Multi-step Time Series Forecasting of Electric Load Using Machine Learning Models

Shamsul Masum(✉), Ying Liu, and John Chiverton

School of Engineering, University of Portsmouth,
Anglesea Building, Anglesea Road, Portsmouth PO1 3DJ, UK
{shamsul.masum,ying.liu,john.chiverton}@port.ac.uk

Abstract. Multi-step forecasting is very challenging and there are a lack of studies available that consist of machine learning algorithms and methodologies for multi-step forecasting. It has also been found that lack of collaborations between these different fields is creating a barrier to further developments. In this paper, multi-step time series forecasting are performed on three nonlinear electric load datasets extracted from Open-Power-System-Data.org using two machine learning models. Multi-step forecasting performance of Auto-Regressive Integrated Moving Average (ARIMA) and Long-Short-Term-Memory (LSTM) based Recurrent Neural Networks (RNN) models are compared. Comparative analysis of forecasting performance of the two models reveals that the LSTM model has superior performance in comparison to the ARIMA model for multi-step electric load forecasting.

Keywords: Time series analysis · Multi-step forecasting · ARIMA
LSTM

1 Introduction

Electric load forecasting plays a vital role in overall operation and planning of power systems. Accurate electric load forecasting helps to run the power system efficiently and effectively. Areas such as cause of power interruptions, coordination between supply and demand, operating costs, maintenance and infrastructure development can benefit from electric load forecasting. Electric load forecasting has therefore been a research subject of great interest for the past few decades [1]. However, electric load data is univariate which makes time series forecasting more challenging which can make it a difficult model to learn [2].

Duration of electric load forecasting can be mainly classified into two categories, short-term load forecasting and long-term load forecasting [3]. Single-step forecasting and multi-step forecasting are useful for short-term and long-term load forecasting respectively [4]. Several machine learning approaches have been considered for forecasting which can be divided into two broad categories, statistical techniques and soft computing techniques [5]. This paper will consider

L. Rutkowski et al. (Eds.): ICAISC 2018, LNAI 10841, pp. 148–159, 2018.
https://doi.org/10.1007/978-3-319-91253-0_15

Auto-Regressive Integrated Moving Average (ARIMA) as a statistical technique and Recurrent Neural Networks (RNN) as a soft computing technique applied here to forecasting electric load.

ARIMA is considered to be the most common approach for time series forecasting. The technique was introduced by Box and Jenkins. An ARIMA model considers the past values of the time series along with the errors in forecasting. It has been found to be effective for short-term forecasting [6,7]. However, it was found that ARIMA model performs better on linear time series data and stationary data compared to nonlinear and nonstationary data [8,9]. The ARIMA approach has been used by several researchers for short-term electric load forecasting [10–13].

RNN is a class of artificial neural network, which has been found to be an effective model for solving many forecasting problems in different applications. Researchers also found RNN as an effective model for electric load forecasting [14,15]. However, RNN seems to have difficulties in learning "long-term dependencies" as explored by Bengio et al. [16] in 1994. Long Short-Term Memory (LSTM) is a special kind of RNN introduced by Hochreiter and Schmidhuber in 1997, which solves the issue of learning long-term dependencies [17]. Applications of LSTM can be found in different areas. Sak et al. in 2014 used LSTM for speech recognition and concluded that LSTM outperforms a deep feed-forward neural network [18]. Marino et al. in 2016 compared standard LSTM and LSTM-based Sequence to Sequence (S2S) architecture for forecasting energy load and found LSTM-based Sequence to Sequence (S2S) performs better than the standard LSTM [19]. Wu et al. in 2016 found LSTM performs better than traditional deep neural network for forecasting wind power [20].

