Time Series Forecasting on Solar Irradiation using Deep Learning

Murat Cihan Sorkun[1], Christophe Paoli[1], Özlem Durmaz Incel[1]

[1]Galatasaray University, Ortakoy, Istanbul 34349, Turkey
mcsorkun@gmail.com, cpaoli@gsu.edu.tr, odincel@gsu.edu.tr

Abstract

Time series forecasting is currently used in various areas. Energy management is also one of the most prevalent application areas. As a matter of fact, energy suppliers and managers have to face with the energy mix problem. Electricity can be produced from fossil fuels, from nuclear energy, from bio-fuels or from renewable energy resources. Concerning electricity generation system based on solar irradiation, it is very important to know precisely the amount of electricity available for the different sources and at different horizons: minutes, hours, and days. Depending on the horizon, two main classes of methods can be used to forecast the solar irradiation: statistical time series forecasting methods for short to midterm horizons and numerical weather prediction methods for medium to long-term horizons. On this paper we focus only on statistical time series forecasting methods. The aim of this study is to assess if deep learning can be suitable and competitive on the solar irradiation data time series forecasting. In this context, studies using deep learning and other machine learning methods for time series forecasting were investigated. A special Recurrent Neural Network variations Long Short-Term Memories and Gated Recurrent Unit models are introduced.

1. Introduction

While the energy demand on the earth increases day by day, the resources on the earth are inadequate to meet this demand. The reserves of non-renewable energy sources such as coal, oil and natural gas are steadily declining. That is why the trend towards renewable energies in the global sense has emerged. When it can be used efficiently, solar energy has the highest potential from these sources. The development of technology and the reduction of the infrastructure costs required to make use of solar energy are increasing the trend towards this area.

The most essential handicap of systems based on solar energy is that the amount of solar irradiation cannot be estimated depending on many variables. At this point time series forecasting plays a key role in short prediction horizon: a couple of hours. There are number of models that are being conducted to forecast solar irradiation with time series, such as combination of autoregressive and dynamical system models [1], Autoregressive Integrated Moving Average (ARIMA) model [3], Nonlinar Autoregresive (NAR) neural network [2], and Multilayer Perceptrons (MLP) [4, 5].

The disappearance of the hardware boundaries and the production of large volumes of data have led to the widespread use of deep models. In the literature, there are many deep learning models are applied for time series forecasting. These include: Deep Belief Networks (DBN) [7, 8, 9], Stacked Auto Encoders [10, 13], and Long Short-Term Memory (LSTM) units [11, 12, 13]. In this study, a special Recurrent Neural Network variations LSTM and Gated Recurrent Unit (GRU) models are used to time series forecast on solar irradiation data.

This study is organized as follows. In the next section, time series forecasting concepts and methods are described. Deep Learning and Recurrent Neural Network variations are introduced in section 3 and section 4 contains the data and the experiments. Finally, section 5 concludes the findings of the study.

2. Time Series Forecasting

Time series are sequential data that are measured at certain intervals with respect to any process. The intervals used in the time series may be of different sizes, provided that they are equally divided. They are usually measured at intervals such as hourly, daily, monthly and yearly. The annual population, the daily number of passengers on the metro and the hourly exchange rate are examples of time series. In the time series, records must be ordered chronologically. Each record may contain information about one or more features. Univariate term is used for time series containing single information, and multivariate is used for data containing more than one data.

In this context time series forecasting can be defined as the prediction of the future using time series data of the past. Time series data are used for creating model by different methods. The process of creating the model by formulating the data appropriately is called time series analysis. Forecasting is carried out through these models. In this section, time series forecasting models are explained.

2.1. Forecasting Methods

A lot of research has been done on the time series forecasting and many studies are still ongoing. In these studies, different methods have been used for forecasting. Different methods for time series in different characteristics can be successful. In this section, state of the art forecasting methods in the literature are introduced.

2.1.1. Naïve Method

The simplest and cost-effective method that can be used in time series is the naive approach. This method uses the last observed data as the next prediction. Naïve method just ignores the historical data as it uses only the last instance. Although it can achieve successful results over low variance time series, it does not yield reliable results. Generally it is used as a reference to compare with other methods [15, 16, 17].

