ISSN 2278-3091
Volume 11, No.1, January - February 2022
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse051112022.pdf
https://doi.org/10.30534/ijatcse/2022/051112022

A Hybrid Deep Neural Network Model for Time Series Forecasting

Jyoti Verma
Department of Computer Science and Engineering
Suresh Gyan Vihar University
Jaipur, Rajasthan, India
jyoti.18223@mygyanvihar.com

Sohit Agarwal
Department of Computer Science and Engineering
Suresh Gyan Vihar University
Jaipur, Rajasthan, India
Sohit.agarwal@mygyanvihar.com

Received Date : December 10, 2021 Accepted Date :January 10, 2022 Published Date : February 06, 2022

ABSTRACT

Deep neural networks have proven to perform optimal forecasts even with the presence of noisyand non-linear nature of time series data. In thispaper, a hybrid deep neural network consisting of Convolutional Neural Networks (CNN) and Long Short Term Memory (LSTM) architecture have been proposed. The model combines the convolutional layer's capability of feature extraction along with the LSTM's feature of learning long term sequential dependencies. The experiments were performed on two datasets and compared with four other approaches: Recurrent Neural Network (RNN), LSTM, Gated Recurrent Unit (GRU) and Bidirectional LSTM. All five models are evaluated and compared with one step ahead forecasting. The proposed hybrid CNN-LSTM outperformed other modelsfor both datasets showing robustness against error.

Key words : Recurrent Neural Networks, Long Short Term Memory, Gated Recurrent Units, Bidirectional, Convolutional Neural Networks, Time Series Forecasting

1. INTRODUCTION

Time series refers to sequential data in which order is required to be maintained. It is observations recorded in successive intervals of time. Time series data can be frequently observed in the domains of econometrics such as stock prices, currency exchange rates as well assignal processing and meteorology records of wind speeds, temperatures and rainfall. These data are prevalently used forecasting, which is predicting the future values by utilization of the past values.Now, forecasting can be performed using the traditional statistical methods or neural network models. Statistical models such as ARIMA, ARIMAX, GAS is the prevalently used time series forecasting techniques in majority of the domains [1]. Artificial neural networks have also been used along with these for achieving better forecast results. Recently, Recurrent Neural Networks are being used for sequential data problems [2]. These have been widely used for Natural Language Processing as well as time series forecasting. Along with the recurrent neural networks, hybrid models consisting of a convolutional component have also evolved recently. Here, we perform forecasting using four prevalent architectures of recurrent neural networks and analyze their performance. We also develop a hybrid CNN-LSTM architecture and compare its efficiency with other networks. The main issue here is to perform analysis and forecasting of time series datasets and develop the qualitative forecasting models

RNNs are a special class of neural networks characterized by internal self-connections in any nonlinear dynamical system. Prominent architectures of RNN include Deep RNNs with Multi-Layer Perceptron, Bidirectional RNN, Recurrent Convolutional Neural Networks, Multi-Dimensional Recurrent Neural Networks, Long-Short Term Memory, Gated Recurrent Unit, Memory Networks, Structurally Constrained Recurrent Neural Network, Unitary Recurrent Neural Networks, Gated Orthogonal Recurrent Unit and Hierarchical Subsampling Recurrent Neural Networks [3]. However, vanilla RNN is known to be having the underlying issue of vanishing as well as exploding gradients in order to tackle which various clipping strategies as well as other variants of RNN are proposed [4]. The LSTM variant of RNN have been analyzed for eight of its variants concluding that forget gate and the output activation function are the most critical component. Also, the learning rate is found to be the most crucial hyperparameter [5]. Now, RNN and its variants have been widely used for time series forecasting tasks in a wide range of domains. Long short term memory has been used as a novel forecasting technique for solar energy forecasting proving LSTM as being robust and performing better than GBR and FFNN [6]. Petroleum time series data which are characterized by high dimensionality, non-stationary being highly non-linear in nature have also been used to test the performance of LSTM [7]. Furthermore, a deep architecture of RNN has been used to extract deep invariant daily features of financial time series outperforming other models in predictive accuracy and profitability performance[8].A combination of the auto-encoder of convolutional neural network and the long short-term memory unit has also been proposed for the task of wind speed forecast[16].Recently, a black-box CNN-LSTM architecture was proposed forindoor temperature modeling [17].

