned and merged in one single set.
2) Wind and humidity parameters have been discarded since they have been shown to have a mostly irrelevant impact on our problem.
3) Outliers and missing data has been interpolated.

The whole dataset is then normalized in order to have zero mean and unit variance.

III. NEURAL NETWORK ARCHITECTURES AND TinyML

A large number of different ANN models have been proposed in the literature and have been used in a large number of different fields and applications; see, e.g., [24] for a good review and application to water quality prediction, [25] for Convolutional Neural Networks applied to image recognition tasks, or [26] for a general review of applications concerning energy prediction.

In this paper a typical *time series* prediction problem is considered. This means that each ANN model will be presented with a window of known "past" data, where N_p is the size of this window in terms of the number of data samples, and will be required to predict the "future" value of AP after a given time lapse Δt corresponding to N_f data samples. The length of the past data window N_p and the time lapse N_f will affect the final results in terms of accuracy and memory requirements.

In this paper three different basic ANN models will be considered:

1) Multi Layer Perceptrons (MLPs) that are the foundation of DNNs and essentially the ultimate evolution of the architecture proposed by Rosenblatt in 1958 [27].
2) CNNs based on the DNN idea, but with one or more *hidden* (i.e. not directly connected to the input or output) convolution layer [28].
3) Recurrent Neural Networks (RNNs) in the forms of
   a) Long Short Term Memory (LSTM) first introduced in [29] by Hochreiter et al.
   b) Gate Recurring Unit (GRU) introduced by Cho et al. in [30]

More details on the specific instances of these models and how they are used in this study are given in the following sections.

A. INPUT DATA FOR SHORT AND MID TERM PREDICTION

The features chosen for the prediction of short-term active power are eight: GHR, DHR, DR, Temperature, F^s_{dp}, F^s_{yp}, F^c_{dp} and F^c_{yp} while the feature that is predicted, acting as *label*, is AP; thus, the total width of the input feature F_w is nine. The data are sampled with a period of 5 minutes, which means that each 1h data window is represented, in the nominal case, by $N_h = 12$ data samples. Interesting results have also been obtained by downsampling the data, thus using smaller values of N_h. In the network descriptions that follow, it is assumed that predictions are made based on a set of different *past* data windows with length $w \in \mathcal{W} \equiv \{1\text{h}, 2\text{h}, 4\text{h}, 8\text{h}, 12\text{h}, 24\text{h}\}$ so that the total number of samples is $N_p = wN_h$, while the predicted data are in all cases a single sample of AP with different values of Δt at 1h, 6h and 12h in the future corresponding to $N_f = 12, 72$ and 144, respectively.

For all networks the input data shape is therefore $N_p \times F_w$, that is, $wN_h \times 9$ and obviously varies depending on how many past hours of data are considered. Since in this phase only short and mid term prediction is considered, a subset of the whole available data for the chosen PV plant has been considered.

B. TRAINING, VALIDATION AND TEST SUBSETS

In order to train the ANN models, the whole dataset has to be divided into subsets to be used for training, validation, and testing. Training data are actually used to train the model, validation data to check results while training in order to avoid possible overfitting, and finally, the test subset is used to validate the model performance on data that have not been used in the training process.

For short term prediction the full data from years 2012 and 2013 has been extracted from the plant database and used as train dataset, the first 182 days from 2014 have been used for the validation subset, while the remaining portion of 2014 for the test set.

C. BASIC MULTI LAYER PERCEPTRON NETWORK

The simplest network that has been considered is an instance of MLP with a first *flatten* layer, that transforms the two-dimensional input data in a one-dimensional vector, and four dense hidden layers of decreasing size all using the *ReLu* activation function. Each neuron of these layers computes a simple weighted linear combination of its inputs, eventually adds a scalar value, and finally evaluates the activation function on the result. In this case, the commonly used and computationally simple *ReLu* activation function is used; this function will simply set to zero all negative values and leave positive ones unchanged. A final single neuron output layer, in this case without an activation function, equivalent to a simple *linear* combination of all inputs, yields the final scalar output. The basic structure of this network is presented in Fig. 6. As shown, the hidden dense layers have been chosen to be composed by a decreasing number of neurons, namely 64, 32, 16 and 8 for each layer from input to output. In the network definition *dropout* layers have been added between each dense layer. These are actually *pseudolayers* that are active only during training and are responsible for randomly setting inputs to zero to a small fraction of neurons in the layer. This action reduces the chance of *overfitting* i.e. having a network that is essentially specialized on the training set and not on more general and never before seen input data. Tests using different layer configurations did not significantly improve the accuracy of the results obtained.

