Received October 25, 2020, accepted November 18, 2020, date of publication November 20, 2020, date of current version December 7, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3039733

Deep Neural Networks for Multivariate Prediction of Photovoltaic Power Time Series

FEDERICO SUCCETTI[1], ANTONELLO ROSATO[1], (Member, IEEE), RODOLFO ARANEO[2], (Senior Member, IEEE), AND MASSIMO PANELLA[1], (Senior Member, IEEE)

[1]Department of Information Engineering, Electronics and Telecommunications, Sapienza University of Rome, 00184 Rome, Italy
[2]Electrical Engineering Division, Department of Astronautical, Electrical and Energy Engineering, Sapienza University of Rome, 00184 Rome, Italy

Corresponding author: Massimo Panella (massimo.panella@uniroma1.it)

ABSTRACT The large-scale penetration of renewable energy sources is forcing the transition towards the future electricity networks modeled on the smart grid paradigm, where energy clusters call for new methodologies for the dynamic energy management of distributed energy resources and foster to form partnerships and overcome integration barriers. The prediction of energy production of renewable energy sources, in particular photovoltaic plants that suffer from being highly intermittent, is a fundamental tool in the modern management of electrical grids shifting from reactive to proactive, with also the help of advanced monitoring systems, data analytics and advanced demand side management programs. The gradual move towards a smart grid environment impacts not only the operating control/management of the grid, but also the electricity market. The focus of this article is on advanced methods for predicting photovoltaic energy output that prove, through their accuracy and robustness, to be useful tools for an efficient system management, even at prosumer's level and for improving the resilience of smart grids. Four different deep neural models for the multivariate prediction of energy time series are proposed; all of them are based on the Long Short-Term Memory network, which is a type of recurrent neural network able to deal with long-term dependencies. Additionally, two of these models also use Convolutional Neural Networks to obtain higher levels of abstraction, since they allow to combine and filter different time series considering all the available information. The proposed models are applied to real-world energy problems to assess their performance and they are compared with respect to the classic univariate approach that is used as a reference benchmark. The significance of this work is to show that, once trained, the proposed deep neural networks ensure their applicability in real online scenarios characterized by high variability of data, without requiring retraining and end-user's tricks.

INDEX TERMS Photovoltaic power time series, multivariate prediction, energy management system, deep learning, long short-term memory network, convolutional neural network, time series embedding.

NOMENCLATURE

n	Time step.
k	Prediction distance.
$S[n]$	Scalar sample of time series S (univariate case) at time n.
$\widetilde{S}[n]$	Estimated sample of $S[n]$.
$\mathbf{x}[n]$	Embedded input vector.
T	Time lag.
D	Embedding dimension.
f_u	Univariate regression model.
$\mathcal{I}_{n-1}$	Past information associated with the time series up to $n-1$.
H	Number of hidden units in the basic network's LSTM layer.
$h_n^{(m)},\ c_n^{(m)}$	Hidden/cell state (scalar) of the mth LSTM unit at time n.
$\mathbf{h}_n,\ \mathbf{c}_n$	Hidden/cell states (column vector) of all LSTM units at time n.

The associate editor coordinating the review of this manuscript and approving it for publication was Ahmed F. Zobaa.

$i_n^{(m)}$, $f_n^{(m)}$, $g_n^{(m)}$, $o_n^{(m)}$	Input/forget/cell candidate/output gate's value (scalar) in the mth LSTM unit at time n.
$\mathbf{i}_n$, $\mathbf{f}_n$, $\mathbf{g}_n$, $\mathbf{o}_n$	Input/forget/cell candidate/output gate's values (column vector) in all LSTM units at time n.
$\mathbf{W}_i$, $\mathbf{W}_f$, $\mathbf{W}_g$, $\mathbf{W}_o$	Input/forget/cell candidate/output gate's embedded input weights (matrix).
$\mathbf{w}_i$, $\mathbf{w}_f$, $\mathbf{w}_g$, $\mathbf{w}_o$	Input/forget/cell candidate/output gate's scalar input weights (column vector).
$\mathbf{R}_i$, $\mathbf{R}_f$, $\mathbf{R}_g$, $\mathbf{R}_o$	Input/forget/cell candidate/output gate's recurrent weights (matrix).
$\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_g$, $\mathbf{b}_o$	Input/forget/cell candidate/output gate's biases (column vector).
$\mathbf{w}_d$	Fully connected layer's weights (column vector).
b_d	Fully connected layer's bias (scalar).
$\sigma_g(\alpha)$	$= \left(1+e^{-\alpha}\right)^{-1}$. Sigmoid activation function.
$\sigma_c(\alpha)$	$=$ tanh(α). Hyperbolic tangent activation function.
$S_1[n]$	Scalar sample of time series S_1 (multivariate case) at time n.
$S_q[n]$	Scalar sample of the qth time series correlated to S_1 (multivariate case) at time n.
$\widetilde{S}_1[n]$	Estimated sample of $S_1[n]$.
$\mathbf{x}^{(m)}[n]$	Generic multivariate input to the adopted DNN model.
f_m	Generic multivariate regression model.
$\mathbf{x}^{(a)}[n]$	Multivariate input frame of the C-LSTM network.
f_a	Non-linear recurrent regression model implemented by the C-LSTM network.
$F^{(a)}$	Number of 2-D convolutional filters in the first convolutional layer of the C-LSTM network.
$G^{(a)}$	Number of 2-D convolutional filters in the second convolutional layer of the C-LSTM network.
$H^{(a)}$	Number of hidden units in the LSTM layer of the C-LSTM network.
$\mathbf{x}^{(b)}[n]$	Multivariate input frame of the Conv-LSTM network.
f_b	Non-linear recurrent regression model implemented by the Conv-LSTM network.
$F^{(b)}$	Number of 2-D convolutional filters in the convolutional layer of the Conv-LSTM network.
$H^{(b)}$	Number of hidden units in the LSTM layer of the Conv-LSTM network.
$\mathbf{x}_q^{(c)}[n]$	Embedded vector associated with the qth time series of the Multi-LSTM network.
f_c	Non-linear recurrent regression model implemented by the Multi-LSTM network.
$H_{q\text{A}}^{(c)}$	Number of hidden units in the LSTM_qA layer of the Multi-LSTM network.
$H^{(c)}$	Number of hidden units in the LSTM layer of the Multi-LSTM network.
$\mathbf{x}^{(d)}[n]$	Multivariate input vector of the Stacked-LSTM network.
f_d	Non-linear recurrent regression model implemented by the Stacked-LSTM network.
$H_1^{(d)}$	Number of hidden units in the first LSTM layer of the Stacked-LSTM network.
$H_2^{(d)}$	Number of hidden units in the second LSTM layer of the Stacked-LSTM network.

I. INTRODUCTION

Predicting the future values of time series is a frequently studied problem in many fields of science and technology [1]. Over the past few years, electrical power systems have progressively evolved form centralized systems to a Distributed Generation (DG) scenario characterized by decentralized systems, in which smaller generation units are directly connected to distribution networks near the consumer. The diffusion of DG in the energy market, in particular that coming from Renewable Energy Sources (RESs), has been facilitated by various economic and environmental reasons and by government financial incentives, such as feed-in tariffs [2] and net-metering [3] schemes.

The large-scale penetration of DG, and more generally of distributed energy resources, into conventional electricity systems has posed numerous challenges [4], for operators involved in the operation and maintenance of modern grids, e.g., islanding detection [5], voltage and frequency regulation [6], harmonic distortion [7], electromagnetic interference [8], optimal demand side management of prosumers [9], as well as low-environmental impact routing of overhead power lines for the connection of renewable energy plants [10]. Yet, a large number of opportunities are offered by future proactive [11] and transactive [12] energy systems. One of the main problems is related to the integration of high intermittent and stochastic energy production of RESs, such as solar power and wind energy, into deterministic energy systems, thus demanding an improvement of their own flexibility [13].

This is the reason why the prediction of power output has become a necessary tool for all the actors involved [14]: producers need predictions to prepare their respective offer strategies; consumers need predictions to maximize their profit; Transmission (TSO) and Distribution (DSO) System Operators need predictions to optimize short and medium term decisions for energy regulation and dispatching. Furthermore, an accurate prediction system, which is the key element within automatic modeling tools for data analytics and intelligent operation control, may enable prosumer-oriented home energy management systems [15] and reduce both energy and operation costs.

Practical applications in this scenario largely rely on power output forecasting, which becomes particularly important for producers, network operators and market players [16], especially when it comes from RESs whose expected power production is intrinsically intermittent, as in the case of photovoltaic (PV) power sources [17]. This intermittent nature of RESs entails considerable uncertainty problems mainly related to the difference between predicted and real power, known as imbalance [18]. These problems need to be solved not only for stability and dispatchability reasons, but also for market problems concerning operators offering electricity to the day-ahead market [19]. The advantage is that energy offers from RESs are generally granted, since they are usually lower than the market-clearing price due to the low levelized cost of energy. The drawback is that the low predictability of these outputs increases potential costs to compensate for the imbalance between offers and real production.

In the past years, a huge amount of literature has been published on power forecasting models for PV systems, both with regards to the prediction of the primary source (i.e., solar radiation) [20] and the prediction of the produced power output [21], with attention focused recently on metaheuristic and machine learning methods [22]. Among them, Neural Networks (NNs) offer a suitable solution to the PV forecasting problem, since NN-based models show good results with energy time series for different time horizons and distributed scenarios [23]. In literature, several approaches have been proposed based on various declination of NNs, including Deep Neural Network (DNN) architectures [24], non-parametric machine learning approaches with multi-site framework [25], physical hybrid NN approaches [26], and hybrid models [27].

Most of the aforementioned approaches classically involving energy time series are usually carried out in a univariate way, considering a single time series to obtain the prediction of the time series itself. Instead, in some situations, different observations relating to two or more time series can be considered simultaneously, generating a more complex system with a broader generalization capability. Given these multivariate data, it becomes necessary to develop a suitable multivariate model in order to describe the relationships among time series and then use this complex model for prediction. In the energy field, this reasoning can be done by analyzing the time series collected from different physical variables related to different physical phenomena such as wind, solar radiation, humidity, air pressure and so on [28].

Recently, particular attention has been devoted to Long Short-Term Memory (LSTM) networks, a type of Recurrent Neural Networks (RNNs) able to deal with long-term dependencies. In fact, LSTM networks work tremendously well due to their powerful learning capacity, and have been used in a large number of tasks of different kinds. In [29], LSTM networks are used for predicting PV power output by integrating synthetic weather forecast into the statistical knowledge of historical solar irradiance; in [30], LSTM networks are tested under strong weather conditions introducing the clearness sky-index as an input data; in [31], an attenuation mechanism is adopted to facilitate the LSTM networks to adaptively focus on input features that are more significant in forecasting.

In this context, this article proposes four different multivariate models based on LSTM-stacked DNNs. The novel models are denoted as ‘C-LSTM network’, ‘Conv-LSTM network’, ‘Multi-LSTM network’, and ‘Stacked-LSTM network’. The first two models are developed by using a Convolutional Neural Network (CNN) [32] in order to obtain data representation at a higher level of abstraction. The significance of the present study is to assess how the multivariate analysis produces better results than the univariate analysis in most of the practical cases related to PV power output prediction, while showing that not all the methods based on multivariate analysis work equally well. In fact, differences are highlighted among the multivariate models, whose accuracy is proved in terms of several error metrics. This article will show that the Multi-LSTM network generally results as the most accurate model.

