Received July 28, 2015, accepted September 9, 2015, date of publication October 1, 2015, date of current version October 27, 2015.

Digital Object Identifier 10.1109/ACCESS.2015.2485943

Short-Term Electric Load Forecasting Using Echo State Networks and PCA Decomposition

FILIPPO MARIA BIANCHI[1], ENRICO DE SANTIS[1], ANTONELLO RIZZI[1], (Member, IEEE), AND ALIREZA SADEGHIAN[2], (Senior Member, IEEE)
[1]Department of Information Engineering, Electronics, and Telecommunications, Sapienza University of Rome, Rome 00184, Italy
[2]Department of Computer Science, Ryerson University, Toronto, ON M5B 2K3, Canada

Corresponding author: F. M. Bianchi (filippomaria.bianchi@uniroma1.it)

ABSTRACT In this paper, we approach the problem of forecasting a time series (TS) of an electrical load measured on the Azienda Comunale Energia e Ambiente (ACEA) power grid, the company managing the electricity distribution in Rome, Italy, with an echo state network (ESN) considering two different leading times of 10 min and 1 day. We use a standard approach for predicting the load in the next 10 min, while, for a forecast horizon of one day, we represent the data with a high-dimensional multi-variate TS, where the number of variables is equivalent to the quantity of measurements registered in a day. Through the orthogonal transformation returned by PCA decomposition, we reduce the dimensionality of the TS to a lower number k of distinct variables; this allows us to cast the original prediction problem in k different one-step ahead predictions. The overall forecast can be effectively managed by k distinct prediction models, whose outputs are combined together to obtain the final result. We employ a genetic algorithm for tuning the parameters of the ESN and compare its prediction accuracy with a standard autoregressive integrated moving average model.

INDEX TERMS Time-series, forecasting, electric load prediction, echo state network, genetic algorithm, PCA, dimensionality reduction, smart grid.

I. INTRODUCTION

In this paper we consider the problem of Time-Series Forecasting (TSF) [1] concerning the prediction of future values of a time-series (TS) of electric load, leveraging previously observed history. An accurate Short-Term Load Forecast (STLF) method can reduce operating costs, keep power markets efficient and provide a better understanding of the dynamics of the monitored system. On the other hand, a wrong prediction could cause either a load overestimation, which leads to the excess of supply and reserve and consequently more costs and contract curtailments for market participants, or a load underestimation resulting in failures in gathering enough provisions, thereby more costly supplementary services [2], [3].

Recently, the Echo State Network (ESN) have been employed in a wide range of applications such as intrusion detection [4], adaptive control [5], financial credit rating [6], harmonic distortion measurements [7] and speech recognition [8]. In power grid applications, ESN have been adopted as stand-alone forecaster with a minimum number of inputs for STLF with promising results, achieving high accuracy relatively to the problem of electricity load forecast [9], [10]. ESN not only benefits from the presence of feedbacks like any other RNN (Recurrent Neural Network), a feature which gives to the system the capability to model any complex dynamic behavior, but also gains a sparsely interconnected reservoir of neurons that leads to a very fast and simple training procedure, unlike the complicated and time consuming training process required by the RNNs without reservoir. On the other hand, because of the property of ‘‘short-term memory’’, ESN is not suitable for long-term predictions [11]; for long forecasting horizons, a very basic averaging model in general outperforms many more complex alternatives [12].

In this work we face the problem of STLF considering two different methodologies. In the first approach we directly predict the next value on the TS (1-step ahead forecast). When the forecasting horizon is longer, i.e. when the forecast step m is high ($m \gg 1$), trying to directly predict the values of the TS become more difficult. In this case, we represent the TS through a matrix M, with a number of columns equal to the leading time m: in this way the problem consists in predicting the values in the next row of the matrix. Using a Principal Component Analysis (PCA) decomposition, the m columns of the matrix can be approximated by the first k independent components, with $k < m$, which can be predicted independently, possibly in distinct parallel threads. The m-step ahead forecast can therefore be evaluated with k different 1-step ahead forecasts. In this paper we process a novel dataset, relative to the demand of electricity in the distribution network of Rome. We explain how to retrieve and how to dynamically update the coefficients in the PCA decomposition, as long as new values of the TS are observed. We forecast the values of the components using k different predictors and we show how to combine them for retrieving the final forecast of the electricity load. Finally, we compare the results obtained using an ESN as forecast model or with standard ARIMA (Auto-Regressive Integrated Moving Average) predictor.

The remainder of the paper is structured as follows: in Sect. II we discuss previous works in the literature which approached the TSF problem through ESNs, by considering multi-variate TS and by applying techniques for dimensionality reduction. In Sect. III we present the dataset examined in this study and we discuss the properties of the TS under analysis. In Sect. IV we explain how to generate the sets used for training and for testing the prediction model in order to face two different STFL problems, characterized by a short and a long forecast horizons respectively. In the case of long leading time, we explain how we represented the data with a multi-variate TS. In Sect. V we firstly introduce the PCA procedure, then we discuss how it can be used for reducing the number of variables in the multi-variate TS, which can be approximated with a small number of independent components. Finally, we explain how to effectively take advantage of this representation of the TS for forecasting future loads. In Sect. VI we describe the standard procedure for forecasting a TS using an ESN. In Sect. VII we discuss how we tested our system considering different setups and we show the obtained results, while in Sect. VIII we report our conclusions.

II. RELATED WORKS

In the Smart Grid era, where scientific research from one hand and industry and regulation from the other are transforming the actual power grid in a more intelligent technological ecosystem, the electric load forecasting became a fundamental issue in a myriad of applications. Information Communication Technologies (ICT) with smart metering infrastructures provide a huge amount of load data that can be analyzed and modeled with advanced learning algorithms and data mining techniques [13]–[17].

Approaching the TSF with an Artificial Neural Network (ANN), has been widely explored in the past years [1], [18]–[20]. Recently, the RNN has become one of the most used type of ANN as a prediction system, especially for chaotic TS [21], [22], and many works present a direct comparison of the prediction performances obtained using a simple RNN or a standard ARIMA model [23]. Varshney and Verma [24] use a RNN as forecast model, affirming that the commonly used feed-forward back-propagation network offers good performance, but this performance could be improved by using recurrence or reusing past inputs and outputs. The motivation behind considering recurrence is that some patterns may repeat over time. A network which remembers previous inputs or feedback of previous outputs may have greater success in determining these time dependent patterns. In [20] a novel feed-forward NN is proposed which, similarly to the Fourier decomposition method, introduces in every layer a Cosine function to capture the remaining unexpressed non-linearity. The number of layers is set dynamically and it depends upon the complexity of the underlying process and the desired level of accuracy, while the number of nodes at each layer is fixed. A relevant work dealing with ESN in TSF is [9], where the authors train a different ESN for predicting each hour of the day. They perform both a 1-hour and a 24-hours ahead forecast. The 24-h forecast is done calling recursively the 1-h ahead forecast. In the training of the ESN only a specific hour of the day is taken into account. In [25] the ESN is used for a multi-variate TS prediction and the reservoir is trained with a Bayesian regularization technique. In order to avoid overfitting in the regression step, neurons and redundant connections are pruned. In the context of chaotic TS predictions, Li *et al.* [22] propose an alternative to the Bayesian regression for estimating the regularization parameter. The method uses a Laplacian likelihood function rather than a Gaussian, which has demonstrated to be more robust to noise and outliers. In [26] chaotic TS are predicted with ESNs using a decentralized training algorithm over a network of agents, which achieved performances comparable to the centralized case. An hybrid model using ESN in conjunction with ARIMA is proposed in [11], where a TS with multiple seasonality is forecast. The TS is decomposed according to a wavelet multi-resolution analysis in a sum of different wavelets, modeling the seasonal part of the TS, and a residual smooth term. The periodicities that the TS exhibits at different cycles can be isolated using wavelet terms, which are extremely regular and can be predicted using an ARIMA model. The smooth component, instead, is processed with an ESN and the final prediction is obtained integrating all the components. An interesting forecasting model using ESN on a multivariate TS can be found in [25], where the prediction is performed using a special ESN with a different reservoir for each variable: this allows to better catch the dynamic of each single variable. Because of the presence of multiple reservoirs, the number of output connections from the internal matrices to the output is huge and could lead to overfitting during the training; for this reason authors propose a method for pruning these connections.

In previous researches, the complexity of a multi-variate TS prediction problem was reduced by applying singular value decomposition (SVD) prior to time-series forecasting in order to decrement the number of variables in the TS.

The original TS is re-arranged in a $n \times m$ matrix, being m the leading time in the forecast and n the number of periods of length m in the TS. Each column of this matrix constitutes a TS of observations for a particular interval in the time period. SVD is used to extract the main underlying components in these data and thus reducing the number of variables to be forecast. Taylor [27], [28] applies a SVD to obtain a set of $k < m$ coefficients, which are successively forecast using exponential smoothing approaches. Shen and Huang [29] used an AR(1) model for infra-day and inter-day load forecasting. They represented the TS with a matrix where each row stores the load registered in a day and the columns the forecast for each half-hour; the 48 columns are reduced using a SVD. Taylor *et al.* [30] compare the accuracy of six univariate methods for short-term electricity demand forecasting for lead times up to a day ahead. This work analyzes the following methodologies: a multiplicative seasonal ARIMA; an exponential smoothing; an artificial neural network implementation and a Principal Component Analysis (PCA) approach that is strictly related to SVD decomposition. The PCA with the intrinsic dimensionality reduction capability is also used in [31] for STLF of intraday electricity demand, considering data from several European countries.

