Soft Computing (2023) 27:12943–12957
https://doi.org/10.1007/s00500-022-07055-2

FOCUS

Comparing deep and shallow neural networks in forecasting call center arrivals

Andrea Manno[1] · Fabrizio Rossi[1] · Stefano Smriglio[1] · Luigi Cerone[1]

Accepted: 7 March 2022 / Published online: 25 April 2022

Abstract

Forecasting volumes of incoming calls is the first step of the workforce planning process in call centers and represents a prominent issue from both research and industry perspectives. We investigate the application of Neural Networks to predict incoming calls 24 hours ahead. In particular, a Machine Learning deep architecture known as Echo State Network, is compared with a completely different rolling horizon shallow Neural Network strategy, in which the lack of recurrent connections is compensated by a careful input selection. The comparison, carried out on three different real world datasets, reveals better predictive performance for the shallow approach. The latter appears also more robust and less demanding, reducing the inference time by a factor of 2.5 to 4.5 compared to Echo State Networks.

Keywords Call center arrivals · Time series forecast · Machine learning · Artificial neural networks · Echo state networks

1 Introduction

Call centers represent major components for today's organizations, with a huge market size (Statista Research Department 2021) and millions of employees worldwide. The set of activities that a company carries out to optimize employee productivity is called *Workforce management* (WFM). WFM plays a key role in call center industry, as labour costs typically comprise the largest portion of the total operating budget. Furthermore, it offers a surprisingly rich application context for several operations management methodologies, such as forecasting, capacity planning, queueing and personnel scheduling. In fact, there is a huge scientific literature concerning with WFM in call centers and we refer the reader to surveys by Koole and Li (2021), Aksin et al. (2007) and Gans et al. (2003). The general goal of call centers WFM is to find a satisfactory trade-off between the *Level of Service* (LoS) provided to customers and personnel costs.

This is a complex task due to several sources of uncertainty, such as intra-day/intra-week seasonality and variability of the actual number of employees. These fluctuations cannot be managed at operational (real-time) level unless some (expensive) overcapacity has been introduced at planning level. Therefore, the first fundamental step of a successful WFM consists in making an accurate forecast of the highly uncertain incoming/outgoing call volumes.

The prediction of the incoming/outcoming call volumes in a given time horizon (for instance, one day), subdivided into equal time intervals (say one hour) can be naturally modeled as a *Time Series Forecast* problem (Box et al. 2015). It can refer to monthly or yearly forecasts (i.e., *long-term forecasts*) to guide strategic planning decisions, or to *short-term forecasts*, i.e., weekly, daily or hourly forecasts, mainly devoted to support staffing decisions and work-shift scheduling.

Call volume forecasts can also be categorized according to three different features: the call *type* (incoming or outgoing); the *lead time*, i.e. how much time in advance the forecast is performed with respect to the prediction time and the *duration*, that is, the time granularity of the forecast (e.g., hourly or daily). For example, we address the forecasting of incoming calls with 24 hours lead time and hourly duration.

Today's call centers are part of *multi-channel contact centers*, in which the customer, before interacting with an

Communicated by Dario Pacciarelli.

✉ Andrea Manno
andrea.manno@univaq.it

Fabrizio Rossi
fabrizio.rossi@univaq.it

Stefano Smriglio
stefano.smriglio@univaq.it

[1] Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica, Università degli Studi dell'Aquila, Via Vetoio, 67100 L'Aquila, Italia

agent by phone, is assisted through a dedicated website, can send/receive emails and can even be engaged in a virtual conversation through an interactive chatbot. All of these interactions are a valuable source of information to support accurate call volumes forecasts and, although several researches concern time series forecasts, relatively few attempts have been made to harness these *exogenous factors* in call volumes prediction.

In this paper, after reviewing the literature on call center forecasting (Sect. 2), we investigate two algorithms based on *Neural Networks* (Sect. 3). These algorithms are well suited to capture the complex nonlinear relationships between time series to be predicted and values observed on the exogenous factors, leading to better forecasting accuracy. They embody two opposite methodological lines. The first method, proposed in Bianchi et al. (2015), is based on *Echo State Networks* (ESN), a deep architecture with strong representational power which is very suitable for time series (see Sect. 3.1), but also highly sensitive to hyperparameters tuning. Indeed, ESNs combine the high computational power of its architecture with an easy training phase, but require an expensive external optimization of the network hyperparameters.

The second method is a shallow *Single Layer Feedforward Network* (SLFN) (e.g., Grippo et al. (2015)), that is, a Neural Network with a simpler structure, preferable for practical implementations thanks to its stability and ease of use. In general, the structure of SLFNs is not suited to reproduce complicated temporal patterns. Inspired by the recent work of Manno et al. (2022), we show that with an appropriate input selection strategy, based on simple autocorrelation analyses (Sect. 3.2), one can make the performance of the shallow SLFN fully competitive with that of ESNs while requiring less computation workload and maintaining ease of implementation. These findings are drawn by first comparing the two approaches on a benchmark dataset from the Machine Learning literature (Sect. 4), then by experimenting the methods on two publicy available call centers datasets (Sect. 5). Finally, in Sect. 6 some concluding remarks are presented.

2 Literature review

Far from being exhaustive (the interested reader is referred to Gans et al. (2003); Aksin et al. (2007); Ibrahim et al. (2016)), we review some known approaches so as to highlight the main features of the various classes of forecasting models.

The process of inbound calls has been often modeled as a Poisson arrival process (see e.g., Garnett et al. 2002; Gans et al. 2003; Wallace and Whitt 2005), by assuming a very large population of potential customers where calls are statistically independent low-probability events. However, these assumptions may not be consistent with real world call centers, characterized by evident daily and weekly seasonality. Moreover, it has been observed that call centers arrivals exhibit a relevant *overdispersion* (see e.g., Jongbloed and Koole 2001; Steckley et al. 2005; Aldor-Noiman et al. 2009), in which the variance of arrivals tends to dominate its expected value.

Two typical ways to overcome the previous drawbacks are: a nonhomogeneous Poisson process, with non-constant arrival rate which is a deterministic function of the time, may be adopted to cope with the seasonalities (see e.g., Brown et al. 2005; Green et al. 2007); while a doubly stochastic Poisson process (see e.g., Avramidis et al. 2004; Liao et al. 2012; Ibrahim and L'Ecuyer 2013), in which the arrival rate is itself a stochastic process, can be shown to have variance larger than mean (Ibrahim et al. 2016).

In Mandelbaum et al. (1999) and Aguir et al. (2004) call center arrival processes have been approximated as fluid dynamic models described as systems of differential equations, which can be solved by specific numerical methods (see e.g., Arqub and Abo-Hammour 2014).

Although stochastic processes have the advantage of being computed analytically, they are not suited to include important relationships between the target time series and some highly correlated exogenous factors.

Moving towards time series forecasting approaches, many Authors adopted autoregressive linear models, like *seasonal moving average* (e.g., Tandberg et al. 1995), *autoregressive integrated moving average* (ARIMA) (e.g., Andrews and Cunningham 1995; Bianchi et al. 1998; Antipov and Meade 2002; Taylor 2008) and *Exponential Smoothing* (e.g., Taylor 2010; Barrow and Kourentzes 2018), in order to predict future incoming call volumes as linear combinations of past occurrences of the target time series. These methods are both simple and easy to implement. Moreover, they can include important external factors (like calendar effects or advertising campaigns) by considering additive/multiplicative terms, covariates and transfer functions. However, they require a certain degree of users' knowledge of the system in order to make some *a priori* modeling decisions. Consequently, they are not flexible with respect to environmental changes and they are not able to capture hidden and often highly nonlinear relationships involving correlated exogenous factors. For these reasons, they are more suited to forecasting regular working periods.

Only recently, to overcome the previous drawbacks, some *Neural Networks* (NNs) methodologies have been proposed. While poorly interpretable, NNs prediction models have a strong flexibility and representational power, which have stimulated their application in many different fields (e.g., Biswas et al. 2016; Dong et al. 2018; Avenali et al. 2017; Sun et al. 2003; Chelazzi et al. 2021). In these contexts, NNs allow to automatically capture/model nonlinear interactions between the target time series and the correlated exogenous factors, exempting the user from making deliberate modeling choices. This aspect is crucial whenever exogenous factors strongly affect the target time series observations in a non-trivial way, as it is often the case in call centers arrivals. Moreover, NNs quickly adapt to changes in the system. Thus, they represent a promising instrument for call volumes forecasting in contact centers.

