Energy Load Forecasting using Deep Learning Approach-LSTM and GRU in Spark Cluster

Sumit Kumar[1*], Lasani Hussain[2*], Sekhar Banarjee[3], Motahar Reza[4]
*B. Tech, CSE 3rd Year Students,
High Performance Computing Lab,
School of Computer Science and Engineering
National Institute of Science & Technology, Berhampur, India, 761008
[1]checksumitkumar@gmail.com, [2]lasanihussain@gmail.com, [3]sekharbanarjee14@gmail.com, [4]reza@nist.edu

Abstract— **Electric load forecasting has a significant role in power grids in order to facilitate the decision making process of energy generation & consumption. Long term forecasting is not feasible as there might be an uncertainty in the prediction because of irregular increase in the demand for power with the growing population and dependency on electric power. Since the behaviour of electric load time series is very much non-linear and seasonal, Neural Networks are best suited model for learning the Non-Linear behaviour within the data and for forecasting purpose. This paper deals with the Recurrent Neural Networks based Models: Long-Short-Term-Memory (LSTM) and Gated- Recurrent-Unit (GRU) to deal with this challenge. Observations have been made based on the distributed implementation of various configurations of LSTM-RNN & GRU-RNN on spark clusters for hyper parameter tuning purpose and deploying best suited configuration with least RMSE value using apache memos resource management.**

Keyword: Deep Learning, Electric load, forecasting, LSTM, GRU, Distributed Computing, Spark.

I. INTRODUCTION

Since the energy wastage is a threat to sustainability, making the process of energy generation and consumption for future, efficient is extremely crucial. Based on the interval of time period, energy load forecasting can be grouped into three categories: (1) Short-term-load -forecasting (STLF) which is usually from 1 hour to 1 week (2) Medium-term-load forecasting (MTLF), ranging from 1 week to 1 year, and (3) Long-term load forecasting (LTLF) which is longer than one year [1-2].

Various models have been developed for realizing forecasting accuracy such as Regression , Statistical and state space methods [3-4].AI based approaches have been explained based on Expert Systems, Fuzzy Logic Systems, Artificial Neural Networks(ANNs). Hybrid approaches to Time-Series analysis utilizing ANN are not uncommon [4], presents a model for Time-Series forecasting using ANN and ARIMA models. Electric load forecasting is primarily a discrete & univariate time series[3], many statistical time series models[5]can be applied for electric load forecasting such as Autoregressive (AR), Moving Average (MA), Autoregressive Integrated Moving Average (ARIMA) models [5] and lot of their variants. For more information on time series and deep learning approach to it, refer [6].

The main focus of the presented paper is to solve Electric load forecasting problem using deep learning approach on a Spark Cluster. Deep learning allows models composed of multiple layers to learn pattern representations within the data.

II. MULTI-LAYER NEURAL NETWORK

In this section, we will discuss a special type of network called a multi-layer perceptron (MLP) feedforward neural network. It consists of three layers: one input layer, one hidden layer, and one output layer [5-6]. The units in the hidden layer are fully connected to the input layer, and the output layer is fully connected to the hidden layer, respectively. Such a network having more than one hidden layer is called a deep artificial neural network. A Deep Layer Network is shown in the below Figure 1. It is a well-defined technique to solve real life problems such as speech recognition, image classification, and video analysis, and weather forecasting, prediction in stock market and of energy consumption.

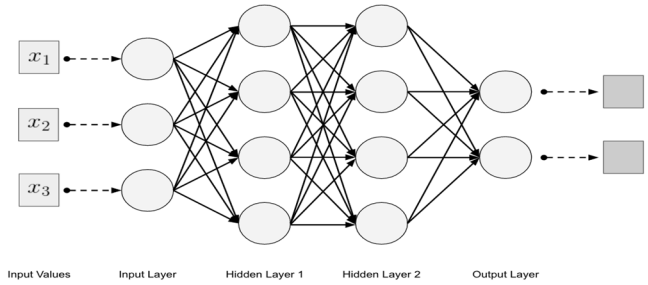

Fig 1. Deep Layer Network

2.1 RECURRENT NEURAL NETWORK

Recurrent Neural Networks (RNNs) are neural networks that employ recurrence, which basically uses information from a previous feedforward pass over the neural network (see Fig 2). RNNs are best suited and recent research papers shows that RNNs have been incredibly successful when applied to problems where the input data's are in the form of a sequence for which predictions are to be made [6-7].

The decision a recurrent net takes at time step **t,** will be affected by the decision made at time step **t-1**.The recurrent networks are having two inputs, the current time step input $\mathbf{x_t}$ and the hidden state of past $\boldsymbol{h_{t-1}}$.

