Short-Term Load Forecasting: An Intelligent Approach based on Recurrent Neural Network

Atul Patel [1], Monidipa Das [2⋆], and Soumya K. Ghosh [2]

[1]Department of Electrical Engineering
[2]Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur, India-721302
e-mails: atulp.iitkgp@gmail.com; monidipadas@hotmail.com; skg@cse.iitkgp.ac.in

Abstract. With the evolution of smart grids in recent years, load forecasting has received more research focus than ever before. Several techniques, especially based on artificial neural network and support vector regression, have been proposed for this purpose. However, due to lack of appropriate modeling of external influences over the load data, the performance of these techniques remarkably deteriorates while making forecast for the peak load values, especially on short-term basis. In this paper, we present a strategy to forecast hourly peak load using Recurrent Neural Network with Long-Short-Term-Memory architecture. The novelty lies here in improving the forecast accuracy by an intelligent incorporation of available domain knowledge during the forecast process. Experimentation is carried out to forecast hourly peak load in five different zones in USA. The experimental results are found to be encouraging.

Keywords: Short-term load forecasting, RNN-LSTM, Peak load, Smart grid, Domain knowledge

1 Introduction

Load forecasting is an active area of research since 1960s. This provides insight into future consumption of power load, based on observed data as well as consumer behavior, and eventually, helps a lot in pricing, utility planning and distribution of power in effectual way [2]. Even a fractional increase in load forecast accuracy can have a significant effect on improving a country's economy. As a consequence, the power load forecasting still remains a popular area of research in the present background of twenty first century.

One of the key factors playing important role in power load forecasting is the *timescale*. On the basis of timescale, the load forecasting can be divided into three broad categories [7], namely *Short-term load forecast* (STLF), *Mid-term load forecast* (MTLF), and *Long-term load forecast* (LTLF). STLF is done for very short duration of time. It can be a few minutes, hours, a day, or even a week. The primary aim of STLF is planning of power exchange and optimal generator

⋆ Corresponding author

unit commitment. It can also aid in addressing real-time control and security assessments of the plant. MTLF is made for a month to a year or two. This helps in scheduling maintenance, coordinating load dispatches, and also maintaining a balance between supply and demand. LTLF is done for few years (> 1 year) to 10–20 years ahead. Major decisions regarding generation, transmission, and distribution of power are made based on the results of LTLF.

Problem Statement and Challenges: In the present paper, we focus on short-term forecast of peak load on hourly basis. Given the time series of hourly peak load data $y_1, y_2, \cdots, y_t$ over t time stamps (hours), the goal is to predict the peak load for the next m time stamps, i.e. $y_{(t+1)}, y_{(t+2)}, \cdots, y_{(t+m)}$ on hourly basis. The task is not as trivial as it seems. Though extensive research efforts have been made so far to improve the performance of peak load forecasting, the area still retains substantial research importance because of a number of challenges prevailing over here. Apart from the highly complex and non-linear nature of the electric load data, the other challenges in short-term peak load forecasting arise due to its dependency on seasonal and social factors. In majority of the cases, it becomes difficult to acquire the relevant data on influencing factors (such as change in temperature, humidity, customer behavior etc.) and accurately fit these into a forecasting model. Hence, the current research thrust is to come up with a complementary method that can better utilize the available data and can help improving the model performance even when the data on influencing factors are unavailable.

Our Contributions: In the present work, we attempt to address the above-mentioned issues by exploiting the power of computational intelligence and available domain knowledge. Our major contributions in this context are as follows:

- proposing an hourly peak load forecasting approach based on RNN with long-short-term-memory (LSTM) architecture;
- devising an intelligent way of improving RNN performance with incorporated domain knowledge;
- proposing a mechanism for dynamic updating of rule base in a knowledge based system;
- validating the effectiveness of the proposed approach with respect to forecasting hourly peak load in five different zones in USA;

The rest of the paper is organized as follows. Section 2, reviews the existing works on short-term load forecasting. Section 3 discusses on the fundamentals of the recurrent neural network with long-short-term memory architecture. Section 4 thoroughly describes our proposed approach for hourly peak load forecasting. The details of experimentation along with the results of hourly load forecast are presented in Section 5, and finally, we conclude in Section 6.