Comparison between ARIMA and RNN has been explored by many researchers in different fields. Ho et al. in 2002 compared ARIMA and RNN models for time series prediction and concluded that RNN performs better than ARIMA [21]. Fu et al. also found RNN model performed better than ARIMA model for predicting traffic flow [22] in 2016. Cao et al. explored RNN models and found that they outperform ARIMA models in forecasting wind speed [23]. Ma et al. compared LSTM with an ARIMA model in forecasting traffic speed prediction in 2015 and found LSTM outperforms the ARIMA model [24]. Tian et al. forecasted short-term traffic flow using LSTM and showed that LSTM performs better than most non-parameteric models [25].

It has been found that both ARIMA and RNN models have mostly been used for short-term electric load forecasting [10–15]. It has also been found that comparisons between ARIMA and RNN model have mostly been conducted for single-step ahead forecasting which is short-term [21–25]. In contrast to this here in this paper, the proposed ARIMA and LSTM forecasting models provide flexibility to forecast over both short-term and long-term. Long-term forecasting in particular, is considered to be a challenging task [26]. Comparison of the two models helps to identify the superiority between two models whilst also considering forecasting using multi-step.

2 Time Series Forecasting Strategies

Time series forecasting is considered to be an important area of machine learning with an ultimate goal of predicting the future. It is also called "*forecasting by exploring the pattern from past data*" [27]. Forecasting of future samples plays a vital role in guiding the decision making of selected areas. Data that is required for time series forecasting are classified into two types: one is time series data and another one is data with time points [28]. Time series data can be described as

$$X = (x_t; t = 1, \ldots, N) \tag{1}$$

where X is the time series, t is time over N observations during that time. Measurement at an individual time point is x_t for time point t. Shumway and Stoffer [29] defined time series as "*a collection of random variables indexed according to the order they are obtained in time*".

2.1 Single-Step Forecasting

Selection of forecasting class and strategy depends on the requirement of the application field and frequency of collected data. Single-step forecasting is applicable where short-term forecasting is required. For example durations of several minutes, hours or days could all be considered short-term. For such a scenario, computing a one step ahead forecast is useful. One step forecast $(t+1)$ is achieved by passing the current and past observations $(t, t-1, \ldots, t-n)$ to a chosen model,

$$F(t+1) = M(o(t), o(t-1), o(t-2), \ldots, o(t-n)) \tag{2}$$

where $F(t+1)$ is the forecast for time $(t+1)$, M is the model and $o(t)$ is an observation at time t.

2.2 Multi-step Forecasting

Multi-step forecasting is useful where the field of application requires long-term duration forecasting. For multiple steps ahead forecast computation Ben Taieb et al. [30] described five multi-step strategies, among them Direct H Step Strategy is considered in the work here.

Direct Strategy. Direct strategy develops N separate forecasting models to forecast N steps. For example, to forecast the next two points of any scenario using a Direct strategy then the first forecast point $F(t+1)$ needs to be calculated through a model and then a different model would be used to forecast the second point $F(t+2)$. However, importantly, the second point is not dependent on the first point estimate. This example can be seen below.

$$F(t+1) = M_1(o(t), o(t-1), o(t-2), \ldots, o(t-n)) \tag{3}$$

$$F(t+2) = M_2(o(t), o(t-1), o(t-2), \ldots, o(t-n)). \tag{4}$$

The direct multi-step strategy can be expressed as

$$y_{t+h} = f_h(y_t, \ldots, y_{t-n+1}) \tag{5}$$

where h is the number of steps to forecast into the future, n is the autoregressive order of the model, f_h is any arbitrary learner.

The direct strategy does not consist of any accumulated errors because it does not use any forecast value as an input. However, it does not guarantee any statistical dependence between forecast points as every model is trained independently [31].

3 ARIMA Based Time Series Forecasting

The acronym of ARIMA is meaningful as it represents the key characteristics of the model which are [32]: AR(Autoregression), relying on a relationship between the current observation and past observations; I(Integrated): differencing of actual observations in order to make the time series stationary; MA(Moving Average): lags of the forecast errors of the moving average model.