This paper aims to analyse the RNN, LSTM, GRU and Bidirectional architectures along with the proposal of a deep neural network architecture for the task of univariate time series forecasting. Datasets from the electricity and air quality domain are being used. The length of both the datasets vary, however the fluctuations can be found to be similar. As such the problems from different domains are chosen in order to analyze the influence on the results of the proposed architecture. The paper is organized as follows. Section II discusses the architecture and mathematical formulation of the RNN models analyzed and the proposed model. Section III states the experimental details of the methodology adopted and the results while Section IV concludes the paper with future directions.

2.RECURRENT NEURAL NETWORK ARCHITECTURES

This section summarizes the basics of the RNN architectures which are analyzed in this paper. RNN is the most basic of all the three and GRU and LSTM are the variants which were introduced at a later time.

A. Recurrent Neural Networks

Recurrent Neural Networks [9] are in the family of feedforward neural networks. They are different from other feedforward networks in their ability to send information over time-steps. Recurrent Neural Networks are considered Turing complete and can simulate arbitrary programs (with weights). If we view neural networks as optimization over functions, we can consider Recurrent Neural Networks as optimization over programs. Recurrent neural networks are well suited for modeling functions for which the input and/or output is composed of vectors that involve a time dependency between the values. Recurrent neural networks model the time aspect of data by creating cycles in the network (hence, the recurrent part of the name). RNN is a special type of Neural Network that accounts for the dependencies between data nodes. It preserves the sequential information in an inner state, allowing them to persist the knowledge accrued from subsequent time steps.

$$z_t = W^{xh}x_t + W^{hh}h_{t-1}$$

$$h_t = \tanh(z_t)$$
$$y_t = W^{hy}h_t$$
$$p_t = softmax(y_t) \quad (1)$$

Figure 1 represents an RNN cell with $x_t$as present input, h_{t-1} the previous state, W^{xh}as weight between inputs to hidden unit, W^{hh}being weight between hidden to hidden unit, i.e., the recurrent weight and bias b. $z_t$is the output of the hidden unit before application of activation function φ. Then, $h_t$is the hidden unit output that is sent to the next recurrent units and also used in computation of final output of that RNN unit. The final output $y_t$is computed by applying another activation function to the hidden unit output and W^{hy} weight between hidden to output unit. The selection and application of activation function depends on the task being performed [hands on].

B. Long Short Term Memory Networks

LSTM [10], as in Figure 2, introduces additional computation components to the RNN, the input gate, the forget gate and the output gate. The recurrence equation for the hidden vector is changed for LSTM with the use of long-term memory. The operations of the LSTM are designed to have fine-grained control over the data written into this long-term memory. The equations for the forward pass are stated below:

$$a_t = \tanh(W_c x_t + R_c h_{t-1})$$
$$i_t = \sigma(W_i x_t + R_i h_{t-1})$$
$$f_t = \sigma(W_f x_t + R_f h_{t-1})$$
$$o_t = \sigma(W_o x_t + R_o h_{t-1})$$
$$c_t = i_t \odot a_t + f_t \odot c_{t-1}$$
$$h_t = o_t \odot \tanh(c_t) \quad (2)$$

The current input and the previous state are worked upon by a_t after which the input gate $i_t$decides upon which parts of $a_t$are required to be added to the long term state c_t. The forget gate $f_t$makes a decision as to which parts of $c_{t-1}$are to be erased and erases unnecessary parts, the output gate $o_t$decides on the parts of $c_t$to be read and shown as output. There exists a short term state $h_t$between the cells and a long term state $c_t$in which the memories are dropped and added by the respective gates.