III. THE ANALYZED TIME SERIES

In this work we analyze a TS of the electricity load provided by ACEA (*Azienda Comunale Energia e Ambiente*), the company which manages the electric and hydraulic distribution in the city of Rome, Italy. The ACEA power grid covers 10,490 km of Medium Voltage (MV) lines, while the Low Voltage (LV) section covers 11.120 km. It is constituted of backbones of uniform section, exerting radially and with the possibility of counter-supply if a branch is out of order. Each backbone is fed by two distinct Primary Stations (PS) and each half-line is protected against faults through the breakers; we refer the reader to [32] for more details. The values in the TS concern the quantity of electricity supplied in the distribution network, measured on a MV feeder. Data are collected every 10 minutes for 3 years of activity, from 2009 to 2011. In Fig. 1 we report the electricity load profile registered in 3 consecutive days.

Before processing the TS, a common procedure consists in applying some kind of normalization to the data, such as standardization or rescaling, which is successively reversed when the forecast values must be provided. Transforming the data with a non-linear function like square-root, logarithm or hyperbolic tangent [20], [29], [33]–[35] is useful for stabilizing the variance in the data, without altering the underlying structure. A log-transformation allows also to capture a multiplicative seasonal pattern by models which are inherently additive. Then, in presence of long term trends or when the residuals show a marked increment in variance over time of the amplitudes of the seasonal cycles, a non-linear transformation should be preferred.

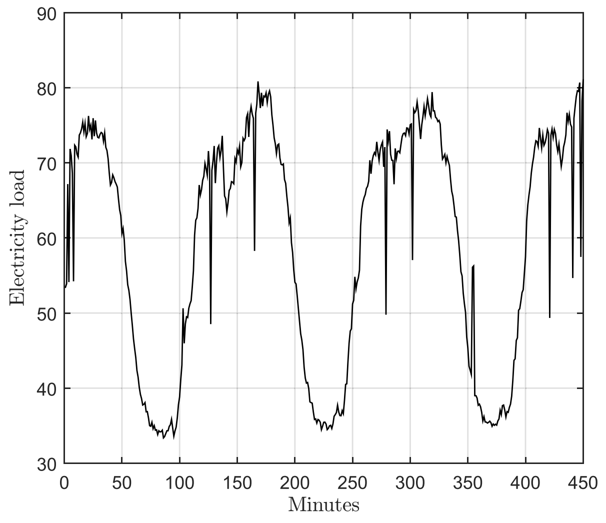

FIGURE 1. **The load profile of the TS in 450 time intervals (approximately 3 days).**

A preemptive analysis of the TS can help us to understand the nature of the seasonality and the variance in the data. In this preprocessing phase it is also important to identify the presence of multiple seasonalities, hidden daily and/or hourly patterns, which could be better treated using multiple forecasting systems [9], [25] or with models designed to deal with multiple seasonality [27], [28], [36]. Fourier frequency analysis and the auto-correlation function (ACF), evaluated up to sufficient high number of lags, are the most commonly used tools for seasonality study. As expected, in the ACF plot in Fig. 2(a), we can notice that the TS shows an obvious seasonality pattern every 144 time intervals (one day); this seasonality is slightly additive, but not multiplicative and the TS is likely to contain a monotonous trend. In fact, if a signal exhibits a monotonous trend it lacks a tendency to return to its mean value, or the values in its ACF are positive up to a high number of lags and the standard deviation of the residuals is greater than in the differentiated signal. For identifying a second, less obvious seasonality we apply a seasonal differencing [37] to the TS and we examine its ACF, which is reported in Fig. 2(b). As we can see, the absence of cyclic correlations shows that the gross pattern of seasonality has been removed and then we can exclude the presence of a second seasonality in the data.

We consider the average daily load in the TS in order to check if in different days the load profile changes. In Fig. 3 we report the mean values of the TS in a time-interval of 1 day, computed over 1 month of activity. We also report the standard deviation from the mean value (represented by gray dashed lines in the figure) which, as can be seen from the plot, is relatively small, confirming a stable load profile over different days. Since there are not multiplicative seasonalities in the data nor overdispersion, i.e. the variance is not greater than the mean value [38], a non-linear transformation for stabilizing the variance is *not* required and a rescaling or a standardization are suitable for normalizing the values

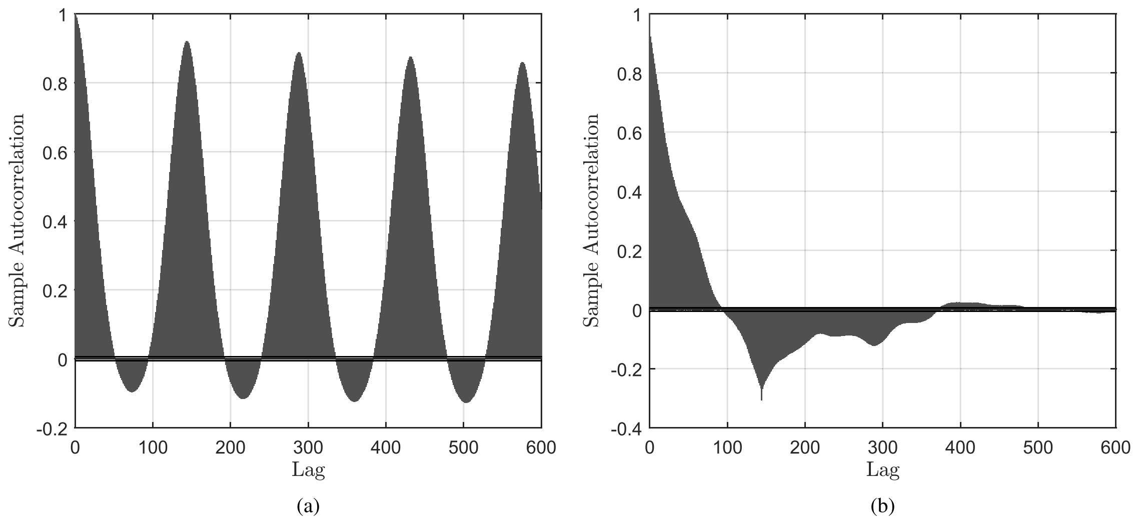

FIGURE 2. In (a) is drawn the autocorrelation of the TS; as we can notice there is a strong correlation component at every 144 lags. In (b) the autocorrelation of the same TS after a seasonal differencing. We can notice that TS now is deseasonalized and there are not other evident seasonalities.

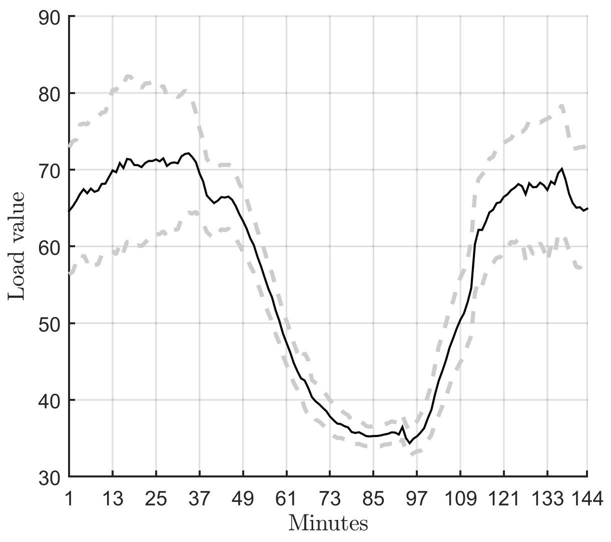

FIGURE 3. With the black line we represent the daily mean of the values in the TS, relatively to 1 month of activity. The standard deviation from the mean is reported in the graphic with gray dashed lines. As we can see, the deviation is small, in particular in the intervals when the electricity demand is lower.

in the TS. In particular, we choose to rescale the values in the interval [0, 1], using a unity-based normalization. Finally, since we do not observe the presence of multiple seasonality in the data or well defined weekly patterns, a single forecasting model can be adopted for our STFL task.

IV. PREDICTION LEADING TIME

In a TSF problem, the forecast step m defines how far ahead the prediction is performed; in particular given $X(\tau)$, which represents all the values assumed by the TS X up to time τ, namely $x_0, x_1, .., x_{\tau-1}, x_\tau$, we want to forecast $x_{\tau+m}$. A generic prediction model which takes as input a series of the observed values is trained to return as output the series of future values assumed by the TS m time intervals ahead. The training set is then composed of an ordered set of pairs of values $\{(x_0, x_m), (x_1, x_{1+m}), .., (x_T, x_{T+m})\}$, being T the number of time samples that we consider in the training data and the pair $(x_\tau, x_{\tau+m})$ the input and the desired output of the system at time τ. A test set (with the same structure) is used to evaluate the prediction accuracy of the model. Note that the value assumed by $x_{\tau+m}$ does not depends only on x_τ, but also on all the previous inputs, which are taken into account by the memory of the system.

In the following we explain the procedure implemented for splitting the data in order to obtain a training and a test set, used for synthesize a prediction model and for evaluating its performances respectively. Note that this procedure is general and it is adopted for both the forecast models considered in this paper (ARIMA and ESN) and the two different approaches used for representing the data, which are discussed at the end of this section.