The process of carrying memory mathematically can be shown as:

$h_t = f(W * x_t + U * h_{t-1})$, where f(.)=activation function

The weight input and hidden state are combinedly compressed by a **logistic sigmoid function or hyperbolic tangent (tanh)** in order to make ***gradients workable for backpropagation.***

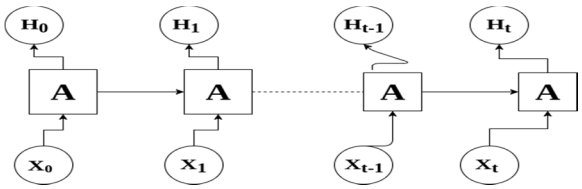

Figure 2 Architecture of Recurrent Neural Network

2.1.1 Long-Term Dependencies in Recurrent Net

Backpropagation algorithm propagates the final layer errors backward from the output layer to the weights and inputs of each hidden layer, updating those weights assigned earlier by calculating their partial derivatives $-\partial J/\partial W$.
In this case, recurrent net needs to back propagate the error through all the previous time steps, which is not feasible at all. The calculation of gradients with respect to the features in all hidden layers of the earlier time step which is too behind will need a lot of computation and have to remember all parameters from each time step visited before.

2.1.2 Vanishing & Exploding Gradient Problem

The vanishing and exploding gradient problem is an obstacle to RNN performance, i.e. gradient of some of the weights starts to become too small or too large, if the network is unfolded for too many time steps as discussed in paper [9].Since the gradient becomes too large in case of exploding gradient, it can be squashed by using some functions such as logistic, etc. But, the major obstacle was vanishing gradients as the gradients become so small to propagate and reflect any changes in the parameters to learn. LSTM solves this problem [10].

2.1.3. Long Short Term Memory (LSTM)

In the mid-90s, Sepp Hochreiter & Juergen Schmidhuber [9] proposed a variation of existing RNN having Long Short-Term Memory units, or LSTMs, as a solution to the problem stated in section 2.1.2. However the full gradient can instead be calculated with backpropagation through time [11]. LSTMs help preserve the error that can be back propagated through time and layers. LSTMs contain information in a gated cell. The cell makes decisions about what to store, and when to allow reads, writes and erases, via gates that open and close. The cells learn when to allow/leave/delete the data, through the iterative guesses, back propagating errors, and adjusting weights. The Fig 3 illustrates data flow & controlling through a memory cells and gates.

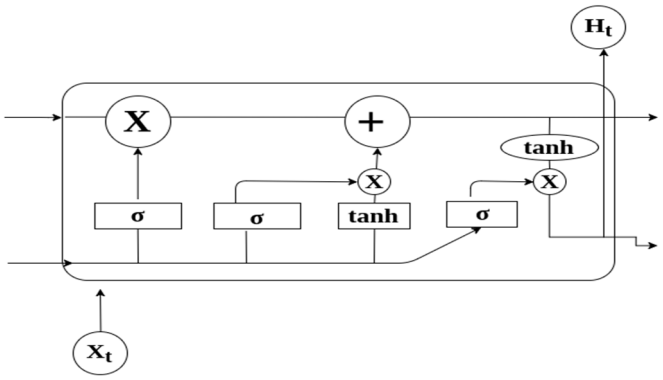

Figure 3 Repeating module in an LSTM containing four layers

2.1.4 Working of LSTM Cell

Step 1:- *D*ecide which information needs to be thrown away using a sigmoid layer by looking at $\mathbf{h_{t-1}}$ and $\mathbf{x_t}$, and outputs a number between 0 & 1 for each number in the cell state $\mathbf{C_{t-1}}$. 1 represents "completely keep this" while a 0 represents "completely get rid of this."

$f_t = \sigma(W_f * (h_{t-1}, x_t) + b_f)$, where ***σ(.)***=sigmoid function

Step 2:-Decide what new information we're going to store in the cell state. First, a sigmoid layer called the "input gate layer" decides which values need to be updated. Then, a **tanh** layer creates a vector of new candidate values, $\mathbf{C'_t}$, that could be added to the state. Then combine these two to create an update to the state.

$$i_t = \sigma(W_i * (h_{t-1}, x_t) + b_i)$$
$$C'_t = tanh(W_c * (h_{t-1}, x_t) + b_c)$$
$$C_t = f_t * C_{t-1} + i_t * C'_t$$
$$O_t = \sigma(W_0 * (h_{(t-1)}, x_t) + b_0)$$
$$h_t = O_t * tanh(C_t)$$

2.1.5. Gated Recurrent Unit

The GRU is a variant of the LSTM and was introduced by K, Cho [12]. The GRU was inspired by the LSTM unit but is considered simpler to compute and implement. It retains the LSTM's resistance to the vanishing gradient problem, but its internal structure is simpler and therefore easier to train since fewer computations are needed to make update to its hidden state.