These component are included in an ARIMA model as a set of parameters. The standard notation for the ARIMA model is usually given as ARIMA(p, d, q); where p is the number of lag observations, d is the degree of differencing and q is the size of the moving average window. In the ARMA model, the forecast value is a linear compound of past values and past errors, expressed as follows [32],

$$y_t = \theta_0 + \varphi_1 y_{t-1} + \ldots + \varphi_p y_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \ldots - \theta_q \varepsilon_{t-q} \tag{6}$$

y_t is the current measured values at time t; ε_t is the random error at time t; φ_i and θ_j are known as coefficients; and p and q are the autoregressive and moving average specific parameters respectively. If the process of ARMA is dynamic and non-stationary, then a transformation of the series is introduced by Box and Jenkins to make it stationary resulting in the ARIMA model. This is achieved by replacing the measured values y_t with the results of a recursive differencing process $\nabla^d y_t$ where d is the number of times the differencing process has been applied. The first order differencing can be expressed as

$$\nabla^d y_t = \nabla^{d-1} y_t - \nabla^{d-1} y_{t-1}. \tag{7}$$

4 LSTM Based Time Series Forecasting

A traditional neural network considers all inputs and outputs as being independent of each other. For time series forecasting it would be unwise to use such a network. An alternative is to use a Recurrent Neural Network (RNN) which includes consideration of the dependencies for past observations and thus has been found to be more effective for time series forecasting [33]. The typical architecture of an RNN is shown in Fig. 1 which consists of some inputs which are fed into a neural network block A with an output O.

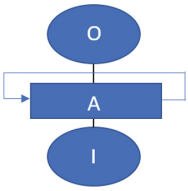

Fig. 1. Simple architecture of RNN

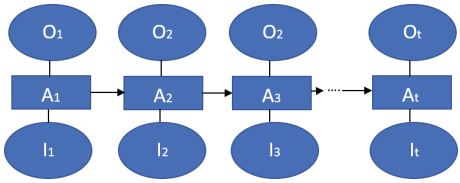

Fig. 2. Unrolled architecture of RNN

NN pass information through a loop. This looping process can be unrolled. The unrolled architecture of a full RNN network is illustrated in Fig. 2 for time steps 1, 2, 3 up to time t; where I_1, I_2, I_3, I_t are the inputs; A_1, A_2, A_3, A_t are the hidden states or the memory of the network; and O_1, O_2, O_3, O_t are the outputs. For example, A_2 is the hidden state at time step 2, where A_2 is calculated using a function (normally tanh) in which the function computes using a previously hidden state A_1 and the current input I_2 with,

$$A_2 = f(UI_2 + WA_1) \tag{8}$$

where U and W are parameters. It can therefore be concluded that RNN does capture information that has been calculated and uses it for long term forecasting. Based on the above description of an RNN it appears to passes some potentially useful properties for long-term forecasting and perhaps it is even capable of handling long-term dependencies. In practice however, these characteristics do not hold as shown by Bengio et al. [16,17]. LSTM introduced by Hochreiter and Schmidhuber [17] in 1997 is bale to learn long-term dependencies better than the RNN architecture.

The motivation behind developing LSTM was to remove the vanishing gradients issues that occur with RNN when processing long-term dependencies. The Standard RNN consists of a chain of repeating modules of the neural network, where each module consists of a structure. For example, in Fig. 3 module has a single tanh layer. Such module structures are found to be simple. Although LSTM has the same chain of repeating modules as an RNN except the LSTM module structure is relatively more complex. Each module consists of four layers rather a single layer as for an RNN module. Figure 4 consists of a simple LSTM network architecture. There modules or memory blocks consist of an input gate, a forget gate, an output gate and the cell state. All these layers interact in a particular way. Information that will be added or removed to the cell state is controlled by three gates. An LSTM network computes a mapping from an input sequence $\boldsymbol{x} = (x_1, \ldots, x_t)$ to an output sequence $\boldsymbol{y} = (y_1, \ldots, y_t)$, where initially, the input gate activation vector $\boldsymbol{i}_t$ and the candidate values of the memory cell state $\tilde{\boldsymbol{c}}_t$ are calculated with,