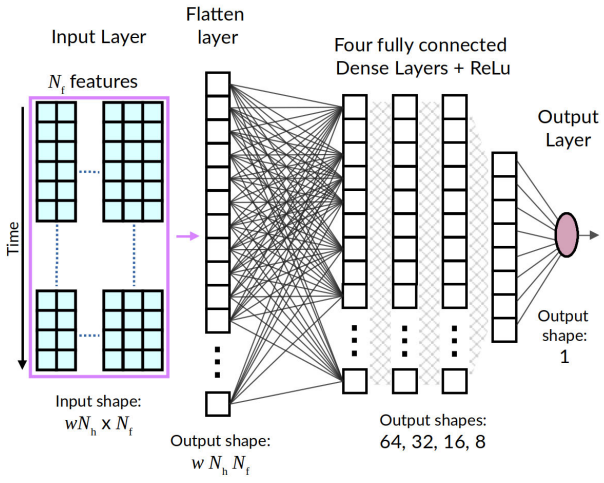


FIGURE 6. Architecture of the MLP neural network used.

D. CONVOLUTIONAL NEURAL NETWORK

CNNs have been shown to yield good results in time series forecasting (see e.g. [31]). The CNN implemented in this study is composed of a first one-dimensional convolution layer with 32 filters and a kernel size equivalent to the number of input time steps N_p, i.e. from 12 to 288 depending on the size of the past window, and the *ReLu* activation function. Two hidden dense layers with, respectively, 32 and 16 neurons and again the *ReLu* activation follow, a final single linear

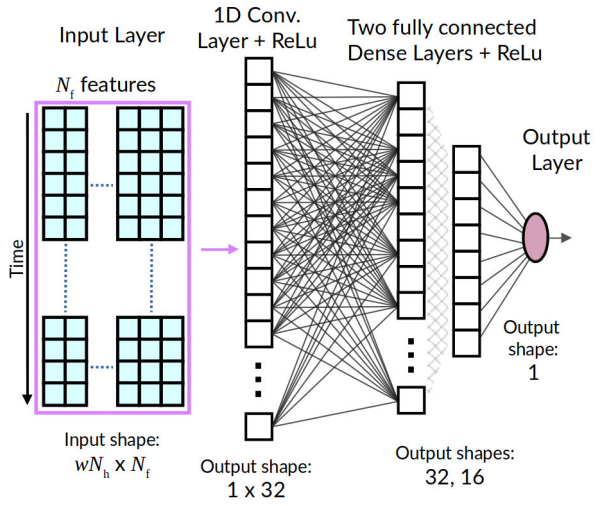


FIGURE 7. Architecture of the CNN neural network used.

neuron output layer computes the scalar prediction. The basic CNN network architecture considered in this paper is shown in Fig. 7. While the behaviour of the neurons in the dense layers is exactly as for the MLP model, the initial convolution layer performs a different task. This layer builds a number of filters, in our case 32, obtained through a cross-correlation operation with each input batch. Each filter provides one output value.

E. LONG SHORT TERM MEMORY NETWORK

LSTM networks, as all RNNs, are well suited for time series prediction. Each unit in a LSTM layer has two states, one representing long-term memory and called *cell state* and a second one, called *hidden state* that is adjusted at each iteration and represents a form of short-term memory. The hidden state at the previous step and the current input are used to adjust the cell state slowly and to determine the new value for the hidden state. The activation function used for this layer is the Hyperbolic Tangent and the usual linear output dense layer yields the output. The basic LSTM network architecture considered in this paper is shown in Fig. 8. The number of neurons (units) in the LSTM layer is 32. Once again, larger sizes did not significantly improve the results.

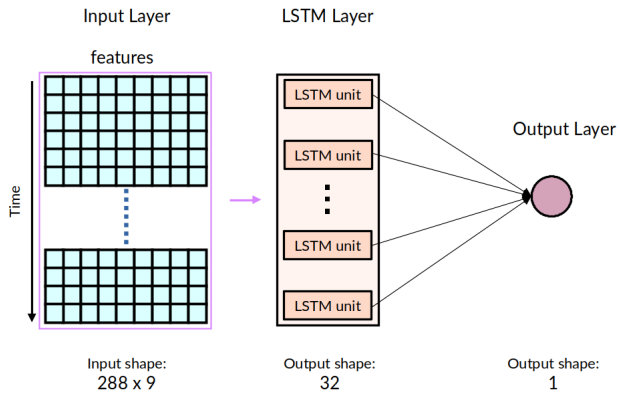


FIGURE 8. Architecture of the LSTM neural network used.