Table 1: DATASET DESCRIPTION

Dataset	Source	No. of Observations	Description
Dataset 1	[12]	2826	Half Hourly values of Electricity Demand ranging from 01-01-1991 to 01-03-1999
Dataset 2	UCI Repository [14]	9352	Hourly Air Quality dataset ranging from 10-03-2004 to 04-04-2005

C. Gated Recurrent Units

The Gated Recurrent Unit [11] can be viewed as a simplification of the LSTM, which does not use explicit cell states. The main simplifications are that both state vectors are merged into a single vector. Figure 3 represents a GRU cell and (3) the equations followed during forward pass.

$$z_t = \sigma(W_{xz}^T . x_{(t)} + W_{hz}^T . h_{(t-1)})$$
$$r_t = \sigma(W_{xr}^T . x_{(t)} + W_{hr}^T . h_{(t-1)})$$
$$g_t = tanh(W_{xg}^T . x_{(t)} + W_{hg}^T . (r_{(t)} \otimes h_{(t-1)}))$$
$$h_t = (1 - z_t) \otimes tanh(W_{xg}^T . h_{(t-1)} + z_t \otimes g_t) \quad (3)$$

A single gate controller controls both the forget gate and the input gate. If the gate controller outputs a 1, the input gate is open and the forget gate is closed. If it outputs a 0, the opposite happens. In other words, whenever a memory must be stored, the location where it will be stored is erased first. This is actually a frequent variant to the LSTM cell in and of itself. There is no output gate; the full state vector is output at every time step. However, there is a new gate controller that controls which part of the previous state will be shown to the main layer.

D. Bidirectional LSTM

Bidirectional long-short term memory allows the neural network to have sequential information in both directions backwards and forward, i.e., past to future as well as future to past.

Since the input flows in two directions, a bidirectional LSTMis different from the regular LSTM. With vanilla LSTM, the input flows either in forward direction or backwards. However, in bidirectional the input flows in both directions. This helps to preserve not only the past information but also the future data.

3.EXPERIMENT AND RESULTS

The methodology followed for the proposed work and the results obtained is being discussed in this section. Figure 4 gives a more descriptive interpretation of the scheme followed. The time series datasets are first preprocessed for making it trainable using the neural network model. The models are further optimized, regularized and properly tuned for attaining generalized results avoiding underfitting as well as overfitting. The performance evaluation of the four variants of RNN and the proposed model is done using evaluation metrics which finally decides the best forecast model for the problem at hand. The experiments were carried out using the keras library with tensorflow backend and python programming language.

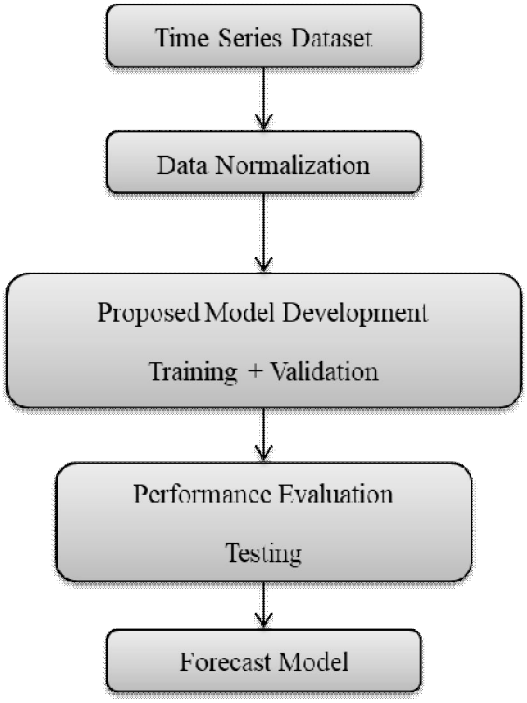

Figure 1: Methodology

A. Dataset Description

The analysis is carried out on two real world datasets from varying domains and lengths. The description of the datasets is given in Table I.

From the figures of the dataset, it can be observed that Dataset 1 follows a particular pattern repeating itself, however the number of observations is low. Dataset 2 is not as complex but it consists of the maximum number of observations. The aim is to analyze the performance of the models in differing set of scenarios of datasets and also prove the efficiency of the proposed model.