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i \boldsymbol{x}_t + \boldsymbol{U}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i); \text{ and} \tag{9}$$

$$\tilde{\boldsymbol{c}}_t = \tanh(\boldsymbol{W}_c \boldsymbol{x}_t + \boldsymbol{U}_c \boldsymbol{h}_{t-1} + \boldsymbol{b}_c) \tag{10}$$

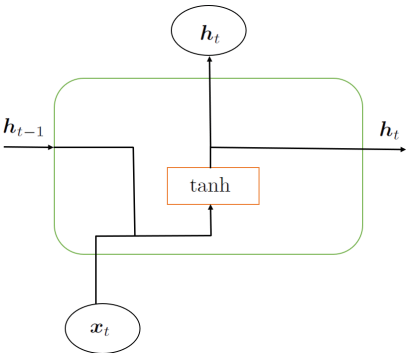

Fig. 3. RNN architecture

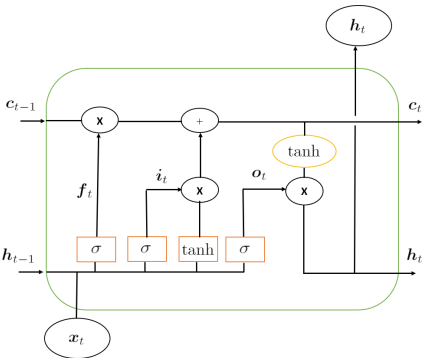

Fig. 4. LSTM based RNN architecture

respectively, where σ is the logistic sigmoid function. The input gate activation vector helps to store new information in the cell state. $\tilde{\boldsymbol{c}}_t$ is a vector of new candidate values created by each tanh layer to be added to the state. The forget gate activation vector $\boldsymbol{f}_t$ is then calculated using,

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f \boldsymbol{x}_t + \boldsymbol{U}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f). \tag{11}$$

The forget gate helps to throw away information from the cell state and it also helps to reset the memory cells. The calculated values of $\boldsymbol{i}_t$, $\tilde{\boldsymbol{c}}_t$ and $\boldsymbol{f}_t$ are then used to calculate the new state of the memory cell $\boldsymbol{c}_t$,

$$\boldsymbol{c}_t = \boldsymbol{i}_t \times \tilde{\boldsymbol{c}}_t + \boldsymbol{f}_t \times \boldsymbol{c}_{t-1}. \tag{12}$$

The cell state works like a conveyor belt which runs through the entire chain. The cell state vector $\boldsymbol{c}_t$ is then used to compute the output gate activation vector $\boldsymbol{o}_t$ via [17,35],

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o \boldsymbol{x}_t + \boldsymbol{U}_o \boldsymbol{h}_{t-1} + \boldsymbol{V}_o \boldsymbol{c}_t + \boldsymbol{b}_o); \text{and} \tag{13}$$

which can then be used to determine the output vector of the LSTM

$$\boldsymbol{h}_t = \boldsymbol{o}_t \tanh(\boldsymbol{c}_t). \tag{14}$$

The output gate helps to compute the output using the cell state along with filtering the cell activations. The weight matrices $\boldsymbol{W}_i$, $\boldsymbol{W}_c$, $\boldsymbol{W}_f$, $\boldsymbol{W}_o$, $\boldsymbol{U}_i$, $\boldsymbol{U}_c$, $\boldsymbol{U}_f$, $\boldsymbol{U}_o$ and $\boldsymbol{V}_o$ are used in Eqs. 9–14 which have to be learnt in the training stage along with $\boldsymbol{b_i}$, $\boldsymbol{b_c}$, $\boldsymbol{b_f}$ and $\boldsymbol{b_o}$ which represent bias vectors. The described LSTM structure does resolve the vanishing gradients issues and has been found to be suitable for long-term dependencies problems [34].