2.1.2. Linear Models

Linear models try to predict future data with a linear function using past time series data. The most commonly used linear methods are Auto Regressive (AR), Mean Average (MA), and methods used as a hybrid, such as ARMA and ARIMA.

ARMA is a combination of AR and MA models. It is used to forecast stationary and univariate time series. In an AR model the future value of a variable is assumed to be a linear combination of past observations and a random error together with a constant [14].The MA model predicts the future by taking the average of the nearest number of data from the past data.

ARIMA model is a more generalized version of the ARMA model. ARIMA model can also be applied on non-stationary time series with differencing method. It is the most widely used linear model and has been used as a comparison reference [18, 19].

2.1.3. Nonlinear Models

When time series show a linear structure, linear prediction models can be used. However, many time series have a nonlinear structure. Many nonlinear models have been proposed in the literature in order to come up to the limitations of linear models: Support Vector Machine, Artificial Neural Network, and Random Forest.

Support Vector Machine (SVM) is a supervised learning algorithm that has been successfully implemented on classification, approximation, and time series forecasting problems. It is possible to separate two groups by drawing a border between two groups in the plane. The place where this boundary can be drawn is that the two groups must be the most distant and apart from each other. The SVM determines how this boundary is drawn. SVM models generated for multiple inputs can produce nonlinear solutions. In many studies that forecast time series, use SVM [9, 18].

Artificial Neural Network (ANN) is a model inspired from the architecture of the human brain. ANN provides complex quadratic and polynomial functions to be easily represented. It is one of the most commonly used models for predicting nonlinear time series. The ANN architecture consists of three layers: the input layer, the hidden layer, and the output layer. Each layer has fully connected nodes to the next layer. These connections called weights are calculated by backpropagation method. There are many studies in the literature on time series forecasting using ANN [4, 5, 20].

Another structure that is often used in forecasting time series is decision trees. Random Forest is an ensemble decision tree method in which multiple prediction trees are constructed with different random variables [21]. Random Forest is a fast and extremely resistant method to over fitting problem. The system can be created using as many trees as desired. There are many time series forecasting studies in the literature using Random Forest [22, 23].

3. Deep Learning

Deep Learning (DL) is a machine learning method using multi-layer deep ANN architectures. As specified in Section 2.3.3.2, ANN consists of three layers, the input layer, the hidden layer and the output layer. DL is the ANN model with multiple hidden layers. The first study involving the DL structure was carried out by Ivakhnenko and Lapa in 1965 [24]. Even if the DL concept was based on the 1960s, the rise took place in recent years. This is because there is no processor power to train the deep architects, and there is no sufficient amount of data. Nowadays, with the increase of processor power and generation of enormous dimensions of data generated by digitization has provided the necessary infrastructure for the DL. These developments have enabled DL to become widely used in the field, such as computer vision, text processing, translation, and time series prediction.

3.1. Recurrent Neural Networks

Recurrent Neural Network (RNN) is a special kind of neural network developed for processing sequential data. First studies were conducted in the 1980s [25, 26]. In traditional structures, each sample is trained independently of each other, but this training is not a sufficient method for text, sound, image, and other data related to time. Independent training is not enough to preserve this knowledge because sequence information is also contained within sequence data. RNN offers this probing solution by taking inputs sequentially.

Unlike other FNNs, the RNN has feedback connections in the hidden layer units. With this feature, it can perform temporal processing and learn sequences. The hidden layer acts as a memory and can store sequential information. The RNN architecture can be transformed into an Feedforward structure by spreading over time (see Fig. 1.). On this track, the RNN can be trained with a backpropagation version, a Backpropagation Through Time (BPTT) learning algorithm.

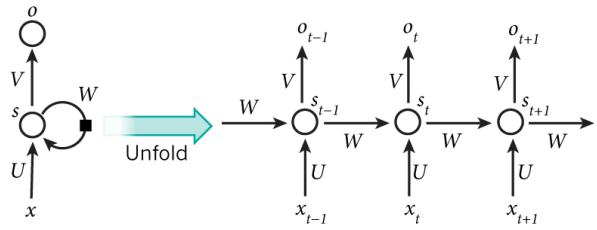

Fig. 1. Unfolding RNN over Time [6]

In addition to the inputs from the previous layer, each unit in the hidden layer receives as input the output value of the previous phase result. This feedback value is multiplied by its own weight, just like other inputs, and sent to the activation function. The function for the output calculation in the units in the hidden layer is as follows:

$$h(t)=f_H(W_{HH} * x(t) + W_{IH} * h(t-1)) \quad (1)$$

where f_H is the activation function of the hidden layer, $x(t)$ is the input from the previous layer, W_{IH} is the weight of the links from previous layer, $h(t-1)$ is the feedback output calculated in the previous time step and W_{HH} is the weight of this feedback output.

