RESEARCH ARTICLE OPEN ACCESS

Deep Recurrent Neural Networks for Forecasting

Jyoti Verma, Chanchal Sharma
Department of Computer Science and Engineering Suresh GyanViharUnivarsity Jaipur, Rajasthan, India
Department of Computer Science and Engineering Suresh GyanViharUnivarsity Jaipur, Rajasthan, India
Corresponding Author: Jyoti Verma

ABSTRACT—Recurrent neural networks (RNNs) are the neural network architectures developed for sequential and time-series data. RNN can model dynamical systems with many of its variants solving the problem of long term dependency that it faces. In this paper, we present an analysis of the three variants of RNN namely, the vanilla RNN, Long Short term Memory (LSTM) and Gated Recurrent Units (GRU). The fundamentals of the architectures are explained along with experiments performed on three real world datasets. The impact of deeper networks and the RNN variants on forecast performance is performed.

Index Terms—Recurrent Neural Networks, Long Short Term Memory, Gated Recurrent Units, Time Series Forecasting

Date of Submission: 05--12-2019 Date Of Acceptance: 16-12-2019

I. INTRODUCTION

Time series refers to observations recorded in successive intervals of time. It is sequential data in which order is required to be maintained. Time series data can be frequently observed in signal processing, econometrics such as stock prices, currency exchange rates as well as meteorology records of wind speeds, temperatures and rainfall. These data are prevalently used for time series forecasting, which is prediction of the future values utilizing the past values. As such, time series forecasting is useful in many domains of day to day life. Now, forecasting can be performed using the traditional statistical methods or neural network models. Statistical models such as Nave method, Holts Winter, Exponential smoothing, ARIMA, ARIMAX, GAS are 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. Here, we perform forecasting using three prevalent architectures of recurrent neural networks and analyze their performance. There exists plenty of forecasting methods with each requiring their own corresponding conditions. The main issue here is to perform analysis and forecasting of time series datasets and develop the qualitative forecasting models. For solving this task, the recurrent neural network architectures of vanilla RNN, LSTM and GRU are used.

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].

This paper aims to analyse the RNN, LSTM and GRU architectures for the task of univariate time series forecasting. Datasets from the electricity, stock and air quality domain are being used for deciding how length and problems from different domains influence the results of the three RNN architectures. The impact of deep neural network architecture is also analyzed for all three architectures. The paper is organized as follows. Section II discusses the architecture and mathematical formulation of the RNN models analyzed. Section III states the experimental details of the methodology adopted and the results while Section IV concludes the paper with future directions.

II. 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.

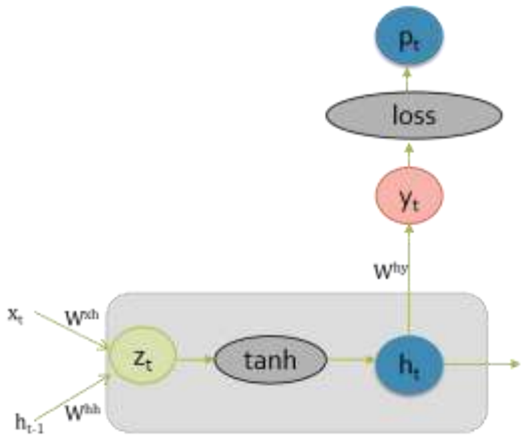

Fig. 1. RNN

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.

$$z_t = W^{xh} x_t + W^{hh} h_{t-1}$$
$$h_t = \tanh(z_t)$$
$$y_t = W^{hy} h_t$$
$$p_t = \mathrm{softmax}(y_t) \qquad (1)$$

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].

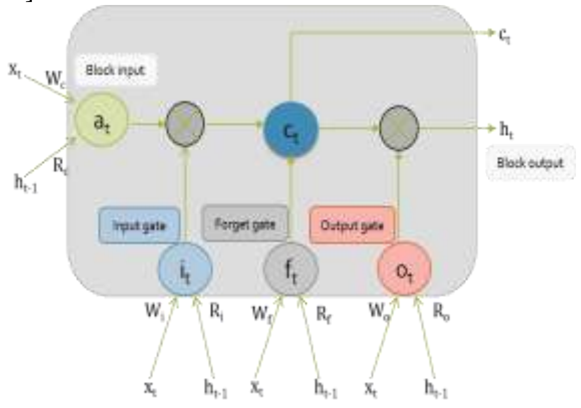

Fig. 2. LSTM

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:

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

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.

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	[13]	5671	Daily Yahoo stock price dataset ranging from 09-08-1996 to 22-02-2019
Dataset 3	UCI Repository [14]	9352	Hourly Air Quality dataset ranging from 10-03-2004 to 04-04-2005

TABLE I DATASET DESCRIPTION

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.