In the following of this article, Sect. II illustrates the basic LSTM networks along with the standard univariate approach. A brief discussion regarding general approaches to prediction in this context is reported in Sect. III. Then, Sect. IV introduces the four different multivariate models based on LSTM-stacked DNNs. All the focused prediction systems are evaluated and compared in Sect. V, by considering a real application case study regarding the active power produced by a PV plant located in Oak Ridge, Tennessee, USA. All performances will be compared with respect to the ones obtained by the classic univariate approach, which is considered as the reference benchmark in the present context as it has been demonstrated to achieve state-of-the-art performances in almost all applications. Finally, some concluding remarks are offered in Sect. VI.

II. LSTM NETWORKS IN THE CLASSIC UNIVARIATE APPROACH

This work focuses on the LSTM network as the reference prediction model also adopted for the successive extensions, because it is a special type of RNN able to learn long-term dependencies among data, avoiding optimization error due to gradient descent combined with back-propagation algorithm. It is known that RNNs can also connect past information to the present task but, when dealing with long-term dependencies, they tend to perform poorly [33]. In fact, when the

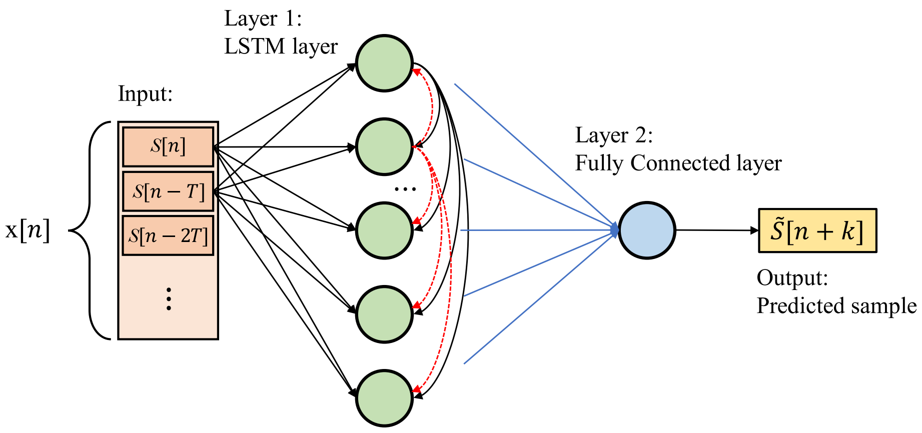


FIGURE 1. **Architecture of the basic LSTM network for univariate prediction fed by the vector x[*n*].**

data present long-term dependencies, the location of relevant information can be found very far from the current time step.

In this regard, LSTM-based networks can be considered a valid and solid solution to the long-term prediction problem, since they have the ability to control the amount of information that passes in a single cell through 'gates'. For this reason, LSTMs are well suited for DNN architectures, where several LSTM layers and other types of layers can be connected, and therefore are used as an integral part of the prediction scheme in this work.

Let us first introduce the basic univariate approach that is used for benchmarking and serves as the basis of the models presented hereafter. Atop of that, in a time series forecasting problem it is important to study which samples of the sequence are relevant to the prediction task. In this regard, a fruitful approach consists in selecting the samples via embedding of the sequence [34].

Let $S = \{S[n]\}$, $n > 0$, be the scalar time series to be predicted. Suppose that the current sample at n and all of the previous ones are known, and that $S[n+k]$, with $k > 0$, is the sample to be predicted. In a univariate approach, this prediction problem can be expressed as a regression problem by using past samples of the *sole* time series under analysis. Two different types of input data can be considered in this regard. The first one implies that $\widetilde{S}[n+k]$ is obtained by determining the parameters of a regression model through the so-called 'embedding technique', where the input vector to the regression model is based on past samples of the time series [35]:

$$\mathbf{x}[n] = \left[S[n]\ S[n-T]\ \ldots\ S[n-(D-1)T]\right]^t . \quad (1)$$

Then, $\widetilde{S}[n+k]$ can be written as

$$\widetilde{S}[n+k] = f_u\big(\mathbf{x}[n];\ \mathcal{I}_{n-1}\big) , \quad (2)$$

for which it can be observed that $f_u(\cdot)$ has its own parameters to be determined by a data driven learning procedure, while $\mathcal{I}_{n-1}$ is associated with the internal state of any dynamical RNN model, namely with the hidden states of LSTM layers.

The second univariate approach implies that the estimated value is obtained by using as a simple scalar input the current sample of the series:

$$\widetilde{S}[n+k] = f_u\big(S[n];\ \mathcal{I}_{n-1}\big) . \quad (3)$$

As said, the regression model adopted in the following is represented by a stacked DNN based on one or more LSTM layers together with other layers that can be adopted for improving learning (i.e., *dropout* layer, *normalization* layer, etc.) or for processing data at the input and/or output of the network itself. The simplest univariate network that can be implemented by using embedding is illustrated in Fig. 1, while the one implemented by using the scalar input is shown in Fig. 2. The layers in this architecture are:

- LSTM layer, which is fed by the input time series and makes use of H hidden units;
- Fully connected layer, which is a standard feed-forward layer connecting the H hidden states of LSTM layer to the scalar output that represents the predicted sample.

At every time-step, the LSTM layer takes as input either $\mathbf{x}[n]$ or $S[n]$ and, through the H internal units, it computes $h_n^{(m)}$ and $c_n^{(m)}$, $m = 1 \ldots H$, which are grouped in $\mathbf{h}_n \in \mathbb{R}^H$ and $\mathbf{c}_n \in \mathbb{R}^H$, respectively. Each LSTM unit computes both its hidden and cell states by a recursive scheme, taking as input either $\mathbf{x}[n]$ or $S[n]$ and the previous states $\mathbf{h}_{n-1}$ and $\mathbf{c}_{n-1}$ coming from itself and from other units; thus, the said past information $\mathcal{I}_{n-1}$ will coincide with vectors $\mathbf{h}_{n-1}$ and $\mathbf{c}_{n-1}$. A complete description of how the cells of the LSTM work in the framework of energy time series prediction can be found in [36].

The flow of information in each of the H units is controlled via four specific gates to model very long-term dependencies in the data: (*i*) the *input gate i* decides how much of the current input will be passed to compute the cell state; (*ii*) the *forget gate f* decides what information from the current input will be thrown away or kept; (*iii*) the *cell candidate g* controls the selective memory of the past; (*iv*) the *output gate o* decides

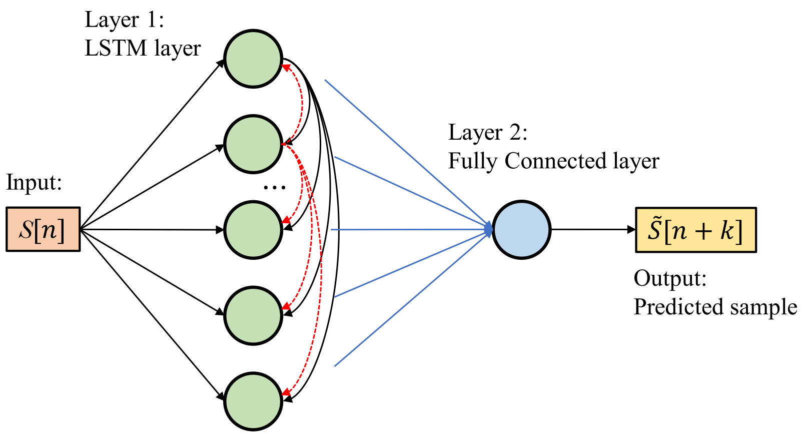


FIGURE 2. **Architecture of the basic LSTM network for univariate prediction fed by the scalar value *S*[*n*].**

what the next hidden state should be, regulating what of the internal state is carried over to the external network.

In case of an embedded input, the gate outputs $\{i_n^{(m)}, f_n^{(m)}, g_n^{(m)}, o_n^{(m)}\}$ can be arranged in vector form as:

$$\mathbf{i}_n = \sigma_g\left(\mathbf{W}_i\mathbf{x}[n] + \mathbf{R}_i\mathbf{h}_{n-1} + \mathbf{b}_i\right), \quad (4a)$$
$$\mathbf{f}_n = \sigma_g\left(\mathbf{W}_f\mathbf{x}[n] + \mathbf{R}_f\mathbf{h}_{n-1} + \mathbf{b}_f\right), \quad (4b)$$
$$\mathbf{g}_n = \sigma_c\left(\mathbf{W}_g\mathbf{x}[n] + \mathbf{R}_g\mathbf{h}_{n-1} + \mathbf{b}_g\right), \quad (4c)$$
$$\mathbf{o}_n = \sigma_g\left(\mathbf{W}_o\mathbf{x}[n] + \mathbf{R}_o\mathbf{h}_{n-1} + \mathbf{b}_o\right), \quad (4d)$$

while in case of scalar input, the gate outputs are obtained as:

$$\mathbf{i}_n = \sigma_g\left(\mathbf{w}_iS[n] + \mathbf{R}_i\mathbf{h}_{n-1} + \mathbf{b}_i\right), \quad (5a)$$
$$\mathbf{f}_n = \sigma_g\left(\mathbf{w}_fS[n] + \mathbf{R}_f\mathbf{h}_{n-1} + \mathbf{b}_f\right), \quad (5b)$$
$$\mathbf{g}_n = \sigma_c\left(\mathbf{w}_gS[n] + \mathbf{R}_g\mathbf{h}_{n-1} + \mathbf{b}_g\right), \quad (5c)$$
$$\mathbf{o}_n = \sigma_g\left(\mathbf{w}_oS[n] + \mathbf{R}_o\mathbf{h}_{n-1} + \mathbf{b}_o\right). \quad (5d)$$

In both cases, input weights, recurrent weights, and biases are obtained during the network training. The cell and hidden states of the LSTM layer are finally obtained as:

$$\mathbf{c}_n = \mathbf{f}_n \odot \mathbf{c}_{n-1} + \mathbf{i}_n \odot \mathbf{g}_n \quad (6a)$$
$$\mathbf{h}_n = \mathbf{o}_n \odot \sigma_c(\mathbf{c}_n), \quad (6b)$$

where $\odot$ denotes the Hadamard product.

The last layer in any LSTM-based architecture is used to compute the final output, which is $\widetilde{S}[n+k]$; it is obtained by the fully connected layer making use of the hidden states of the LSTM layer:

$$\widetilde{S}[n+k] = \mathbf{w}_d^t\mathbf{h}_n + b_d, \quad (7)$$

where $\mathbf{w}_d$ and b_d are also determined by the training algorithm.

III. RELATED WORKS

Energy time series forecasting, in particular solar power, has attracted a considerable interest in recent years due to the new legislation encouraging the use of solar energy and the large-scale deployment of photovoltaic plants. Prior to deep learning and LSTM networks, the literature has been filled by statistical and data driven modeling tools where the univariate approach represented by (2) or (3) has been pursued by solving prediction through data regression. In machine learning, both recurrent and feed-forward models have been investigated by using shallow neural networks, possibly hybridized with fuzzy modeling and evolutionary computing for complex optimization [14], [37].

There are two main groups of forecasting approaches in the considered field: indirect and direct [38]. Indirect forecasts firstly predict solar irradiation and then, using a PV performance model of the plant, obtain the power produced. Conversely, direct forecasts directly calculate the power output of the plant. Although solar radiation is closely related to PV output power, producers are more interested in knowing the amount of available PV output power rather than the solar radiation, in order to propose profitable offers on the consumer market. For this reason, our proposed methods fall into the second group.