Studies have shown that Simple RNN can store limited historical data [27]. Some data may be dependent on remote history data, which has led to a long-term dependency problem. To solve this problem, a customized RNN structures have been developed: Long Short-Term Memory Unit (LSTM) and Gated Recurrent Unit (GRU).

3.1.1. Long Short-Term Memory Unit

LSTM is a customized RNN model developed by Hochreiter and Schmidhuber in 1997 [28]. LSTMs can keep track of long-term information through the gates they contain. In an LSTM unit there are basically three gates, which determine what information to store, which are input gate, forget gate and output gate. The input gate specifies which of the incoming input values will be stored in the next state. Forget gate determines which of the previous state information will no longer be stored. The output gate specifies which of the information in the new state will be sent as output. The gates that make up the LSTM unit are shown in Fig. 2., where i, f and o represent input gate, forget gate and output gate, respectively, C represents the unit state, and $\hat{C}$ represents the next candidate state.

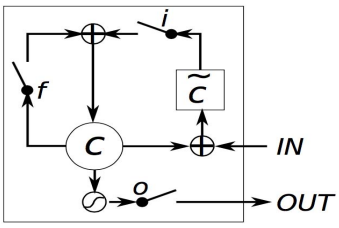

Fig. 2. LSTM Unit [29]

The equations used to calculate the next output and state values in the LSTM unit are as follows:

$$f_t = \sigma\,(W_f * [x(t), C(t-1), h(t-1)] + b_f) \quad (2)$$

$$i_t = \sigma\,(W_i * [x(t), C(t-1), h(t-1)] + b_i) \quad (3)$$

$$o_t = \sigma\,(W_o * [x(t), C(t-1), h(t-1)] + b_o) \quad (4)$$

$$C(t) = C(t-1) + f_t + \hat{C} * i_t \quad (5)$$

where σ is the activation function, $x(t)$ is the input, $h(t-1)$ is the previous output, W_i, W_f, W_o and b_i, b_f, b_o are the weights and biases of input gate, forget gate and output gate, respectively.

3.1.2. Gated Recurrent Unit

Gated Recurrent Unit (GRU) is another gated recurrent unit that can learn the long-term dependencies proposed by Bahdanau at al. [30]. Unlike the LSTM, there is no memory unit and there are 2 gates instead of 3 gates (see Fig. 3.). Having a simpler architecture requires less computation and can be trained faster. Despite having a less complex structure, studies have shown that performance is comparable to LSTM [6].

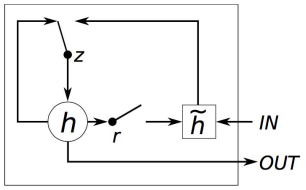

Fig. 3. Gated Recurrent Unit [29]

GRU unit contains update z and reset r gates. The update gate specifies what information to keep in the next state. The reset gate specifies how information from the previous state and the new input are to be combined. The equations used to calculate the next output and state values in the GRU unit are as follows:

$$z_t = \sigma\,(W_z * [x(t), h(t-1)]) \quad (6)$$

$$r_t = \sigma\,(W_r * [x(t), h(t-1)]) \quad (7)$$

$$\hat{h}(t) = \sigma\,(W_h * [x(t), (r_t * h(t-1))]) \quad (8)$$

$$h(t) = (1-z_t) * h(t-1) + z_t * \hat{h}(t) \quad (9)$$

where σ is the activation function, $x(t)$ is the input, $h(t-1)$ is the previous output, W_z, W_r and W_h are the weights of update gate, reset gate and candidate output, respectively.