Millán-Ruiz et al. (2010) applied *momentum backpropagation* NN algorithm (LeCun et al. 2012) to forecast incoming volumes in a multi-skill call center. As NN inputs, they selected some lagged observations of the incoming volumes and some static exogenous factors (like the day of the week), based on a relevance criterion computed via the Mann-Whitney-Wilcoxon test (e.g., Nachar et al. 2008). Their proposed NN approach seems to outperform the ARIMA method in all the skills' groups. Barrow and Kourentzes (2018) adopted a single layer feedforward network (SLFN) (e.g., Grippo et al. 2015) method in which sinusoidal waves and special dummy variable inputs are combined to cope with respectively the seasonalities and the outliers (like special days or singular) of the target time series (see also Kourentzes and Crone 2010). This SLFN methodology has been shown to outperform linear models like ARIMA and exponential smoothing, however it requires a prior knowledge of outliers periods, which are often unpredictable.

As an alternative to shallow NNs, *Deep Learning* (DL) methods may be considered. In particular, *Recurrent Neural Networks* (RNNs) (e.g., Goodfellow et al. 2016) fit very well time series forecasting problems. Indeed, their feedback connections are able to reproduce temporal patterns which may repeat over time. Nonetheless, the training of RNNs is a very complicated task, due essentially to vanishing/exploding gradient phenomena as highlighted in Pascanu et al. (2013), and to bifurcation problems (see e.g., Doya 1992).

Jalal et al. (2016) adopted an ensemble NN architecture by combining an *Elman* NN (ENN) (Elman 1990), a semi-recurrent NN structured as a SLFN enriched with feedback connections where the latter may not be trained, and a *Non-linear Autoregressive Exogenous Model* (NARX) NN, i.e. a SLFN in which the inputs are lagged values of the target time series and exogenous factors (Chen et al. 1990). They favorably compared their ensemble method with a time-lagged feed-forward network (a SLFN where the only inputs are lagged values of the target time series) to forecast the incoming volumes on 15 minutes intervals for an Iranian call center. However, the combination of ENN with NARX-NN does not look straightforward and also efficiency may become an issue as ENNs are known to suffer from slow training convergence (see e.g., Cheng et al. 2008).

Bianchi et al. (2015) used *Echo State Networks* (ESNs) to the 24 hours ahead forecast of the call volumes hourly accessing an antenna of a mobile network, where data are taken from the Data for Development (D4D) challenge (Blondel et al. 2012). As will be better specified in Sect. 3, ESNs are DL methods which try to retain the benefits of RNNs while mitigating their training issues. In Bianchi et al. (2015) ESNs outperform ARIMA and exponential smoothing on the D4D dataset by *automatically* leveraging 6 exogenous time series (as well as the target time series), where an external genetic algorithm is used to determine the better network configuration.

Motivated by the aforementioned advantages of NNs-based methods, mainly in terms of prediction accuracy and flexibility, aim of this work is to identify suitable NNs able to leverage exogenous factors to forecast call volumes in practical settings. The first method considered is the ESN-based method by Bianchi et al. (2015). It has the high computational power of a deep architecture along with an easy training phase. However, it requires a careful and costly (in terms of CPU time) external optimization of network hyperparameters, preventing straightforward implementation by unskilled users. The second is a new NARX-like shallow SLFN approach, in which a simpler and easy-to-use model is adopted. Despite its simplicity implies less representational power, we are able to combine the SLFN model with a simple but effective input selection strategy that keeps the SLFN performance competitive with respect to ESNs, speeding-up significantly the inference time and being less sensitive to hyperparameters tuning.

3 Methodologies

In this section we review ESNs general principles and describe the competing shallow SLFN methodology, with particular focus on the lagged input selection.

Let us first introduce the supervised learning framework, where we assume to have a dataset of $p+1$ time series $(x_1[t], x_2[t], \dots, x_p[t], \hat{y}[t])$, with $t = 1, \dots, H$; where $\hat{y}[t]$ is the target time series we are aimed to predict, while $x_1[t], x_2[t], \dots, x_p[t]$ are exogenous time series that may provide information to build the prediction model. The latter may have an aleatory nature (e.g. the number of outgoing calls from a call center division), or may be deterministic (e.g. the time series representing the timestamp in which the time series values are observed).

The goal is to train a ML model to predict a lagged value of the target time series at timestamp $t+k$ with $k \in \mathbb{N}_+$ by leveraging all observations of exogenous and target time series collected until the current timestamp t. Formally, let us consider a *training set* TR of available data

$$TR := \{(x_1[t], x_2[t], \dots, x_p[t], \hat{y}[t], \hat{y}[t+k]), \quad t = 1, \dots, H_{TR}\}, \tag{1}$$

where H_{TR} is the length of the training horizon. The predictive model is trained on the basis of TR and tested on a similar set of data, referred to as *testing set* (TS), for which the lagged target time series values are also known but whose samples have not been used during the training phase. As it is often the case in time series forecasting applications, the samples of TR are temporarily consecutive and provided in an increasing order, and TS consists in a subsets of data corresponding to the tail of the whole period of data collection, i.e.

$$TS := \{(x_1[t], x_2[t], \dots, x_p[t], \hat{y}[t], \hat{y}[t+k]),\quad t = H_{TR}+1, \dots, H_{TR}+H_{TS}\}, \tag{2}$$

where H_{TS} is the length of the testing horizon. As will be clarified later, ESNs, being dynamical systems, require data to be provided with such a temporal order both in TR and TS. Differently, for SLFNs, this requirement is not strictly necessary for TR while it is for TS, as a sequential rolling horizon strategy is adopted in the testing phase.

In this work we consider hourly timestamps and a prediction advance of 24 hours, i.e. $k = 24$.

3.1 Brief summary on ESNs

ESNs are composed of three main layers:

- A standard input layer as in SLFNs,
- A deep hidden layer, denoted as *reservoir*,
- A standard linear output layer, denoted as *readout*.

The reservoir is a recurrent layer made up of a large variety of connection patterns (including feedforward, feedback and loop connections) connecting r hidden units and reproducing multiple temporal dynamics. ESNs act as a dynamic system in which, at current time step t, the input signal $x[t] \in \mathbb{R}^p$ is propagated from the input layer to the reservoir by means of feedforward weighted connections, with weights $W_p^r \in \mathbb{R}^{p\times r}$. In the reservoir, the signals pass through a sparse set of weighted recurrent connections (with weights $W_r^r \in \mathbb{R}^{r\times r}$), as the reservoir connectivity should be very low to avoid signal explosions after a certain number of time steps. Finally, the signals exiting the reservoir reach the output layer through further feedforward weighted connections (with weights $w_r^o \in \mathbb{R}^r$), producing the output of the network $y[t]$. More formally, the current output $y[t]$ of a ESN is computed as follows. At first, the state of the units of the reservoir, say $h[t] \in \mathbb{R}^r$, is updated as

$$h[t] = f({W_p^r}^T x[t] + W_r^r h[t-1]), \tag{3}$$

where ${W_p^r}^T x[t]$ is the result of propagating the current input signal $x[t]$ through the input-reservoir weighted connections, $W_r^r h[t-1]$ represents the propagation of the previous

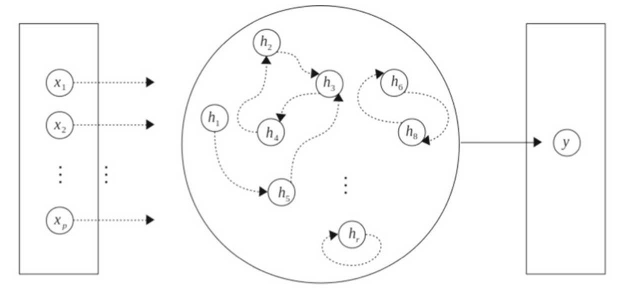

Fig. 1 A schematic depiction of a simplified ESN

reservoir states, i.e. the memory of the network, and $f : \mathbb{R}^r \to \mathbb{R}^r$ is a nonlinear function (typically sigmoidal). The values of W_p^r and W_r^r are randomly assigned in a previous initialization phase. Notice that the update of the states is univocally determined by the selected f and the random assignment for W_p^r and W_r^r.