2 Related Works

Short-term load forecasting (STLF) is quite an widely investigated research area. As per the recent surveys [5], most of the existing STLF models are defined either

on classical auto-regressive models or on artificial neural network (ANN) based machine learning techniques. The classical models mostly suffer from the issue of modeling non-linearity within load time series data, which is addressed by the ANN models [1, 8] with their ability to efficiently analyze non-linear problems. Unfortunately, due to the over-fitting and curse-of-dimensionality issues, the ANN-based load forecasting models are often found to produce poor prediction performance in many of the load forecasting scenarios [2]. In order to tackle these issues, a number of support vector machine (SVM)-based models [10] have been proposed in recent days. Nevertheless, the modeling of influences from external factors still remains a challenge for accurate load forecasting. Incidentally, to the best of our knowledge, the issue of modeling external influences in absence of the relevant data has not yet been addressed in any of the existing works.

3 An Overview of Recurrent Neural Networks

3.1 Recurrent Neural Networks (RNNs)

RNNs are the exclusive cases of feed forward neural networks, where the hidden units are connected in such a way that it forms a directed cycle (recurrent connection) thus allowing the network models to exhibit dynamic temporal behavior. One of the major benefits of having such recurrent connection is that the memory of previous inputs remains within the networks internal state. This makes RNNs applicable to various complex problems of sequence to sequence learning. Typically, the current input x_t is multiplied with weight u and then is added to the product of the previous output y_{t-1} and corresponding weight w. This value is passed through $tanh$ nonlinearity to generate the current output.

$$y_t = tanh(wy_{t-1} + ux_t) \tag{1}$$

The simplest RNN can be visualized by unrolling the time axis of a fully connected neural network (refer to Fig. 1).

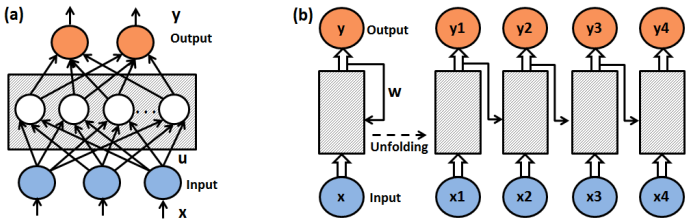


Fig. 1: Neural network variants: (a) Feed-forward model, (b) Recurrent model

3.2 RNN with Long-Short-Term Memory (LSTM)

The Long-Short-Term Memory (LSTM) architecture [6] of RNN is primarily proposed to overcome the issues of vanishing/exploding gradient and lack of ability to capture the long-term dependencies in standard RNN. LSTM can enforce constant error flow through constant error carousels within special units by bridging minimal time lags in excess of 1000 discrete time steps. Typically, the LSTM architecture consists of three gates, as illustrated next.

- **Forget Gate:** The forget gate concatenates the previous hidden state (h_{t-1}) at $t-1$ and the current input (x_t) at time t into a single tensor. It then passes it through a sigmoid function (σ) after applying a linear transformation. If the output of the forget gate is 1, then it completely forgets the previous state, otherwise if it is 0, then the previous internal state is passed as it is. Accordingly, the return vector of forget gate can be represented as follows:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2}$$

  where, b_f and W_f are the bias and weight vector for the forget gate.