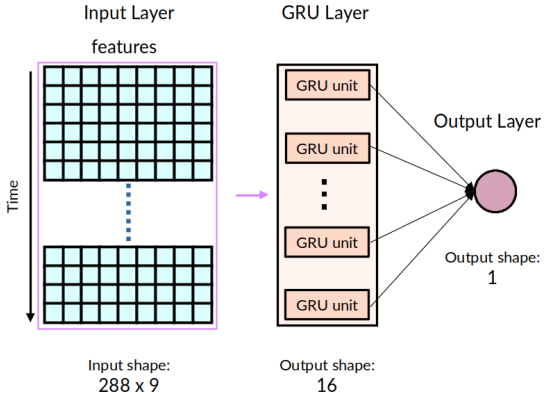


FIGURE 9. Architecture of the GRU neural network used.

F. GATE RECURRING UNIT NETWORK

GRU networks are very similar to the LSTM networks but require a reduced number of tensor products and each unit has only one internal state. Since in many cases GRUs yield results as good as those of LSTMs, and that the specific application envisioned in this paper is to be implemented using TinyML on a small micro-controller platform, these lighter requirements in terms of processing power are very attractive. Also for this layer the activation function used is the hyperbolic tangent, and a linear output dense layer is used to compute the output. The basic GRU network architecture considered in this paper is shown in Fig. 9.

IV. RESULTS AND EVALUATION

A. COMPLEXITY PROFILING

The trained neural networks will be embedded onto a low power MCU, model name STM32F302VCTx, integrated on an ad-hoc fabricated industrial board provided by SECO. The memory embedded in the core system MCU consists of 32kB RAM and 256kB ROM, with an ARM Cortex-M4 Core @ 72 MHz. STM32Cube.AI software tool (version 7.1.0) [32] was used to convert pre-trained models into optimized ANSI C code, which also provides analysis of memory usage both on the target board and on the desktop.

The models can also be quantized to reduce memory requirements. This operation has some effects on the quality of the prediction results. The tool used to convert the pre-trained models was STM32Cube.AI software tool (version 7.1.0), which automatically converts the ANN models into optimized ANSI C code for embedded MCUs, allowing, at the same time, their performances analysis. Model validation can be run on both the PC and the MCU board.

The same tool can then be used to *quantize* pre-trained models based on 8 bit integers. This operation can significantly improve memory requirements and power consumption, since the ANN runtime is reduced. Specifically, NN weights and associated activation values are converted

from 32-bit floating-point to 8-bit integer precision, with, as will be shown, essentially irrelevant degradation in terms of model accuracy. The algorithm used to generate the quantized values is the standard "Minmax" method included in STM32Cube.AI, which bases the quantization process on the minimum and maximum values of all weights and activations. The floating-point values of weights and activations are first scaled and then shifted:

$$x_I = \frac{x_F}{S} + Z$$

where the original floating point values are denoted by x_F while x_I are the new 8-bit integer values; S is the scaling coefficient, a floating point number, and Z the "zero point" shift value, an 8-bit integer that corresponds to the quantized value of floating point zero [33]. Using this algorithm the quantized values x_I are linearly distributed around Z, and final precision depends on the scale factor. In the STM32Cube.AI implementation the weights can be asymmetric, with $Z \neq 0$ or symmetric, in this second case $Z = 0$, activations, due to their nature, are only asymmetric. Post-quantized TensorFlow Lite models use the symmetric and signed integer mode for the weights, and asymmetric and signed integer for the activation values. The same scheme (coded as "ss/sa" in the STM32 documentation) was chosen for the quantization process. Quantization of recursive layers such as LSTM and GRU is not supported, and therefore their execution remains in floating-point.

1) MLP

A first analysis of complexity obtained through STM32 Cube.AI has been performed on the MLP architecture for different values of $w \in \mathcal{W}$ (i.e. hours of past data used both in training and in evaluation). Results of this complexity analysis are shown in Table 2 where the required RAM and Flash memory are expressed in KiB and computation complexity by number of Multiply ACCumulate (MACC) operations in 32bit floating point arithmetic.

TABLE 2. Complexity of the non-quantized MLP model for different values of $w \in \mathcal{W}$ (* values to large for the chosen MCU).

	MLP non-quantized		
w	RAM	Flash	MACC
1h	0.82 kiB	38.0 kiB	9850
2h	1.22 kiB	65.0 kiB	16762
4h	2.07 kiB	119.0 kiB	30586
8h	3.75 kiB	227.0 kiB	58234
12h	5.44 kiB	335.0* kiB	85882
24h	10.50 kiB	659.0* kiB	168826

Note that the requirements in terms of RAM, ROM and MACC depend essentially only on the value of w and not of Δt; this is due to the fact that only the size of input data influences the number of internal weights and the number of MACC operations needed to obtain a prediction. Therefore, these complexity results are the same for all values of Δt that have been considered. This is true also for all other ANN models considered.