Then, the current output is computed as

$$y[t] = {w_r^o}^T h[t]. \tag{4}$$

The weights w_r^o are the only trained ones. Figure 1 represents a schematic structure of an ESN, where the dotted connections correspond to the weights which are assigned at random in the initial phase, and the solid connection represents the only involved in the training phase. Some ESNs may have also further random feedback connections from the output layer to the reservoir, and trainable feedforward connections from the input to the output layers, but for the sake of simplicity they are not considered here.

Concerning the whole training process, consider a training set of m input-output pairs

$$(x[1], \hat{y}[1+h]), (x[2], \hat{y}[2+h]), \dots, (x[m], \hat{y}[m+h]), \tag{5}$$

where the input signals $x[i]$ with $i = 1, 2, \dots, m$ are associated to m consecutive time instants, h is the a priori selected forecast lead time, and $\hat{y}[i+h]$ is the observed output at time instant $i+h$. As anticipated before, in the first *initialization* step the input and reservoir weights W_p^r and W_r^r are assigned randomly (remaining unaltered during the training process), so that, in the second step denoted as *warming*, the sequence of reservoir states $h[1], h[2], \dots, h[m]$ can be computed according to (3) and (4). The warming phase is performed to get rid of the transient, which is due to the fact that the initial state of the ESN, $h[0]$, is meaningless. In other words, the first internal states and the first outputs are dropped, as they are influenced by $h[0]$.

The sequence of states can be organized into a $m \times r$ matrix H as follows

$$H = \begin{bmatrix} h[1]^T \\ h[2]^T \\ \vdots \\ h[m]^T \end{bmatrix}, \tag{6}$$

and the observed outputs can be arranged into the m-dimensional vector $\hat{y}$ as

$$\hat{y} = \begin{bmatrix} \hat{y}[1+h] \\ \hat{y}[2+h] \\ \vdots \\ \hat{y}[m+h] \end{bmatrix}. \tag{7}$$

Then, in the final *learning* step, the training problem can be conveniently formulated as the one of determining the output weights w_r^o by solving the following regularized linear least squares (RLLS) convex problem

$$w_r^o = \underset{w \in \mathbb{R}^r}{\operatorname{argmin}} \frac{1}{2} \|Hw - \hat{y}\|^2 + \frac{\lambda}{2} \|w\|^2, \tag{8}$$

with $\lambda \in \mathbb{R}^+$.

Therefore, the ESN training involves computationally irrelevant initialization and warming steps, along with the solution of a relatively easy convex unconstrained optimization problem in the learning step. Training the readout weights can be interpreted as the problem of determining the proper linear combination of the nonlinear reservoir states in order to reproduce the desired temporal patterns of the target time series.

Alternative strategies have been considered to train the readout weights in the learning phase, for example by replacing (8) with an Elastic Net Penalty problem (Zou and Hastie 2005) to obtain sparse solution, or to use a further ML algorithm like Support Vector Machines (e.g., Shi and Han 2007; Manno et al. 2016). However, only the RLLS formulation is considered in this work.

Once trained, the ESN can be used to the step by step forecast associated to the future time instants.

ESNs, retaining the advantages of complex nonlinear RNNs models with a much simpler training, are well suited for the short-term forecast of time series (for long-term forecast simpler models may perform better), mainly when some correlated exogenous time series are provided to the model.

On the other hand, ESNs need a careful selection of many hyperparameters to produce accurate forecast. Specific attention should be devoted to the matrix of reservoir weights W_r^r. Although randomly determined, W_r^r should respect the *echo state property* (Lukoševičius and Jaeger 2009), i.e. the influence of current input signal $x[t]$ and state $h[t]$ on future states should vanish in a finite number of time steps. This property is ensured if the spectral radius of W_r^r, that is $\rho(W_r^r)$, satisfies

$$\rho(W_r^r) < 1. \tag{9}$$

From a practical point of view (9) can be enforced by simply rescaling W_r^r after the initialization, nonetheless $\rho(W_r^r)$ should not be too far from 1 to avoid the opposite behavior, i.e., a too fast vanishing influence of the current state. It can be shown that if echo state property is satisfied, ESNs have a universal approximation property in the sense that they can realize every nonlinear filter with bounded memory arbitrarily well (Maass et al. 2007).

Further important hyperparameters are the number of the reservoir units, the reservoir connectivity (i.e. the number of nonzero weights connections), the scaling of the input weights W_p^r, and the subset of exogenous time series to be considered as input.

To determine the optimal value of these critical hyperparameters, we use the genetic algorithm applied in Bianchi et al. (2015). Such a choice is driven by the fact that the performance evaluation for a given combination of hyperparameters is a result of an ESN training process, so it has to be treated as a black-box function.

3.2 Shallow SLFNs with lagged inputs

As for the shallow SLFN approach, a NARX strategy has been adopted where the network is provided with the current values of exogenous factors along with the lagged inputs of the target and exogenous time series, the selection of which strongly affects the prediction quality.

Differently from other sophisticated and/or time-consuming approaches (e.g.,Weng and Liu 2006; Kourentzes and Crone 2010; Karasu et al. 2020), the adopted methodology is based on automated simple filters applied to correlations and autocorrelations coefficients (Box et al. 2015), whose thresholds can be easily adapted to the specific application domain, without particular optimization/forecasting skills.

The method is based on two main steps:

1. A two-phase preprocessing for exogenous factors and lagged input selection;
2. A SLFN rolling horizon training.

3.2.1 Preprocessing

The preprocessing step is divided into two sequential phases denoted, respectively `SelectExogenous` and `SelectLag` (see Algorithm 1).

In `SelectExogenous` the exogenous time series are selected, based on their correlations with the target time series. Firstly, a test is performed on $\hat{y}[t]$ to check stationarity (e.g. MacKinnon 1994). Two cases are possible.

i) $\hat{y}[t]$ *is stationary*.
In such a case a normality test is applied to determine if $\hat{y}[t]$ is also normally distributed.[1] Then, for each exogenous time series $x_i[t]$ the pair $x_i[t], \hat{y}[t]$ is analysed and if both the series passed the normality test, their correlation coefficient c is computed via the parametric Pearson test (Pearson 1895), otherwise via the rank-order Spearman test (Zar 2005). If c is greater than a certain threshold $\sigma > 0$ then $x_i[t]$ is selected by adding its index i to the (initially empty) output list L.

ii) $\hat{y}[t]$ *is non-stationary*.
Then, for each exogenous time series $x_i[t]$, a cointegration test (e.g. MacKinnon 1994), which is more suited for non-stationary time series, is applied to the pair $x_i[t], \hat{y}[t]$. If they are cointegrated, $x_i[t]$ is selected by adding its index i to the (initially empty) output list L.

After the execution of `SelectExogenous`, `SelectLag` is applied to the target time series and to the exogenous time series returned by `SelectExogenous` in order to select their lagged values to feed the SLFN. By observing the autocorrelation plot (Box et al. 2015) of this kind of time series (see e.g., Fig. 2) it is possible to appreciate two different kind of intuitive temporal patterns: a daily pattern, i.e. high correlations at lags which are multiple of 24 hours, and high correlations with adjacent/proximal lags. This suggest a "time-window logic" in the selection of the lagged inputs. In particular, considering a time series, the idea is to iteratively select the past lagged values of multiple of 24 hours ($t-24, t-48, t-72, \ldots$), referred to as daily-lags, as long as autocorrelation values are above a certain threshold $\gamma_1 > 0$. Then, for each selected daily-lag, a window is constructed by iteratively select forward and backward adjacent lags (F and B in the scheme) as long as their autocorrelation values are above threshold $\gamma_2 > 0$. At the end of `SelectLag` a window of consecutive lagged inputs is determined for each selected daily-lag (W in the scheme).