- **Input Gate:** In the input gate, the current input (x_t) and the previous hidden state (h_{t-1}) are concatenated and passed through another sigmoid layer. The return vector of this gate can be represented as follows:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{3}$$

  where, b_i and W_i are the bias and the weight vector for the input gate. Once the input return vector is determined, the candidate layer applies a *tanh* nonlinearity to the current input and the previous output in the LSTM cell and generates a candidate vector in following manner:

$$\widehat{C}_t = tanh(W_c[h_{t-1}, x_t] + b_c) \tag{4}$$

  where, b_c is the bias and W_c is the weight vector for the candidate layer. After the current candidate value is determined, it is added to the fraction of the old cell state $C_{(t-1)}$ as allowed by the forget gate, to produce the updated cell state: $C_t = f_t * C_{(t-1)} + i_t * \widehat{C}_t$

- **Output Gate:** The output gate controls what fraction of the internal state is passed to the output. The return vectors from the output gate is expressed below, where, b_o and W_o are the corresponding bias and the weight vector.

$$O_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{5}$$

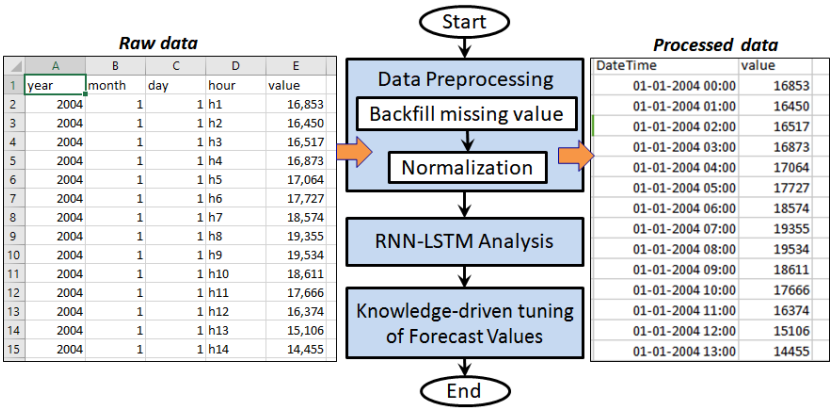
$$h_t = O_t * tanh(C_t) \tag{6}$$

4 Hourly Peak Load Forecasting: Proposed Approach

The overall flow of the proposed forecast model is depicted in Fig. 2. As shown in the figure, the approach is comprised of three major steps: 1) *Data pre-processing*, 2) *RNN-LSTM analysis*, and 3) *Knowledge-driven tuning of forecast value.*

4.1 Data Pre-processing

The primary objective of this step is to convert the input dataset into desired format: $< dd{-}mm{-}yyyy \;\; hh{-}mm, peakLoadValue >$ (refer to Fig. 2). Further, the step also processes for the missing instances in the dataset. Accordingly, the whole dataset is re-sampled at one hour frequency and the missing values are filled out using backfill method [9] in order to have consistency in the dataset. Since RNN depends on scale of data, in the pre-processing step, we also normalize the dataset to have values in the range of 0 to 1.



	A	B	C	D	E
1	year	month	day	hour	value
2	2004	1	1	h1	16,853
3	2004	1	1	h2	16,450
4	2004	1	1	h3	16,517
5	2004	1	1	h4	16,873
6	2004	1	1	h5	17,064
7	2004	1	1	h6	17,727
8	2004	1	1	h7	18,574
9	2004	1	1	h8	19,355
10	2004	1	1	h9	19,534
11	2004	1	1	h10	18,611
12	2004	1	1	h11	17,666
13	2004	1	1	h12	16,374
14	2004	1	1	h13	15,106
15	2004	1	1	h14	14,455

DateTime	value
01-01-2004 00:00	16853
01-01-2004 01:00	16450
01-01-2004 02:00	16517
01-01-2004 03:00	16873
01-01-2004 04:00	17064
01-01-2004 05:00	17727
01-01-2004 06:00	18574
01-01-2004 07:00	19355
01-01-2004 08:00	19534
01-01-2004 09:00	18611
01-01-2004 10:00	17666
01-01-2004 11:00	16374
01-01-2004 12:00	15106
01-01-2004 13:00	14455