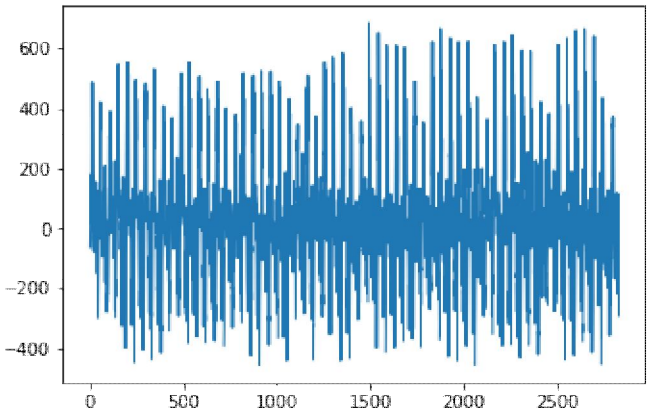

Figure 2: Electricity Demand Data

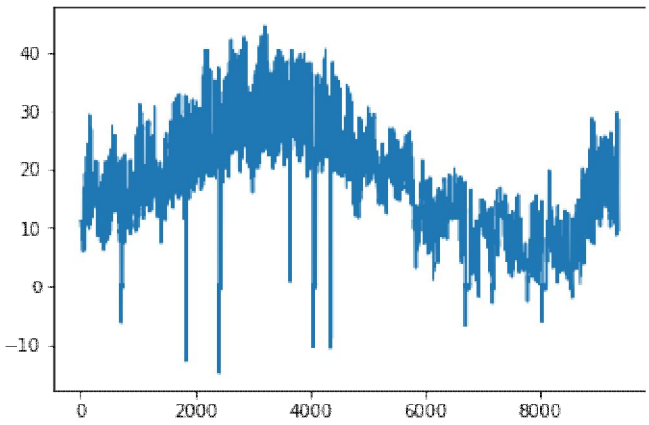

Figure 3: Air Quality Data

The datasets described are raw datasets, i.e., preprocessing is required for making them usable. Since, neural network architecture is being used, converting the series to a stationary set is not a mandatory step. The values range go high as well as low due to which data normalization is performedin order to standardize the inputs and approach faster towards the global minima. It also ensures that larger inputs do not overwhelm or become dominant. Min-max normalization, as described in (4), is performed in a range of 0 to 1. It does not change the data pattern or characteristics but only readjusts the scale of the data.

$$x^{normalized} = \frac{x-min}{max-min} \quad (4)$$

Data pre-processing step also includes the conversion to a supervised data format since time series datasets are being dealt with in here. One-step ahead forecast is to be done where the next time step (t+1) is predicted. Originally the univariate time series dataset consists of only one feature column. We divide this time series into input (x) and output (y) using lag time method. Lags differ in all three datasets. Dataset 1 consists of lag size 6and Dataset 2 of size 4.

B. Network Modelling

Modeling the neural network architecture such that it performs optimally requires setting and tuning different configurations of the network. The supervised data is split into a train-validation-test split for proper estimation of error. Optimization is the minimization of loss function with respect to the parameters of our model. Here, ADAM optimizer is used as stated in [15]. ADAM optimizer is robust and is used frequently used for training RNN architectures. Now, optimizers mainly aim to decrease the training error. But, sometimes this results in overfitting, i.e., the model fits well on the training data but unable to fit on the test data. approximately high value for the parameters governing the capacity of the model and then controlling it by adding a regularization term to the error function. In our work, we have used Dropout regularization as and when required [2]. A dropout layer blocks a random set of cell units in one iteration. Blocked units do not receive and do not transmit information. Removing connections in the network reduces the number of free parameters to be estimated during training and the complexity of the network. Consequently, dropout helps to prevent over-fitting. Dropout ratio of 0.2 is used in our work in the hidden layer. Hyperparameters are the settings that are not adapted by the learning algorithm since that would result in model overfitting. Hidden layers of size 2 and 3 were experimented upon. The number of hidden nodes was set to form a narrow architecture. Number of epochs is tuned for the problem at hand.

Figure 4 represents the proposed network model denoting the input, output and the hidden layers. The convolution operations are performed in the initial layers for automatic feature extraction rather than doing it manually.Pooling is a process of down-sampling, which can effectively reducethe dimension of the matrix window, while retaining the deep informationat the same time. In this work, the max pooling was used. Then, we have the LSTM layers to preserve the sequential information. Finally, we have the output layer which gives the one step ahead forecast results.