Example 1 Suppose one wish to predict the time series value at hour 13 of a generic day d (the black slot in Figs. 3 and 4). The outer loop in `SelectLag` selects the first two autocorrelation values lagged at 24 hours multiples as they are larger than γ_1. Then, the time series values at hour 13 of days $d-1$ and $d-2$ are added to the set of lagged inputs W (gray slots in Fig. 3). The loop ends at $d-3$ where $a_{3\cdot 24} < \gamma_1$. Now, assuming that for each of the two selected lagged inputs only the adjacent autocorrelation values (the previous and the next) are larger than γ_2, the inner loop adds to W the time series values at hour 12 and 14 of days $d-1$ and $d-2$ as lagged inputs (they correspond to the light gray slots in Fig. 4). The final set of lagged inputs W is represented by all gray slots in Fig. 4.

Algorithm 1: Preprocessing

```
Phase 1: SelectExogenous
Data: $TR$ set as in (1), parameter $\sigma > 0$
begin
    $L \leftarrow \emptyset$;
    apply stationarity_test to $\hat{y}[t]$ for
      $t = \{1, \ldots, H_{TR}\}$;
    if $\hat{y}[t]$ is stationary then
        apply normal_test to $\hat{y}[t]$ for $t = \{1, \ldots, H_{TR}\}$;
    end
    for $i \in \{1, \ldots, p\}$ do
        if $\hat{y}[t]$ is stationary then
            apply normal_test to $x_i[t]$ for
              $t = \{1, \ldots, H_{TR}\}$;
            if ($x_i[t]$ is normal) and ($\hat{y}[t]$ is normal) then
                $c$ =Pearson_corr_coeff($x_i[t], \hat{y}[t]$);
            else
                $c$=Spearman_corr_coeff($x_i[t], \hat{y}[t]$);
            end
            if $c > \sigma$ then
                $L := L \cup \{i\}$;
            end
        else
            apply cointegration_test to $x_i[t]$, $\hat{y}[t]$
              for $t = \{1, \ldots, H_{TR}\}$;
            if $x_i[t]$, $\hat{y}[t]$ are cointegrated then
                $L := L \cup \{i\}$;
            end
        end
    end
end
return $L$;

Phase 2: SelectLag
Data: Time series $S[t]$, parameters $\gamma_1, \gamma_2 > 0$;
      maxLags $\in \mathbb{N}_+$
begin
    $W \leftarrow \emptyset$, $k := 1$;
    determine the autocorrelation values vector
      $a \in [-1, 1]^{\text{maxLags}}$ by computing the autocorrelation
      values of $S[t]$ for maxLags time lags;
    while $a_{k \cdot 24} > \gamma_1$ do
        $B \leftarrow \emptyset$, $F \leftarrow \emptyset$, $k_B, k_F := 1$ ;
        while $a_{k \cdot 24 - k_B} > \gamma_2$ do
            $B := B \cup k \cdot 24 - k_B$;
            $k_B := k_B + 1$ ;
        end
        while $a_{k \cdot 24 + k_F} > \gamma_2$ do
            $F := F \cup k \cdot 24 + k_F$;
            $k_F := k_F + 1$ ;
        end
        $W := W \cup \{B, k \cdot 24, F\}$;
        $k := k + 1$;
    end
end
return $W$;
```

[1] Generally time series are not normally distributed due to their temporal dependencies so the normality test can be removed. However, to have a more general method we have included it in the algorithm.

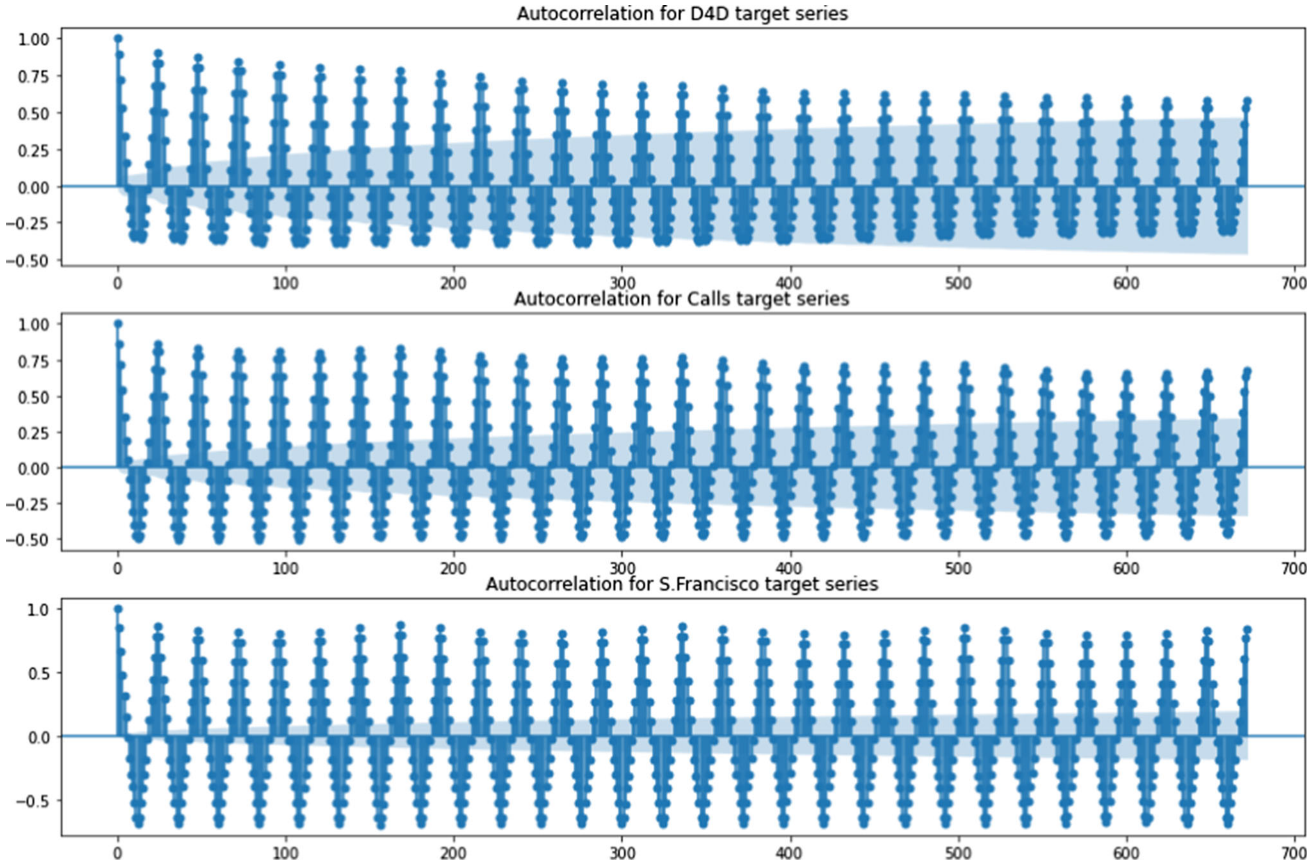

Fig. 2 The autocorrelation plots for the target time series of the case studies. In the x-axis the lagged timestamps (hours) are reported, while the y-axis represent the autocorrelation value

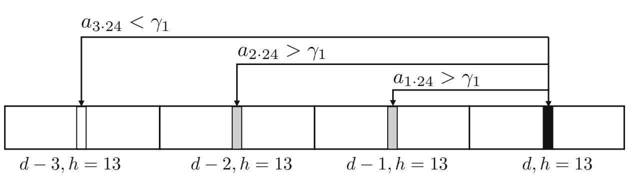

Fig. 3 Example of lagged inputs selected at 24 hours multiples

Notice that, for notation simplicity, the `SelectLag` scheme is reported for a single generic time series $S[t]$. `SelectLag` is not applied to the deterministic exogenous series for which lagged inputs would provide no information.

3.2.2 SLFN rolling horizon training

Once `SelectExogenous` and `SelectLag` have selected the exogenous factors, and the lagged values for both the target and exogenous time series, sets TR and TS can be built accordingly. Clearly, besides the lagged values, also the current values of the target and of the exogenous factors are included in each sample. Then, the hyperparameters of the SLFN are heuristically estimated by means of a simple grid-search method with k-fold cross-validation (see e.g., Bishop 2006). Similarly to ESNs, an expensive external optimization procedure could be applied to determine the optimal values of those hyperparameters. Nevertheless, as shown in the performed experiments, the SLFNs are more robust with respect to hyperparameters selection, so that they obtain good performance even if a cheaper grid-search procedure is applied.