Fig. 2: Flow of the proposed forecast approach

4.2 RNN-LSTM Analysis

The initial forecast for the hourly peak load is obtained using the RNN-LSTM, as described in Section 3.2. Thus, the forecast for the next time stamp $(t+1)$ becomes as follows:

$$y_{t+1} = softmax(O_{t+1}) \tag{7}$$

$$O_{t+1} = \sigma(W_o[h_t, x_{t+1}] + b_o) \tag{8}$$

where, O_{t+1} is the un-normalized output which is further normalized using $softmax$ function to obtain the forecast value $y_{(t+1)}$ of the hourly peak load. The value of h_t is determined by following the eq. 6 as illustrated in Section 3.

4.3 Knowledge-driven tuning of Forecast values

As established in literature, various factors, including the weather condition and customer behavior can have significant influence on short-term load forecasting. However, the relevant data are not always available in practice. In this context, we propose a novel technique of utilizing our generic domain knowledge to indirectly extract such influence pattern from the given load time series data. As per the domain knowledge, the load demand during Summer and Winter is more, compared to that in Autumn and Spring (refer Fig. 3 (a)). Even, the load demand is different during different hours of the day. For example, the load demand during normal working office hours is higher compared to other time (refer Fig. 3 (c)). Also, it is evident from Fig. 3 (b) that the load demand on Weekends is less than the load demand on weekdays. This is so because most of the workplaces are closed on weekends. Accordingly, we use this knowledge to indirectly determine the effect of weather factors and customer behavior on the power load variation of any zone.

Feature Creation: In order to utilize the domain knowledge available with us, we create four new features so as to tune the forecast values, and to further improve our forecast accuracy. The new features are as follows:

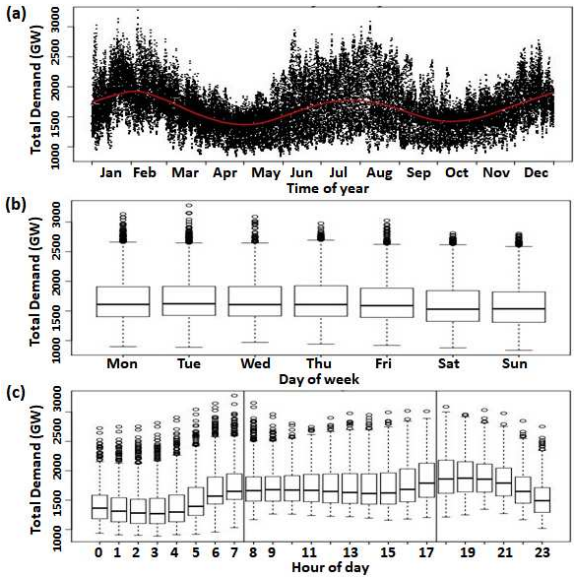


Fig. 3: Total demand on (a) daily basis, (b) weekly basis , (c) hourly basis[11]

- Difference = load_value[i+1] - load_value[i]
- Season (S) : Winter ⇒ December, January, February
  Summer ⇒ June, July, August
  Autumn ⇒ September, October, November
  Spring ⇒ March, April, May
- Time of the day (T) = Morning, Day, Evening, Night
- Weekend (W) = Binary variable indicating the given day is a weekend or not.

$$W = \begin{cases} 1 & : Weekend \\ 0 & : Otherwise \end{cases}$$

Fine-tuning forecast values: In order to fine-tune the forecast value obtained from RNN-LSTM analysis, first we calculate the average difference between all consecutive pairs of observed load values, considering all the possible combinations of possible *Season, Time of the day,* and whether it is *Weekend* or not. Then these average values are added to the forecast values based on a *dynamically* changing rule-base. Typically, each rule is generated in following form:

If Season=S & Time=T & Weekend=W **Then** tuning-component=Dict(S,T,W)

where $Dict(S, T, W)$ is a function of $Difference$ (see feature creation). Even though it looks trivial, the process helps extracting and incorporating domain knowledge in our forecast model, and thereby, helps in improving the forecast accuracy. In general, the electricity demand is significantly affected by customer consumption behavior that changes almost arbitrarily with the change in weather conditions (temperature, humidity etc.), random occurrence of social events (e.g. special game series, festivals, party etc.), change in individual work-load over a

week, and so on. However, the lack of data availability often becomes the biggest issue for direct modeling of influences of these factors in a load forecast model. Contrarily, the present process employs a data-driven technique for an *indirect* as well as *smart* extraction of the pattern of how load demand changes with the customer-behavior change according to the variation in seasonal factors and day-to-day social activities. Our approach for knowledge-driven fine-tuning of forecast value is presented in Algorithm 1.

Algorithm 1 Knowledge-driven Forecast Value Tuning

```
/* Initialization
Season := [Winter, Spring, Summer, Autumn]
Time := [Morning, Day, Evening, Night]
Weekend := [0, 1]
for S ∈ Season do
    for T ∈ Time do
        for W ∈ Weekend do
            Count = 0; Sum = 0;
            for i ∈ 1:nrow(TrainData) do
                if Season=S & Time=T & Weekend=W then
                    Sum += Difference[i];
                    Count++;
                end if
            end for
            Dict(S,T,W) = Sum/Count;  /* Dynamically changing rule-base
        end for
    end for
end for

for S ∈ Season do
    for T ∈ Time do
        for W ∈ Weekend do
            for i ∈ 1:nrow(TestData) do
                if Season=S & Time=T & Weekend=W then
                    Forecast[i] += Dict(S,T,W);  /* Fine-tuning forecast value
                end if
            end for
        end for
    end for
end for
```

5 Experimental Evaluation

5.1 Study Area and Dataset

The effectiveness of our proposed knowledge-driven RNN-LSTM model is validated with respect to hourly load forecasting using the dataset from Kaggle "Global Energy Forecasting Competition" held in 2012[1]. The reason for using this publicly available and well known load forecasting dataset is to allow other researchers to easily compare their models to our proposed method. The dataset consists of zone-wise load history of USA, among which we have considered the data of 5 zones (Zone-1 to Zone-5) to build and test our model.

5.2 Experimental Setup

The proposed model is evaluated in comparison with four baselines, namely statistical ARIMA, NARNET (non-linear autoregressive neural network), RNN,

[1] `https://www.kaggle.com/c/global-energy-forecasting-competition-2012-load-forecasting`

and RNN-LSTM models. The proposed forecasting model (knowledge-driven RNN with LSTM architecture) along with normal RNN, RNN-LSTM, and ARIMA models are implemented using python (Flavor: Anaconda Python and IDE: Jupyter notebook). For the deep learning part, Keras[2] is used. On the other side, for NARNET, we have used the library function of MATLAB NN-Toolbox[3]. The model building, training and testing are carried out in a 64-bit PC with windows 10 OS and 4GB RAM. The typical configuration of our model is as follows: one hidden layer having 16 LSTM blocks; one output layer, predicting the hourly load. The dataset is split into 67% training set and 33% test set. For data pre-processing, LSTM-based forecast, and knowledge-driven tuning, we follow the same convention as exemplified in the respective subsections within Section 4.

All the considered models are evaluated with respect to two popular statistical goodness-of-fit criteria, namely NRMSD (normalized root mean squared deviation)[3] and MAPE (mean absolute percentage error)[4]. Additionally, we also perform correlation study over the forecast from the considered NN-based models and the actual load values.