5 Experiments and Results

Multi-step time series forecast analyses on electric load datasets are now performed using the ARIMA model and the LSTM model. Forecast performance of multi-step electric load forecasting of both models are also compared.

5.1 Dataset

Three datasets were obtained from the Open Power System Data on electric load for the Great Britain (GB), Poland (PL) and Italy (IT) [36]. The datasets consist of data of electric load from 2010-02-01 to 2014-01-31 each comprising of 35064 data points with a sampling frequency of 60 min. The sampling frequency of the time series is too granular so downsampling of time series data from 60 min to 1 day was performed using resampling technique [37] for datasets.

5.2 ARIMA Model Formulation

The Dickey-Fuller (ADF) test [4,32] was applied and it was found that the time series data of all three datasets is non-stationary whereas it is important to have the time series data to be stationary for application of the ARIMA forecasting model [4,32]. Seasonal differencing was applied on the three datasets to make the time series stationary as outlined in [4,32]. It was found that before doing the first order differencing the ADF Statistic was found to be more than the 1% critical value and the p-value was close to 0.05 so the null hypothesis of the ADF test could not be rejected. Whereas after the seasonal difference ADF Statistic to be found it was less than the 1% critical value and the p-value was much lower than 0.05 which suggests that the d parameter of the ARIMA model should at least be a value of 1 for better performance. Next, for the other two parameters p and q of the ARIMA model, the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) were computed and plotted following the procedure as outlined in [4,32]. Examining the ACF and PACF plots, the parameters p and q were identified for all three datasets. From the ACF and PACF plots, the 'p' and 'q' values were identified as follows:

- GB Electric Load: $p = 2$ and $q = 2$.
- PL Electric Load: $p = 2$ and $q = 1$.
- IT Electric Load: $p = 2$ and $q = 1$.

Following this, ARIMA $(2, 1, 2)$, ARIMA $(2, 1, 1)$ and ARIMA $(2, 1, 1)$ models were used for the GB Electric Load, PL Electric Load and IT Electric Load datasets respectively.

5.3 LSTM Model Formulation

Selected datasets need to be transformed before fitting in an LSTM model [38]. The transformation of the datasets has been done in three steps. In the first step, the selected time series datasets were divided into input and output components to enable supervised learning to be undertaken. In the Python ecosystem, this was achieved using the shift() function of pandas where, the Previous time steps $(t - n)$ used as an input and current time step (t) used as an output for the observed data [37]. For the second step, the selected time series datasets were then transformed to a stationary time series data set as it is easier to model and it was expected to produce a better forecast. This has been achieved by

applying seasonal differencing of the data as the seasonal trend is visible in the actual data [38]. For the third step, the time series data sets were re-scaled to values between −1 and 1, this is because LSTM model requires data to be within the scale of the activation function of the network. This is achieved using the 'MinMaxScaler class' from scikit-learn Python library [37,38].

Data shape needs to be reshaped as the LSTM requires the input to be in a matrix form with the dimensions: [samples, time steps, features]. The original data set is a 2D array [samples, features] which required transforming to a 3D array [samples, timesteps, features] [38]. This was achieved by fixing the time steps at a value of 1. Simple LSTM have been designed where a network structure consisting of 1 hidden layer with 1 LSTM unit, then an output layer with a linear activation and N output values. Value of N was varied according to the number of time steps over which a forecast was required. The network also represented the 'Root Mean Squared Error (RMSE)' as a loss function and the 'ADAM algorithm' [39] as an optimizer. LSTM is stateful in designed network and the network was fitted with 1 epochs for simplicity. Batch size of the network was set to 1, which is also known as online learning for both training and prediction. All these parameters were used to configure the LSTM model in preparation for forecasting.