The training phase has been organized by a rolling horizon procedure. In particular, at each rolling horizon step k corresponding to a certain current day d, a SLFN model is trained on the basis of the current training set, say TR^k and it is used to generate the 24 predictions for day $d+1$. Then, at step $k+1$, the 24 samples associated to day $d+1$ are included in the training set, while samples form the less recent day are removed. Formally, being A_d the set of 24 hourly samples associated to day d, $TR^{k+1} = \left(TR^k \setminus A_{d-H_{TR}+1}\right) \cup A_{d+1}$, and so on. In the experimental setting presented in the sequel, starting from the basic ordered and consecutive TR and TS in the first rolling horizon step the SLFN is trained on a training set $TR^0 = TR$ and the predictions are generated for the 24 hours associated to the first 24 samples of TS, i.e. $A_{H_{TR}+1}$ (see (2)). Then, $TR^1 = \left(TR^0 \setminus A_{d-H_{TR}+1}\right) \cup A_{H_{TR}+1}$, the SLFN is trained again, and the predictions are produced for the hours associated to samples $A_{H_{TR}+2}$, and so on.

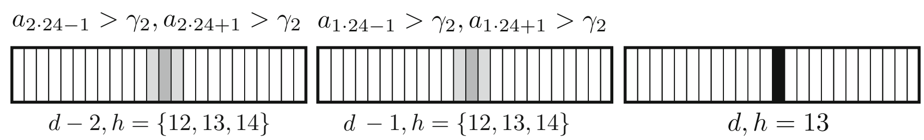

Fig. 4 Example of lagged inputs selected by procedure `SelectLag`

Table 1 Fields description for the D4D dataset

Name	Description
ts1	Number of hourly incoming calls
ts2	Volume in minutes of hourly incoming calls received by the selected antenna
ts3	Number of hourly outgoing calls sent by the selected antenna
ts4	Volume in minutes of hourly outgoing calls sent by the selected antenna
ts5	Hour of the day to which this CDR refers
ts6	Day of the week to which this CDR refers

It is worth noticing that, in our setting, the grid-search procedure is performed only once on the initial TR, even if alternative choices can be considered.

4 Benchmarking on a challenge dataset

The two considered methodologies are first calibrated and assessed on a benchmarking dataset known as D4D.

4.1 D4D dataset description

D4D is an open data challenge on anonymous call patterns of Orange's mobile phone users in Ivory Coast. It contains anonymized Call Detail Records (CDR) of phone calls and SMS exchanges between customers in a cell covered by an antenna of a telecommunication company. Data are collected in the period ranging from December 2011 to April 2012. The dataset is taken from Blondel et al. (2012) and, in particular, it is the one labeled with as "antenna-to-antenna traffic on an hourly basis". We restricted our attention on the CDR related to the antenna labeled 1. Although this dataset did not originate from a call center, its features are fully comparable with the features associated to call volumes, making the experiment meaningful and, at the same time, directly comparable with results in the literature. The time series of the D4D dataset are reported in Table 1.
ts1 is the target time series, **ts2**, **ts3** and **ts4** are the exogenous aleatory time series, while **ts5** and **ts6** are the exogenous deterministic time series.

The dataset is organised so that data of the first 109 days are inserted in TR (2616 hourly samples), while data of the last 20 days (480 hourly samples) are inserted in TS.

4.2 D4D experimental settings and preprocessing outputs

Concerning the ESN, following the standard practice adopted in Bianchi et al. (2015), a constant fictitious time series **ts0** with all values set to 1 is added, acting as a bias for the individual neurons of the network. The ESN hyperparameters have been determined by the same genetic algorithm used in Bianchi et al. (2015), i.e. the one presented in Deep et al. (2009) which is able to handle both continuous and discrete variables, and whose implementation is available at https://github.com/deap. The hyperparameters optimization has been carried out by extracting a validation set consisting in the last 10% of samples of TR to compute the fitness function, and training the ESN on the remaining TR data. Table 2 reports the adopted values for the genetic algorithm parameters, together with their short descriptions.

Table 3 reports the ESN hyperparameters optimized through the genetic procedure, together with their lower and upper bounds and best value found. For details of all the hyperparameters meaning the reader is referred to Løkse et al. (2017) or to the website https://github.com/siloekse/PythonESN where the `Python` ESN implementation adopted in this work is freely available. A ridge regression method has been used to train the output weights.

Concerning the SLFN, it has been implemented by exploiting the method `MLPRegressor()` of the `scikit-learn Python` package. The stationarity, normality and cointegration tests of the `SelectExogenous` procedure have been carried out through the `statsmodel` and `scipy Python` packages. In particular, the stationarity Augmented Dickey-Fuller test (Fuller 2009) has been performed through the `adfuller()` function, the normality D'Agostino-Pearson test (Pearson et al. 1977) through function `normaltest()`, and the cointegration Augmented Engle-Granger test (Engle and Granger 1987; MacKinnon 1994), through function `coint()`. All time series resulted

Table 2 Parameters used for the genetic algorithm

Param. name	Description	Value
Population_Size	size of the population	50
n_Generations	number of generation	20
n_Offsprings	number of children to produce at each generation	35
Mutpb	Probability that an offspring is produced by mutation	0.2
Cxpb	Probability that an offspring is produced by crossover	0.5
Mu_1	Mean of the Gaussian distribution used to mutate individuals	0.0
Mu_2	Number of individuals to select for the next generation	50
Parallel	Enable multi-threading	false

Table 3 Hyperparameters spaces for ESN model

Hyperparameter name	Lower bound	Upper bound	Best
Connectivity	0.25	0.25	0.25
n_drop	50	50	50
Input_shift	0	0	0
Teacher_shift	0	0	0
n_internal_units	100	500	429
Teacher_scaling	0.1	0.9	0.758
Noise_level	0.0	0.1	0.0065
Input_scaling	0.1	0.9	0.349
Spectral_radius	0.5	1.4	0.799
Feedback_scaling	0.0	0.6	0.580
Regularization	0.001	1.0	0.238

to be stationary and not normal with a 5% significance level (this is valid for all the three considered case-studies). The hyperparameters of the SLFN have been obtained by a simple grid-search procedure with a 5-fold cross-validation in which the validation set has been determined by extracting randomly the 10% of samples of TR. the correlation threshold σ in `SelectExogenous`, has been set to 0.3 as, commonly, correlations coefficients below this value are considered not relevant (see e.g., Asuero et al. 2006). As far as for `SelectLag`, high correlation thresholds above the values 0.7 have been considered, i.e $\gamma_1 = 0.8$ and $\gamma_2 = 0.7$. The remarkably high γ_1 value is to avoid to go back too many days in the lagged input selection, which may cause potential training and applicability issues, as the autocorrelation daily pattern remains stably high for a long time (see Fig. 2).

When applied to the D4D, the preprocessing phases selected all exogenous time series except for **ts6**, and selected lagged inputs made up of 3 values (the central, one backward and one forward) up to 9 days before.

Concerning the data normalization, all data have been normalized in the interval [0, 1], except for the **ts5** time series which, in SLFNs, has been transformed with a one-hot-encoding. The considered hyperparameters for the SLFN grid-search are reported in Table 4.

For ESNs, all experiments have been carried out with and without seasonal differencing all aleatory time series, as a very light first-order trend can be noticed from the autocorrelation plots.

4.3 Performance measures

Analogously to the code used in Bianchi et al. (2015), the *Normalized Root Mean Squared Error* (NRMSE) has been considered as performance measure. This can be computed as

$$\mathrm{NRMSE}(\hat{y}, y) = \frac{\sqrt{\frac{1}{n}\sum_{i=0}^{n}\left(\hat{y}_i - y_i\right)^2}}{std(\hat{y})}, \tag{10}$$

where $\hat{y}$ is the vector of observed values, y is the vector of predicted ones, $std(\hat{y})$ is the standard deviation of $\hat{y}$, and n is the number of samples of $\hat{y}$ and y .