3.1.3. Backpropagation Through Time

Traditional neural networks are trained via the backpropagation algorithm [31]. The backpropagation algorithm aims to minimize the cost function by updating the weights between layers. The algorithm consists of two repetitive phases, propagation and weight update. In the first phase, input vector propagated forward through layers and output vector is obtained. In the second stage, the error value is calculated through the cost function by comparing the expected output values with the output value obtained. The error values are propagated backwards and the weights are updated to minimize the cost function. This operation is repeated until the error function reaches the desired small size.

BPTT is a customized backpropagation algorithm used to train RNNs. Since RNNs use sequenced data, it is necessary to unfold them over time as illustrated in Fig. 1. to train them. After unfolding, each time step corresponds to a layer. This structure is trained by backpropagation like traditional FNNs. As the number of time steps used in RNN increases, the training time also increase due to the represented network is getting deeper.

4. Experiments

4.1. The Data

In this study, hourly global horizontal solar radiation data is used. The data used covers the 10-year period from January 1998 to December 2007. Measurements of data provided by French meteorological organization (Météo-France) were carried out at the meteorological station of Ajaccio (Corsica Island, France, 41° 55'N, 8° 44'E, 4 m above mean sea level). The station is located between the Mediterranean Sea and mountains. Sensors used in the station can work in the range of 0 - 90000 J / m² and annual maintenance is done regularly. The data has a yearly seasonal pattern as shown in Fig. 4.

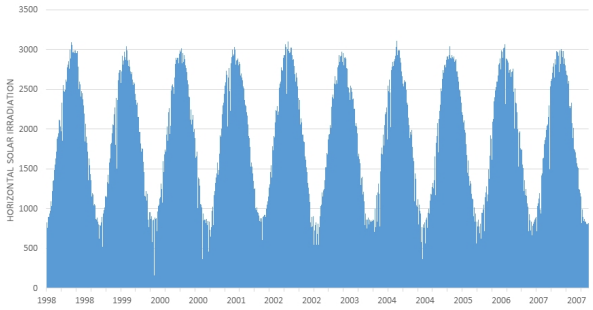

Fig. 4. Daily Horizontal Solar Irradiation

4.2. Experiments

In this section, LSTM and GRU models are used for 1 hour horizon time series forecasting on the data mentioned in section 4.1. Hyper-parameters are tuned to optimize the models. Single layer and two layer models were formed and the effect of depth was investigated. The obtained optimized single layer and two layer models were compared with naive method and simple RNN models.

4.2.1 Hyper-parameters Tuning

There is no golden method known to optimize artificial neural networks. In order to optimize the models constructed in this study, the effect of the parameters through the experiments on the model performance was observed. Examined parameters are window size, number of neurons and number of epochs. These parameters have been increased by doubling in order to obtain best value that satisfies the most successful result. The experiments were run 6 times for the same configuration then best and mean results are obtained. The optimization experiments for the single-layer LSTM model are shown in Table 1, Table 2 and Table 3, respectively.

Table 1. Single-layer LSTM window size tuning

Number of Neurons	Number of Epochs	Window Size	Best (nRMSE)	Mean (nRMSE)
30	40	1	0.36387	0.364279
30	40	2	0.276411	0.277801
30	40	4	0.249072	0.24746
30	40	8	0.236184	0.24281
30	40	16	0.224898	0.228282
30	40	32	0.216498	0.218615
30	40	64	**0.214726**	0.21828
30	40	128	0.217557	0.219357

Table 2. Single-layer LSTM number of neurons tuning

Number of Neurons	Number of Epochs	Window Size	Best (nRMSE)	Mean (nRMSE)
1	40	64	0.236154	0.241564
2	40	64	0.226201	0.227503
4	40	64	0.220388	0.222954
8	40	64	0.21748	0.220124
16	40	64	0.214506	0.217394
40	40	64	0.214726	0.21828
64	40	64	**0.213652**	0.219843
128	40	64	0.22216	0.248149

Table 3. Single-layer LSTM number of epochs tuning

Number of Neurons	Number of Epochs	Window Size	Best (nRMSE)	Mean (nRMSE)
64	8	64	0.22085	0.229908
64	16	64	0.217642	0.223031
64	32	64	0.217799	0.225571
64	40	64	**0.213652**	0.219843
64	64	64	0.214807	0.216060

The optimization experiments for the two-layer LSTM model are shown in Table 4, Table 5 and Table 6, respectively.