Fig 4. Shows that the GRU has an update gate & a reset gate similar to the forget & input gates in the LSTM unit. The update gate defines how much previous memory to keep around and the reset gate defines how to combine the new input with the previous memory. The major difference is that the GRU fully exposes its memory content using only integration (but with an adaptive time constant controlled by the update gate).

These are the mathematical functions used in controlling the gating mechanism in GRU cell:-

$$z = \sigma(W_z h_{t-1} + U_z x_t)$$
$$r = \sigma(W_r h_{t-1} + U_r x_t)$$

$$c = tanh(W_c(h_{t-1} \otimes r) + U_c x_t)$$
$$h_t = (z \otimes c) + \big((1 - z) \otimes h_{t-1}\big)$$

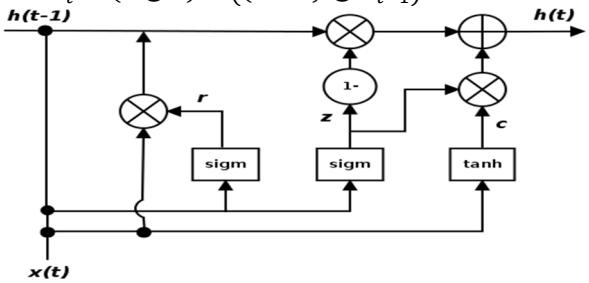

Figure 4. ***A Gated recurrent unit cell***

III. EXPERIMENTAL SETUP

In the presented paper we use Spark and a cluster of 7 machines of core i5 processor with 4 GB primary memory to improve deep learning based model's performance .The models selected for experimental analysis include LSTM and GRU. Distributing the computations among the 7-node cluster, we were able to train 7 different configuration of models concurrently.

3. 1 Dataset & Experimental Results

This Section deals about the dataset chosen for testing and training the models discussed earlier in the presented paper using Spark and a Cluster of machines .It then discusses the results obtained for the two models with different configurations investigated both on Sequential machine and on Spark Cluster .

3. 2. Dataset

The selected methods were implemented on a dataset of electric power consumption in household with a one -minute sampling rate .The data set contained power consumption measurements collected between mid-December 2006 to mid-April 2008 .The First 14 months data was chosen to train both the model and the last two month data was used as testing.

To improve the performance of deep neural models hyper-parameter tuning was done firstly on single machine chosen 1 configuration at a time. Then each different configuration was trained parallel on an individual node. Using Spark for computation we find the best set of hyper-parameters for RNN, LSTM & GRU training, resulting in 6X reduced training time and 23% lower error rate.

IV. EXPERIMENTAL RESULTS

4.1. RNN Network

The very first experiment was conducted using standard RNN network to provide a base result for further comparison with improved models[Table 1].

Table 1 Errors for various Configurations of RNN Network model

Configuration	Layers	Units	RMSE (Training)
1.	1	20	0.683
2.	1	30	0.644
3.	1	50	0.635
4.	2	20	0.631
5.	2	50	0.599
6.	3	20	0.600
7.	3	30	0.598

4.2. LSTM Network

The second experiment was conducted using standard LSTM network to predict few next hours load based on last 24 hours. The results show that it is an easy task for the standard LSTM network architecture, and hence give low error ratios with the test dataset [Table 2].

Table 2 Errors for various Configurations of LSTM Network model

Configuration	Layers	Units	RMSE (Training)
1.	1	20	0.661
2.	1	30	0.636
3.	1	50	0.604
4.	2	20	0.618
5.	2	50	0.597
6.	3	20	0.607
7.	3	30	0.592

4.3. GRU Network

The above mentioned configurations of deep neural nets were implemented with the standard GRU network to predict few steps ahead. The results shows that GRU model performed much better than the standard LSTM network architecture, and hence give less error ratios with both training & test dataset[Table 3].

In RNN, LSTM and GRU, the configuration with 3 Hidden Layers and 30 units in each hidden unit's promises to give the least RMSE 0.592, 0.584 & 0.562 respectively. The best configuration with least error value was chosen for testing purpose.

Deploying models: Apply the selected trained RNN/LSTM and GRU model for forecasting on a large amount of new data on a Spark Cluster.

Table 3 Errors for various Configurations of GRU Network model

Configuration	Hidden Layers	Units	RMSE(Training)
1.	1	20	0.621
2.	1	30	0.616
3.	1	50	0.592
4.	2	20	0.601
5.	2	50	0.564
6.	3	20	0.579
7.	3	30	0.562

A Graph of the computing times w.r.t the number of nodes on the cluster is shown in Fig 5. Comparison error analysis was displayed in Table 4. Fig 6 shows the predicted vs actual data plot using RNN, Fig 7 shows the predicted vs actual data plot using LSTM and Fig 8 shows that same analysis by GRU.