From the original data X we generate 2 different overlapping TS of the same length, X_{obs} and X_{fut}, where $X_{\text{obs}}(\tau) = \{x_0, \ldots, x_\tau\}$ and $X_{\text{fut}}(\tau) = \{x_m, \ldots, x_{\tau+m}\}$. Note that X_{obs} and X_{fut} can be simply constructed by removing from X the last and the first m elements respectively. The length of the 2 new time-series is $N - m$, being N the entries in X. Successively, X_{obs} and X_{fut} are split in 2 different sets each, which represent the training set $(X^{tr}_{\text{obs}}, X^{tr}_{\text{fut}})$ and the test set $(X^{ts}_{\text{obs}}, X^{ts}_{\text{fut}})$. The procedure is illustrated in Fig. 4.

When the forecast horizon is very short (e.g. $m = 1$), a variety of different models and techniques can successfully predict the future values of X. In particular, an ESN-based prediction model, like the one considered in this work, obtains very high prediction accuracy when dealing with short leading times. As long as the value m increases, the forecast horizon stretches, the TSF problem become harder and the

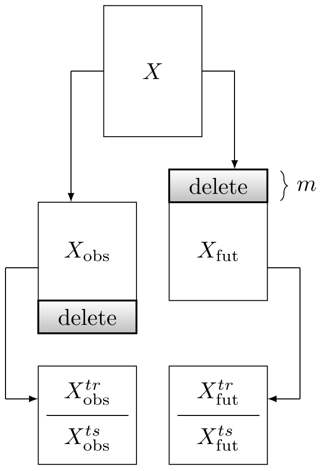

FIGURE 4. X_{obs} contains the observed values on the TS and is obtained removing the last m values from X, while X_{fut} contains the values that must be predicted by the system and is obtained by removing the first m values from X. Both X_{obs} and X_{fut} are split in a training and test set, according with the ordering described in the figure.

accuracy in the prediction decreases. A common case of study is forecasting the load in the next 24 hours. It is noted that forecasting the electricity load for the next 24 hours is an important task in grid operation and planning, primarily in the Smart Grid application field, where intermittent renewable resources such as wind or solar plants are widely integrated in the power system, making such tasks more difficult. Forecasting next-day electricity demand can be useful in unit commitment applications, especially in presence of energy sources characterized by a long start-up time. Since in our case the TS contains data collected every 10 minutes, predicting the values of the following day corresponds to a leading time $m = 24 \times 6 = 144$. Note that 144 corresponds also to the main seasonality in our TS (see Sect. III).

Rather than directly predicting the next m values of X, the STFL problem can be approached by representing the TS in a matrix form $\mathbf{M} \in \mathbb{R}^{n \times m}$, with m columns and n rows, being $n = \lceil \frac{N}{m} \rceil$ the total number of time intervals (days in our case) in X of length m. The t-th row of $\mathbf{M}$ contains the electricity load registered in the t-th day. Then, given a row t, the prediction problem consists in determining the values of the next row $t+1$, which is equivalent to perform a 1-step ahead forecast of a multivariate TS. The prediction of a multivariate TS is a widely studied problem, which is commonly approached using models such as vector autoregressive models or, more generally, vector autoregressive and moving average models [39]. Alternatively, more forecasting models are used, each one dedicated and individually trained on a single variable of the TS, and their outcomes are combined together in order to obtain the final forecast [25]. However, those systems are computationally demanding and they are usually applied when the number of variables in the TS is small. In our case, we have a TS with 144 variables and then a dimensionality reduction is required for representing each row in $\mathbf{M}$ with a lower number k of coefficients. This operation can be done using the PCA decomposition and it is described in detail in the following section.

V. PREDICTION WITH PCA DECOMPOSITION

The Principal Component Analysis (PCA) is the well-known statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The first component accounts for as much of the variability in the data as possible and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. Also known as discrete Karhunen-Loève transform (KLT) in signal processing, the PCA is desirable in applications which require a dimensionality reduction [40]. Linear filtering methods, known in signal processing as "subspace filtering", use the PCA to separate the "signal subspace" from the "noise subspace" in an ensemble of measured signals, i.e. a sampled signal under the hypothesis of uncorrelated additive noise. An important feature of orthogonal transformations is the tendency to redistribute the energy contained in the signal so that most of energy is contained in a small number of components. In the following we discuss the properties of the PCA decomposition and how they can be exploited for approximating the analyzed TS with a lower number of variables.

A. PROPERTIES OF PCA DECOMPOSITION

Given the matrix representation of the TS $\mathbf{M} \in \mathbb{R}^{n \times m}$ with column-wise zero empirical mean, the estimated covariance matrix $\mathbf{S}_M \in \mathbb{R}^{m \times m}$ can be written as:

$$\mathbf{S}_M = \frac{1}{n-1}\mathbf{M}^T\mathbf{M}, \tag{1}$$

that is a square symmetric $m \times m$ matrix where the diagonal terms represent the variance of the column vectors M_j, $j = 1, .., m$ of $\mathbf{M}$, while the off-diagonal terms are the covariance between the column vectors M_i, M_j for $i \neq j$ and $i, j = 1, .., m$. Algebraically, the PCA can be obtained by identifying an orthogonal matrix $\mathbf{C} \in \mathbb{R}^{m \times m}$ (rotation matrix) that diagonalizes the given covariance matrix $\mathbf{S}_M$, whose columns are the eigenvectors and are called *loadings* of the original matrix $\mathbf{M}$. It can be shown that $\mathbf{C}$ originates from the following orthogonal decomposition:

$$\mathbf{A} = \mathbf{C}\mathbf{D}\mathbf{C}^T, \tag{2}$$

where $\mathbf{A} = \mathbf{M}^T\mathbf{M} \in \mathbb{R}^{m \times m}$ is the positive semi-definite scatter matrix. $\mathbf{D} \in \mathbb{R}^{m \times m}$ is a diagonal matrix whose diagonal real-valued elements $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m$, placed in decreasing order, are the eigenvalues of $\mathbf{M}^T\mathbf{M}$. Thus, the diagonal elements are proportional to the variance (up to a constant factor $(n-1)^{-1}$) of principal components in the directions given by the corresponding eigenvectors represented by the columns of $\mathbf{C}$. The matrix $\mathbf{C}$ is an instance of a linear operator that projects $\mathbf{M}$ in a new matrix $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_m] \in \mathbb{R}^{n \times m}$ that lies in a new coordinate system whose components $\beta_j, j = 1, .., m$ are the PC *scores*. Thus, for $\mathbf{M}$ the following equality holds:

$$\mathbf{M} = \boldsymbol{\beta}\mathbf{C}^T. \tag{3}$$

being $\mathbf{C}$ an orthogonal matrix ($\mathbf{C}^T = \mathbf{C}^{-1}$). The score matrix $\boldsymbol{\beta}$ can easily computed as:

$$\boldsymbol{\beta} = \mathbf{MC} \tag{4}$$

Since the eigenvalues of the covariance matrix measure the "generalized variance" in $\mathbf{M}$, an high eigenvalue means that the related PC accounts for more variability in data. Conversely, a low variance accounts for less variability. In general, the latter case indicates the presence of redundancy in the available data, allowing a reduction of the dimensionality. A measure able to catch the explained variance by the PCs is the Cumulative Variance Ratio R_k defined as the ratio of the sum of the first k, $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k$ eigenvalues and the sum of all m eigenvalues, $k \geq m$, of the covariance matrix, that is:

$$R_k = \frac{\sum_{j=1}^{k} \lambda_j}{\sum_{i=1}^{m} \lambda_i}. \tag{5}$$

An approximation of $\mathbf{M}$ using the first k components with the highest variance is defined as:

$$\bar{\mathbf{M}}^{(k)} = \boldsymbol{\beta}^{(k)}\mathbf{C}^{(k)T}, \boldsymbol{\beta}^{(k)} \in \mathbb{R}^{n\times k}, \mathbf{C}^{(k)} \in \mathbb{R}^{m\times k} \tag{6}$$

where $\boldsymbol{\beta}^{(k)} = [\beta_1, \ldots, \beta_k]$ and $\mathbf{C}^{(k)} = [c_1, \ldots, c_k]$ are the first k columns of $\boldsymbol{\beta}$ and $\mathbf{C}$ respectively.

B. DIMENSIONALITY REDUCTION WITH PCA

The t-th row of $\bar{\mathbf{M}}^{(k)}$ can be expressed as $\bar{\mathbf{M}}^{(k)}(t) = \sum_{j=1}^{k} \beta_j(t)c_j^T$, which is an approximation of the t-th period of length m of the original TS. Given the t-th period, represented by the coefficients $\beta_j(t), j = 1, \ldots, k$, the m-step ahead forecasting problem can be solved with the 1-step ahead prediction of the k coefficients, i.e. the values of the period $t+1$ can be approximated as $\sum_{j=1}^{k} \hat{\beta}_j(t+1)c_j^T$, where the coefficients $\hat{\beta}_j(t+1), j = 1, \ldots, k$ are the 1-step ahead predictions. In fact, thanks to the orthogonality of the principal components $\beta_1, \ldots, \beta_k$, they can be considered independent TS which can be predicted using k distinct forecast models, each one trained to forecast a specific principal component β_j.