For many years, statistical approaches to data regression have been considered for the PV output forecasting. Among them, there are Auto-Regressive (AR) and Auto-Regressive eXogenous (ARX) models [39], Moving Average (MA) models [40], Auto-Regressive Moving Average (ARMA) models [41], Auto-Regressive Moving Average with eXogenous variables (ARMAX) models [40], Auto-Regressive Integrated Moving Average (ARIMA) models [42], and Seasonal Auto-Regressive Integrated Moving Average (SARIMA) models [43]. Under normal conditions, these models deliver good prediction results. However, sudden changes in weather conditions can significantly affect energy usage models and, consequently, make these techniques unable to provide accurate predictions.

In this regard, Artificial Neural Networks (ANNs) are one of the most used approaches due to their ability in solving complex and non-linear forecasting models [27]. In literature there are many studies concerning ANNs used for the prediction of PV output. Recent approaches are based on the promising tool of Random Forests in the framework of ensemble learning, as in [44] regarding irrelevant inputs selection methods, [45] for feature extraction of radiation data, and [46] for mitigating climatic effects. However, most of these studies use a univariate approach to solve the problem, as they only consider the PV output time series.

Recently, researchers have started to consider a multivariate approach to solve the PV prediction problem with good results. Regarding the indirect forecasts, the authors in [47] compare several multivariate ANN models for a week ahead of solar power generation forecasting in Gorakhpur, India. Nine parameters (time, average temperature, minimum temperature, maximum temperature, rain, wind, dew, atmospheric, and azimuth) are selected as input variables to ANNs. The mathematical analysis, carried out in terms of Root Mean Square Error (RMSE), shows that the Model Average NN presents the best results whereas the Back Propagation NN presents the worst results.

Another research focused on forecasting the monthly global solar radiation in India is presented in [48]. The authors propose two ANN models which use three input variables collected for 10 years: temperature, wind speed, and humidity. They identify the best model based on minimum Mean Absolute Error (MAE), RMSE, and maximum linear correlation coefficient. They also observe that the accuracy of the ANN forecasting model depends on the quantity and quality of the training data.

Authors in [49] still focus on solar intensity forecasting by proposing an Adaptive Learning Hybrid Model (ALHM) integrated with a Time-varying Multiple Linear Model (TMLM), which catch the linear and dynamic property of data, and a Genetic Algorithm Back Propagation (GABP) NN which is used to learn non-linearity in the data. Simulation results show that ALHM outperforms several benchmarks in both short-term and long-term solar intensity forecasting by using several meteorological data (temperature, humidity, dew point, wind speed, precipitation) as additional variables.

Still remaining on indirect forecasting, in [50] authors propose a Back-Propagation ANN (BP-ANN) methodology to predict hour-ahead photovoltaic irradiation by using average temperature and average relative humidity as additional variables. The results prove that the performance of the BP-ANN model is better when compared to the ARMA model in terms of RMSE, MAE, Mean Absolute Percentage Error (MAPE), Mean Bias Error (MBE), and correlation coefficient.

Regarding the direct forecasting, in [51] a two-layer feed-forward network trained with Levenberg-Marquardt algorithm is used to estimate the profile of the output power of a grid-connected 20kWp solar power plant in a reputed manufacturing industry located in Tiruchirappalli, India. An experimental database comprising the PV output power and atmospheric temperature time series was used for training the ANN. The performance are evaluated in terms of Mean Squared Error (MSE) and are within the range of 0.019 to 0.025, for the day-ahead forecasting.

Another approach, which uses the back-propagation ANN to forecast the PV output power of the day ahead, is presented in [52]. Four time series are taken into account: temperature, humidity, wind speed, and Aerosol Index (AI). Results prove that the proposed model improves the accuracy of PV output power forecasting compared with the ANN method which does not use the AI as an additional variable with respect to the others.

Considering different types of ANN, authors in [53] created three different NN ensemble networks, namely feed-forward NN (FNN), Elman back-propagation network (ELM) and Cascade-Forward back-propagation (NewCF) trained with input data from meteorological variables (solar irradiance, PV module temperature, humidity, and wind speed). Results highlight an improvement in the forecasting accuracy of the PV output power for one day ahead compared to benchmark methods.

The research proposed in [54] introduces an ANN model for solar power generation forecasting using two additional weather parameters: air temperature and wind speed. Authors conducted an application study based on the Buruthakanda solar park. Results show that the PV output power forecasts obtained with their ANN model can be used to forecast solar power generation with high accuracy under clear conditions.

A further research on direct forecasting is reported in [55], where three PV power output prediction methods are used: ANN, DNN, and LSTM based models. Three additional variables are taken into account in order to help the prediction: temperature, humidity and cloudiness. Experimental results prove that the LSTM based model performs better than the ANN based forecasting system in terms of MAE.

The study in [56] focuses on the use of ANN in short-term prediction (1 hour to 6 hours) of global solar irradiance based on observations made in parallel by neighboring sensors measuring different variables (temperature, humidity, pressure, wind and other estimates). The results of the power generation forecasting prove that the time frame between 4 hours and 6 hours provides the best result for the ANN model.

In [57], the authors present a weather-based hybrid method for 1-day ahead hourly forecasting of PV power output. They use three variables: temperature, probability of precipitation and solar irradiance. The numerical results demonstrate that the proposed method significantly improves the prediction power generation compared to the Support Vector Regression (SVR) method.

An ANN method based on the MultiLayer Perceptron (MLP) with back-propagation to predict the solar radiation is proposed in [58]. Air temperature, minimum temperature, maximum temperature, relative humidity, wind speed and solar radiation are used as network inputs, reaching an accuracy of 0.1169 in terms of MSE.

In [59], a feed-forward ANN and an Adaptive Neuro-Fuzzy Inference System (ANFIS) are considered by using as inputs nine environmental parameters: global horizontal irradiance, global diffused irradiance, ambient temperature, precipitation, wind speed, air pressure, sunshine duration, relative humidity and surface temperature. Results prove that the ANN model is better than the fuzzy neural network in predicting the PV power output.

IV. PROPOSED MULTIVARIATE MODELS

Let $S_1 = \{S_1[n]\}$, $n > 0$, be the scalar time series to be predicted and let $S_q = \{S_q[n]\}$, $q = 2 \dots M$, be further $M - 1$ scalar time series that bring a correlated information regarding the underlying prediction problem and thus they can be used for the prediction of $S_1[n]$. Suppose that all samples previous to n are known, including the latter, and that $S_1[n+k]$, $k > 0$, is the sample to be predicted. The estimated output is obtained using these past samples in a delayed-input prediction model $f_m(\cdot)$ as:

$$\widetilde{S}_1[n+k] = f_m\left(\mathbf{x}^{(m)}[n];\, \mathcal{I}_{n-1}\right), \tag{8}$$

where $\mathbf{x}^{(m)}[n]$ takes samples from all of the considered time series, $\mathcal{I}_{n-1}$ is the information associated with the internal states of the involved recurrent LSTM layers.

The remainder of this section analyzes the four deep neural models proposed in this article and the suited embedding technique applied to each of them. The C-LSTM network is presented in Sect. IV-A, while the Conv-LSTM is illustrated in Sect. IV-B. Then, the Multi-LSTM is introduced in Sect. IV-C and, finally, the Stacked-LSTM in Sect. IV-D.

A. C-LSTM NETWORK

In order to optimize the LSTM model structure, in C-LSTM network a 2-D CNN layer is stacked before the actual LSTM layer. This way, it is possible to obtain data representation (i.e., feature extraction) at a higher level of abstraction, as introduced in [60]. Consequently, the input of the system must be adapted to the network architecture. For this matter, the embedded data is handled as a sequence of video frames, representing the input of the prediction system which is then fed into the 2-D CNN layer.

In the case $M = 2$, two scalar time series are considered, $S_1[n]$ and $S_2[n]$. To predict $S_1[n+k]$, the input frame $\mathbf{x}^{(a)}[n]$ is built in four consecutive steps:

1) set the number of past samples D that are used (for each time series) to predict the new one, i.e., $S_1[n+k]$;
2) replicate the input sequences D times and obtain the two frames $\mathbf{x}_1[n]$ and $\mathbf{x}_2[n]$ of size $D \times D$;
3) transpose the frame $\mathbf{x}_2[n]$ for obtaining $\mathbf{x}_2^t[n]$;
4) make the dot product between $\mathbf{x}_1[n]$ and $\mathbf{x}_2^t[n]$ for obtaining $\mathbf{x}^{(a)}[n]$.

This way, the deep architecture is able to recognize and efficiently exploit the temporal correlation between the considered embedded sequences. Furthermore, through the use of numerous convolutional filters, different temporal correlation structures are inferred by using the back-propagation learning algorithm in the training phase. This process of inferring data structures in $\mathbf{x}^{(a)}[n]$ consists in mixing different time series and autonomously selecting which and how many past samples are to be used in the learning phase. The procedure employed to create $\mathbf{x}^{(a)}[n]$ is illustrated in Fig. 3.

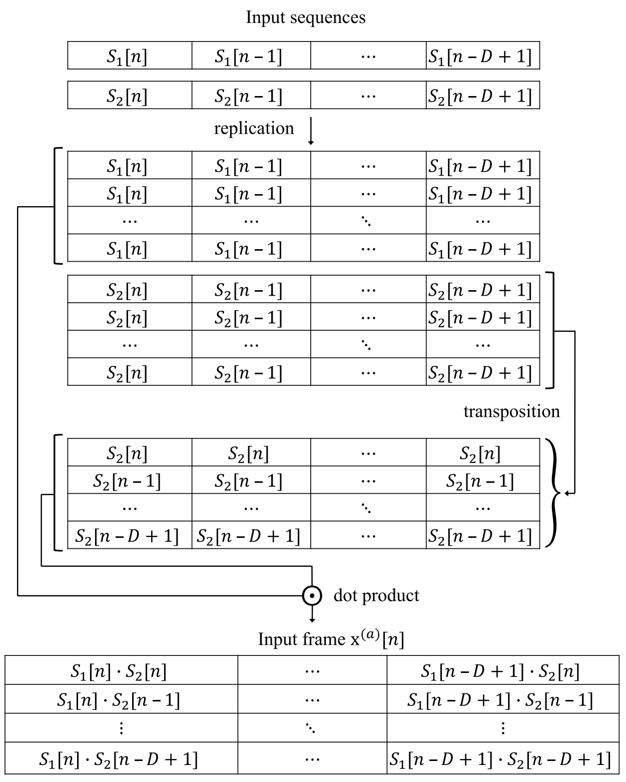

Input sequences
$S_1[n]$ | $S_1[n-1]$ | ⋯ | $S_1[n-D+1]$
$S_2[n]$ | $S_2[n-1]$ | ⋯ | $S_2[n-D+1]$
replication
transposition
dot product
Input frame $\mathbf{x}^{(a)}[n]$
$S_1[n]\cdot S_2[n]$ | ⋯ | $S_1[n-D+1]\cdot S_2[n]$
$S_1[n]\cdot S_2[n-1]$ | ⋯ | $S_1[n-D+1]\cdot S_2[n-1]$
$S_1[n]\cdot S_2[n-D+1]$ | ⋯ | $S_1[n-D+1]\cdot S_2[n-D+1]$

FIGURE 3. Procedure to create $\mathbf{x}^{(a)}[n]$. The two input sequences of length D are replicated D times, obtaining the two frames $\mathbf{x}_1[n]$ and $\mathbf{x}_2[n]$ of size $D \times D$. The frame $\mathbf{x}_2[n]$ is transposed, obtaining $\mathbf{x}_2^t[n]$, which is multiplied element by element with frame $\mathbf{x}_1[n]$ therefore obtaining $\mathbf{x}^{(a)}[n]$.