NRMSE tends to penalize large errors more than other measures like the *Mean Absolute Error* (MAE) (computed as $\frac{1}{n}\sum_{i=0}^{n}|\hat{y}_i - y_i|$), resulting more meaningful in call centers operations, where large errors are undesirable as they greatly affect service level. However, both NRMSE and MAE appear not completely suited to compare call centers prediction models and for operational purposes as they are sensitive to call volumes, which can have a large variability range between heterogeneous call centers, and between week days of the same call center. For this reason, we have also considered a specific version of the better "interpretable" *Mean Absolute Percentage Error* (MAPE) (see e.g., Manno et al. 2022), denoted as *daily Mean Absolute Percentage Error* (dMAPE) and computed as

$$\mathrm{dMAPE}(\hat{y}, y) = \frac{1}{n}\sum_{i=0}^{n}\left|\frac{\hat{y}_i - y_i}{M(d_i)}\right|, \tag{11}$$

where d_i is the day corresponding to observation i, and $M(d_i)$ is the mean of the observed values in the 24 hours associated

Table 4 The considered hyperparameters for the SLFN grid-search

Hyperparam. name	Description	Grid values	Best
Hidden_layer_sizes	Number of hidden layer neurons	{25, 50, 100}	25
Alpha	Regularization coefficient	{0, 0.1, 0.01, 0.001}	0.1
Solver	Loss function optimizer	{lbfgs,adam}	lbfgs
Activation function	Hidden layer neurons activation function	{Logistic,tanh,relu}	tanh
Max_iter	Maximum number of optimizer iterations	{200, 400}	200

to d_i and computed as

$$M(d_i) = \frac{1}{24} \sum_{j \in d_i} \hat{y}_j. \tag{12}$$

dMAPE overcomes the "infinite error" issue affecting the MAPE. Indeed, being the latter computed as $\frac{1}{n} \sum_{i=0}^{n} \left| \frac{\hat{y}_i - y_i}{\hat{y}_i} \right|$, it generates huge or infinite errors when one or more observations tend to the zero value, and this may be the case in hourly call center data. Moreover, dMAPE seems to be more suited for daily scheduling decisions, as normalizing the error with respect to the daily mean of the hourly calls seems to have a more relevant impact from an operational point of view.

We finally look at a third performance measure, namely, the *Mean of Maximum Daily Error* (MMDE), chosen for its operational relevance. It is computed as

$$\text{MMDE}(\hat{y}, y) = \frac{1}{m} \sum_{j=0}^{m} \text{max\_p\_err}_j(\hat{y}, y), \tag{13}$$

where m is the number of days associated to vectors $\hat{y}$ and y, and $\text{max\_p\_err}_j(\hat{y}, y)$ is the maximum daily percentage error associated to day j, say d^j, and computed as

$$\text{max\_p\_err}_j = \max \left\{ \left| \frac{\hat{y}_i - y_i}{M(d_i)} \right| : i \in d^j \right\}. \tag{14}$$

Being an upper bound on the forecasting error, MMDE is a good measure for the reliability of the prediction models, and it is also a robust indicator for daily staffing decisions.

Other commonly used performance indexes, like the aforementioned MAE, seem more suited for different kinds of applications (see e.g., Adnan et al. 2021).

4.4 Numerical results on D4D

The results of the experiments on the D4D dataset are reported in Table 5. It is worth mentioning that in Bianchi et al. (2015) ESNs have been favorably compared with standard forecasting methods like ARIMA and Exponential Smoothing on the same D4D dataset.

Since both the ESNs and SLFNs trained models vary, in general, with the initialization, all the reported results are averaged on 30 runs characterized by different random selections of the reservoir weights for ESNs, and different random starting solutions for the SLFNs training optimization problem.

Table 5 Comparison results for D4D

D4D	NRMSE	dMAPE %	MMDE %
ESN	0.3769	17.60	54.27
SLFN	**0.3691**	**16.75**	**50.65**
SN	0.3909	17.82	60.89

The best results are highlighted in bold

The performance of the ESN and SLFN models are compared with those of a simple Seasonal Naive (SN) (see e.g., Barrow and Kourentzes 2018) with seasonal cycle equal to 24, in which the 24 hourly predictions of a certain day correspond to the 24 values observed in the previous day. The motivation of comparing with SN is to verify if the more complicated NNs-based models are able to leverage exogenous factors to improve the predictive performance with respect to an extremely simple prediction model requiring no computation. Moreover, as also specified below, all target time series considered here exhibit a low correlation with the day of the week and a high correlation with the hour of the day, and this may imply good performance of the SN model with a 24 hours seasonal cycle.

All the experiments have been carried out on a laptop Intel(R) Core(TM) i7-6700HQ CPU @ 2.60G with 8 GB of RAM.

The reported results for the ESN correspond to the seasonal differencing case as they were slightly better than the ones without differencing.

First of all we notice that the NRMSE obtained with ESN are comparable with those reported in Bianchi et al. (2015) on the same dataset. Concerning the three performance, SLFN achieves always the best ones and SN the worst ones. However, a little surprisingly, the performance difference between the more sophisticated ESN and SLFN models against the SN is not so remarkable in terms of dMAPE (around 1%), and this can be connected with the aforementioned fact the day of the week (**ts5**) has no relevant correlation with the target time series, making all days quite similar. Nonetheless, a certain difference may be noticed in terms of NRMSE and mainly in

Table 6 Fields description for the ACD dataset

Time series name	Description
ts1	Number of hourly received calls by the call center.
ts2	Number of hourly handled calls by all the operators in the call center.
ts3	Number of hourly answered calls by all the operators in the call center.
ts4	Number of hourly routed or queued per operator.
ts5	Number of hourly received calls by satisfying.
ts6	Hour of the day to which this CDR refers.
ts7	Day of the week to which this CDR refers.

Table 7 Fields description for the SF dataset

Time series name	Description
ts1	Number of hourly received calls by the call center.
ts2	Number of hourly received calls belonging to category *Passing Call*.
ts3	Number of hourly received calls belonging to category *Traffic Stop*.
ts4	Number of hourly received calls belonging to other categories.
ts5	Hour of the day to which this CDR refers.
ts6	Day of the week to which this CDR refers.

terms of MMDE. Considering the latter, which is very critical from an operational point of view, ESN improves the performance up to more than 6% with respect to the SN, while SLFN up to more than 10%. In terms of CPU-time, the ESN requires about 40 minutes to determine the hyperparameters through the genetic algorithm, and the SLFN grid-search about 15 minutes. For both methods, the training times are irrelevant with respect to the hyperparameter selection phase. Clearly, the SN CPU-time can be considered null.

5 Comparisons on call center datasets

In this section we compare the performance of the considered forecasting techniques on two real-world datasets, which are freely available on the Kaggle website (https://www.kaggle.com/).

5.1 Call center datasets description

The first dataset contains incoming calls information for a tech support call center. The CDR are in the form of a dump from an Automatic Calls Distribution (ACD) software. Hence, this dataset is shortly referred to as ACD. The available data are relative to 2019 year, starting from the 1st of January to the 31th of December.

The time series of the ACD dataset are reported in Table 6. **ts1** is the target time series, time series from **ts2** to **ts5** are the aleatory exogenous ones, while **ts6** and **ts7** the deterministic.

By considering that a subset of data related to the final weeks of the year have been removed as their values were anomalous (probably due to the holidays), the final training set TR corresponds to 258 days (6192 hourly samples) and the testing set TS corresponds to 45 consecutive days (1080 hourly samples).

The second dataset, which is referred to a SF, contains information about incoming calls to a San Francisco police switchboard. The SF dataset spans across different years, starting from March 2016 to July 2019. The dataset has been selected to see how the models behave with a bigger data set. The time series involved in SF are reported in Table 7. **ts1** is the target time series, time series from **ts2** to **ts4** are the aleatory exogenous ones, while **ts5** and **ts6** the deterministic.

The final training set TR corresponds to 1095 days (26280 hourly samples) and the testing set TS corresponds to 193 consecutive days (4632 hourly samples).

5.2 Call centers experimental settings and preprocessing outputs

The experimental settings are the same as those described in Sect. 4.