We apply to $\mathbf{M}$ the same procedure described in Sect. IV: at first, we generate the 2 matrices $\mathbf{M}_{\text{obs}}$ and $\mathbf{M}_{\text{fut}}$ containing the observed and the future values of the TS. Then, with a split, we generate the training ($\mathbf{M}^{tr}_{\text{obs}}$, $\mathbf{M}^{tr}_{\text{fut}}$) and the test set ($\mathbf{M}^{ts}_{\text{obs}}$, $\mathbf{M}^{ts}_{\text{fut}}$).

In order to retrieve the first k principal components in each matrix, we firstly apply the PCA decomposition on $\mathbf{M}^{tr}_{\text{obs}}$ and $\mathbf{M}^{tr}_{\text{fut}}$ which give us the components used for training the prediction system. During the test phase, as soon as a new test element is presented as input to the prediction system, we can approximate the values of its first k principal components, according to the property defined in Eq. 4. Let $\boldsymbol{\beta}^{tr(k)}_{\text{obs}}$, $\mathbf{C}^{tr(k)}_{\text{obs}}$ and $\boldsymbol{\beta}^{tr(k)}_{\text{fut}}$, $\mathbf{C}^{tr(k)}_{\text{fut}}$ be the first k principal components and loadings of $\mathbf{M}^{tr}_{\text{obs}}$ and $\mathbf{M}^{tr}_{\text{fut}}$ respectively. As we observe, given the t-th row $\mathbf{M}^{ts}_{\text{obs}}(t)$ of the test set $\mathbf{M}^{ts}_{\text{obs}}$, the associated values of the k principal components can be approximated as follows:

$$\bar{\boldsymbol{\beta}}^{ts(k)}_{\text{obs}}(t) = \mathbf{M}^{ts}_{\text{obs}}(t)\mathbf{C}^{tr(k)}_{\text{obs}}. \tag{7}$$

where $\bar{\boldsymbol{\beta}}^{ts(k)}_{\text{obs}}(t)$ is an approximation of $\boldsymbol{\beta}^{ts(k)}_{\text{obs}}(t)$, i.e. the t-th value of the k principal components which would result from the PCA decomposition on $\{\mathbf{M}^{tr}_{\text{obs}}, \mathbf{M}^{ts}_{\text{obs}}(1), \ldots, \mathbf{M}^{ts}_{\text{obs}}(t)\}$. Analogously, we can estimate $\bar{\boldsymbol{\beta}}^{ts(k)}_{\text{fut}}(t)$ using $\mathbf{C}^{tr(k)}_{\text{fut}}$.

C. PREDICTING THE k COMPONENTS

Because of the property of orthogonality, each column β_j, $j = 1, \ldots, k$ can be considered a different TS where the n-th value $\beta_j(n)$ depends only on the previous values $\beta_j(t)$, $t = 1, \ldots, n-1$ observed on the same TS. In fact, the orthogonality property ensures every TS to be linearly independent from the others and the final forecast is the combination of results of the single predictions. Each model j is trained using the TS $\beta^{tr}_{\text{obs}_j}$ (which is the j-th column of the matrix β^{tr}_{obs}) as input and the TS $\beta^{tr}_{\text{fut}_j}$ as desired output. For what concerns the prediction of the t-th test element, we retrieve the values $\bar{\boldsymbol{\beta}}^{ts(k)}_{\text{obs}}(t)$ of the k principal components, according to the procedure described in the previous section. Then, for each component $\bar{\beta}^{ts}_{\text{obs}_j}(t)$ we evaluate the m-step ahead forecast $\hat{\beta}^{ts}_{\text{fut}_j}(t)$.

In order to obtain the predicted values of the $t+m$ interval in the original TS, whose ground-truth values are contained in $\mathbf{M}^{ts}_{\text{fut}}(t)$, we use the rotation matrix $\mathbf{C}^{tr(k)}_{\text{fut}}$ from the training set. In particular, the forecast will be the following:

$$\hat{\mathbf{M}}^{ts}_{\text{fut}}(t) = \hat{\boldsymbol{\beta}}^{ts(k)}_{\text{fut}}(t)\mathbf{C}^{tr(k)}_{\text{fut}} \tag{8}$$

Note that as long as more rows of the test set are processed, the approximation error increases, since the orthogonality property among the k components is no longer guaranteed because in the last values of the columns β_j some linear correlations with the other TS $\beta_q, q \neq j$ are introduced. This produces an increasing decay of the accuracy in the forecast, since the prediction relies exclusively on the past values observed on the j-th TS, according to the assumption of orthogonality which is now violated. For this reason, we adopt a mechanism for updating the matrices of the loadings $\mathbf{C}^{tr(k)}_{\text{obs}}$ and $\mathbf{C}^{tr(k)}_{\text{fut}}$ during the forecasting procedure, in order to modify the internal state of the forecast model for taking into account the most recent observations to be considered in the future predictions and for recovering the geometric properties of the principal components. Every p time-intervals we recompute $\mathbf{C}^{tr(k)}_{\text{obs}}$ and $\mathbf{C}^{tr(k)}_{\text{fut}}$ on the new $\mathbf{M}^{tr}_{\text{obs}}$ and $\mathbf{M}^{tr}_{\text{fut}}$, which are updated by inserting the data of the test set that have been observed in the last p intervals. In order to obtain the highest accuracy, p should be selected equal to 1; however every time we recompute the PCA, we obtain a set of completely different k principal components, meaning that we have to retrain every forecast model on the new set of TS; a good compromise between prediction accuracy and computational effort should be chosen.

VI. FORECAST WITH ESN

In this work we consider two different forecasting methods for predicting the electricity load values: the first is a classic ARIMA model that we use for benchmarking purposes, the second is an Echo State Network, whose structure and training procedure is described in this section.

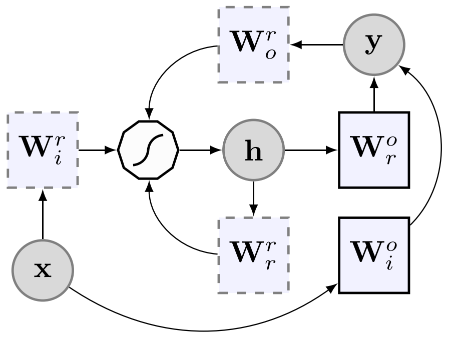

FIGURE 5. Schematic depiction of an ESN. The circles represent the input variables x, the state variables h and the output variables y. The squares depicted with solid lines, $\mathbf{W}_r^o$ and $\mathbf{W}_i^o$, are the trainable weight matrices of the readout, while the squares with dashed lines, $\mathbf{W}_r^r$, $\mathbf{W}_o^r$ and $\mathbf{W}_i^r$, are random initialized weight matrices. The polygon represents the non-linear transformation of the neurons in the network.

A schematic depiction of an ESN is shown in Fig. 5. It is divided in three components, namely the input layer, a recurrent reservoir, and a readout. The current output of an ESN is computed in two distinct phases. First, the N_i-dimensional input vector $\mathbf{x}[n] \in \mathbb{R}^{N_i}$ is given as input to the recurrent reservoir, whose internal state $\mathbf{h}[n-1] \in \mathbb{R}^{N_r}$ is updated according to the state equation:

$$\mathbf{h}[n] = f_{\text{res}}(\mathbf{W}_i^r \mathbf{x}[n] + \mathbf{W}_r^r \mathbf{h}[n-1] + \mathbf{W}_o^r y[n-1]), \quad (9)$$

where $\mathbf{W}_i^r \in \mathbb{R}^{N_r \times N_i}$, $\mathbf{W}_r^r \in \mathbb{R}^{N_r \times N_r}$ and $\mathbf{w}_o^r \in \mathbb{R}^{N_r}$ are randomly initialized at the beginning of the learning process, and they remain unaltered afterward. $f_{\text{res}}(\cdot)$ in Eq. (9) is a suitable non-linear function, typically of sigmoid shape, and $y[n-1] \in \mathbb{R}$ is the previous scalar output of the network. In our case, we have $f_{\text{res}}(\cdot) = \tanh(\cdot)$. In the second phase, the ESN's prediction is computed according to:

$$y[n] = \left(\mathbf{W}_i^o\right)^T \mathbf{x}[n] + \left(\mathbf{W}_r^o\right)^T \mathbf{h}[n], \quad (10)$$

where $\mathbf{W}_i^o \in \mathbb{R}^{N_i}$, $\mathbf{W}_r^o \in \mathbb{R}^{N_r}$ are trainable connections. The difference between fixed and adaptable weight matrices is shown in Fig. 5 with the use of continuous and dashed lines, respectively. Additionally, to increase the overall stability, it is possible to insert a small uniform noise term to the state update in Eq. (9), before computing the non-linear transformation $f_{\text{res}}(\cdot)$ [41].

A few words should be spent on the choice of the matrix $\mathbf{W}_r^r$. According to the ESN theory, the reservoir must satisfies the so-called 'echo state property' (ESP) [42]. This means that the effect of a given input on the state of the reservoir must vanish in a finite number of time-instants. A widely used rule-of-thumb is to rescale the matrix $\mathbf{W}_r^r$ to have $\rho(\mathbf{W}_r^r) < 1$, where $\rho(\cdot)$ denotes the spectral radius operator. We use this strategy and we refer the interested reader to [43] for recent theoretical studies on the subject. If the ESP is satisfied, an ESN with a suitably large reservoir can approximate any non-linear filter with bounded memory to any given level of accuracy [42].