The equation reported in (8) can thus be written as:

$$\widetilde{S}_1[n+k] = f_a\left(\mathbf{x}^{(a)}[n];\, \mathcal{I}_{n-1}\right), \tag{9}$$

where the regression model is implemented by the recurrent DNN shown in Fig. 5(a), whose general functioning is summarized in Fig. 4. The layers in the architecture are:

- Convolutional layer 1: it applies $F^{(a)}$ sliding 2-D convolutional filters to every input frame.
- Max pooling layer 1: it performs down-sampling by dividing the input into square pooling regions and computing the maximum of each region.
- Convolutional layer 2: it uses $G^{(a)}$ sliding 2-D convolutional filters.
- Max pooling layer 2: it uses pooling regions.
- Flatten layer: it collapses the spatial dimensions of the input to the LSTM layer, which is a vector time series.
- LSTM layer: it represents the actual LSTM layer with $H^{(a)}$ hidden units.
- Fully connected layer: it is a standard feed-forward layer connecting the $H^{(a)}$ hidden states to the scalar output that represents the sample to predict.

B. CONV-LSTM NETWORK

The same concepts expressed in Sect. IV-A for the C-LSTM network are used herein for the Conv-LSTM network, i.e., a 2-D CNN layer is stacked before the actual LSTM layer. Also

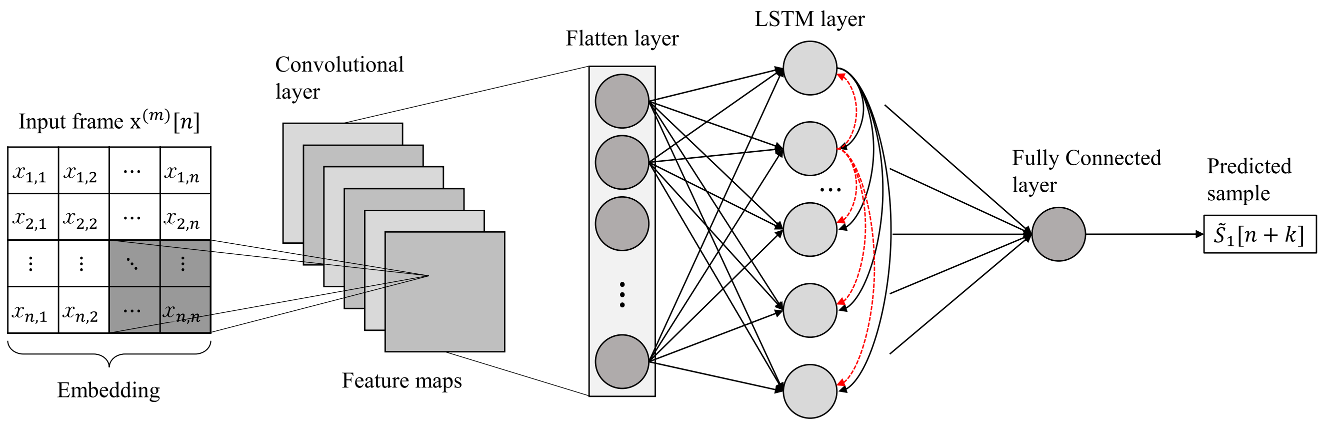


FIGURE 4. General structure and functioning of the C-LSTM and Conv-LSTM networks.

in this case, the input of the system must be adapted to the network architecture, encapsulating the embedded data in a frame-like structure.

To predict $S_1[n+k]$, the input frame $\mathbf{x}^{(b)}[n]$ is built as follows:

$$\mathbf{x}^{(b)}[n] = \begin{bmatrix} S_1[n] & S_1[n-1] & \dots & S_1[n-D+1] \\ S_2[n] & S_2[n-1] & \dots & S_2[n-D+1] \\ \vdots & \vdots & \ddots & \vdots \\ S_M[n] & S_M[n-1] & \dots & S_M[n-D+1] \end{bmatrix}. \tag{10}$$

This way, the input to the CNN layer is represented by a 1-channel $M \times D$ frame and hence, the equation reported in (8) can be rewritten as:

$$\widetilde{S}_1[n+k] = f_b\left(\mathbf{x}^{(b)}[n]; \mathcal{I}_{n-1}\right), \tag{11}$$

where the regression model is implemented by the recurrent DNN reported in Fig. 5(b). Its general operation is the same as for C-LSTM network and thus, it can be still summarized as in Fig. 4. The layers in the architecture are:

- Convolutional layer: it applies $F^{(b)}$ sliding 2-D convolutional filters, each of dimension $M \times M$, to every frame $\mathbf{x}^{(b)}[n]$.
- Batch normalization layer: it normalizes the convolution results to reduce the sensitivity to network initialization.
- Flatten layer: it collapses the spatial dimensions of the input to the LSTM layer, which is a vector time series.
- LSTM layer: it is the actual LSTM layer with $H^{(b)}$ hidden units.
- Fully connected layer: it is a standard feed-forward layer connecting the $H^{(b)}$ hidden states to the scalar output that represents the sample to predict.

It is important to underline that 2-D convolutional filters are used, whose size varies according to the number of the considered time series. This means that the convolutional filters move in one direction only, as in the case of the well-known 1-D CNN [61], [62]. However, a 2-D CNN layer is used to feed the DNN with a more generalized embedded data, handled similarly to a sequence of video frames.

C. MULTI-LSTM NETWORK

The use of multiple LSTMs can significantly improve the generalization capability of DNNs. In the proposed Multi-LSTM network, the key idea lies in the use of a specific LSTM for each sequence, not only for prediction purposes but also to extract deep information on each input time series. This information is then passed through the sequent layers of the network, where it is aggregated and used to predict the univariate output sequence. The proposed architecture is illustrated in Fig. 5(c) in the case of two input sequences, it can be generalized to any number of input time series.

In order to predict $S_1[n+k]$, the embedding technique is applied on all the time series involved in the multivariate prediction problem, therefore obtaining:

$$\mathbf{x}_1^{(c)}[n] = \left[S_1[n]\ S_1[n-1] \dots S_1[n-D+1]\right], \tag{12a}$$
$$\mathbf{x}_2^{(c)}[n] = \left[S_2[n]\ S_2[n-1] \dots S_2[n-D+1]\right], \tag{12b}$$
$$\vdots$$
$$\mathbf{x}_M^{(c)}[n] = \left[S_M[n]\ S_M[n-1] \dots S_M[n-D+1]\right]. \tag{12c}$$

Each sequence is then used as input to the respective channel in order to extract deep information. For example, considering Fig. 5(c), $\mathbf{x}_1^{(c)}[n]$ can be sent to the lower channel and $\mathbf{x}_2^{(c)}[n]$ to the upper channel (or vice versa).

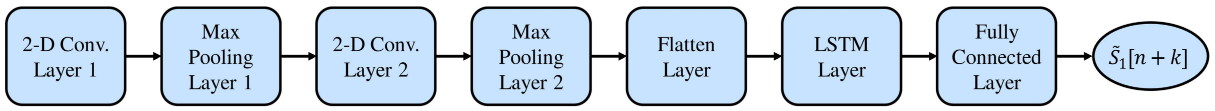
The equation reported in (8) can be written as:

$$\widetilde{S}_1[n+k] = f_c\left(\mathbf{x}_1^{(c)}[n], \mathbf{x}_2^{(c)}[n], \dots, \mathbf{x}_M^{(c)}[n]; \mathcal{I}_{n-1}\right), \tag{13}$$

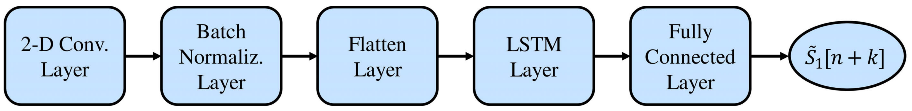
where the regression model is implemented by the recurrent DNN in Fig. 5(c).

The network architecture consists of the following layers:

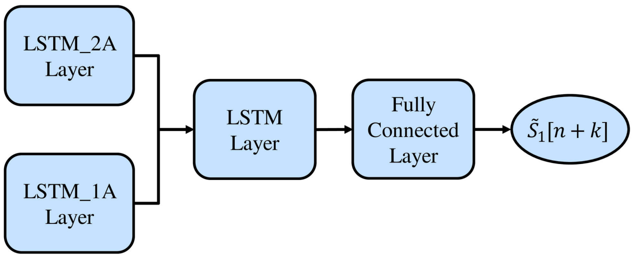
- LSTM_qA layer, $q = 1 \dots M$: it is the LSTM layer with $H_{q\mathrm{A}}^{(c)}$ hidden units that extracts information from the single sequence selected from the multivariate input.
- LSTM layer: it is the LSTM layer with $H^{(c)}$ hidden units that exploits the aggregate information received



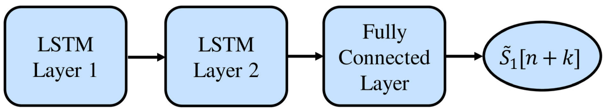
(a)



(b)



(c)



(d)

FIGURE 5. Architectures of the proposed DNNs for multivariate prediction: (a) C-LSTM network; (b) Conv-LSTM network; (c) Multi-LSTM network, where the number of parallel channels varies upon the adopted time series; (d) Stacked-LSTM network.

from the previous layers. Namely, the layer's input is a column vector that serializes (i.e., aggregates) all the hidden states computed by the previous LSTM layers.

- Fully connected layer: it is a standard feed-forward layer connecting the $H^{(c)}$ hidden states to the scalar output that represents the sample to predict.

D. STACKED-LSTM NETWORK

The basic idea in the proposed Stacked-LSTM network is to fully exploit the potential of the LSTM model by concurrently using multiple time series. The input to the prediction system is represented by several time series. The latter are fed to the LSTM layer in the form of a column vector, where each element corresponds to a different time series:

$$\mathbf{x}^{(d)}[n] = \left[S_1[n] \;\; S_2[n] \;\; \ldots \;\; S_M[n] \right]^t . \tag{14}$$

This way, the DNN shown in Fig. 5(d) implements a regression model where $\widetilde{S}_1[n+k]$ depends upon $\mathbf{x}^{(d)}[n]$, which embeds the multivariate nature of different time series:

$$\widetilde{S}_1[n+k] = f_d\left(\mathbf{x}^{(d)}[n];\ \mathcal{I}_{n-1}\right) . \tag{15}$$

The input of the system is represented by the vector $\mathbf{x}^{(d)}[n]$ reported in (14). The layers in the architecture are:

- LSTM layer 1: it is the first LSTM layer with $H_1^{(d)}$ hidden units, it is fed by the input $\mathbf{x}^{(d)}[n]$.
- LSTM layer 2: it is the second LSTM layer with $H_2^{(d)}$ hidden units.
- Fully connected layer: it is a standard feed-forward layer connecting the $H_2^{(d)}$ hidden states to the scalar output that represents the sample to predict.

V. EXPERIMENTAL RESULTS

In order to assess the performance of the proposed forecasting models, a specific application case is considered pertaining to the PV power production plant of the 'Oak Ridge National Laboratory' located in Oak Ridge, TN, USA, whose geographic coordinates are 35°92′99.6″ N, 84°30′95.2″ W, elevation 245 m.

A. DATA SETUP

Irradiance data is measured through the use of a LICOR LI-200 pyranometer sensor mounted on the roof of a two story building at 12.04 m above ground level. The output power to be predicted (i.e., S_1 in kW) is computed by using a balance of system of 0.9 and by applying an inverter curve with Maximum Power Point Tracking (MPPT).