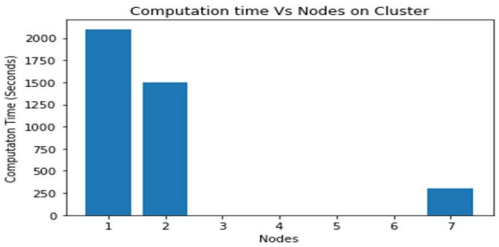

Figure 5: Comparison the computer time

Table 4 Errors of LSTM & GRU Model on Test Data

	Hidden Layers	Units	RMSE (Test)
RNN	3	30	0.623
LSTM	3	30	0.602
GRU	3	30	0.589

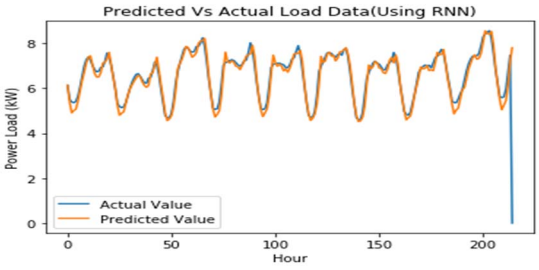

Figure 6 Plot of data fit using RNN model

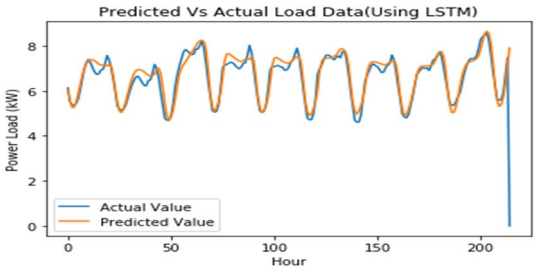

Figure 7 Plot of data fit using LSTM model

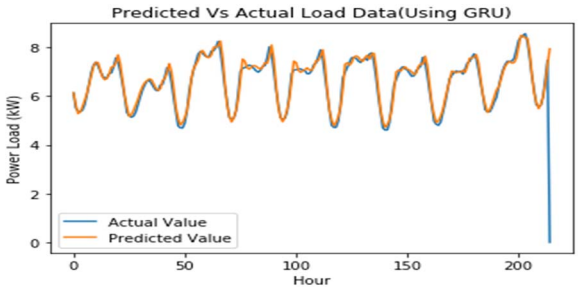

Figure 8 Plot of data fit using GRU model

V. CONCLUSIONS

The goal of the presented work was to investigate the effectiveness in using LSTM and GRU based neural network architectures for energy load forecasting. Both of them were trained and tested on a Spark Cluster. The accuracy of forecasting load with the initial set of hyper-parameters is 99.1%. The best chosen configuration of LSTM & GRU has 99.39% accuracy on the test set, which is around **34% reduced test error**. We were able to train 7 models concurrently using a 7 - node cluster, which is 6**x faster** compared to training time on a single node sequential machine. GRU model was better at forecasting load as compared to LSTM.

REFERENCES

[1] Eugene A. Feinberg and Dora Genethliou,"Chapter 12 Load Forecasting" Weather(2006), Issue: August, Publisher:Springer, pp. 269-285

[2] Vahid Mansouri and Mohammad E. Akbari ,Efficient Short-Term Electricity Load Forecasting Using Recurrent Neural Networks,Journal of Artificial Intelligence in Electrical Engineering, Vol. 3, No. 9, June 2014

[3] N. Hubele, et al., "Identification of Seasonal Short-term Load Forecasting Models Using Statistical Decision Functions," IEEE Transactions on Power Systems, Vol. 5, No. 1, 1990, pp. 40-5. doi:10.1109/59.49084

[4] [4]Khashei, M., Bijari, M.: A novel hybridization of artificial neural networks and arima models for time series forecasting. Applied Soft Computing 11(2), 2664–2675 (2011)

[5] Peter J. Brockwell and Richard A. Davis ,"Chapter 1 & 4 " Introduction to Time Series and Forecasting,Second Edition , Springer

[6] [6]Deep Learning for Time-Series Analysis ,John Gamboa ,University of Kaiserslautern, Kaiserslautern, Germany

[7] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back- propagating errors. Cognitive modeling 5, 3 (1988)

[8] Schmidhuber, J.: Deep learning in neural networks: An overview. Neural Networks,61, 85–117 (2015)

[9] Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. Neural Networks, IEEE Transactions on 5(2), 157–166 (1994)

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

[11] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidi-rectional LSTM and other neural network architectures. Neural Networks,18:602–610, 2005.

[12]] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In NIPS 2014 Workshop on Deep Learning, December 2014.