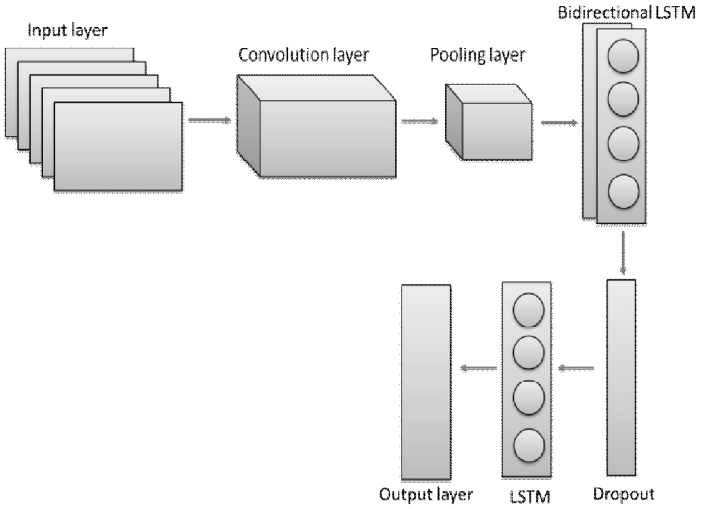

Figure 4 : Network Model

C. Performance Metrics

Four performance evaluation metrics are used to assess forecast accuracy. These error metrics are frequently used for assessing model accuracy. After evaluating all the forecast models according to the above stated metrics, the best configured forecast model is decided upon.These are shown in Table II below:

Table 2: PERFORMANCE EVALUATION METRICS

Metric		Description
MAE	Mean Absolute Error	$\frac{1}{N}\sum_{n=1}^{N}(y_n^{actual} - y_n^{predicted})$
MSE	Mean Squared Error	$\frac{1}{N}\sum_{n=1}^{N}(y_n^{actual} - y_n^{predicted})^2$
RMSE	Root Mean Squared Error	$\sqrt{\frac{1}{N}\sum_{n=1}^{N}(y_n^{actual} - y_n^{predicted})^2}$
$\boldsymbol{R^2}$ **score**	R^2score	$1 - \frac{\sum_{n=1}^{N}(y_n^{actual} - y_n^{predicted})^2}{\sum_{n=1}^{N}(y_n^{actual} - \vec{y}_n^{predicted})^2}$

Here, *N* is the total number of observations for which the error is evaluated. $y_n^{actual}$is the actual observation in nth position and $y_n^{predicted}$the predicted value.

D. Results

As stated earlier, the experiments are carried out on two datasets from different real-world domains. Four architectures of Recurrent Neural Networks, being RNN, LSTM, GRU, Bidirectional LSTM, and proposed hybrid deep neural network model areusedfor forecasting one step-ahead result. The results shown below in Table III and Table IV show the performance evaluation parameters for both the datasets with optimal parameters.

Table 3:PERFORMANCE EVALUATION FOR DATASET 1

Models	MAE	MSE	RMSE	R^2 score
RNN	0.09277	0.01489	0.1221	54.33
LSTM	0.09273	0.01462	0.1209	55.18
GRU	0.09185	0.01461	0.1209	55.19
Bidirectional	0.09182	0.01452	0.1205	55.48
Proposed	0.09082	0.01436	0.1198	55.97

Table 5: PERFORMANCE EVALUATION FOR DATASET 2

Models	MAE	MSE	RMSE	R^2 score
RNN	0.02097	0.0006274	0.02505	94.27
LSTM	0.02003	0.0005952	0.02439	94.56
GRU	0.01972	0.0005882	0.02425	94.63
Bidirectional	0.01580	0.0005341	0.02311	95.13
Proposed	0.01566	0.0004176	0.02044	96.19

The results are stated for well-tuned models achieved after experiments for generalizing the model for a better test performance on unknown samples. From the table of results, many observations can be made about the performance of the forecast models. In the case of network performance, it can be seen that the proposed hybrid model performs the best out of all five models in both the scenarios. The convolutional layer does enhance the forecast performance when combined with recurrent layers of LSTM network. The bidirectional LSTM also indicates better performance than other variants due to its property of preserving both past and future information.