As it can be seen in Table 2, in the non-quantized version of the model values of $w > 8$ can not be implemented on standard MCU boards.

After quantization it can be seen that, while RAM requirements and MACC operations vary only slightly, the amount of ROM needed is reduced by values ranging from 74% to 75%. The results are shown in Table 3.

TABLE 3. Complexity of the quantized MLP model for different values of $w \in \mathcal{W}$.

	MLP quantized		
w	RAM	Flash	MACC
1h	1.00 kiB	9.86 kiB	9729
2h	1.32 kiB	16.61 kiB	16641
4h	1.95 kiB	30.11 kiB	30465
8h	3.22 kiB	57.11 kiB	58113
12h	4.49 kiB	84.11 kiB	85761
24h	8.28 kiB	165.12 kiB	168705

Note that, in this case, all w values are compatible with the chosen MCU board (and with most other low power boards).

2) CNN

The CNN model, as will be shown in the next section, yields good results in terms of accuracy. Performing the same complexity analysis in exactly the same conditions as with the MLP just seen, results for the non-quantized CNN model are shown in Table 4. As it can be seen the CNN model that has been considered has lager RAM requirements but significantly smaller Flash memory requirements with respect to the non-quantized MLP model. With quantization, like in the previous case, requirements in terms of RAM and complexity in terms of MACC change very little; flash memory requirements, on the other hand, are again drastically reduced by values ranging from 73% to 75%.

TABLE 4. Complexity of the non-quantized model CNN for different values of $w \in \mathcal{W}$ (* values to large for the chosen MCU).

	CNN non-quantized		
w	RAM	Flash	MACC
1h	5.71 kiB	20.23 kiB	5169
2h	6.00 kiB	33.98 kiB	8625
4h	6.98 kiB	61.46 kiB	15537
8h	8.59 kiB	116.43 kiB	29361
12h	10.11 kiB	171.40 kiB	43185
24h	16.93 kiB	336.32* kiB	84657

3) LSTM

The LSTM model is not quantizable by the STM32Cube.AI system. The complexity related to the standard non-quantized model is thus reported in Table 6. Note that, due to the

TABLE 5. Complexity of the quantized CNN model for different values of $w \in \mathcal{W}$.

	CNN quantized		
w	RAM	Flash	MACC
1h	6.00 kiB	5.21 kiB	5089
2h	6.52 kiB	8.58 kiB	8545
4h	7.58 kiB	15.33 kiB	15457
8h	9.69 kiB	28.83 kiB	29281
12h	11.80 kiB	42.33 kiB	43105
24h	18.13 kiB	82.83 kiB	84577

TABLE 6. Complexity of the nonquantized LSTM model for different values of $w \in \mathcal{W}$.

	LSTM non-quantized		
w	RAM	Flash	MACC
1h	1.43 kiB	25.63 kiB	66017
2h	1.85 kiB	25.63 kiB	130913
4h	2.69 kiB	25.63 kiB	260705
8h	4.38 kiB	25.63 kiB	520289
12h	6.07 kiB	25.63 kiB	779873
24h	11.13 kiB	25.63 kiB	1558625

peculiar architecture of LSTM, the required amount of Flash memory does not change for different values of w, on the other hand, the complexity of the recursive layer yields very large values for the MACC attribute.

4) GRU

Also for the GRU model quantization is not possible and, due to its nature, the Flash memory requirements do not change with w. Complexity results for this model are reported in Table 7. As can be seen, this model also requires a large number of operations, only slightly smaller than those required by LSTM.

TABLE 7. Complexity of the non-quantized GRU model for different values of $w \in \mathcal{W}$.

	GRU non-quantized		
w	RAM	Flash	MACC
1h	1.30 kiB	20.38 kiB	49121
2h	1.72 kiB	20.38 kiB	97121
4h	2.57 kiB	20.38 kiB	193121
8h	4.25 kiB	20.38 kiB	385121
12h	5.94 kiB	20.38 kiB	577121
24h	11.00 kiB	20.38 kiB	1153121

B. EFFECTS OF QUANTIZATION

Quantization has, in general, some effect on the accuracy of the ANN models. One would expect that, after quantization, some decay in accuracy should be observed. In the case here considered, if the Root Mean Square Error (RMSE) of the MLP and CNN models are compared before and after quantization, it can be seen that the difference is negligible, and in some cases the quantized models even yield better results. In Fig. 10 the RMSE for MLP in the case $\Delta t = 1$h is shown, as can be noted, the impact of quantization is not very relevant. Similar results can be seen from the CNN model, as shown in Fig. 11. Quantization effects on the accuracy of predictions are negligible, and this is true also for different values of Δt; in the following, only the quantized models will be considered for MLP and CNN.