`SelectExogenous` discarded the time series related to the day of the week for both ACD and SF, while `SelectLag` selected lagged inputs made up of windows of 3 values and up to 9 days before for ACD and 44 for SF.

The hyperparameters determined (by the genetic algorithm) for the ESN are reported in Table 8, while the ones obtained by the grid-search for the SLFN are reported in Table 9.

Table 8 The selected hyperparameters for the ESN model

Hyperparameter name	ACD	SF
Connectivity	0.25	0.25
n_drop	50	50
Input_shift	0	0
Teacher_shift	0	0
n_internal_units	375	436
Teacher_scaling	0.493	0.211
Noise_level	0.0009	0.003
Input_scaling	0.886	0.196
Spectral_radius	0.557	1.000
Feedback_scaling	0.137	0.282
Regularization	0.115	0.372

Table 9 The selected hyperparameters for the SLFN model

Hyperparameter name	ACD	SF
Hidden_layer_sizes	50	100
Alpha	0.1	0.001
Solver	adam	adam
Activation function	relu	relu
Max_iter	400	400

Table 10 Comparison results for ACD and SF datasets

ACD	NRMSE	dMAPE %	MMDE %
ESN	0.3834	21.49	83.23
SLFN	**0.3502**	**19.28**	**73.63**
SN	0.4237	23.45	89.24
SF			
ESN	0.3775	12.60	39.19
SLFN	**0.3582**	**11.97**	**38.04**
SN	0.4808	16.00	51.58

The best results are highlighted in bold

5.3 Numerical results on the call center datasets

The results for the call center datasets are reported in Table 10. The best performance of the ESN (the ones reported) are obtained with the seasonal differencing for ACD and without seasonal differencing for SF.

The results show that, similarly to the D4D benchmark case, the SLFN model performs better with respect to all the performance measures. In terms of dMAPE ESN and SLFN are comparable (mainly in SF) and they both perform better than SN (for both datasets SLFN improves the SN results by almost 4%). However, the more significant difference between the NNs-based approaches and the SN is found in MMDE, where SLFN improves the SN performance by more than the 15% for ACD and more than 13% in SF (the improvements for the ESN are respectively of the 6% and of the 12%). As mentioned before, a MMDE improvement is very relevant from an operational point of view, as it may incide on the personnel sizing policy. Also the NRMSE of the ESN and SLFN models are significantly better than SN, and comparable to each other, even if SLFN is always preferable.

Table 11 The time needed by each method to determine the values of the hyperparameters (genetic algorithm for ESN, grid-search for SLFN)

Dataset	ESN time (min.)	SLFN time (min.)
D4D	40	15
Calls	90	30
S.Francisco	400	84

Table 12 Comparison on the outliers days for SF

SF outliers	dMAPE	MMDE
SLFN	**13.02**	**36.80**
SN	19.24	54.87

The best results are highlighted in bold

Comparing ESNs and SLFNs, another important aspect from an operational point of view is the inference time, intended as the sum of the time required to set the hyperparameters, train the model and generate the forecast. In the considered case-studies the inference time can be approximated with the time needed only to set the hyperparameters (through genetic optimization for ESNs and grid-search for SLFNs), as the latter dominates the times needed for training and forecasting. Table 11 reports the CPU-time in minutes required to set the hyperparameters by both NNs-based models on the three datasets. The computational workload for SLFN is significantly lower than for ESN.

Summing up, by considering all the accuracy performance and the time needed to run the method, the SLFN strategy appears to be the most promising one. Moreover, its simplicity and ease of implementation make it appealing from an industrial point of view.

Concerning the SLFN, we remark that, for datasets Calls and SF, we adopted the same threshold values σ, γ_1, and γ_2 as in D4D, highlighting the robustness and stability of the method.

5.4 Numerical results for outlier days

A further analysis has been performed to compare the SLFN model (which is the most performing one) with the SN model (the cheapest one) on potential outlier days for the SF dataset.

Since no information about outliers days were available for the investigated datasets, we have tried to impute them by using simple statistics techniques. In particular a *sigma-clipping* procedure (e.g., Akhlaghi and Ichikawa 2015) has been used to determine the days of TS whose overall daily calls belong to the tails of an estimated normality distribution with a 95% of confidence. The normality assumption for the overall daily calls has been confirmed by a standard normality test.

The size of the investigated testing sets made this type of analysis significant only for the SF dataset, for this reason it has been applied only to the latter.

The method detected 5 outlier days out of 193 (4 days corresponding to very low values and 1 day to a very high value). The results reported in Table 12, show that the performance difference between the SLFN model and the SN one tends to increase for difficult-to-handle outliers days. This suggests a certain robustness of the SLFN model with respect to the outliers, even if the special days are not explicitly handled by the method, as done in Barrow and Kourentzes (2018).

6 Concluding remarks

Exploiting exogenous factors may be an option to improve the accuracy of call forecasts in call centers, mostly in modern multi-channel contact centers where many such factors are available. We have shown that both ESNs and SLFNs are able to leverage exogenous time series to improve the 24 hours ahead forecast accuracy of hourly incoming calls with respect to the simple SN method, which only uses data of the target time series. The improvement is particularly significant for the Mean of Maximum Daily Error (MMDE), which is a relevant indicator for staffing decisions. Moreover, the performance gap between the SLFN method and the SN seems to grow in the prediction of statistically detected anomalous days.

We have also documented that, implementing a careful, yet relatively simple, input selection procedure, SLFNs slightly outperform ESNs, which, in principle, have stronger computational power and are more structured to reproduce complex temporal patterns. Additionally, SLFNs turn out to be 2.5 to 4.5 times faster than ESNs (in terms of inference time).

This outcome is also relevant in practice, as the SLFN method looks like a good candidate for industrial implementations. In fact, it is less sensitive to hyperparameter configuration, requires limited computational effort, and can properly be managed by ML non-experts. As an open direction, it would be interesting to investigate the application of the proposed SLFN methodology to different industrial contexts.

Author Contributions AM, FR and SS contributed to the study conception and design. AM and LC contributed to data collection and code development.

Funding Open access funding provided by Università degli Studi dell'Aquila within the CRUI-CARE Agreement.

Data availability All datasets used in the study are freely available on the websites specified in the paper.

Code Availability Custom code.

Declarations

Conflict of interest No conflict of interest or competing interests.

References

Adnan RM, Liang Z, Parmar KS, Soni K, Kisi O (2021) Modeling monthly streamflow in mountainous basin by mars, gmdh-nn and denfis using hydroclimatic data. Neural Comput Appl 33(7):2853–2871

Aguir S, Karaesmen F, Akşin OZ, Chauvet F (2004) The impact of retrials on call center performance. OR Spectr 26(3):353–376

Akhlaghi M, Ichikawa T (2015) Noise-based detection and segmentation of nebulous objects. Astrophys J Suppl S 220(1):1

Aksin Z, Armony M, Mehrotra V (2007) The modern call center: a multi-disciplinary perspective on operations management research. Prod Oper Manage 16(6):665–688

Aldor-Noiman S, Feigin PD, Mandelbaum A et al (2009) Workload forecasting for a call center: methodology and a case study. Ann Appl Stat 3(4):1403–1447

Andrews BH, Cunningham SM (1995) Ll bean improves call-center forecasting. Interfaces 25(6):1–13

Antipov A, Meade N (2002) Forecasting call frequency at a financial services call centre. J Oper Res Soc 53(9):953–960

Arqub OA, Abo-Hammour Z (2014) Numerical solution of systems of second-order boundary value problems using continuous genetic algorithm. Inf Sci 279:396–415

Asuero AG, Sayago A, Gonzalez A (2006) The correlation coefficient: an overview. Crit Rev Anal Chem 36(1):41–59

Avenali A, Catalano G, D'Alfonso T, Matteucci G, Manno A (2017) Key-cost drivers selection in local public bus transport services through machine learning. WIT Trans Built Env 176:155–166

Avramidis AN, Deslauriers A, L'Ecuyer P (2004) Modeling daily arrivals to a telephone call center. Manage Sci 50(7):896–908