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

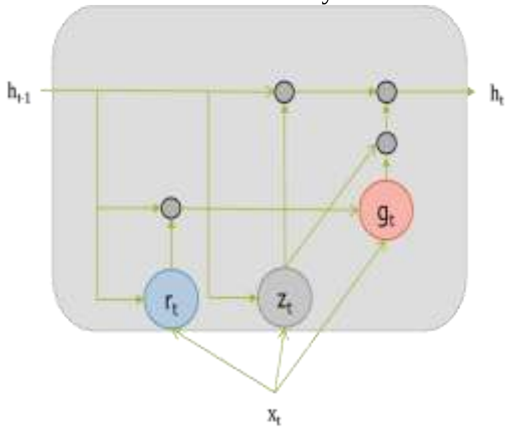

Fig.3.GRU

III. 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 three variants of RNN 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.

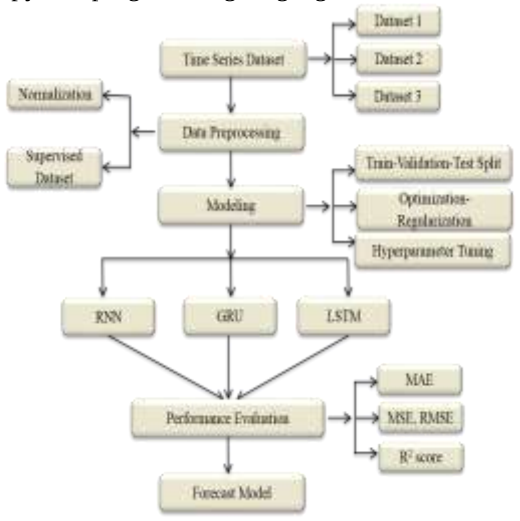

Fig. 4. Methodology

A. **Dataset Description**

The analysis is carried out on three 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 are low. Dataset 2 seems to be fluctuating in unequal intervals indicating more complexity. Dataset 3 is not as complex but it consists of the maximum number of observations out of the three. The aim is to analyze the performance of the models in such differing set of scenarios of datasets.

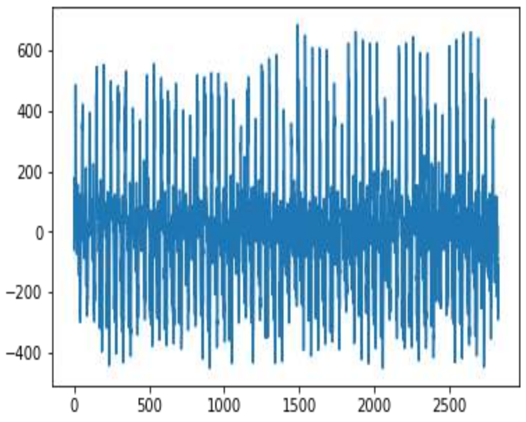
Fig. 5. Electricity Demand Data

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

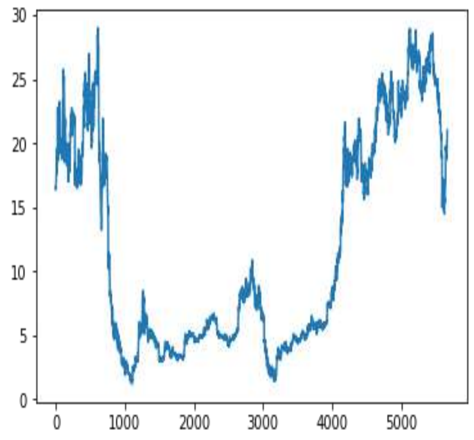
Fig. 6. Stock Price Data

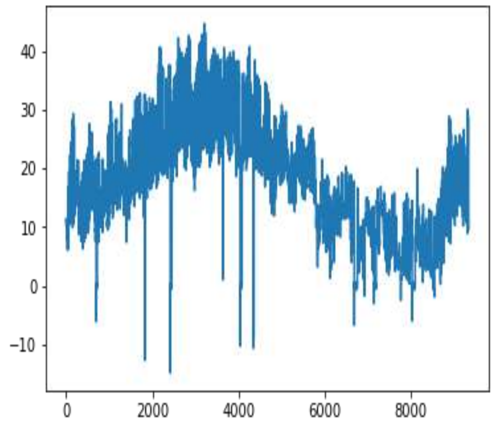
Fig. 7. Air Quality Data

Regularization tries to overcome overfitting by choosing anlow due to which data normalization is performed in 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.