Concerning the datasets, more amount of data leads to better performance. Dataset 1 has lesser amount of data samples than Dataset 2 which results in the models performing better in Dataset 2. The fluctuating data requires the complex structure of these RNN variants in order to learn the data patterns and dependencies required for forecasting. The proposed CNN-LSTM architecture enjoys the advantage of efficiency.

4.CONCLUSION

In the paper, four variants of RNN, namely RNN, LSTM, GRU and Bidirectional LSTM has been analyzed on two different real-world datasets. It was observed that Bidirectional LSTM performed more efficiently amongst all. Further, a hybrid deep neural network mechanism comprising of convolutional layers along with recurrent layers and dropout regularization have been proposed. The proposed model is able provide optimal forecast results in both the datasets. Hence, the convolutional neural network's properties can be utilized along with recurrent neural network architectures to develop efficient time series forecast mechanisms.

REFERENCES

[1] P. J. Brockwell and R. A. Davis, Introduction to Time Series and Forecasting - Second Edition. 2002.

[2] Ian Goodfellow, Y. Bengio, and A. Courville, Deep learning. 2017.

[3] H. Salehinejad, J. Baarbe, S. Sankar, J. Barfett, E. Colak, S. Valaee, "Recent advances in recurrent neural networks," 2017, [Online] Available: https://arxiv.org/abs/1801.01078

[4] R. Pascanu, T. Mikolov, and Y. Bengio, On the difficulty of training recurrent neural networks, in 30th International Conference on Machine Learning, ICML 2013, 2013, no. PART 3, pp. 23472355.

[5] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, LSTM: A Search Space Odyssey, IEEE Trans. Neural Networks Learn. Syst., vol. 28, no. 10, pp. 22222232, 2017.

[6] S. Srivastava and S. Lessmann, A comparative study of LSTM neural networks in forecasting day-ahead global horizontal irradiance with satellite data, Sol. Energy, vol. 162, pp. 232247, 2018.

[7] A. Sagheer and M. Kotb, Time series forecasting of petroleum production using deep LSTM recurrent networks, Neurocomputing, vol. 323, pp. 203213, 2019.

[8] W. Bao, J. Yue, and Y. Rao, A deep learning framework for financial time series using stacked autoencoders and long-short term memory, PLoS One, vol. 12, no. 7, 2017.

[9] K. Unnikrishnan and K. P. Venugopal, Alopex: A correlation-based learning algorithm for feedforward and recurrent neural networks, Neural Computation, vol. 6, no. 3, pp. 469490, 1994.

[10] F. A. Gers, J. Schmidhuber, F. Cummins,"Learning to forget: Continual prediction with LSTM", Neural Comput., vol. 12, no. 6, pp. 2451-2471, 2000.

[11] K. Cho, D. Bahdanau, F. Bougare, H. Schwenk and Y. Bengio, "Learning Phrase Representations using RNN EncoderDecoderfor Statistical Machine Translation," arXiv, 2014.

[12] I. Koprinska, M. Rana, and V. G. Agelidis, Yearly and seasonal models for electricity load forecasting, in Neural Networks (IJCNN), The 2011 International Joint Conference on. IEEE, 2011, pp. 14741481.

[13] [Online]. Available:https://finance.yahoo.com/quote/CSV/history/

[14] M. Lichman, UCI machine learning repository, 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[15] D. P. Kingma and J. L. Ba, Adam: A method for stochastic optimization, 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc., 2015.

[16] Y. Chen, Y. Wang, Z. Dong, J. Su, Z. Han, D. Zhou, Y. Zhao, Y. Bao, "2-D regional short-term wind speed forecast based on CNN-LSTM deep learning model," Energy Conversion and Management, vol. 244,2021.

[17] F. Elmaz, R. Eyckerman, W. Casteels, S. Latré, P. Hellinckx, "CNN-LSTM architecture for predictive indoor temperature modeling," Building and Environment, vol. 206, 2021.