5.4 Forecast Performance of the Models

The ARIMA and LSTM models were developed using the Python ecosystem [37]. Statsmodel library was used to fit the ARIMA model by calling ARIMA() along with the p, d, and q parameters [41]. Then fit() and predict() functions were called to train the model and make predictions respectively. SciPy environment with Keras deep learning library using the TensorFlow backend was used for the LSTM model [40]. To perform out of sample forecasting using the ARIMA and LSTM models, the datasets were split into train and test. The training data were used to train the models and the test data were used for performance characterization. Ten different ranges of time points were selected for training, simulating a sweeping action through the data sets across time. This sweeping action helps to determine the potential applicability of the models to a real time prediction scenario. Furthermore, it enables the mean error to be calculated across these different prediction scenarios to determine the generalised performance of the discussed models.

The sampling frequency of the data was a day. For example, if 1 day ahead forecast was required then single-step forecasting was performed and the value of N was 1. However, to forecast 10 days ahead multi-step forecasting was performed and the value of N was 10. To measure the performance of the ARIMA and LSTM models, the RMSE performance measurement technique was used. RMSE can be expressed and calculated using

$$\bar{e} = \sqrt{\frac{\sum_{i=1}^{m} (y_i - \widehat{y}_i)^2}{m}} \tag{15}$$

where, y_i are actual values, $\widehat{y}_i$ are forecast values and m is the number of target output data. Forecast performance of both models on the three datasets are provided in Table 1.

Table 1. Multi-step forecast response of GB, PL and IT electric load

Forecast step (N)	GB		PL		IT	
	ARIMA model	LSTM model	ARIMA model	LSTM model	ARIMA model	LSTM model
1	5269	4009	2154	1416	4615	4087
2	4245	2829	1913	1218	4374	3176
3	4471	2950	2135	1991	5548	3798
4	4821	3491	2328	1788	6441	4887
10	4587	3765	2461	1922	6263	4876

5.5 Discussion

The RMSE performance indicator on various forecast steps of the ARIMA and LSTM models using the three datasets are plotted in Figs. 5, 6 and 7.

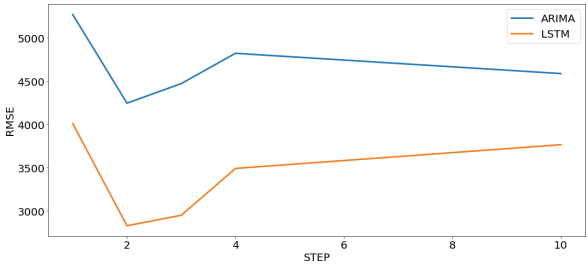

Fig. 5. LSTM and ARIMA model performance on predicting the GB load data.

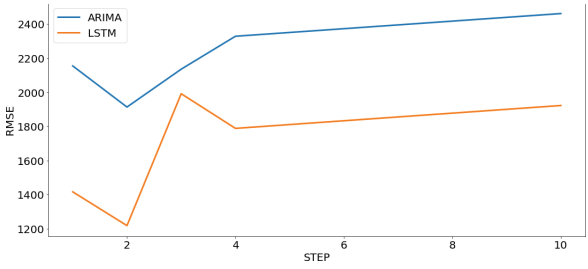

Fig. 6. LSTM and ARIMA model performance on predicting the PL load data.

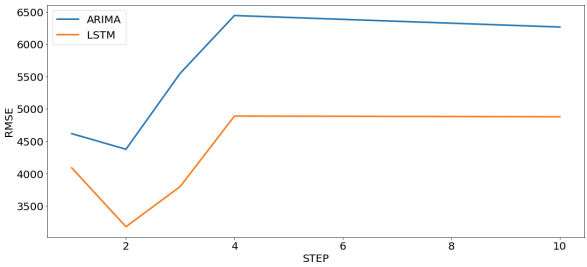

Fig. 7. LSTM and ARIMA model performance on predicting the IT load data.