In addition to the main time series, two further variables are considered: the air temperature (i.e., S_2 in °C), measured with a thermometer inside a naturally aspirated radiation shield mounted at 11.45 m above ground level on roof; the wind speed (i.e., S_3 in m/s), measured by an RM Young 3-cup anemometer and vane mounted at 12.80 m above ground level on roof. All time series are obtained from the Measurement and Instrumentation Data Center (MIDC) database; they are collected in the same plant and sampled at hourly rate. It is important to remark that these physical parameters are chosen as they can be measured easily in residential PV prosumers plants. However, the present study can be generalized to any number of correlated time series that can be used for this task, independently of their physical nature.

The considered time series refer to year 2018 and they are all aligned in time in order to ensure to properly infer time-dependent structures on data. Before applying the learning procedures, a linear mapping is used for normalization between 0 and 1, using the minimum and maximum value of each sequence as extremes for normalization, i.e.:

- S_1: 0 corresponds to 0 kW and +1 to 75 kW;
- S_2: 0 corresponds to −16 °C and +1 to 37 °C;
- S_3: 0 corresponds to 0.2 m/s and +1 to 4.1 m/s.

A training set of 2 months (i.e., 60 consecutive days) is used for the experiments. The training set contains past samples used for predicting the future ones. The latter are associated with two test sets whose lengths are 1 day (i.e., 24 samples) and 3 days (i.e., 72 samples) after the last available sample of the training set, respectively. Based on a preliminary analysis and on the results assessed in past published researches, in all of the experiments the number of past samples and the time lag are set to $D = 24$ and $T = 1$, respectively, when the embedding technique is used.

The two considered test sets start in May 2018 and October 2018, they are composed by the 24 samples of the first day of the month (i.e., 1-day test set) and by the 72 samples of the first three days of the month (i.e., 3-days test set). As the samples must be predicted all together at the same time, for operation purposes in the electrical grid, the prediction distance in all models is set to $k = 24$ or $k = 72$, respectively. By the way, although two training sets refer to a same period, by changing the prediction distance they will have different target values that affect the computation of the loss function.

In order to evaluate the performance of the proposed multivariate forecasting approaches, three different cases are considered: an approach considering the pairs $\{S_1, S_2\}$ or $\{S_1, S_3\}$ and an approach considering all the time series $\{S_1, S_2, S_3\}$. A separate discussion must be made for the C-LSTM network due to the method used to create the input frame described in Sect. IV-A. It uses only two time series at a time, limiting the potential of the network in case more than two sequences are used in the prediction. To solve the problem, three input frames are created, considering the pairs $\{S_1, S_2\}$, $\{S_1, S_3\}$ and $\{S_2, S_3\}$. These frames are then sent to the first CNN layer concurrently: one for each channel. In this way, the C-LSTM network can efficiently exploit the temporal correlation among all the considered time series.

B. MODEL SETTINGS

All DNNs have been trained using the ADAM algorithm [63] with gradient decay factor 0.9 and mini batch size equal to 1. Additional training options and model hyperparameters have been set by using a grid search procedure for cross-validation on the training data in order to avoid overfitting [64].

This model optimization has been performed by using the ensemble of 1-day and 3-days datasets, of both May 2018 and October 2018, for any combination of input time series (i.e., univariate with or without embedding, multivariate with two or more time series). This way, a unique setup of training options and model hyperparameters is determined and thus, a unique DNN model is adopted for testing all cases and for evaluating its robustness to generalization in the different situations of a real operation scenario. The general optimization procedure is summarized in Algorithm 1 and the results are listed in the following:

- Basic LSTM network: $H = 30$ hidden units in the sole LSTM layer; number of epochs 200; L2 regularization factor 10^{-4}; initial learning rate $6 \cdot 10^{-3}$ without dropping.
- C-LSTM network: $F^{(a)} = 12$ filters of size 6×6 in the Convolutional layer 1; $G^{(a)} = 9$ filters of size 3×3 in the convolutional layer 2; pooling regions of size 4×4 in the max pooling layer 1; pooling regions of size 2×2 in the max pooling layer 2; $H^{(a)} = 60$ hidden units in the LSTM layer; number of epochs 200; L2 regularization factor 10^{-4}; initial learning rate $6 \cdot 10^{-3}$ without dropping.
- Conv-LSTM network: $F^{(b)} = 5$ filters of size $M \times M$ in the convolutional layer and $H^{(b)} = 50$ hidden units in the LSTM layer; number of epochs 100; L2 regularization factor $5 \cdot 10^{-4}$; initial learning rate $5 \cdot 10^{-3}$; learning rate drop factor 0.9; learning rate drop period 20 epochs.
- Multi-LSTM network: $H_{qA}^{(c)} = 30$ hidden units; $q = 1 \ldots M$; for each input LSTM layer and $H^{(c)} = 80$ hidden units in the final LSTM layer; number of epochs 200; L2 regularization factor $5 \cdot 10^{-4}$; initial learning rate $6 \cdot 10^{-3}$; learning rate drop factor 0.9; learning rate drop period 20 epochs.
- Stacked-LSTM network: $H_1^{(d)} = 25$ hidden units in the first LSTM layer and $H_2^{(d)} = 60$ hidden units in the second LSTM layer; number of epochs 200; L2 regularization factor 10^{-4}; initial learning rate $6 \cdot 10^{-3}$ without dropping.

C. NUMERICAL RESULTS: PERFORMANCE

All the experiments were performed using Matlab® R2020a on a machine provided with an Intel® Core™ i7-3820 64-bit CPU at 3.60 GHz and with 32 GB of RAM, using for training and inference an NVIDIA® GeForce™ GTX 680 GPU at 1.006 GHz and 2048 MB GDDR5 RAM. As the training outcome and the consequent performance on test set depend upon random initialization of network parameters, 10 different runs are performed for each test and in the following the average result of MAE over the different runs is reported. Standard deviation is not reported for the sake of conciseness, it is quite stable in a confidence interval of ±5% for all tests.

Algorithm 1 Pseudocode of the Cross-validation Procedure Adopted to Optimize the Training Options and the DNN Hyperparameters

```
input: observed time series $\{S_1, S_2, S_3\}$ of the whole 2018; a
    generic DNN model $N$ to be trained and optimized, which
    is associated with one of the proposed architecture (i.e., C-
    LSTM, Conv-LSTM, Multi-LSTM, or Stacked-LSTM).[a]
 1: data preparation: time series normalization, data filling,
    etc.
 2: experiment setup: define a grid search space $\mathcal{G}$, where
    each point $p \in \mathcal{G}$ is a specific set of hyperparameters
    related to the training algorithm (i.e., epochs, learning rate,
    regularization factor, etc.) and to the network architecture
    (LSTM states, CNN filter dimensions, etc.). Experiments
    are performed by varying input data and test conditions.[b]
 3: loop {for each $s$, $m$, $d$:}
 4:     training/test setup: prepare training and test set for
        each sequence, let them be denoted as $\mathcal{R}^{(s,m,d)}$ and
        $\mathcal{T}^{(s,m,d)}$, respectively;
 5:     training/validation setup: extract from $\mathcal{R}^{(s,m,d)}$ a val-
        idation set $\mathcal{V}^{(s,m,d)}$ to assess model performance during
        training, the remaining data $\mathcal{R}_{\text{red}}^{(s,m,d)}$ will be actually
        used to train the DNN.[c]
 6: end loop
 7: loop {for each point $p \in \mathcal{G}$:}
 8:     loop {for each $s$, $m$, $d$:}
 9:         network training: train the adopted DNN model by
            using $\mathcal{R}_{\text{red}}^{(s,m,d)}$ and the hyperparameters specified in
            $p$, let the trained network be denoted as $N_p^{(s,m,d)}$;
10:         network validation: evaluate the performance of
            $N_p^{(s,m,d)}$ through $\mathcal{V}^{(s,m,d)}$, let $e_p^{(s,m,d)}$ be the MAE
            performance of the DNN.
11:     end loop
12:     model performance: evaluate the model performance
        as the average $\bar{e}_p$ over all trained DNNs on different
        $\{s, m, d\}$ datasets.
13: end loop
14: network optimization: let $\hat{p} = \arg\min_{p\in\mathcal{G}}\{\bar{e}_p\}$ be the
    optimal set of hyperparameters to be used for final train-
    ing.
15: loop {for each $s$, $m$, $d$:}
16:     final training: let $N_{\text{opt}}^{(s,m,d)}$ be the optimal DNN model
        trained (see Table 2) by using the setup $\hat{p}$ and the whole
        training set $\mathcal{R}^{(s,m,d)}$;
17:     final inference: compute the final DNN performance
        $\tilde{e}_{\hat{p}}^{(s,m,d)}$ for the specific test set $\mathcal{T}^{(s,m,d)}$ by using $N_{\text{opt}}^{(s,m,d)}$.
18: end loop
output: model performance $\tilde{e}_{\hat{p}}$ for all tests (see Table 1).
```

[a]For the sake of conciseness, we do refer to the multivariate case only. In the univariate approach, the same steps hold considering in that case S_1 only either with or without embedding.

[b]Let the superscript 's' denote the specific multivariate approach (i.e., two or more time series adopted as input data), superscript 'm' denote the month (May 2018 or October 2018), and superscript 'd' denote the 1-day or 3-days test option.

[c]The validation set is obtained by extracting from the training set the samples of the latest day or the samples of latest three days as target values and, according to the specific DNN model, the preceding samples as inputs. The reduced training set will end one day before in both cases.

Table 1 reports the numerical results for both univariate and multivariate approaches. Considering the C-LSTM network, when the input data are $\{S_1, S_2, S_3\}$ it means that three frames (one for each channel) are considered. Bold numbers evidence the best results for every experiment. Considering the different combination of the time series in the multivariate approach, it can be observed that the accuracy varies depending on which DNN model is used. In C-LSTM, the pair $\{S_1, S_2\}$ gives better results than $\{S_1, S_3\}$ for all test conditions. In particular, the relative improvements range from 38% up to 60% in three cases out of four. The only exception is for the 3-days test set in October, where the performance is quite similar. This difference is not evident when comparing the results obtained using $\{S_1, S_2\}$ with respect to $\{S_1, S_2, S_3\}$.

In Conv-LSTM, the use of $\{S_1, S_2\}$ gives the best results with respect to $\{S_1, S_3\}$ and $\{S_1, S_2, S_3\}$ considering the May test sets. The relative improvements range from 65% up to 72% for the 1-day test set and from 39% up to 54% for the 3-days test set, respectively. This is not true for the October test sets, where the best performance is obtained by using $\{S_1, S_2, S_3\}$ in the 1-day test set and the pair $\{S_1, S_3\}$ in the 3-days test set. It is important to underline that in the 1-day test set in May the use of $\{S_1, S_2\}$ gives better performance than the basic LSTM which uses the embedding technique, with a relative improvement of 4.4%.

In Multi-LSTM the pair $\{S_1, S_2\}$ gives the best results with respect to $\{S_1, S_3\}$ and $\{S_1, S_2, S_3\}$ in three cases out of four, with a relative improvement from 1.0% up to 33%. The only exception is for the 3-days test set in October. The Multi-LSTM is the network that has obtained the best results compared to the other DNN models. The performance is the best in all tests for any combination of the input data except for the 1-day test set in October when $\{S_1, S_2, S_3\}$ are used. In this case, the best result is obtained through the Conv-LSTM. It is important to remark that in three cases out of four the Multi-LSTM performs better than the basic LSTM which uses the embedding technique, with a relative improvement

TABLE 1. Average MAE (kW) for 1-day and 3-days test sets considering both univariate and multivariate approaches.