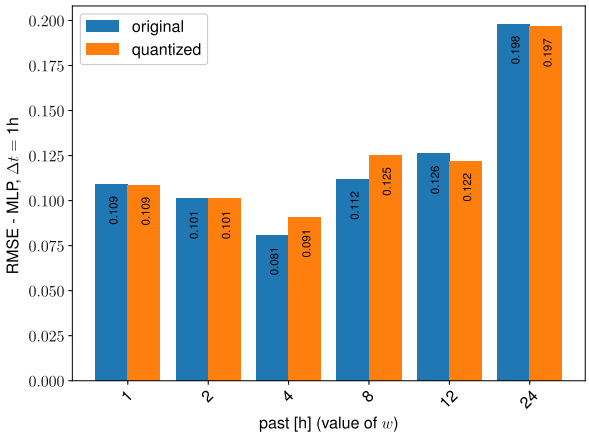


FIGURE 10. RMSE of the original and quantized MLP model for different values of w.

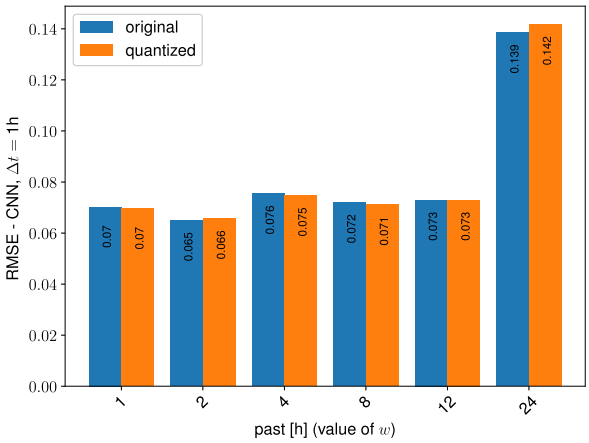


FIGURE 11. RMSE of the original and quantized CNN model for different values of w.

C. PREDICTION ACCURACY

There are various metrics that can be used to evaluate the accuracy of ANN. The most used are RMSE and Mean Absolute Error (MAE). These metrics are apparently quite similar but have some interesting and useful differences; in fact, we have:

$$\text{RMSE} \equiv \sqrt{\frac{1}{n}\sum_{i=1}^{N}(y_i^2 - \hat{y}_i^2)}$$
$$\text{MAE} \equiv \frac{1}{n}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{4}$$

where $\hat{y}_i$ are the predictions and y_i the actual observations. Both metrics are based on the sum of some function of prediction errors, have the same dimensions as the observed quantities, and do not consider the sign of each error value; nevertheless, since in RMSE the sum is averaged *before* taking the square root, it will always be RMSE $\geq$ MAE with the difference between the two metrics related to the variance of the distribution of error values, since RMSE is more sensitive to large error values.

The results in terms of MAE for all ANN models for different values of w and for $\Delta t = 1, 6$ and 12 are shown in Fig.12, Fig.13 and Fig.14 respectively. As expected, larger values of Δt yield larger values of MAE. As can be seen, an interesting first result is that a value of $w = 1$ yields better results than using a larger number of past samples. This effect may seem surprising, but shows the importance of adding an artificial feature depending on time of year and time of day, as outlined in section II-B, effects of local perturbations, such as clouds or temperature variations end up being more diluted if more past data is fed to the model.

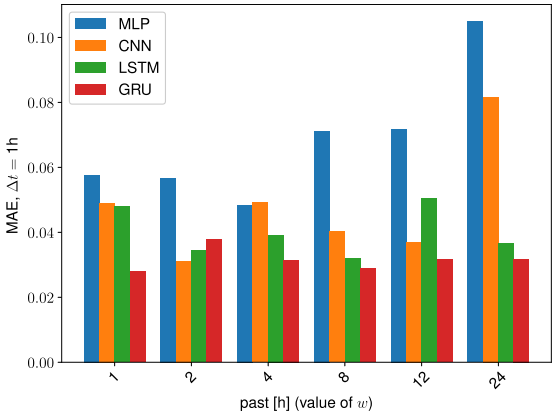


FIGURE 12. MAE of all the models for different values of w at $\Delta t = 1$.

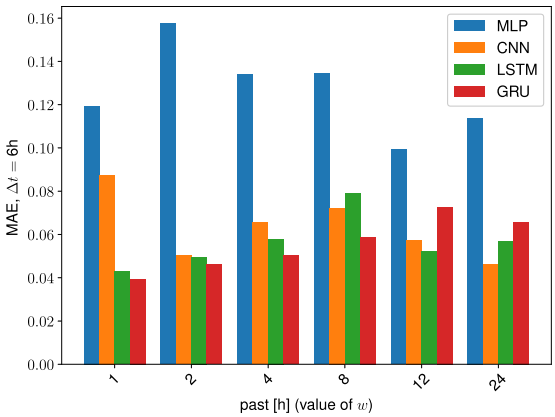


FIGURE 13. MAE of all the models for different values of w at $\Delta t = 6$.