These show that for every forecast step LSTM provides better performance compared to the ARIMA model for each dataset. It can be concluded that in forecasting single-step and multi-step ahead of selected electric load data sets, LSTM model outperforms ARIMA model.

6 Conclusions

Multi-step forecasting of electric load has been presented in this paper. The experimental results obtained with electric load datasets on the performance of ARIMA and LSTM model are also compared. The comparative analysis revealed that LSTM model outperformed ARIMA model in forecasting multi-step ahead of electric load.

References

1. Mandal, P., Haque, A.U., Meng, J., Srivastava, A.K., Martinez, R.: A novel hybrid approach using wavelet, firefly algorithm and fuzzy ARTMAP for day-ahead electricity price forecasting. IEEE Trans. Power Syst. **28**(2), 1041–1051 (2013)
2. Du Preez, J., Witt, S.F.: Univariate versus multivariate time series forecasting: an application to international tourism demand. Int. J. Forecasting **19**(3), 435–451 (2003)
3. Nataraja, C., Gorawar, M., Shilpa, G., Harsha, J.S.: Short term load forecasting using time series analysis: a case study for Karnataka, India. Int. J. Eng. Sci. Innov. Technol. **1**, 45–53 (2012)
4. Masum, S., Liu, Y., Chiverton, J.: Comparative analysis of the outcomes of differing time series forecasting strategies. In: 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery. IEEE Press (2017)
5. Wang, J.J., Wang, J.Z., Zhang, Z.G., Guo, S.P.: Stock index forecasting based on a hybrid model. Omega **40**(6), 758–766 (2012)
6. Meyler, A., Kenny, G., Quinn, T.: Forecasting Irish inflation using ARIMA models. Published in Central Bank and Financial Services Authority of Ireland Technical Paper Series, vol. 3, p. 148 (1998)
7. Tabachnick, B.G., Fidell, L.S.: Using Multivariate Statistics, 4th edn. Pearson Education, Upper Saddle River (2001)

8. Kam, K.M.: Stationary and non-stationary time series prediction using state space model and pattern-based approach. The University of Texas at Arlington (2014)
9. Lineesh, M., Minu, K., John, C.J.: Analysis of nonstationary nonlinear economic time series of gold price: a comparative study. In: International Mathematical Forum, vol. 5, no. 34, pp. 1673–1683. Citeseer (2010)
10. Liu, K., Subbarayan, S., Shoults, R.R., Manry, M.T., Kwan, C., Lewis, F.L., Naccarino, J.: Comparison of very short-term load forecasting techniques. IEEE Trans. Power Syst. **11**(2), 877–882 (1996)
11. Hagan, M.T., Behr, S.M.: The time series approach to short term load forecasting. IEEE Trans. Power Syst. **PWRS-2**(3), 785–791 (1987)
12. Yang, H.T., Huang, C.M., Huang, C.L.: Identification of ARMAX model for short term load forecasting: an evolutionary programming approach. IEEE Trans. Power Syst. **11**(1), 403–408 (1996)
13. Espinoza, M., Joye, C., Belmans, R., Moor, B.D.: Short-term load forecasting, profile identification, and customer segmentation: a methodology based on periodic time series. IEEE Trans. Power Syst. **20**(3), 1622–1630 (2005)
14. Mandal, J.K., Sinha, A.K., Parthasarathy, G.: Application of recurrent neural network for short term load forecasting in electric power system. In: IEEE International Conference on Neural Networks, vol. 5, pp. 2694–2698 (1995)
15. Senjyu, T., Takara, H., Uezato, K., Funabashi, T.: One-hour-ahead load forecasting using neural network. IEEE Trans. Power Syst. **17**(1), 113–118 (2002)
16. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**(2), 157–166 (1994)
17. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
18. Sak, H., Senior, A., Beaufays, F.: Long short-term memory recurrent neural network architectures for large scale acoustic modelling. In: Fifteenth Annual Conference of the International Speech Communication Association (2014)
19. Marino, D.L., Amarasinghe, K., Manic, M.: Building energy load forecasting using deep neural networks. In: 42nd Annual Conference of the IEEE Industrial Electronics Society (IECON), pp. 7046–7051 (2016)
20. Wu, W., Chen, K., Qiao, Y., Lu, Z.: Probabilistic short-term wind power forecasting based on deep neural networks. In: International Conference on Probabilistic Methods Applied to Power Systems (PMAPS), pp. 1–8 (2016)
21. Ho, S.L., Xie, M., Goh, T.N.: A comparative study of neural network and Box-Jenkins ARIMA modeling in time series prediction. Comput. Ind. Eng. **42**, 371–375 (2002)
22. Fu, R., Zhang, Z., Li, L.: Using LSTM and GRU neural network methods for traffic flow prediction. In: 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), pp. 324–328 (2016)
23. Cao, Q., Ewing, B., Thompson, M.: Forecasting wind speed with recurrent neural networks. Eur. J. Oper. Res. **221**, 148–54 (2012)
24. Ma, X., Tao, Z., Wang, Y., Yu, H., Wang, Y.: Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. Transp. Res. Part C: Emerg. Technol. **54**, 187–197 (2015)
25. Tian, Y., Pan, L.: Predicting short-term traffic flow by long short-term memory recurrent neural network. In: IEEE International Conference on Smart City, Chengdu, pp. 153–158 (2015)
26. Cheng, H., Tan, P.N.: Semi-supervised learning with data calibration for long-term time series forecasting. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, p. I-9. ACM (2008)