Table 4. Two-layer LSTM window size tuning

Num. of Neurons (L1)	Num. of Neurons (L2)	Number of Epochs	Window Size	Best (nRMSE)	Mean (nRMSE)
32	8	32	1	0.363869	0.365237
32	8	32	2	0.257817	0.260344
32	8	32	4	0.243725	0.245013
32	8	32	8	0.233748	0.237458
32	8	32	16	0.224112	0.225946
32	8	32	32	0.21716	0.21965
32	8	32	64	**0.214083**	0.217465
32	8	32	128	0.215939	0.217726

Table 5. Two-layer LSTM number of neurons tuning

Num. of Neurons (L1)	Num. of Neurons (L2)	Number of Epochs	Window Size	Best (nRMSE)	Mean (nRMSE)
16	8	32	64	0.214445	0.219393
32	8	32	64	0.214083	0.217465
64	8	32	64	0.213144	0.216074
128	8	32	64	0.212346	0.216435
256	8	32	64	0.211936	0.214954
128	4	32	64	**0.211575**	0.214416
128	16	32	64	0.214556	0.217934
256	4	32	64	0.212674	0.218435

Table 6. Two-layer LSTM number of epochs tuning

Num. of Neurons (L1)	Num. of Neurons (L2)	Number of Epochs	Window Size	Best (nRMSE)	Mean (nRMSE)
128	4	8	64	0.218595	0.221475
128	4	16	64	0.213716	0.216859
128	4	32	64	**0.211575**	0.214416
128	4	64	64	0.214511	0.216623

4.2.2 Results

Same optimization method with LSTM model is implemented for single-layer and two-layer GRU model and the results are compared with Naive method and simple RNN model in Table 7. According to the results, LSTM and GRU models are not superior to each other. It has been observed that LSTM and GRU models are much more successful than conventional simple RNN architectures. Two layer architectures have shown slight success compared to single layer architectures.

Table 7. Comparison of models

Model	Number of Layers	Result (nRMSE)
Naïve Method	-	0.37
Simple RNN	1	0.222499
Simple RNN	2	0.219836
LSTM	1	0.213652
LSTM	2	0.211575
GRU	1	0.214065
GRU	2	0.211165

5. Conclusion

In this study, time series forecasting experiments on solar irradiation data using LSTM and GRU models, which are a special type of RNN are conducted. The performances of the deep architectures of these models have been investigated. The results are compared with conventional simple RNN and Naive models. The results show that LSTM and GRU models can be suitable and competitive on the solar irradiation data time series forecasting.

Acknowledgement

Thanks to the University of Corsica, UMR CNRS 6134 for providing the data used in this study.

References

[1] Huang, J., Korolkiewicz, M., Agrawal, M., Boland, J. "Forecasting solar radiation on an hourly time scale using a Coupled AutoRegressive and Dynamical System (CARDS) model", *Solar Energy*, 2013, 87, 136-149.

[2] Benmouiza, K., Cheknane, A. "Forecasting hourly global solar radiation using hybrid k-means and nonlinear autoregressive neural network models", *Energy Conversion and Management*, 2013, 75, 561-569.

[3] Yang, D., Jirutitijaroen, P., Walsh, W. M. "Hourly solar irradiance time series forecasting using cloud cover index", *Solar Energy*, 86(12), 2012, 3531-3543.

[4] Paoli, C., Voyant, C., Muselli, M., Nivet, M. L. "Forecasting of preprocessed daily solar radiation time series using neural networks", *Solar Energy*, Elsevier, 2010, 84(12), 2146-2160.

[5] Mihalakakou, G., Santamouris, M., Asimakopoulos, D. N. "The total solar radiation time series simulation in Athens, using neural networks", *Theoretical and Applied Climatology,* 2000, 66(3), 185-197.

[6] Britz, D. (2015). "Recurrent Neural Networks Tutorial", [online]. URL: http://www.wildml.com/2015/09/recurrent-neural-networks- tutorial-part- 1-introduction- to-rnns/ [accessed July 12, 2017].