To determine the weight matrices in the readout, let us consider a sequence of Q desired input-outputs pairs given by:

$$(\mathbf{x}[1], d[1]) \ldots, (\mathbf{x}[Q], d[Q]) \quad (11)$$

In our case, the input vector is the original time-series X or the i-th principal component β_i, while the output is given by:

$$d[t] = x[t+m], \quad (12)$$

where m define the forecast horizon. In the initial phase of training, called 'state harvesting', the inputs are fed to the reservoir in accordance with Eq. (9), producing a sequence of internal states $\mathbf{h}[1], \ldots, \mathbf{h}[Q]$. Since, by definition, the outputs of the ESN are not available for feedback, the desired output is used instead in Eq. (10) (so-called 'teacher forcing'). The states are stacked in a matrix $\mathbf{H} \in \mathbb{R}^{Q \times N_i + N_r}$ and the desired outputs in a vector $\mathbf{d} \in \mathbb{R}^Q$:

$$\mathbf{H} = \begin{bmatrix} \mathbf{x}^T[1], & \mathbf{h}^T[1] \\ & \vdots \\ \mathbf{x}^T[Q], & \mathbf{h}^T[Q] \end{bmatrix}, \quad (13)$$

$$\mathbf{d} = \begin{bmatrix} d[1] \\ \vdots \\ d[Q] \end{bmatrix}. \quad (14)$$

The initial D rows from Eq. (13) and Eq. (14) should be discarded, since they refer to a transient phase in the ESN's behavior. We refer to them as the *dropout* (or washout) elements.[1]

At this point the resulting training problem is a standard linear regression, which can be solved in a large variety of ways. We used the least-square regression (LSR), which is the algorithm originally proposed for training the readout [44]. It consists in the following regularized least-square problem:

$$\mathbf{w}_{\text{ls}}^* = \underset{\mathbf{w} \in \mathbb{R}^{N_i+N_r}}{\arg\min} \ \frac{1}{2} \|\mathbf{H}\mathbf{w} - \mathbf{d}\|_2^2 + \frac{\alpha}{2} \|\mathbf{w}\|_2^2, \quad (15)$$

where $\mathbf{w}_{\text{ls}} = \left[\mathbf{w}_i^o \ \mathbf{w}_r^o\right]^T$ and $\alpha \in \mathbb{R}^+$ is a positive scalar known as *regularization factor*. A solution of problem (15) can be obtained in closed form as:

$$\mathbf{w}_{\text{ls}}^* = \left(\mathbf{H}^T \mathbf{H} + \alpha \mathbf{I}\right)^{-1} \mathbf{H}^T \mathbf{d}. \quad (16)$$

[1]Not to be confused with the dropout regularization currently in use in the deep learning literature.

Whenever $N_r + N_i > Q$, Eq. (16) can be computed more efficiently by rewriting it as:

$$\mathbf{w}^*_{\text{ls}} = \mathbf{H}^T \left(\mathbf{H}\mathbf{H}^T + \alpha \mathbf{I}\right)^{-1} \mathbf{d}. \tag{17}$$

A. PARAMETERS OPTIMIZATION

The initial configuration of the ESN depends on a set of parameters which need to be properly tuned to achieve the best performances in the task considered. An effective approach consists in using a genetic algorithm (GA) to retrieve an optimal parameter configuration [9]. Since the variables which must be tuned assume both integer and real values, we opted for a mix-integer GA based on MI-LXPM algorithm [45], which optimizes a genetic code composed of real and integer values, defined in a suitable interval. The algorithm follows the standard GA procedure, which begins by creating a random initial population. Then, in each iteration the GA generates a new population using some of the individuals with high fitness in the current generation, called parents, which are chosen using a tournament selection with tournament size equal to 2. Individuals in the next generation are generated either by making random changes to a single parent with a *Gaussian mutation*, which adds a random number taken from a Gaussian distribution with 0 mean to each entry of the parent vector, or by combining parent entries with a *Laplacian crossover* [45], with crossover fraction μ. The algorithm also implements elitism, moving the E individuals with the highest fitness to the next generation, where $E = \lceil \rho P \rceil$, being ρ and P elitism rate and population size respectively. The stop criterion triggers when: (i) a maximum number G of generations is reached; (ii) the average relative change in the value of the fitness function over a number of consecutive generations is below a given threshold τ_{stall}; (iii) the fitness value of the best individual in the current population is less than or equal to the fitness limit – usually 0 (or 1), if the normalized fitness score is minimized (or maximized).

Previous uses of GA for optimizing ESN parameters [9], [46], [47] have shown, in most cases, that the achieved performances are comparable to an exhaustive search and that the ESN itself is robust to small variations in parameter values. We considered the recommendations reported in previous works, e.g. [48], as guidelines to determine the search space of the genetic code. In the following we define the search space for each parameter and the optional resolution of the search.

- For the number of neurons in the reservoir we selected the bounds [100, 1000], with a resolution of 100.
- The weights in $\mathbf{W}^r_r$ are extracted from an uniform distribution in $[-1, +1]$. Then, a given percentage p of values are set to 0, and the matrix is rescaled to obtain a desired spectral radius ρ^*. ρ^* is optimized in [0.5, 0.99], while the percentage p of the connectivity in the reservoir in [0.1, 0, 4].
- The small noise term in Eq. (9) is drawn from a Gaussian distribution with zero mean and variance β^2, which is searched in [0.00001, 0.001].
- To provide additional flexibility, input signal, desired response, and feedback signal are all scaled by three constant factors, namely γ_{in}, γ_{out}, and γ_{feed}. The first two are optimized in [0.1, 1], while the third in [0, 1]. The particular case of $\gamma_{\text{feed}} = 0$ represents an ESN without feedback.
- The search of the parameter regularization α used in the linear regression of the readout training, takes place in the codomain of the exponential function $\{2^c\}$, where the value of c is searched by the GA in $[-10, 10]$ with resolution 1.

In the optimization procedure, the fitness of the genetic code is evaluated using the forecast accuracy obtained on a validation set X^{vs}_{obs} and X^{vs}_{fut}, which is generated by splitting the training set in 2 parts: the first part becomes the new training set, the remaining part the validation set. If the TS is represented in the matrix form $\mathbf{M}$, the coefficients $\boldsymbol{\beta}^{vs}_{\text{obs}}$ and $\boldsymbol{\beta}^{vs}_{\text{fut}}$ are approximated with $\bar{\boldsymbol{\beta}}^{vs}_{\text{obs}}$ and $\bar{\boldsymbol{\beta}}^{vs}_{\text{fut}}$, analogously to $\bar{\boldsymbol{\beta}}^{ts}_{\text{obs}}$ and $\bar{\boldsymbol{\beta}}^{ts}_{\text{fut}}$, as described in Sect. V-B.

Each genetic code c_j defines a network ESN$_j$ and we use $\boldsymbol{\beta}^{tr}_{\text{obs}}$ and $\boldsymbol{\beta}^{tr}_{\text{fut}}$ as input and teacher signal for training the readout. Successively, at each time interval t we present $\bar{\boldsymbol{\beta}}^{vs}_{\text{obs}}(t)$ as input to ESN$_j$ and we compare the predicted values $\hat{\boldsymbol{\beta}}^{vs}_{\text{fut}}(t)$ with the true values $\bar{\boldsymbol{\beta}}^{vs}_{\text{fut}}(t)$. The fitness of the genetic code c_j is evaluated according to the prediction error; once the optimal genetic code c_{opt} is identified, the associated ESN$_{\text{opt}}$ is used for forecasting the values of the test set, as described in Sect. V-C, and the prediction error on the test set represents the generalization capability of the system. A schema of the whole procedure is depicted in Fig. 6.

VII. EXPERIMENTS

The TS contains 137376 values representing the electric load registered on the feeder named "Belsito Prisciano" every 10 minutes during approximately 3 years of activity. Accordingly to the splitting procedure described in Sect. IV, we used the first 80% of the TS as the training set. Of the remaining part the first 15% become the validation set and the last 5% the test set.

In a problem of a m-step ahead prediction, a TS X is serially processed by the system which, for each component $X(t)$, returns the predicted value $\hat{X}(t+m)$. Many different techniques can be used for evaluating the prediction accuracy [1], [49]. The error measurement that we adopted is the Normalized Root Mean Squared Error (NRMSE) function, which is a frequently used error measure that represents the sample standard deviation of the differences between predicted values and observed values. It is defined as follows:

$$\text{NRMSE}(\hat{X}, X) = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{x}_i - x_i)^2}}{x_{\max} - x_{\min}} \tag{18}$$

being $\hat{x}_i$ the i-th prediction and x_i the observed value.