27. Molaei, S.M., Keyvanpour, M.R.: An analytical review for event prediction system on time series. In: 2nd International Conference on Pattern Recognition and Image Analysis (IPRIA), pp. 1–6 (2015)
28. Minaei-Bidgoli, B., Lajevardi, S.B.: Correlation mining between time series stream and event stream. In: Fourth International Conference on Networked Computing and Advanced Information Management, Gyeongju, pp. 333–338 (2008)
29. Soyiri, I.N., Reidpath, D.D.: An overview of health forecasting. Environ. Health Prev. Med. **18**(1), 1–9 (2013)
30. Ben Taieb, S., Bontempi, G., Atiya, A.F., Sorjamaa, A.: A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. Expert Syst. Appl. **39**(8), 7067–7083 (2012)
31. An, N.H., Anh, D.T.: Comparison of strategies for multi-step ahead prediction of time series using neural network. In: International Conference on Advanced Computing and Applications (ACOMP), pp. 142–149 (2015)
32. George, E.P.B., Gwilym, M.J., Gregory, C.R., Greta, M.L.: Time Series Analysis: Forecasting and Control. Wiley Publisher, New Jersey (2015)
33. Medsker, L., Jain, L.: Recurrent Neural Networks, Design and Applications. CRC Press LLC, Boca Raton (2001)
34. Graves, A.: Neural networks. In: Graves, A. (ed.) Supervised Sequence Labelling with Recurrent Neural Networks. SCI, vol. 385, pp. 15–35. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-24797-2_3
35. Olah, C.: Understanding LSTM networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs
36. Open power system data. https://data.open-power-system-data.org/timeseries/
37. McKinney, W.: Python for Data Analysis. O'Reilly, Sebastopol (2013)
38. Lewis, N.D.: Deep Time Series Forecasting with Python. Create Space Independent Publishing Platform (2016)
39. Kingma, D.P., Ba, J.L.: ADAM: a method for stochastic optimization. In: ICLR, pp. 1–15 (2015)
40. Chollet, F.: Keras, GitHub repository (2015). https://github.com/keras-team/keras
41. Seabold, S., Josef, P.: Statsmodels: econometric and statistical modeling with Python. In: The Proceedings of the 9th Python in Science Conference (2010)