The second result is that the GRU model outperforms all other models for all Δt values.

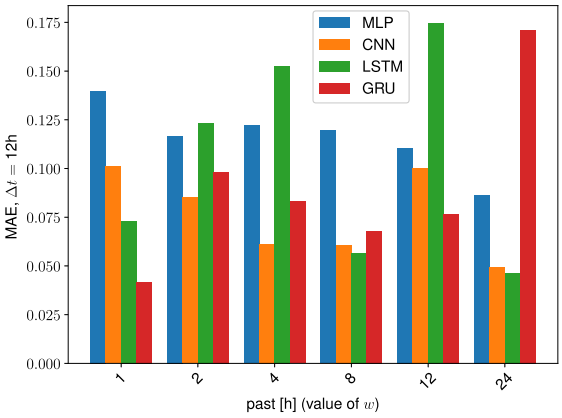


FIGURE 14. MAE of all the models for different values of w at $\Delta t = 12$.

Some sample prediction data, related to 4 consecutive days, are shown in Fig. 15. The predictions in this figure are related to ANNs models trained on data from October 2008 to July 2009, and tested on data from the remaining part on 2009. As can be seen, the actual power data from the solar field are slightly underestimated by all ANNs models.

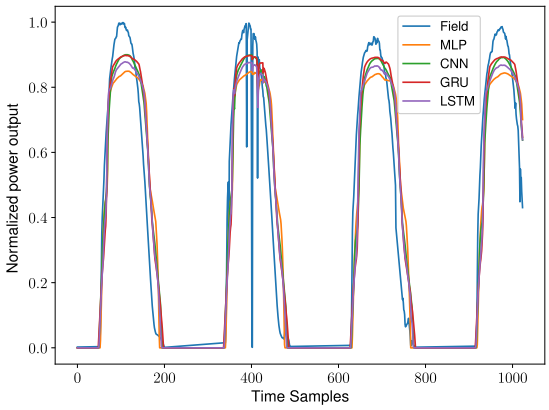


FIGURE 15. Prediction of all ANNs trained on 2009 data compared to measured data from the same year.

D. SOLAR PANEL AGING

To evaluate the effects of aging on solar panels, the ANNs models have been trained on data from the early days of the chosen solar field and used to predict power output using recent environmental data. Recall that the dataset has been chosen so that we could have a long span in terms of years of data and in terms of the availability of similar irradiance and temperature profiles.

With this approach it is expected that the available real power output in recent times by aged panels, compared with data predicted using just deployed ones, will show some decay. Furthermore, the prediction accuracy is expected to be lower.

A first qualitative idea of this performance decay can be immediately seen by comparing Fig. 15 with similar results shown in Fig. 16. The predictions in this figure are obtained

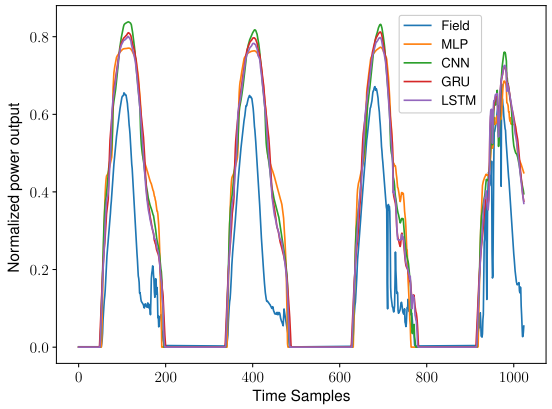


FIGURE 16. Prediction of all ANNs trained on 2009 data compared to measured data from the solar field in 2020.

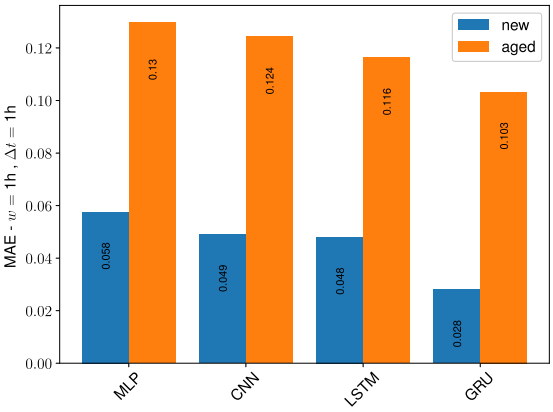


FIGURE 17. MAE for all ANNs trained on 2009 and tested on 2009 data (when the field was 'new') and 2020 data (when the field was 'aged').

using the same ANNs trained on 2008 and 2009 data and predicting power output using 2020 weather data. As expected, all ANNs models overestimate panel performance. Due to ageing effects the solar field considered is delivering much less power than predicted by the models trained on the original solar field data.