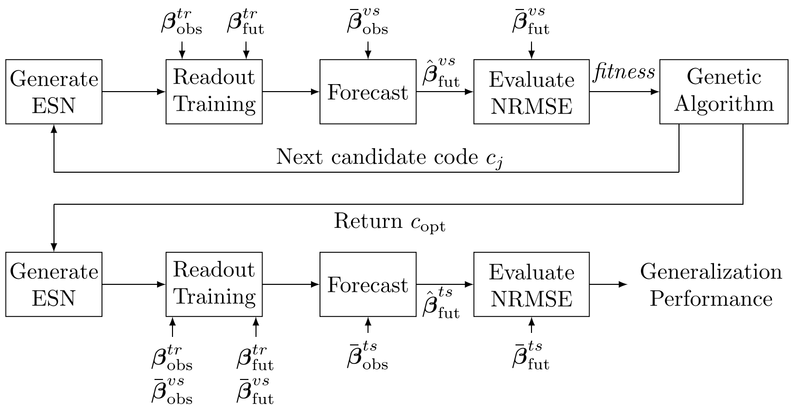
Generate ESN
Readout Training
Forecast
Evaluate NRMSE
fitness
Genetic Algorithm
Next candidate code c_j
Return c_{opt}
Generalization Performance
β^{tr}_{obs}
β^{tr}_{fut}
$\bar{\beta}^{vs}_{\text{obs}}$
$\bar{\beta}^{vs}_{\text{fut}}$
$\hat{\beta}^{vs}_{\text{fut}}$
$\hat{\beta}^{ts}_{\text{fut}}$
$\bar{\beta}^{ts}_{\text{obs}}$
$\bar{\beta}^{ts}_{\text{fut}}$

FIGURE 6. The optimization procedure performed on the training and validation set using the GA. Once the optimal parameters are identified, the optimal configuration of the ESN is used for predicting the values in the test set: the prediction accuracy describes the generalization capability of the system.

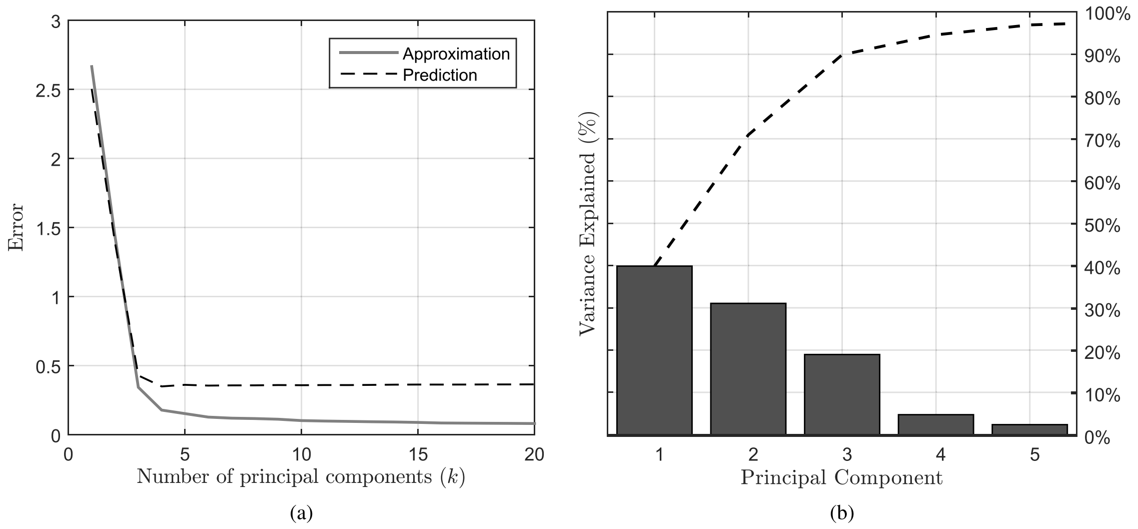

FIGURE 7. In (a) we report the approximation error and the prediction error (both evaluated with the NRMSE measure) when the TS X is approximated using a different number k of principal components. In (b) we report the Pareto chart which reports the percentage of explained variance in the data, considering a different number k of components. As we can see, by using a number of components $k = 5$ we can obtain good performances, lowering significantly the original number of variables, which are 144.

For training the ESN we optimized the network parameters using the GA, as described in Sect. VI-A, configured with a crossover rate $\mu = 0.8$, a population size $P = 50$, a maximum number of generations $G = 100$, a stop threshold $\tau_{\text{stall}} = 0.01$ and an elitism rate $\rho = 0.05$. During the optimization step the prediction accuracy is evaluated on the validation set using the NRMSE function, which represents the reciprocal of the fitness for each chromosome.

In order to select the optimal number of principal components k, we analyzed the approximation error of the original TS and the prediction error as k varies – in this case the prediction error is evaluated directly on X using an ARIMA model and considering a forecast step of 50 time-intervals. As we can see from the graphics in Fig. 7a, when X is reconstructed using more than 5 principal components, both the approximation and the prediction errors start to decrease very slowly. This means that if we consider a number of components $k > 5$, we are adding complexity to the system (a new forecast model is required for any additional principal component) without obtaining a significant decrement of the approximation and prediction error. In the Pareto chart in Fig. 7b we report the explained variance of the first principal components which, according to Eq. 5 in the case of $k = 5$ is:

$$R_5 = \frac{\sum_{j=1}^{5} \lambda_j}{\sum_{i=1}^{144} \lambda_i} = 0.946. \tag{19}$$

As in the previous case, we can see that the first 5 components are sufficient to explain almost all the variance in the data. Thus, in the following experiments we always consider $k = 5$ principal components for approximating $\mathbf{M}$, the matrix representation of X.

We compared the results obtained with the ESN based prediction system using an ARIMA model. We refer to an ARIMA model using the standard notation ARIMA$(\mathbf{p}, \mathbf{d}, \mathbf{q}) \times (\mathbf{P}, \mathbf{D}, \mathbf{Q})_s$, where the term $(\mathbf{p}, \mathbf{d}, \mathbf{q})$ gives the order of the nonseasonal part, $(\mathbf{P}, \mathbf{D}, \mathbf{Q})$ the order of the seasonal part, being s the value of the seasonality [23]. For the direct prediction of the original TS X, we generated an ARIMA model according to a preemptive analysis, whose results are partially reported in Sect. III. The non-stationary TS can be forecast using a moving average term and including both seasonal and non-seasonal first order differencing, i.e. a ARIMA$(0, 1, 1) \times (0, 1, 0)_{144}$ model. For what concerns the prediction of the coefficients $\beta_1, .., \beta_k$, they are non-stationary TS without a defined seasonal pattern and they can be modeled using a single autoregressive term and a first order differencing component for achieving stationarity: we processed the k components returned by the PCA decomposition using k ARIMA(1,1,0) models. Like for the ESN, the parameters of the models are evaluated on the training and the validation set using a standard optimization routine.

The code is fully written in MATLAB. The ESN is implemented using a customized version of the Simple ESN toolbox [50] which can be found at https://bitbucket.org/ispamm/distributed-esn, while for the prediction with ARIMA we used the implementation belonging to the software in the MATLAB financial toolbox.

A. RESULTS

In this section we evaluate the results obtained relatively to the problems of STLF considering two forecast horizons of 1 and 144 step ahead. Because of the random nature of the GA used in the optimization and of the ESN, whose internal reservoir structure is generated randomly when the network is generated, we repeated the initialization, the optimization, the training of the network and the final test 10 different times. Additionally, because of the initial transient phase of the ESN, we discard the first 50 output values as dropout elements, in order to let the system reach a steady behavior.

TABLE 1. Forecast errors evaluated with the NRMSE function on the load of 50 days, using ARIMA and ESN. There are two different leading times in the prediction, which are 1 and 144. The 144-step ahead forecast is done directly on the original time-series, on the *k* components returned by the PCA decomposition with and without updating the loadings.

		ARIMA	ESN
	Lead Time	Forecast Error	Forecast Error
original TS	1	0.1429	0.1164 ± 0.0041
original TS	144	0.5104	0.3254 ± 0.0083
PCA	144	0.3557	0.3015 ± 0.0065
PCA + update	144	0.3043	0.2524 ± 0.005

In Tab. 1 we report the average forecasting accuracy evaluated with the error function defined in Eq. 18 relatively to the two STF problems. Concerning the 144-step ahead forecast case, we present the prediction results of ESN and ARIMA models directly on the TS and also the results obtained combining the k different 1-step ahead predictions of the principal components returned by the PCA. We also show the forecasting error obtained updating the PCA coefficients, as described in Sect. V-C every $p = 5$ days. All the errors concerns the prediction of the electricity load in the 50 days of the test set, for a total of $144 \cdot 50 = 7200$ predictions.

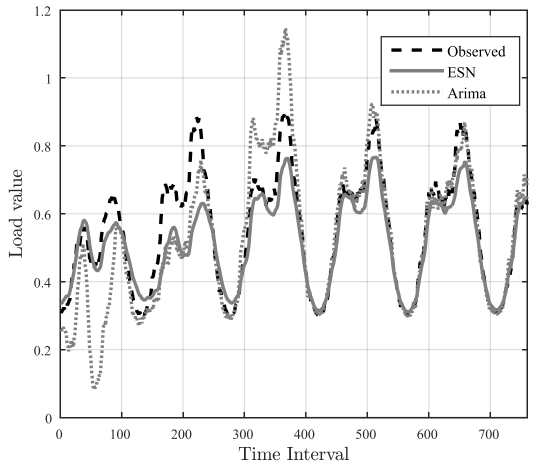

FIGURE 8. Predicted values using ESN and ARIMA, with a forecast horizon of 144 time-intervals.