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

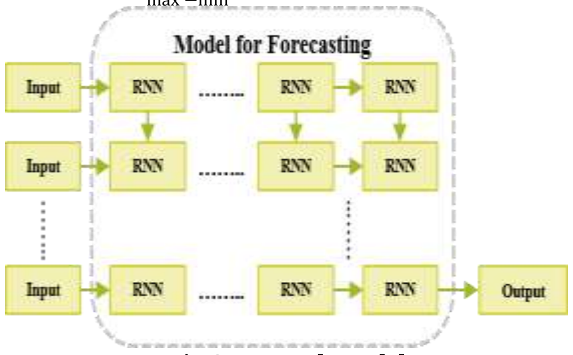

Fig.8.NetworkModel

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 6, Dataset 2 of size 40 and Dataset 3 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 8 represents a network model denoting the input, output and dynamic number of the hidden layers as well as units of the forecast model developed.

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 II
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	$\sqrt{\frac{1}{N}\sum_{n=1}^{N}(y_n^{actual} - y_n^{predicted})^2}$
R^2 score	R^2score	$1 - \frac{\sum_{n=1}^{N}(y_n^{actual} - y_n^{predicted})^2}{\sum_{n=1}^{N}(y_n^{actual} - \bar{y}_n^{predicted})^2}$

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 three datasets from different real world domains. Three architectures of Recurrent Neural Networks, being RNN, LSTM, GRU, areused for forecasting one step-ahead result. The results shown below in Table III show the performance evaluation parameter for a 2 layer architecture while Table IV show for a 3 layered architecture.

TABLE III
PERFORMANCE EVALUATION FOR A 2-LAYERED ARCHITECTURE

	Performance Metrics				
	Dataset	MAE	MSE	RMSE	R^2 score
RNN	Dataset 1	0.0916	0.0149	0.1221	0.5433
	Dataset 2	0.0554	0.0041	0.0638	0.6498
	Dataset 3	0.0209	0.0006	0.0250	0.9427
GRU	Dataset 1	0.0918	0.0146	0.1209	0.5519
	Dataset 2	0.0462	0.0035	0.0595	0.6958
	Dataset 3	0.0200	0.0006	0.0242	0.9463
LSTM	Dataset 1	0.0917	0.0146	0.1209	0.5519
	Dataset 2	0.0442	0.0027	0.0525	0.7631
	Dataset 3	0.0157	0.0004	0.0204	0.9619

TABLE IV
PERFORMANCE EVALUATION FOR A 3-LAYERED ARCHITECTURE

	Performance Metrics				
	Dataset	MAE	MSE	RMSE	R^2 score
RNN	Dataset 1	0.0915	0.0148	0.1217	0.5462
	Dataset 2	0.0619	0.0050	0.0708	0.5701
	Dataset 3	0.0224	0.0007	0.0267	0.9350
GRU	Dataset 1	0.0925	0.0147	0.1212	0.5497
	Dataset 2	0.0429	0.0028	0.0529	0.7594
	Dataset 3	0.0199	0.0006	0.0246	0.9448
LSTM	Dataset 1	0.0935	0.0145	0.1204	0.5553
	Dataset 2	0.0468	0.0033	0.0577	0.7137
	Dataset 3	0.0167	0.0004	0.0214	0.9581

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 model. In the case of network performance, it can be seen that LSTM performs the best out of all three models in all the scenarios. As such, the LSTM architecture is the superior architecture amongst the three variants of RNN which have been analyzed in this paper. Also, deeper network architecture does not always guarantee better performance. Models with less number of hidden layers also perform well when tuned appropriately. Further, deeper networks lead to more complex models and costly time consumption. Therefore, proper tuning of model parameters as well as hyperparameters is a must before adopting a deep architecture.

Concerning the datasets, more amount of data leads to better performance. Dataset 1 has the least number of observations which leads to the model requiring more parameters for a better forecast performance. Both LSTM and GRU performs well on dataset 2 consisting of complex and fluctuating data, hence the complex structure of these RNN variants learn the data patterns and dependencies required for forecasting. The GRU is simpler and enjoys the advantage of greater ease of implementation and efficiency. It might generalize slightly better with less data because of a smaller parameter footprint although the LSTM would be preferable with an increased amount of data. In case of dataset 3, more number of observations and lesser complexity leads to all the three models performing with good accuracy, proving that the simple structure of RNN performs well in some scenarios.

Therefore, the different architectures perform differently according to the problem at hand. Simpler architectures are sought to be used when the datasets have less complexity and more number of observations. However, with increase in data complexity, more complex models are required, the network size being a dynamic issue for every other problem and architecture.

IV. CONCLUSION

In the paper, an analysis is performed on the three variants of RNN, namely RNN, LSTM and GRU on different real world datasets. It was observed that LSTM performs the best in maximum cases with model tuning being of utmost importance before increasing the complexity. In future, more architectures are sought to be analyzed for the multivariate case of forecasting data with more emphasis on the data characteristics.

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.

Jyoti Verma "Deep Recurrent Neural Networks for Forecasting" International Journal of Engineering Research and Applications (IJERA), vol. 9, no. 12, 2019, pp 63-68