Barrow D, Kourentzes N (2018) The impact of special days in call arrivals forecasting: a neural network approach to modelling special days. Eur J Oper Res 264(3):967–977

Bianchi FM, Scardapane S, Uncini A, Rizzi A, Sadeghian A (2015) Prediction of telephone calls load using echo state network with exogenous variables. Neural Netw 71:204–213

Bianchi L, Jarrett J, Hanumara RC (1998) Improving forecasting for telemarketing centers by arima modeling with intervention. Int J Forecast 14(4):497–504

Bishop CM (2006) Pattern recognition and machine learning. Springer, Berlin

Biswas MR, Robinson MD, Fumo N (2016) Prediction of residential building energy consumption: a neural network approach. Energy 117:84–92

Blondel VD, Esch M, Chan C, Clérot F, Deville P, Huens E, Morlot F, Smoreda Z, Ziemlicki C (2012) Data for development: the d4d challenge on mobile phone data. arXiv preprint arXiv:1210.0137

Box GE, Jenkins GM, Reinsel GC, Ljung GM (2015) Time series analysis: forecasting and control. John Wiley & Sons, Hoboken

Brown L, Gans N, Mandelbaum A, Sakov A, Shen H, Zeltyn S, Zhao L (2005) Statistical analysis of a telephone call center: a queueing-science perspective. J Am Stat Assoc 100(469):36–50

Chelazzi C, Villa G, Manno A, Ranfagni V, Gemmi E, Romagnoli S (2021) The new sumpot to predict postoperative complications using an artificial neural network. Sci Rep 11(1):1–12

Chen S, Billings S, Grant P (1990) Non-linear system identification using neural networks. Int J Control 51(6):1191–1214

Cheng YC, Qi WM, Zhao J (2008) A new elman neural network and its dynamic properties. In: CONF CYBERN INTELL S, IEEE, pp 971–975

Deep K, Singh KP, Kansal ML, Mohan C (2009) A real coded genetic algorithm for solving integer and mixed integer optimization problems. Appl Math Comput 212(2):505–518

Dong S, Zhang Y, He Z, Deng N, Yu X, Yao S (2018) Investigation of support vector machine and back propagation artificial neural network for performance prediction of the organic rankine cycle system. Energy 144:851–864

Doya K (1992) Bifurcations in the learning of recurrent neural networks 3. learning (RTRL) 3:17

Elman JL (1990) Finding structure in time. Cognit Sci 14(2):179–211

Engle Granger (1987) Cointegration and error correction: representation, estimation and testing. Econometrica 55:251–276

Fuller WA (2009) Introduction to statistical time series, vol 428. John Wiley & Sons, Hoboken

Gans N, Koole G, Mandelbaum A (2003) Telephone call centers: tutorial, review, and research prospects. Manuf Serv Op 5(2):79–141

Garnett O, Mandelbaum A, Reiman M (2002) Designing a call center with impatient customers. Manuf Serv Op 4(3):208–227

Goodfellow I, Bengio Y, Courville A, Bengio Y (2016) Deep learning, vol 1. MIT press Cambridge, Cambridge

Green LV, Kolesar PJ, Whitt W (2007) Coping with time-varying demand when setting staffing requirements for a service system. Prod Oper Manage 16(1):13–39

Grippo L, Manno A, Sciandrone M (2015) Decomposition techniques for multilayer perceptron training. IEEE T Neur Net Lear 27(11):2146–2159

Ibrahim R, L'Ecuyer P (2013) Forecasting call center arrivals: fixed-effects, mixed-effects, and bivariate models. Manuf Serv Op 15(1):72–85

Ibrahim R, Ye H, L'Ecuyer P, Shen H (2016) Modeling and forecasting call center arrivals: a literature survey and a case study. Int J Forecast 32(3):865–874

Jalal ME, Hosseini M, Karlsson S (2016) Forecasting incoming call volumes in call centers with recurrent neural networks. J Bus Res 69(11):4811–4814

Jongbloed G, Koole G (2001) Managing uncertainty in call centres using poisson mixtures. Appl Stoch Model Bus 17(4):307–318

Karasu S, Altan A, Bekiros S, Ahmad W (2020) A new forecasting model with wrapper-based feature selection approach using multi-objective optimization technique for chaotic crude oil time series. Energy 212:118750

Koole G, Li S (2021) A practice-oriented overview of call center workforce planning. arXiv preprint arXiv:2101.10122

Kourentzes N, Crone SF (2010) Frequency independent automatic input variable selection for neural networks for forecasting. In: The 2010 international joint conference on neural networks (IJCNN), IEEE, pp 1–8

LeCun YA, Bottou L, Orr GB, Müller KR (2012) Efficient backprop. Neural networks: tricks of the trade. Springer, Berlin, pp 9–48

Liao S, Koole G, Van Delft C, Jouini O (2012) Staffing a call center with uncertain non-stationary arrival rate and flexibility. OR Spectr 34(3):691–721

Løkse S, Bianchi FM, Jenssen R (2017) Training echo state networks with regularization through dimensionality reduction. Cogn Comput 9(3):364–378

Lukoševičius M, Jaeger H (2009) Reservoir computing approaches to recurrent neural network training. Comput Sci Rev 3(3):127–149

Maass W, Joshi P, Sontag ED (2007) Computational aspects of feedback in neural circuits. PLoS Comput Biol 3(1):e165

MacKinnon JG (1994) Approximate asymptotic distribution functions for unit-root and cointegration tests. J Bus Econ S 12(2):167–176

Mandelbaum A, Massey WA, Reiman MI, Rider B (1999) Time varying multiserver queues with abandonment and retrials. In: proceedings of the 16th International teletraffic conference, vol 4, pp 4–7

Manno A, Sagratella S, Livi L (2016) A convergent and fully distributable SVMs training algorithm. In: 2016 IEEE IJCNN, IEEE, pp 3076–3080

Manno A, Martelli E, Amaldi E (2022) A shallow neural network approach for the short-term forcast of hourly energy consumption. Energies 15(3):958

Millán-Ruiz D, Pacheco J, Hidalgo JI, Vélez JL (2010) Forecasting in a multi-skill call centre. In: International conference on artificial intelligence and soft computing, Springer, pp 582–589

Nachar N et al (2008) The mann-whitney u: a test for assessing whether two independent samples come from the same distribution. Tutor Quant Methods Psychol 4(1):13–20

Pascanu R, Mikolov T, Bengio Y (2013) On the difficulty of training recurrent neural networks. In: International conference on machine learning, pp 1310–1318

Pearson ES, D'Agostino RB, Bowman KO (1977) Tests for departure from normality: comparison of powers. Biometrika 64(2):231–246

Pearson K (1895) Vii. note on regression and inheritance in the case of two parents. P R Soc London 58(347-352):240–242

Shi Z, Han M (2007) Support vector echo-state machine for chaotic time-series prediction. IEEE T Neural Networ 18(2):359–372

Statista Research Department (2021) call center market. https://www.statista.com/statistics/880975/global-contact-center-market-size/, Accessed: 2021-06-15

Steckley SG, Henderson SG, Mehrotra V (2005) Performance measures for service systems with a random arrival rate. In: Proceedings of the winter simulation conference, 2005., IEEE, pp 10

Sun Y, Peng Y, Chen Y, Shukla AJ (2003) Application of artificial neural networks in the design of controlled release drug delivery systems. Adv Drug Deliver Rev 55(9):1201–1215

Tandberg D, Easom LJ, Qualls C (1995) Time series forecasts of poison center call volume. J Toxicol Clin Toxic 33(1):11–18

Taylor JW (2008) Exponentially weighted information criteria for selecting among forecasting models. Int J Forecast 24(3):513–524

Taylor JW (2010) Exponentially weighted methods for forecasting intraday time series with multiple seasonal cycles. Int J Forecast 26(4):627–646

Wallace RB, Whitt W (2005) A staffing algorithm for call centers with skill-based routing. Manuf Serv Op 7(4):276–294

Weng SS, Liu YH (2006) Mining time series data for segmentation by using ant colony optimization. Eur J Oper Res 173(3):921–937

Zar JH (2005) Spearman rank correlation. Encyclopedia of biostatistics 7

Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. J Roy Stat Soc B 67(2):301–320