Table 1: Comparison of model performance for different zones in USA

Study Zone	**Metrics**	**RNN (LSTM)**	**RNN**	**NARNET**	**ARIMA**	**Proposed Approach**
Zone 1	NRMSD	19.574	32.604	21.325	26.71	**15.916**
	MAPE	7.112	25.321	31.469	23.680	**6.815**
Zone 2	NRMSD	19.246	39.220	31.077	29.898	**18.058**
	MAPE	4.043	15.871	23.598	17.418	**3.728**
Zone 3	NRMSD	18.842	26.875	32.331	29.898	**18.066**
	MAPE	4.035	16.432	23.925	17.418	**3.718**
Zone 4	NRMSD	14.415	37.842	36.839	39.688	**12.297**
	MAPE	6.474	15.079	37.839	14.635	**5.997**
Zone 5	NRMSD	21.172	30.188	30.659	20.818	**18.477**
	MAPE	9.075	20.918	41.121	20.443	**8.392**

5.3 Results and Discussions

The results of comparative study are summarized in Table 1 and in Figs. 4-5. On analyzing the results, the following inferences can be drawn:

- As shown in Table 1, for all the considered study zones, the proposed model produces small NRMSDs and MAPEs, which are even lesser than that of the benchmark RNN-LSTM model. This indicates superiority of our knowledge-driven LSTM variant over all other considered models.
- The high value of correlation (refer Fig. 5) reveals that the hourly load series forecasts made by our model have best match with the observed load time series. [Due to the page limitation, we have included the results of correlation study only for the Zone-1.]
- Finally, as depicted in Fig. 4, the average percentage improvement (in reducing error) for the proposed forecast model with respect to standard RNN-LSTM, RNN, and NARNET models are 9%, 59%, and 63%, respectively.

[2] `https://github.com/keras-team/keras`
[3] `https://se.mathworks.com/help/deeplearning/ref/narnet.html`

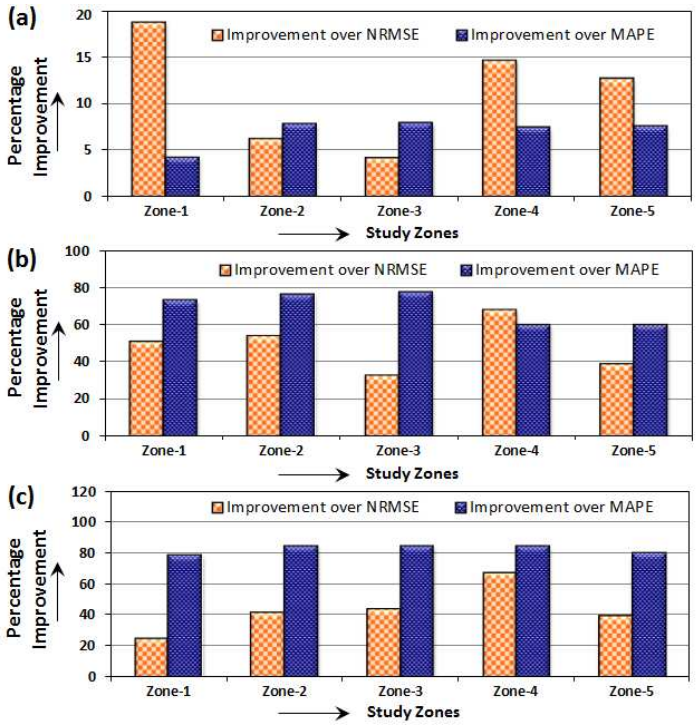


Fig. 4: Percentage improvement in forecast error compared to various NN variants: (a) RNN-LSTM, (b) RNN, (c) NARNET