In Fig. 8 we plot the values predicted by ESN and ARIMA along with the ground-truth values of the TS relatively to the 144-step ahead forecast case. In Fig. 9 instead, the same values are predicted combining the k 1-step ahead forecasts $\hat{\beta}_1, \ldots, \hat{\beta}_k$, related to the components returned by the PCA decomposition of the matrix representation $\mathbf{M}$ of the original time-series. The loadings of the PCA are update every $p = 5$ days, in order to better keep track of the last observed modifications.

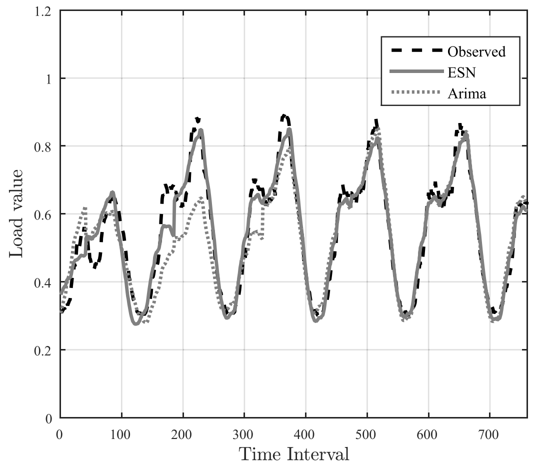

FIGURE 9. The forecast returned by ESN and ARIMA is relative to a leading time of 144 time-intervals. This time the result is obtained combining the *k* different 1-step ahead predictions of the PCA components and applying the updating procedure on the coefficients every 5 time intervals.

As expected, we can see that the best prediction accuracy is obtained relatively to the problem of 1-step ahead forecast which, due to its simplicity, can be computed effectively on the original TS X. However, when we consider the longer leading time of 144 time-intervals ahead, the forecast accuracy obtained directly predicting the next 144 values on X decreases substantially. The same prediction task is approached with the PCA decomposition technique proposed in Sect. V, which permits to greatly increment the prediction performance on the same problem. The prediction capability can be further improved by updating the PCA coefficients as long as new elements of the TS are observed. Even if representing the original TS only with the first k coefficients of the PCA decomposition, introducing an approximation error, the characterizing features of the TS are mostly preserved while the noise component is mostly removed. Since a prediction system is in any case unable to model and predict a random noise affecting the signal, removing it from the original TS eases the forecasting task and the overfitting phenomenon during training is reduced.

In every prediction task the ESN achieves better performances than the ARIMA model, which is a traditional method that often encounters difficulty in properly modeling a complex non-linearity in the TS. Conversely, the hyperbolic tangent transfer functions of the neurons of the reservoir are able to map the input signal in a high dimensional nonlinear space. Thus, methods based on a ESN model have the ability to learn or map those nonlinear relationships so that, if they are properly trained on well suited preprocessed data, the accuracy of the predicted results can be very high. Additionally, the ESN is particularly effective in short term predictions and thus, by reducing the problem to k 1-step ahead predictions, we obtained the ideal application scenario of the ESN model. The results obtained with the ESN are very stable as we can see from the standard deviation in NRMSE (reported in Tab. 1); the ESN prediction accuracy achieved in the 10 different runs is very similar, even if the network is initialized every time with a different random seed (which generates a different internal structure of the reservoir) and the model is trained with a genetic algorithm, which has a stochastic behavior as well.

VIII. CONCLUSIONS AND FUTURE WORKS

In this paper we analyzed a real-world time-series of electric load measured upstream of a MV feeder within the power grid of Rome, Italy. The TS has a well defined seasonality of 144 time intervals and we considered two different Short Term Load Forecasting (STLF) problems, the first with a leading time $m = 1$ and the second with $m = 144$. Relatively to this second case of study, we proposed a method for improving the prediction accuracy, consisting in representing the TS in a matrix form, where each row has m components, each one storing the daily electric load value sampled every 10 minutes. Given the t-th row of the matrix, the m step ahead forecast consists in predicting the value of the next row $t + 1$. In order to ease the prediction task, we retrieve from the matrix the first k principal components using a PCA decomposition and the related rotation matrix. Due to the orthogonality of the components, they can be forecast independently with k different predictors. In this way, the problem of forecasting m steps ahead can be decomposed in k simpler 1-step ahead prediction tasks and the final result is obtained by combining together the solutions found. We described how the values of the principal components can be obtained and updated during the prediction phase. As forecast model we used k different ESNs, one for each of the selected principal components and we determined the optimal set of parameters using a GA.

The method that we propose in this paper can be applied to a multitude of TS, however it is advised to take into account a TS with a sufficiently long seasonality, in order to effectively exploit the reduction of the dimensionality. Furthermore, the total number of time-steps in the original TS X must be high enough. In fact, the length of the multi-variate TS generated by the matrix representation of X, is reduced by m times (being m the length of the seasonality in X) and it should be sufficiently long for a meaningful analysis.

We tested our system by predicting the electricity load values in 50 days, considering two different leading times of 10 minutes and 1 day respectively. For the 1-day ahead prediction we performed a direct prediction on the original TS and we showed how the prediction accuracy can be improved with the alternative method, based on the matrix representation and PCA decomposition. We also considered the procedure for updating the coefficients of the PCA decomposition during the prediction, in order to better keep track of the evolution of the TS, as new values are observed, which demonstrated to significantly increase the prediction accuracy. We compared the performances obtained with a standard ARIMA system and we showed how the ESN can outperform ARIMA in every prediction task.

In this work we used a basic ESN model, which could be improved or modified in future, considering the most recent methodologies for training the readout or for generating the reservoir in a more effective way. Additionally, we plan to explore additional types of ANN for the prediction, like Adaptive NeuroFuzzy Inference Systems (ANFIS) or SLTM deep neural networks, which are gaining a lot of attention recently.

REFERENCES

[1] J. G. De Gooijer and R. J. Hyndman, "25 years of time series forecasting," *Int. J. Forecasting*, vol. 22, no. 3, pp. 443–473, 2006.

[2] D. W. Bunn, "Forecasting loads and prices in competitive power markets," *Proc. IEEE*, vol. 88, no. 2, pp. 163–169, Feb. 2000.

[3] P. A. Ruiz and G. Gross, "Short-term resource adequacy in electricity market design," *IEEE Trans. Power Syst.*, vol. 23, no. 3, pp. 916–926, Aug. 2008.

[4] H.-Y. Ding, W.-J. Pei, and Z.-Y. He, "A multiple objective optimization based echo state network tree and application to intrusion detection," in *Proc. IEEE Int. Workshop VLSI Design Video Technol.*, May 2005, pp. 443–446.

[5] S. I. Han and J. M. Lee, "Fuzzy echo state neural networks and funnel dynamic surface control for prescribed performance of a non-linear dynamic system," *IEEE Trans. Ind. Electron.*, vol. 61, no. 2, pp. 1099–1112, Feb. 2014.

[6] J. Bozsik and Z. Ilonczai, "Echo state network-based credit rating system," in *Proc. 4th IEEE Int. Symp. Logistics Ind. Inform. (LINDI)*, Sep. 2012, pp. 185–190.

[7] J. Mazumdar and R. G. Harley, "Utilization of echo state networks for differentiating source and nonlinear load harmonics in the utility network," *IEEE Trans. Power Electron.*, vol. 23, no. 6, pp. 2738–2745, Nov. 2008.