[7] Merkel, G., Povinelli, R. J., Brown, R. H. "Deep Neural Network Regression for Short-Term Load Forecasting of Natural Gas", *In 37th Annual International Symposium on Forecasting* 2017.

[8] Kuremoto, T., Kimura, S., Kobayashi, K., Obayashi, M. "Time series forecasting using a deep belief network with restricted Boltzmann machines", *Neurocomputing*, 2014, 137, 47-56.

[9] Qiu, X., Zhang, L., Ren, Y., Suganthan, P. N., Amaratunga, G. "Ensemble deep learning for regression and time series forecasting", *In Computational Intelligence in Ensemble Learning (CIEL), IEEE Symposium*, 2014.

[10] Lv, Y., Duan, Y., Kang, W., Li, Z., Wang, F. Y. "Traffic flow prediction with big data: a deep learning approach", *IEEE Transactions on Intelligent Transportation Systems*, 2015, 16(2), 865-873.

[11] Fischer, T., Krauß, C. "Deep learning with long short-term memory networks for financial market predictions", *FAU Discussion Papers in Economics*, 2017.

[12] Hsu, D. "Time Series Forecasting Based on Augmented Long Short-Term Memory", arXiv preprint arXiv: 1707.00666, 2017.

[13] Bao, W., Yue, J., Rao, Y. "A deep learning framework for financial time series using stacked autoencoders and long-short term memory", *PloS one*, 2017, 12(7), e0180944.

[14] Adhikari, R., Agrawal, R. K. "An introductory study on time series modeling and forecasting", LAP Lambert Academic Publishing, Germany, 2013.

[15] Balkin, S.D. Ord, J. K. "Automatic neural network modeling for univariate time series", *International Journal of Forecasting*, 2000, 16, 509–515.

[16] Conejo, A.J., Palazas, M.A., Espimola, R., Molina, A.B. "Day-ahead electricity price forecasting using the wavelet transform and ARIMA models", *IEEE transactıons on power systems,* 2002, 20(2), 1035–1042.

[17] Stergiou, K.I., Chirtou, E.D. "Modelling and forecasting annual fisheries catches: comparison of regression, univariate and multivariate time series methods", Fisheries Research, 1996, 50, 1035–1042.

[18] Vengertsev, D. "Deep learning architecture for univariate time series forecasting", Stanford, Technical Report 2014.

[19] More, A., Deo, M.C. "Forecasting wind with neural networks", *Marine Structures,* 2003, 16(1), 35–49.

[20] Allende, H., Moraga, C., Salas R. "Artıficıal neural networks in time series forecasting: a comparative analysis", Kybernetıka, 2002, 38(6), 685–707.

[21] Breiman, L. "Random forests", Machine learning, 2001, 45(1), 5-32.

[22] Juban, J., Fugon, L., Kariniotakis, G. "Probabilistic short-term wind power forecasting based on kernel density estimators", *European Wind Energy Conference and Exhibition*, 2007.

[23] Dudek, G. "Short-term load forecasting using random forests", *Intelligent Systems*, 2015, 821-828.

[24] Ivakhnenko, A. G., Lapa, V. G. "Cybernetic Predicting Devices", CCM Information Corporation, 1965.

[25] Hopfield, J. J. "Neurons with graded response have collective computational properties like those of two-state neurons", *Proceedings of the national academy of sciences*, 1984, 81(10), 3088-3092.

[26] Rumelhart, D. E., Hinton, G. E., Williams, R. J. "Learning internal representations by error propagation", *Parallel Distributed Processing*, 1986, 1(8) 318–362.

[27] Bengio, Y., Simard, P., Frasconi, P. "Learning long-term dependencies with gradient descent is difficult", *IEEE transactions on neural networks*, 1994, 5(2), 157-166.

[28] Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

[29] Chung, J., Gulcehre, C., Cho, K., Bengio, Y. "Empirical evaluation of gated recurrent neural networks on sequence modeling", arXiv:1412.3555 2014.

[30] Bahdanau, D., Cho, K., Bengio, Y. "Neural machine translation by jointly learning to align and translate", arXiv preprint arXiv:1409.0473 2014.

[31] Rumelhart, D. E., Hinton, G. E., Williams, R. J. "Learning representations by back-propagating errors", *Nature*, 1986, 323(6088), 533-538.