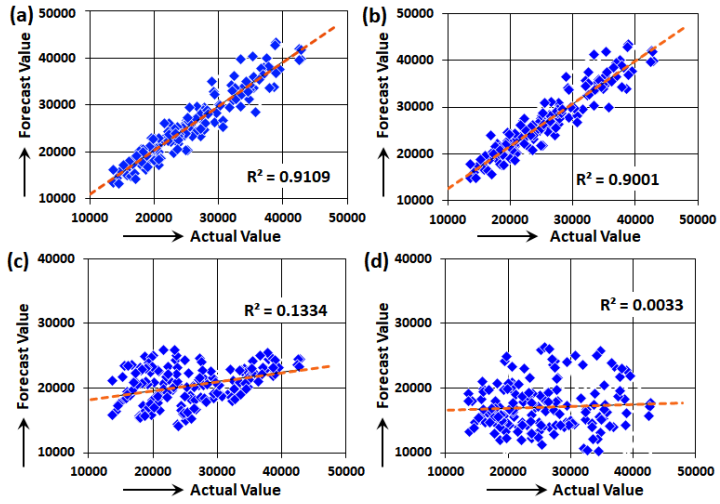


Fig. 5: Correlation between actual and forecast values for the considered NN variants (Zone-1): (a) Proposed model, (b) RNN-LSTM, (c) RNN, (d) NARNET

Overall, in comparison with the baselines, our proposed knowledge-driven RNN-LSTM variant is found to show improved performance with respect to all the considered metrics. This demonstrates that the factor contributing to the increased accuracy of our model is nothing but the intelligent incorporation of the domain knowledge, which is the main contribution of this work.

6 Conclusions

In this paper, we have proposed a novel variant of short-term load forecasting (STLF) strategy based on RNN with LSTM architecture. The uniqueness of the proposed method remains in embedding available domain information to tune the forecast values. The promising results of experimental study demonstrate significant improvement in forecast accuracy due to incorporation of domain knowledge in the forecast process. In future, we plan to upgrade this model with added feature for utilizing spatial auto-correlation among neighboring zones.

References

1. Baliyan, A., Gaurav, K., Mishra, S.K.: A review of short term load forecasting using artificial neural network models. Procedia Computer Science 48, 121–125 (2015)
2. Ceperic, E., Ceperic, V., Baric, A.: A strategy for short-term load forecasting by support vector regression machines. IEEE Transactions on Power Systems 28(4), 4356–4364 (2013)
3. Das, M., Ghosh, S.K.: Deep-STEP: A deep learning approach for spatiotemporal prediction of remote sensing data. IEEE Geoscience and Remote Sensing Letters 13(12), 1984–1988 (2016)
4. Das, M., Ghosh, S.K.: Spatio-temporal prediction of meteorological time series data: An approach based on spatial Bayesian network (SpaBN). In: International Conference on Pattern Recognition and Machine Intelligence. pp. 615–622. Springer (2017)
5. Fallah, S.N., Ganjkhani, M., Shamshirband, S., Chau, K.w.: Computational intelligence on short-term load forecasting: A methodological overview. Energies 12(3), 393 (2019)
6. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)
7. Khuntia, S.R., Rueda, J.L., van der Meijden, M.A.: Forecasting the load of electrical power systems in mid-and long-term horizons: a review. IET Generation, Transmission & Distribution 10(16), 3971–3977 (2016)
8. Khwaja, A., Zhang, X., Anpalagan, A., Venkatesh, B.: Boosted neural networks for improved short-term electric load forecasting. Electric Power Systems Research 143, 431–437 (2017)
9. Koubli, E., Palmer, D., Rowley, P., Gottschalg, R.: Inference of missing data in photovoltaic monitoring datasets. IET Renewable Power Generation 10(4), 434–439 (2016)
10. Mitchell, G., Bahadoorsingh, S., Ramsamooj, N., Sharma, C.: A comparison of artificial neural networks and support vector machines for short-term load forecasting using various load types. In: Manchester PowerTech. pp. 1–4. IEEE (2017)
11. Taieb, S.B., Hyndman, R.J.: A gradient boosting approach to the kaggle load forecasting competition. International journal of forecasting 30(2), 382–394 (2014)