Model	Input Data	May 1-day	May 3-days	Oct. 1-day	Oct. 3-days
Basic LSTM (with embedding)	S_1	1.445	1.744	**1.844**	2.502
Basic LSTM (without embedding)	S_1	5.335	2.961	6.167	4.085
C-LSTM	S_1, S_2	2.222	1.962	2.563	3.506
	S_1, S_3	4.729	4.912	4.170	3.576
	S_1, S_2, S_3	2.250	1.988	2.438	3.520
Conv-LSTM	S_1, S_2	1.381	1.943	2.792	3.117
	S_1, S_3	3.943	3.171	2.340	2.614
	S_1, S_2, S_3	4.994	4.252	1.873	3.379
Multi-LSTM	S_1, S_2	**1.089**	**1.524**	2.148	2.439
	S_1, S_3	1.625	1.806	2.225	2.372
	S_1, S_2, S_3	1.334	1.539	2.273	**2.360**
Stacked-LSTM	S_1, S_2	4.048	2.841	5.737	3.937
	S_1, S_3	5.317	3.091	5.512	3.899
	S_1, S_2, S_3	5.058	2.962	5.973	4.214

from 5.7% up to 25%. The only case where the basic LSTM performs better than the Multi-LSTM is for the 1-day test set in October, with a relative improvement of about 19% (bold numbers).

In Stacked-LSTM, the use of $\{S_1, S_2\}$ gives the best results with respect to $\{S_1, S_3\}$ and $\{S_1, S_2, S_3\}$ considering the May test sets, as for the Conv-LSTM. The relative improvements range from 20% up to 24% for the 1-day test set and from 4.1% up to 8.1% for the 3-days test set, respectively. This is not true for the October test sets, where the pair $\{S_1, S_3\}$ provides the best performance. For the above reasons, it is clear that the use of output power S_1 and air temperature S_2 gives the best performance in most cases compared to the wind speed S_3. The latter does not seem to positively affect the prediction accuracy, even if in the case of 3-days test in October the Multi-LSTM network achieves the best performance by including it. Another important remark is that in each test, for each DNN model, there is at least one combination of the input data that guarantees a better performance than the basic LSTM model which does not use the embedding technique.

Considering the univariate approach, it is clear that the use of the embedding technique significantly improves the performance of the basic LSTM. In fact, the only case when the univariate prediction is better than the multivariate one is in October for a 1-day test when the embedding technique is used. A further remark concerns the difference between the time horizons in terms of accuracy. One might expect the accuracy to be lower for $k = 72$ than for $k = 24$, but this is not always the case. In the univariate approach, the basic LSTM using the embedding technique does exactly this, while the other one does not, which means that the latter performs better over a longer time horizon. The same happens in the multivariate approach. In C-LSTM there is at least one combination of the input data that gives better performance with a longer time horizon across all tests. The same goes for the Conv-LSTM but only for the May test set. The Multi-LSTM does not behave like this: 1-day test sets always give better results than 3-days test sets for any combination of input data. The Stacked-LSTM does exactly the opposite. It works well with a longer time horizon, as the basic LSTM which does not use the embedding technique.

The graphical results of the best combination obtained in each of the considered test conditions are reported in Fig. 6 to Fig. 9, respectively. The visual analysis does spark the discussion on how well the forecasting methods are able to follow the real time series. Actually, the numerical errors are often misleading in this sense, not showing under and overfitting problems and delay-prone solutions. Conversely, in our work, it can be observed that the best models bear a quite remarkable performance in modeling the output power, except for sharp peak variations, which are probably due to short-time meteorological events usually addressed by using more expensive remote sensing techniques for cloud tracking.

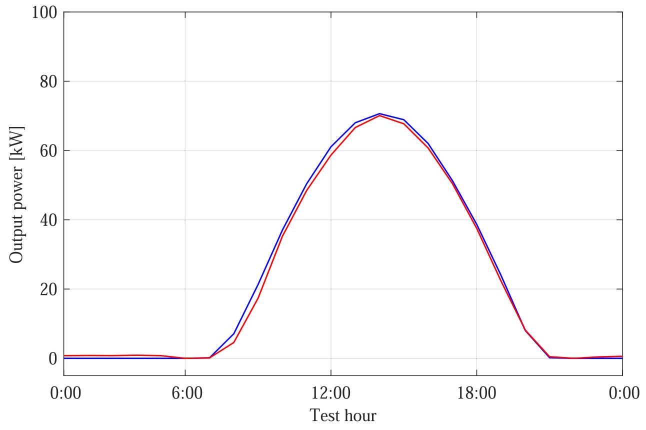


FIGURE 6. Predicted (red) and real (blue) value of output power in May 2018 by using the Multi-LSTM model with $\{S_1, S_2\}$ on 1-day test set.

D. NUMERICAL RESULTS: TRAINING AND COMPLEXITY

The average training times (in seconds) for each of the considered approaches are reported in Table 2; they are relevant to the optimal model obtained by using the hyperparameters for network complexity and for the training algorithm as listed before. Remarkable differences can be observed between the univariate and multivariate approaches, and between the multivariate ones themselves. In all cases, the training time does not exceed 85 seconds by using a very basic CPU and GPU, it is about 60 seconds for the best scores of Multi-LSTM. Therefore, working with time series sampled on a hourly basis, the model can be retrained and even optimized in an operation scenario every hour, as soon as new samples are available.

More in detail, the training times of multivariate approaches vary according to the number of the considered time series. The C-LSTM is faster when two time series are used rather than three, with a relative improvement from 33% up to 37% considering $\{S_1, S_2\}$ and from 35% up to 36% considering $\{S_1, S_3\}$. This difference is not so evident when comparing $\{S_1, S_2\}$ with respect to $\{S_1, S_3\}$. It is important to remark that the C-LSTM is the slowest network compared to the other DNN models.

The Conv-LSTM is also faster when two time series are used rather than three, but in this case the difference is very small. The relative improvements range from 2.4% up to 6.7% considering $\{S_1, S_2\}$ and from 0.3% up to 4.1% considering $\{S_1, S_3\}$. The training times obtained using $\{S_1, S_2\}$ and $\{S_1, S_3\}$ are quite similar. It is important to underline that the Conv-LSTM is the fastest network compared to both univariate and multivariate models.

Looking at the pairs $\{S_1, S_2\}$ and $\{S_1, S_3\}$, the Multi-LSTM has slightly lower training times than the C-LSTM. Again, using fewer time series positively affects network speed. In fact, the training times obtained with $\{S_1, S_2\}$ and $\{S_1, S_3\}$ are faster than those obtained with $\{S_1, S_2, S_3\}$. In the first case the relative improvements range from 22% up to 24% whereas in the second case they range from 22% up to 23%. In Stacked-LSTM, the training times are very similar to each

TABLE 2. Average training time (s) and model complexity (No. of Parameters) considering both univariate and multivariate approaches.

Model	Input Data	Training Time May 1-day	Training Time May 3-days	Training Time Oct. 1-day	Training Time Oct. 3-days	Model Complexity (same model for all tests)
Basic LSTM (with embedding)	S_1	26.404	26.623	26.631	26.466	$6.631 \cdot 10^3$
Basic LSTM (without embedding)	S_1	26.430	26.178	26.292	26.169	$3.871 \cdot 10^3$
C-LSTM	S_1, S_2	52.779	55.560	52.840	53.630	$70.126 \cdot 10^3$
	S_1, S_3	53.493	53.789	53.503	53.224	$70.126 \cdot 10^3$
	S_1, S_2, S_3	84.017	82.617	83.685	82.445	$70.990 \cdot 10^3$
Conv-LSTM	S_1, S_2	18.201	18.530	18.739	17.988	$33.286 \cdot 10^3$
	S_1, S_3	18.703	18.860	18.814	18.697	$33.286 \cdot 10^3$
	S_1, S_2, S_3	19.504	19.265	19.190	18.755	$32.311 \cdot 10^3$
Multi-LSTM	S_1, S_2	52.646	52.675	52.556	51.574	$58.401 \cdot 10^3$
	S_1, S_3	52.529	51.983	52.249	52.116	$58.401 \cdot 10^3$
	S_1, S_2, S_3	68.235	67.558	67.242	67.586	$74.601 \cdot 10^3$
Stacked-LSTM	S_1, S_2	43.673	43.581	43.512	43.076	$23.501 \cdot 10^3$
	S_1, S_3	43.234	43.465	43.348	43.072	$23.501 \cdot 10^3$
	S_1, S_2, S_3	43.528	43.304	43.768	43.318	$23.601 \cdot 10^3$

other in all tests. In three cases out of four the fastest times are obtained with the pair $\{S_1, S_3\}$. Instead, in the 3-days test set in May the use of $\{S_1, S_2, S_3\}$ achieve the fastest performance.

Considering the univariate approach, there are two different behaviors depending on whether or not the embedding technique is used. In the 1-day tests of May, training times are very similar while in all other tests the basic LSTM network which does not use embedding is faster than the one which uses it, with a relative improvement from 1.1% up to 1.7%.

The inference time necessary to evaluate the predicted samples is in all cases of few milliseconds by using the adopted software implementation, so differences among different test sets and among different models are not meaningful. Nonetheless, we report in Table 2 the number of parameters associated with each optimal DNN model (CNN filter coefficients, LSTM and fully connected weights, etc.), as they determine the number of algebraic and nonlinear operations during inference and give information regarding how much the model complexity scales with respect to the training time and, above all, to the related performance.

As discussed before, a same optimized DNN model is used to test all cases and hence, the complexity is independent of the considered test set but it may change for a same DNN according to the size of input data (i.e., embedding procedure and used time series). The univariate LSTM model is the simplest one although its complexity strongly depends upon the embedding procedure and the number of adopted past samples at the DNN's input. The multivariate models have a higher complexity, up to one order of magnitude, which is quite stable with respect to the number of adopted time series for most of them, apart from Multi-LSTM because of the replicated LSTM layers at the first stage.

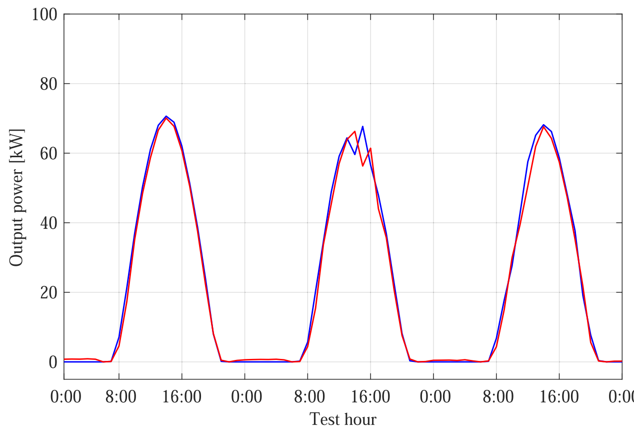


FIGURE 7. Predicted (red) and real (blue) value of output power in May 2018 by using the Multi-LSTM model with $\{S_1, S_2\}$ on 3-days test set.

Stacked-LSTM is the simplest model among the multivariate ones, while C-LSTM has quite a triple complexity due to the number of involved layers in its stacked architecture. By the way, it is not surprising that Conv-LSTM's complexity gets lower when three time series are used instead of two, as we do not use padding at the CNN frame boundaries while the filter size is larger (i.e., 3×3) in this case.