To have a more quantitative measure of this decay effect, the MAE value for all models in the case $w = 1$, $\Delta t = 1$ has been evaluated and is shown in Fig. 17. In this figure, the prediction accuracy for new panels, i.e., the performance of the solar field in 2009, immediately after the data used for training is compared to aged panels, where predictions of the original ANNs using 2020 weather data are compared to actual power output.

V. CONCLUSION

In this article, different ANN architectures are capable of reliable short and medium-term solar power forecasting, and the best results are obtained using limited past data. After training, these architectures can be made compact and implemented in low cost, low-power and small MCU platforms, allowing for distribution forecasting directly on solar plants or to single panel applications. All architectures achieve good results; the one showing the best performance for this application is the one based on the GRU model.

With all models, solar panel aging has relevant effects on prediction accuracy.

ACKNOWLEDGMENT

The authors would like to thank Riccardo Cal Novic for the support.

REFERENCES

[1] M. Pisano, F. Bizzarri, A. Brambilla, G. Gruosso, and G. S. Gajani, "Micro-inverter for solar power generation," in *Proc. Int. Symp. Power Electron. Power Electron., Electr. Drives, Autom. Motion*, Jun. 2012, pp. 109–113.

[2] J. Shi, N. Liu, Y. Huang, and L. Ma, "An edge computing-oriented net power forecasting for PV-assisted charging station: Model complexity and forecasting accuracy trade-off," *Appl. Energy*, vol. 310, Mar. 2022, Art. no. 118456.

[3] C. Feng, Y. Wang, Q. Chen, Y. Ding, G. Strbac, and C. Kang, "Smart grid encounters edge computing: Opportunities and applications," *Adv. Appl. Energy*, vol. 1, Feb. 2021, Art. no. 100006.

[4] M. Yesilbudak, M. Colak, and R. Bayindir, "A review of data mining and solar power prediction," in *Proc. IEEE Int. Conf. Renew. Energy Res. Appl. (ICRERA)*, Nov. 2016, pp. 1117–1121.

[5] K. Jaskie, J. Martin, and A. Spanias, "PV fault detection using positive unlabeled learning," *Appl. Sci.*, vol. 11, no. 12, p. 5599, Jun. 2021.

[6] J. Zeng and W. Qiao, "Short-term solar power prediction using a support vector machine," *Renew. Energy*, vol. 52, pp. 118–127, Apr. 2013.

[7] G. Reikard, "Predicting solar radiation at high resolutions: A comparison of time series forecasts," *Solar Energy*, vol. 83, no. 3, pp. 342–349, Mar. 2009.

[8] H. S. Jang, K. Y. Bae, H.-S. Park, and D. K. Sung, "Solar power prediction based on satellite images and support vector machine," *IEEE Trans. Sustain. Energy*, vol. 7, no. 3, pp. 1255–1263, Jul. 2016.

[9] R. Iqdour and A. Zeroual, "Prediction of daily global solar radiation using fuzzy systems," *Int. J. Sustain. Energy*, vol. 26, no. 1, pp. 19–29, Mar. 2007.

[10] X. Xue, "Daily diffuse solar radiation estimation using adaptive neuro-fuzzy inference system technique," *Numer. Heat Transf., B, Fundam.*, vol. 77, no. 2, pp. 138–151, Feb. 2020.

[11] G. Capizzi, F. Bonanno, and C. Napoli, "A wavelet based prediction of wind and solar energy for long-term simulation of integrated generation systems," in *Proc. SPEEDAM*, Jun. 2010, pp. 586–592.

[12] H. Long, Z. Zhang, and Y. Su, "Analysis of daily solar power prediction with data-driven approaches," *Appl. Energy*, vol. 126, pp. 29–37, Aug. 2014.

[13] M. Mohandes, A. Balghonaim, M. Kassas, S. Rehman, and T. O. Halawani, "Use of radial basis functions for estimating monthly mean daily solar radiation," *Sol. Energy*, vol. 68, no. 2, pp. 161–168, Feb. 2000.

[14] A. Mellit, M. Benghanem, and M. Bendekhis, "Artificial neural network model for prediction solar radiation data: Application for sizing stand-alone photovoltaic power system," in *Proc. IEEE Power Eng. Soc. Gen. Meeting*, Jun. 2005, pp. 40–44.