[8] M. D. Skowronski and J. G. Harris, "Automatic speech recognition using a predictive echo state network classifier," *Neural Netw.*, vol. 20, no. 3, pp. 414–423, 2007.
[9] A. Deihimi and H. Showkati, "Application of echo state networks in short-term electric load forecasting," *Energy*, vol. 39, no. 1, pp. 327–340, 2012.
[10] A. Deihimi, O. Orang, and H. Showkati, "Short-term electric load and temperature forecasting using wavelet echo state networks with neural reconstruction," *Energy*, vol. 57, pp. 382–401, Aug. 2013.
[11] Y. Peng, M. Lei, J.-B. Li, and X.-Y. Peng, "A novel hybridization of echo state networks and multiplicative seasonal ARIMA model for mobile communication traffic series forecasting," *Neural Comput. Appl.*, vol. 24, nos. 3–4, pp. 883–890, 2014.
[12] J. W. Taylor, "A comparison of univariate time series methods for forecasting intraday arrivals at a call center," *Manage. Sci.*, vol. 54, no. 2, pp. 253–265, 2008.
[13] Y.-H. Hsiao, "Household electricity demand forecast based on context information and user daily schedule analysis from meter data," *IEEE Trans. Ind. Informat.*, vol. 11, no. 1, pp. 33–43, Feb. 2015.
[14] C. E. Borges, Y. K. Penya, and I. Fernandez, "Evaluating combined load forecasting in large power systems and smart grids," *IEEE Trans. Ind. Informat.*, vol. 9, no. 3, pp. 1570–1577, Aug. 2013.
[15] E. Terciyanli, T. Demirci, D. Kucuk, M. Sarac, I. Cadirci, and M. Ermis, "Enhanced nationwide wind-electric power monitoring and forecast system," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1171–1184, May 2014.
[16] D. De Silva, X. Yu, D. Alahakoon, and G. Holmes, "A data mining framework for electricity consumption analysis from meter data," *IEEE Trans. Ind. Informat.*, vol. 7, no. 3, pp. 399–407, Aug. 2011.
[17] A. Khosravi and S. Nahavandi, "Load forecasting using interval type-2 fuzzy logic systems: Optimal type reduction," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1055–1063, May 2014.
[18] G. P. Zhang, "An investigation of neural networks for linear time-series forecasting," *Comput. Oper. Res.*, vol. 28, no. 12, pp. 1183–1202, 2001.
[19] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks: The state of the art," *Int. J. Forecasting*, vol. 14, no. 1, pp. 35–62, 1998.
[20] M. Ghiassi, H. Saidane, and D. K. Zimbra, "A dynamic artificial neural network model for forecasting time series events," *Int. J. Forecasting*, vol. 21, no. 2, pp. 341–362, 2005.
[21] Z. Shi and M. Han, "Support vector echo-state machine for chaotic time-series prediction," *IEEE Trans. Neural Netw.*, vol. 18, no. 2, pp. 359–372, Mar. 2007.
[22] D. Li, M. Han, and J. Wang, "Chaotic time series prediction based on a novel robust echo state network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 5, pp. 787–799, May 2012.
[23] S. L. Ho, M. Xie, and T. N. Goh, "A comparative study of neural network and Box–Jenkins ARIMA modeling in time series prediction," *Comput. Ind. Eng.*, vol. 42, nos. 2–4, pp. 371–375, 2002.
[24] S. Varshney and T. Verma, "Half hourly electricity load prediction using echo state network," *Int. J. Sci. Res.*, vol. 3, no. 6, pp. 885–888, 2014.
[25] D. Niu, L. Ji, M. Xing, and J. Wang, "Multi-variable echo state network optimized by Bayesian regulation for daily peak load forecasting," *J. Netw.*, vol. 7, no. 11, pp. 1790–1795, 2012.
[26] S. Scardapane, D. Wang, and M. Panella, "A decentralized training algorithm for echo state networks in distributed big data applications," *Neural Netw.*, Aug. 2015.
[27] J. W. Taylor, "Exponentially weighted methods for forecasting intraday time series with multiple seasonal cycles," *Int. J. Forecasting*, vol. 26, no. 4, pp. 627–646, 2010.
[28] J. W. Taylor, "Short-term load forecasting with exponentially weighted methods," *IEEE Trans. Power Syst.*, vol. 27, no. 1, pp. 458–464, Feb. 2012.
[29] H. Shen and J. Z. Huang, "Interday forecasting and intraday updating of call center arrivals," *Manuf. Service Oper. Manage.*, vol. 10, no. 3, pp. 391–410, 2008.
[30] J. W. Taylor, L. M. de Menezes, and P. E. McSharry, "A comparison of univariate methods for forecasting electricity demand up to a day ahead," *Int. J. Forecasting*, vol. 22, no. 1, pp. 1–16, 2006.
[31] J. W. Taylor and P. E. McSharry, "Short-term load forecasting methods: An evaluation based on European data," *IEEE Trans. Power Syst.*, vol. 22, no. 4, pp. 2213–2219, Nov. 2007.
[32] E. De Santis, L. Livi, A. Sadeghian, and A. Rizzi, "Modeling and recognition of smart grid faults by a combined approach of dissimilarity learning and one-class classification," *Neurocomputing*, vol. 170, pp. 368–383, Dec. 2015.
[33] J. Weinberg, L. D. Brown, and J. R. Stroud, "Bayesian forecasting of an inhomogeneous Poisson process with applications to call center data," *J. Amer. Statist. Assoc.*, vol. 102, no. 480, pp. 1185–1198, 2007.
[34] R. Ibrahim and P. L'Ecuyer, "Forecasting call center arrivals: Fixed-effects, mixed-effects, and bivariate models," *Manuf. Service Oper. Manage.*, vol. 15, no. 1, pp. 72–85, 2013.
[35] B. H. Andrews and S. M. Cunningham, "L. L. Bean improves call-center forecasting," *Interfaces*, vol. 25, no. 6, pp. 1–13, 1995.
[36] J. W. Taylor, "Short-term electricity demand forecasting using double seasonal exponential smoothing," *J. Oper. Res. Soc.*, vol. 54, no. 8, pp. 799–805, 2003.
[37] P. H. Franses, "Seasonality, non-stationarity and the forecasting of monthly time series," *Int. J. Forecasting*, vol. 7, no. 2, pp. 199–208, 1991.
[38] A. N. Avramidis, A. Deslauriers, and P. L'Ecuyer, "Modeling daily arrivals to a telephone call center," *Manage. Sci.*, vol. 50, no. 7, pp. 896–908, 2004.
[39] G. C. Reinsel, *Elements of Multivariate Time Series Analysis*. New York, NY, USA: Springer-Verlag, 2003.
[40] I. T. Jolliffe, *Principal Component Analysis*. New York, NY, USA: Springer-Verlag, 1986.
[41] H. Jaeger, "Adaptive nonlinear system identification with echo state networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 593–600.
[42] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Comput. Sci. Rev.*, vol. 3, no. 3, pp. 127–149, 2009.
[43] I. B. Yildiz, H. Jaeger, and S. J. Kiebel, "Re-visiting the echo state property," *Neural Netw.*, vol. 35, pp. 1–9, Nov. 2012.
[44] H. Jaeger, "The 'echo state' approach to analysing and training recurrent neural networks-with an erratum note," German Nat. Res. Center Inf. Technol., Sankt Augustin, Germany, Tech. Rep. GMD 148, 2001, p. 34.
[45] K. Deep, K. P. Singh, M. L. Kansal, and C. Mohan, "A real coded genetic algorithm for solving integer and mixed integer optimization problems," *Appl. Math. Comput.*, vol. 212, no. 2, pp. 505–518, 2009.
[46] F. Jiang, H. Berry, and M. Schoenauer, "Supervised and evolutionary learning of echo state networks," in *Parallel Problem Solving From Nature*. Berlin, Germany: Springer-Verlag, 2008, pp. 215–224.
[47] D. Xu, J. Lan, and J. C. Principe, "Direct adaptive control: An echo state network and genetic algorithm approach," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, vol. 3. Jul./Aug. 2005, pp. 1483–1486.
[48] G. K. Venayagamoorthy and B. Shishir, "Effects of spectral radius and settling time in the performance of echo state networks," *Neural Netw.*, vol. 22, no. 7, pp. 861–863, 2009.
[49] E. Mahmoud, "Accuracy in forecasting: A survey," *J. Forecasting*, vol. 3, no. 2, pp. 139–159, 1984.
[50] H. Jaeger. (2009). *Simple and Very Simple MATLAB Toolbox for Echo State Networks*. [Online]. Available: http://organic.elis.ugent.be/node/129

FILIPPO MARIA BIANCHI received the M.Sc. (Hons.) degree in artificial intelligence and robotics from the Sapienza University of Rome, in 2012, where he is currently pursuing the Ph.D. degree. He is a Computer Engineer. He is also a Research Assistant with the Department of Computer Science, Ryerson University, Toronto. His main research topics are pattern recognition, time-series forecasting, neural networks, data mining, and soft computing with a focus on the design of supervised and unsupervised data driven modeling techniques, such as clustering and classification systems.

ENRICO DE SANTIS received the M.Sc. (Hons.) degree from the Information Engineering, Electronics and Telecommunications Department, Sapienza University of Rome, Italy, in 2012, where he is currently pursuing the Ph.D. degree. He is a Communications Engineer. He is also a Research Assistant with the Department of Computer Science, Ryerson University, Toronto, Canada. His main research interests are focused on artificial intelligence and computational intelligence methodologies for data science and complex systems, with a particular attention to industrial applications such as smart grids.

ANTONELLO RIZZI (M'98–SM'04) received the Ph.D. degree in information and communication engineering from the Sapienza University of Rome, in 2000. In 2000, he joined the Information and Communication Department, Sapienza University of Rome, as an Assistant Professor. Since 2010, he has been with the Information Engineering, Electronics and Telecommunications Department, Sapienza University of Rome. His research activity concerns the design of automatic modeling systems, focusing on classification, clustering, function approximation, and prediction problems. Since 2008, he has served as the Scientific Coordinator and Technical Director of research and development activities with the Intelligent Systems Laboratory, Research and Technology Transfer Center for Sustainable Mobility of Lazio Region. He is currently involved in different research topics and projects, such as smart grids and microgrids modeling and control, intelligent systems for sustainable mobility, energy storage systems modeling and control, predictive diagnostic systems for condition-based maintenance in smart grids, granular computing, data mining systems in industrial applications, graph and sequence matching, and agent-based clustering. He has co-authored over 100 journal/conference papers and book chapters. His major research interests are in the area of soft computing, pattern recognition, and computational intelligence, including supervised and unsupervised data driven modeling techniques, neural networks, fuzzy systems, and evolutionary algorithms.

ALIREZA SADEGHIAN (S'94–M'00–SM'06) received the B.A.Sc. (Hons.) degree in electrical and computer engineering from Tehran Polytechnic University, Tehran, Iran, and the M.A.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada. He is currently a Professor with the Department of Computer Science, Ryerson University. He has authored over 100 technical papers. His research interests include computational intelligence, neural networks, fuzzy sets of higher order, and data-driven modeling. He is an Associate Editor of the IEEE Access and the *Expert Systems Journal*, and serves on the Editorial Board of *Applied Soft Computing*. He is also actively involved in a number of professional and academic boards, including the North American Fuzzy Information Processing Society Board, and the IEEE Toronto Section Executive Committee.