E. GENERAL DISCUSSION

A comprehensive comparison of the multivariate models considered in this article is illustrated by the radar chart in Fig. 10. The 'performance' reported is the average of MAE results in Table 1 over all combinations of input data and test conditions. The same average is computed for the 'training time' (in seconds) reported in Table 2, while the 'complexity' is the average of parameters over the different

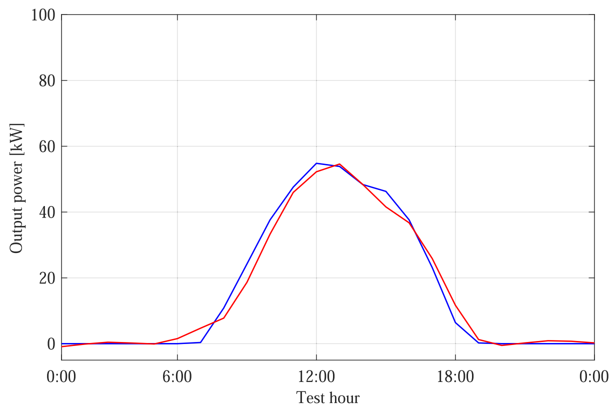


FIGURE 8. Predicted (red) and real (blue) value of output power in October 2018 by using the basic LSTM model with embedding on 1-day test set.

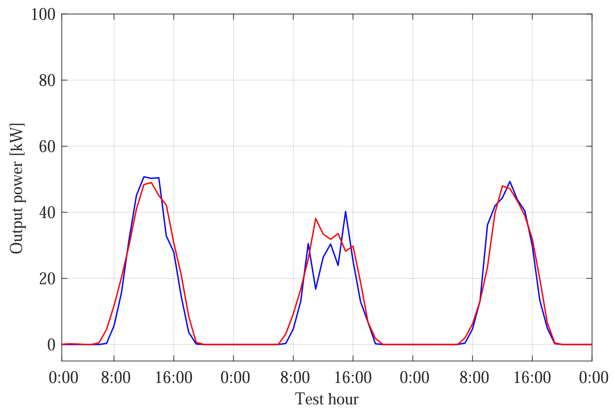


FIGURE 9. Predicted (red) and real (blue) value of output power in October 2018 by using the Multi-LSTM model with $\{S_1, S_2, S_3\}$ on 3-days test set.

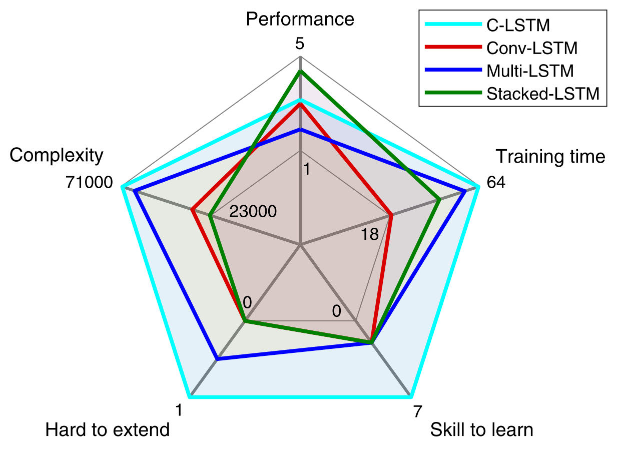


FIGURE 10. A radar chart comparing the optimized multivariate models: the lower the value along each axis the better the quality of the model in terms of performance, complexity and flexibility.

input data. The 'skill to learn' has been evaluated as the number of hyperparameters to be set for each DNN model, which in turn indicates the difficulty in optimizing the model when this number increases. The 'hard to extend' refers to the model scalability, and to the related software implementation, when the number of input time series increases; it has been quantitatively evaluated as 0 when the hardness is low, 1 when the hardness is high, and 0.5 in the middle.

Based on the adopted convention, the smallest the area of the produced chart the better the results in terms of performance, training, complexity, and so on. Although on average the Multi-LSTM model can achieve a better performance while the Stacked-LSTM results as the simplest one, it is evident that Conv-LSTM is the model scoring the best trade-off and the smallest area. Definitely, the final choice among such models depends on the relevance of performance with respect to other factors that basically involve training times, model complexity and software scalability with respect to the number of adopted time series.

Overall, the results reported so far are satisfactory and assess the soundness of the presented forecasting methods. It is clear that, apart from very few cases, the performance of prediction is quite comparable, which in turns spawns a reflection on the effective use of the more complex deep networks for similar tasks. While it is true that DNN training times are way shorter than even a short prediction horizon, the increasing computational complexity of training several LSTM layers is still not negligible and it must be taken into account especially when complex cross-validation procedures are adopted. However, the reported numerical results are very promising and favor the use of more resource expensive models, such as the ones proposed in this work.

VI. CONCLUSION

This article presented four different DNN models for the multivariate prediction of photovoltaic power output time series. They mainly rely on LSTM models to exploit long-term correlation and dependencies among several different time series. Two of the four DNN models also exploit the 2-D convolutional layer in order to obtain a generalized embedding procedure. All DNN models were tested on real world data to show and compare their performance with the basic univariate LSTM model. The numerical results suggest that the multivariate analysis produces better results than the univariate analysis in most of the cases. The results allowed to highlight the differences among the multivariate models and to prove their accuracy in terms of MAE; the Multi-LSTM network resulted as the most accurate. This article is a further step in an ongoing research conducted by the authors about forecasting in renewable energies and smart grids. Future works will consider a different combination of input time series as well as the extension of the proposed approaches to more complex scenarios. Particularly, improvements will be introduced for the management of distributed energy resources where active prosumers need efficient data-driven modeling tools for enabling active participation and diffused coordination tasks in a smart grid.

REFERENCES

[1] J. Donate, X. Li, G. Gutierrez, and A. Sanchis de Miguel, "Time series forecasting by evolving artificial neural networks with genetic algorithms, differential evolution and estimation of distribution algorithm," *Neural Comput. Appl.*, vol. 22, pp. 1–10, Jan. 2011.

[2] T. Couture and Y. Gagnon, "An analysis of feed-in tariff remuneration models: Implications for renewable energy investment," *Energy Policy*, vol. 38, no. 2, pp. 955–965, Feb. 2010.

[3] W. U. Rehman, A. R. Bhatti, A. B. Awan, I. A. Sajjad, A. A. Khan, R. Bo, S. S. Haroon, S. Amin, I. Tlili, and O. Oboreh-Snapps, "The penetration of renewable and sustainable energy in Asia: A state-of-the-art review on net-metering," *IEEE Access*, vol. 8, pp. 170364–170388, 2020.

[4] J. A. P. Lopes, N. Hatziargyriou, J. Mutale, P. Djapic, and N. Jenkins, "Integrating distributed generation into electric power systems: A review of drivers, challenges and opportunities," *Electric Power Syst. Res.*, vol. 77, no. 9, pp. 1189–1203, Jul. 2007.

[5] D.-U. Kim and S. Kim, "Anti-islanding detection method using phase-shifted feed-forward voltage in grid-connected inverter," *IEEE Access*, vol. 7, pp. 147179–147190, 2019.

[6] Y.-Y. Hong and M.-J. Liu, "Optimized interval type-II fuzzy controller-based STATCOM for voltage regulation in power systems with photovoltaic farm," *IEEE Access*, vol. 6, pp. 78731–78739, 2018.

[7] R. Sinvula, K. M. Abo-Al-Ez, and M. T. Kahn, "Harmonic source detection methods: A systematic literature review," *IEEE Access*, vol. 7, pp. 74283–74299, 2019.

[8] R. Araneo, S. Lammens, M. Grossi, and S. Bertone, "EMC issues in high-power grid-connected photovoltaic plants," *IEEE Trans. Electromagn. Compat.*, vol. 51, no. 3, pp. 639–648, Aug. 2009.

[9] H. A. U. Muqeet and A. Ahmad, "Optimal scheduling for campus prosumer microgrid considering price based demand response," *IEEE Access*, vol. 8, pp. 71378–71394, 2020.

[10] R. Araneo, S. Celozzi, and C. Vergine, "Eco-sustainable routing of power lines for the connection of renewable energy plants to the Italian high-voltage grid," *Int. J. Energy Environ. Eng.*, vol. 6, no. 1, pp. 9–19, Mar. 2015.

[11] I. Kaitovic, S. Lukovic, and M. Malek, "Proactive failure management in smart grids for improved resilience: A methodology for failure prediction and mitigation," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2015, pp. 1–6.

[12] M. F. Zia, M. Benbouzid, E. Elbouchikhi, S. M. Muyeen, K. Techato, and J. M. Guerrero, "Microgrid transactive energy: Review, architectures, distributed ledger technologies, and market analysis," *IEEE Access*, vol. 8, pp. 19410–19432, 2020.

[13] F. Chen, D. Liu, and X. Xiong, "Research on stochastic optimal operation strategy of active distribution network considering intermittent energy," *Energies*, vol. 10, no. 4, p. 522, Apr. 2017.

[14] A. Rosato, R. Altilio, R. Araneo, and M. Panella, "Prediction in photovoltaic power by neural networks," *Energies*, vol. 10, no. 7, p. 1003, Jul. 2017.

[15] J. Leitao, P. Gil, B. Ribeiro, and A. Cardoso, "A survey on home energy management," *IEEE Access*, vol. 8, pp. 5699–5722, 2020.

[16] P. Mirowski, S. Chen, T. Kam Ho, and C.-N. Yu, "Demand forecasting in smart grids," *Bell Labs Tech. J.*, vol. 18, no. 4, pp. 135–158, Mar. 2014.

[17] C. Brancucci Martinez-Anido, B. Botor, A. R. Florita, C. Draxl, S. Lu, H. F. Hamann, and B.-M. Hodge, "The value of day-ahead solar power forecasting improvement," *Sol. Energy*, vol. 129, pp. 192–203, May 2016.

[18] R. Pal, C. Chelmis, M. Frincu, and V. Prasanna, "MATCH for the prosumer smart grid the algorithmics of real-time power balance," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 12, pp. 3532–3546, Dec. 2016.

[19] M. B. Rasheed, M. A. Qureshi, N. Javaid, and T. Alquthami, "Dynamic pricing mechanism with the integration of renewable energy source in smart grid," *IEEE Access*, vol. 8, pp. 16876–16892, 2020.

[20] D. S. Kumar, G. M. Yagli, M. Kashyap, and D. Srinivasan, "Solar irradiance resource and forecasting: A comprehensive review," *IET Renew. Power Gener.*, vol. 14, no. 10, pp. 1641–1656, Jul. 2020.

[21] A. Mellit, A. Massi Pavan, E. Ogliari, S. Leva, and V. Lughi, "Advanced methods for photovoltaic output power forecasting: A review," *Appl. Sci.*, vol. 10, no. 2, p. 487, Jan. 2020.

[22] M. N. Akhter, S. Mekhilef, H. Mokhlis, and N. Mohamed Shah, "Review on forecasting of photovoltaic power generation based on machine learning and Metaheuristic techniques," *IET Renew. Power Gener.*, vol. 13, no. 7, pp. 1009–1023, May 2019.

[23] R. Fierimonte, S. Scardapane, A. Uncini, and M. Panella, "Fully decentralized semi-supervised learning via privacy-preserving matrix completion," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 11, pp. 2699–2711, Nov. 2017.

[24] A. Gensler, J. Henze, B. Sick, and N. Raabe, "Deep learning for solar power forecasting—An approach using autoencoder and LSTM neural networks," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2016, pp. 002858–002865.

[25] C. Persson, P. Bacher, T. Shiga, and H. Madsen, "Multi-site solar power forecasting using gradient boosted regression trees," *Sol. Energy*, vol. 150, pp. 423–436, Jul. 2017.

[26] E. Ogliari, A. Dolara, G. Manzolini, and S. Leva, "Physical and hybrid methods comparison for the day ahead PV output power forecast," *Renew. Energy*, vol. 113, pp. 11–21, Dec. 2017.