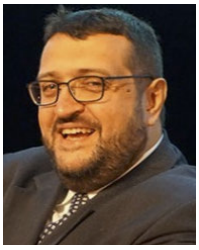
[15] J. Heo, K. Song, S. Han, and D.-E. Lee, "Multi-channel convolutional neural network for integration of meteorological and geographical features in solar power forecasting," *Appl. Energy*, vol. 295, Aug. 2021, Art. no. 117083.

[16] V. Archana, "Tinyml for solar panels: Bringing edge computing applications to solar energy systems," M.S. thesis, NUS, Singapore, 2020.

[17] P. M. Kumar, R. Saravanakumar, A. Karthick, and V. Mohanavel, "Artificial neural network-based output power prediction of grid-connected semitransparent photovoltaic system," *Environ. Sci. Pollut. Res.*, vol. 29, no. 7, pp. 10173–10182, Feb. 2022.

[18] L. Gutiérrez, J. Patiño, and E. Duque-Grisales, "A comparison of the performance of supervised learning algorithms for solar power prediction," *Energies*, vol. 14, no. 15, p. 4424, Jul. 2021.

[19] L. Abualigah, R. A. Zitar, K. H. Almotairi, A. M. Hussein, M. Abd Elaziz, M. R. Nikoo, and A. H. Gandomi, "Wind, solar, and photovoltaic renewable energy systems with and without energy storage optimization: A survey of advanced machine learning and deep learning techniques," *Energies*, vol. 15, no. 2, p. 578, Jan. 2022.
[20] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, pp. 637–646, May 2016.
[21] G. Crocioni, D. Pau, J.-M. Delorme, and G. Gruosso, "Li-ion batteries parameter estimation with tiny neural networks embedded on intelligent IoT microcontrollers," *IEEE Access*, vol. 8, pp. 122135–122146, 2020.
[22] D. Pau, D. Denaro, G. Gruosso, and A. Sahnoun, "Microcontroller architectures for battery state of charge prediction with tiny neural networks," in *Proc. IEEE 11th Int. Conf. Consum. Electron. (ICCE-Berlin)*, Nov. 2021, pp. 1–6.
[23] *Desert Knowledge Australia Solar Centre, Alice Springs*. Accessed: Apr. 1, 2022. [Online]. Available: http://dkasolarcentre.com.au/
[24] Y. Chen, L. Song, Y. Liu, L. Yang, and D. Li, "A review of the artificial neural network models for water quality prediction," *Appl. Sci.*, vol. 10, no. 17, p. 5776, Aug. 2020.
[25] A. Dhillon and G. K. Verma, "Convolutional neural network: A review of models, methodologies and applications to object detection," *Prog. Artif. Intell.*, vol. 9, no. 2, pp. 85–112, 2020.
[26] J. F. Bermejo, J. F. G. Fernández, F. O. Polo, and A. C. Márquez, "A review of the use of artificial neural network models for energy and reliability Prediction. A study of the solar PV, hydraulic and wind energy sources," *Appl. Sci.*, vol. 9, no. 9, p. 1844, May 2019.
[27] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, 1958.
[28] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.
[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[30] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Oct. 2014, pp. 1724–1734.
[31] A. Borovykh, S. Bohte, and C. W. Oosterlee, "Conditional time series forecasting with convolutional neural networks," doi: 10.48550/ARXIV.1703.04691.
[32] S. Microelectronics. *Stm32Cube MCU and MPU Packages*. Accessed: Jul. 2022. [Online]. Available: https://www.st.com/
[33] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2704–2713.



GIAMBATTISTA GRUOSSO (Senior Member, IEEE) was born in 1973. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from the Politecnico di Torino, Italy, in 1999 and 2003, respectively. From 2002 to 2011, he was an Assistant Professor at the Department of Electronics and Informatics, Politecnico di Milano, where he has been an Associate Professor, since 2011. He is the author of more than 80 papers on journals and conferences on the topics. He does research in electrical engineering, electronic engineering, and industrial engineering. His main research interests include electrical vehicles transportation electrification, electrical power systems optimization, simulation of electrical systems, digital twins for smart mobility, factory and city, and how they can be obtained from data.

GIANCARLO STORTI GAJANI (Senior Member, IEEE) received the Dr.Ing. degree in electronic engineering and the Ph.D. degree in electronic systems from the Politecnico di Milano, in 1986, and 1991 respectively. Since 1992, he has been an Assistant Professor at the Politecnico di Milano, where he also has been an Associate Professor of circuit theory, since 2002. He is the author of more than 90 papers in international journals and conferences. His research interests include the development of architectures for signal processing (in particular, for audio and music applications) and neural networks. More recently, his research interests include non linear circuits and non linear dynamics.

Open Access funding provided by 'Politecnico di Milano' within the CRUI CARE Agreement