[27] J. Antonanzas, N. Osorio, R. Escobar, R. Urraca, F. J. Martinez-de-Pison, and F. Antonanzas-Torres, "Review of photovoltaic power forecasting," *Sol. Energy*, vol. 136, pp. 78–111, Oct. 2016.

[28] C.-J. Huang and P.-H. Kuo, "Multiple-input deep convolutional neural network model for short-term photovoltaic power forecasting," *IEEE Access*, vol. 7, pp. 74822–74834, 2019.

[29] M. S. Hossain and H. Mahmood, "Short-term photovoltaic power forecasting using an LSTM neural network and synthetic weather forecast," *IEEE Access*, vol. 8, pp. 172524–172533, 2020.

[30] Y. Yu, J. Cao, and J. Zhu, "An LSTM short-term solar irradiance forecasting under complicated weather conditions," *IEEE Access*, vol. 7, pp. 145651–145666, 2019.

[31] H. Zhou, Y. Zhang, L. Yang, Q. Liu, K. Yan, and Y. Du, "Short-term photovoltaic power forecasting based on long short term memory neural network and attention mechanism," *IEEE Access*, vol. 7, pp. 78063–78074, 2019.

[32] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Neural Inf. Process. Syst.*, Jan. 2012, pp. 1–9.

[33] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[34] A. Rosato, R. Altilio, R. Araneo, and M. Panella, "Embedding of time series for the prediction in photovoltaic power plants," in *Proc. IEEE 16th Int. Conf. Environ. Electr. Eng. (EEEIC)*, Jun. 2016, pp. 1–4.

[35] H. D. I. Abarbanel, *Analysis of Observed Chaotic Data*. New York, NY, USA: Springer-Verlag, 1996.

[36] A. Rosato, M. Panella, R. Araneo, and A. Andreotti, "A neural network based prediction system of distributed generation for the management of microgrids," *IEEE Trans. Ind. Appl.*, vol. 55, no. 6, pp. 7092–7102, Nov. 2019.

[37] A. Rosato, M. Panella, and R. Araneo, "A distributed algorithm for the cooperative prediction of power production in PV plants," *IEEE Trans. Energy Convers.*, vol. 34, no. 1, pp. 497–508, Mar. 2019.

[38] M. Rana, I. Koprinska, and V. G. Agelidis, "Univariate and multivariate methods for very short-term solar photovoltaic power forecasting," *Energy Convers. Manage.*, vol. 121, pp. 380–390, Aug. 2016.

[39] P. Bacher, H. Madsen, and H. A. Nielsen, "Online short-term solar power forecasting," *Sol. Energy*, vol. 83, no. 10, pp. 1772–1783, Oct. 2009.

[40] Y. Li, Y. Su, and L. Shu, "An armax model for forecasting the power output of a grid connected photovoltaic system," *Renew. Energy*, vol. 66, pp. 78–89, Jun. 2014.

[41] Y. Chu, B. Urquhart, S. M. I. Gohari, H. T. C. Pedro, J. Kleissl, and C. F. M. Coimbra, "Short-term reforecasting of power output from a 48 MWe solar PV plant," *Sol. Energy*, vol. 112, pp. 68–77, Feb. 2015.

[42] H. T. C. Pedro and C. F. M. Coimbra, "Assessment of forecasting techniques for solar power production with no exogenous inputs," *Sol. Energy*, vol. 86, no. 7, pp. 2017–2028, Jul. 2012.

[43] M. Bouzerdoum, A. Mellit, and A. Massi Pavan, "A hybrid model (SARIMA–SVM) for short-term power forecasting of a small-scale grid-connected photovoltaic plant," *Sol. Energy*, vol. 98, pp. 226–235, Dec. 2013.

[44] A. Lahouar, A. Mejri, and J. Ben Hadj Slama, "Importance based selection method for day-ahead photovoltaic power forecast using random forests," in *Proc. Int. Conf. Green Energy Convers. Syst. (GECS)*, Mar. 2017, pp. 1–7.

[45] J. Liu, M. Y. Cao, D. Bai, and R. Zhang, "Solar radiation prediction based on random forest of feature-extraction," in *Proc. IOP Conf. Ser., Mater. Sci. Eng.*, vol. 658, Oct. 2019, Art. no. 012006.

[46] M. Meng and C. Song, "Daily photovoltaic power generation forecasting model based on random forest algorithm for north China in winter," *Sustainability*, vol. 12, no. 6, p. 2247, Mar. 2020.

[47] R. Srivastava, A. N. Tiwari, and V. K. Giri, "Forecasting of solar radiation in india using various ANN models," in *Proc. 5th IEEE Uttar Pradesh Sect. Int. Conf. Electr., Electron. Comput. Eng. (UPCON)*, Nov. 2018, pp. 1–6.

[48] N. Premalatha and A. Valan Arasu, "Prediction of solar radiation for solar systems by using ANN models with different back propagation algorithms," *J. Appl. Res. Technol.*, vol. 14, no. 3, pp. 206–214, Jun. 2016.

[49] Y. Wang, Y. Shen, S. Mao, G. Cao, and R. M. Nelms, "Adaptive learning hybrid model for solar intensity forecasting," *IEEE Trans. Ind. Informat.*, vol. 14, no. 4, pp. 1635–1645, Apr. 2018.

[50] T. Laopaiboon, W. Ongsakul, P. Panyainkaew, and N. Sasidharan, "Hour-ahead solar forecasting program using back propagation artificial neural network," in *Proc. Int. Conf. Utility Exhib. Green Energy Sustain. Develop. (ICUE)*, Oct. 2018, pp. 1–7.

[51] R. M. Ehsan, S. P. Simon, and P. R. Venkateswaran, "Artificial neural network predictor for grid-connected solar photovoltaic installations at atmospheric temperature," in *Proc. Int. Conf. Adv. Green Energy (ICAGE)*, Dec. 2014, pp. 44–49.

[52] J. Liu, W. Fang, X. Zhang, and C. Yang, "An improved photovoltaic power forecasting model with the assistance of aerosol index data," *IEEE Trans. Sustain. Energy*, vol. 6, no. 2, pp. 434–442, Apr. 2015.

[53] M. Q. Raza, N. Mithulananthan, and A. Summerfield, "Solar output power forecast using an ensemble framework with neural predictors and Bayesian adaptive combination," *Sol. Energy*, vol. 166, pp. 226–241, May 2018.

[54] G. Amarasinghe and S. Abeygunawardane, "An artificial neural network for solar power generation forecasting using weather parameters," in *Proc. Annu. Sessions IESL*, Oct. 2018, pp. 431–438.

[55] D. Lee and K. Kim, "Recurrent neural network-based hourly prediction of photovoltaic power output using meteorological information," *Energies*, vol. 12, no. 2, p. 215, Jan. 2019.

[56] F.-V. Gutierrez-Corea, M.-A. Manso-Callejo, M.-P. Moreno-Regidor, and M.-T. Manrique-Sancho, "Forecasting short-term solar irradiance based on artificial neural networks and data from neighboring meteorological stations," *Sol. Energy*, vol. 134, pp. 119–131, Sep. 2016.

[57] H.-T. Yang, C.-M. Huang, Y.-C. Huang, and Y.-S. Pai, "A weather-based hybrid method for 1-Day ahead hourly forecasting of PV power output," *IEEE Trans. Sustain. Energy*, vol. 5, no. 3, pp. 917–926, Jul. 2014.

[58] S. A. Binti Jumaat, F. Crocker, M. H. Abd Wahab, and N. H. Binti Mohammad Radzi, "Investigate the photovoltaic (PV) module performance using artificial neural network (ANN)," in *Proc. IEEE Conf. Open Syst. (ICOS)*, Oct. 2016, pp. 59–64.

[59] K. R. Kumar and M. S. Kalavathi, "Artificial intelligence based forecast models for predicting solar power generation," *Mater. Today, Proc.*, vol. 5, no. 1, pp. 796–802, 2018.

[60] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6645–6649.

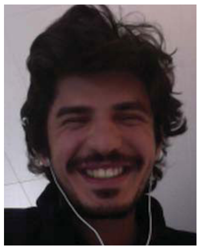
[61] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1D convolutional neural networks and applications: A survey," 2019, *arXiv:1905.03554*. [Online]. Available: https://arxiv.org/abs/1905.03554

[62] S. Kiranyaz, T. Ince, R. Hamila, and M. Gabbouj, "Convolutional neural networks for patient-specific ECG classification," in *Proc. 37th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Aug. 2015, pp. 2608–2611.

[63] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, Dec. 2014, pp. 1–15.

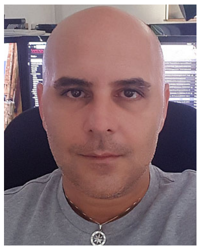
[64] P. M. Lerman, "Fitting segmented regression models by grid search," *Appl. Statist.*, vol. 29, no. 1, pp. 77–84, 1980.

FEDERICO SUCCETTI was born in 1992. He received the M.Sc. degree in electronic engineering from the Sapienza University of Rome, Italy, in 2019. He is currently a Research Fellow with the Department of Information Engineering, Electronics and Telecommunications, Sapienza University of Rome. His current research interests include machine learning and deep learning techniques for management and control of complex systems.



ANTONELLO ROSATO (Member, IEEE) was born in 1990. He received the M.Sc. degree (Hons.) in telecommunication engineering and the Ph.D. degree in information and communications technologies from the Sapienza University of Rome, Italy, in 2015 and 2018, respectively. He is currently a Postdoctoral Research Fellow with the Department of Information Engineering, Electronics and Telecommunications, Sapienza University of Rome. His current research interests include machine learning techniques for prediction of complex systems behavior, distributed clustering algorithms, and neural and fuzzy-neural models for environmental monitoring.

RODOLFO ARANEO (Senior Member, IEEE) was born in Rome, Italy, in 1975. He received the M.S. *(cum laude)* and Ph.D. degrees in electrical engineering from the Sapienza University of Rome, Rome, in 1999 and 2002, respectively. In 1999, he was a Visiting Student with the National Institute of Standards and Technology, Boulder, CO, USA, where he was engaged in TEM cells and shielding. In 2000, he was a Visiting Researcher with the Department of Electrical and Computer Engineering, Missouri University of Science and Technology, where he was engaged in printed circuit boards and finite-difference time-domain techniques. He is currently an Associate Professor with the Department of Electrical Engineering, Sapienza University of Rome. His current research interests include electromagnetic compatibility, numerical and analytical techniques for modeling high-speed printed circuit boards, shielding, transmission-line analysis, frequency selective surfaces, and nanostructures. He was a recipient of the Past President's Memorial Award in 1999 from the IEEE Electromagnetic Compatibility Society. He is currently the General Chair of the IEEE International Conference on Environment and Electrical Engineering.



MASSIMO PANELLA (Senior Member, IEEE) was born in Rome, Italy, in 1971. He received the Dr.Eng. degree (Hons.) in electronic engineering and the Ph.D. degree in information and communication engineering from the Sapienza University of Rome, in 1998 and 2002, respectively. He is currently Associate Professor with the Department of Information Engineering, Electronics and Telecommunications, Sapienza University of Rome, where he holds courses on electrical engineering, circuit theory, circuits and algorithms for signal processing, machine learning, and computational intelligence. His research activities pertain to circuit theory, machine learning, computational intelligence, and quantum computing for modeling, optimization, and control of complex systems, even in distributed environments with multiple data sources as for sensor networks, pervasive systems, and the IoT applications. He was an Associate Editor of the IEEE TRANSACTIONS ON FUZZY SYSTEMS. He is currently an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.