Residential Load Forecasting Using Deep Neural Networks (DNN)

Tareq Hossen, Arun Sukumaran Nair, Radhakrishnan Angamuthu Chinnathambi, Prakash Ranganathan
Department of Electrical Engineering
University of North Dakota
Grand forks, ND, USA
tareq.hossen@und.edu

***Abstract*— Forecasting of consumer electricity usages plays an important role to make total smart grid system more reliable. As the activities of individual residential consumers has many uncertain variables, it is hard to accurately forecast the residential load levels. For planning of the electrical resources and to balance demand and supply, accurate forecasting tasks are critical. This paper presents Deep Neural Network (DNN) based short term load forecasting for Residential consumers. In this work, we compare the Mean Absolute Percentage Error (MAPE) value for residential electricity dataset using different types recurrent neural network (RNN). Our preliminary results indicate that Long short-term memory (LSTM) based RNN performed better compared with simple RNN and gated recurrent unit (GRU) RNN for a single user with 1-minute resolution based on one year of historical data sets.**

Keywords—Deep Neural Network, Recurrent Neural Network, Residential Load Forecasting, Long short term memory, Gated Recurrent Unit.

I. INTRODUCTION & RELATED WORK

Load forecasting has remained an important research area for conducting planning operations in electrical power systems. The advanced metering infrastructure (AMI) technology revolutionized the mass adaptation of smart meters at residential consumer levels by utilities. A significant portion (e.g., 20% to 40%) of the total electricity energy production is consumed by residential loads [1]. Load forecasting based on smart metering datasets that are within 1-minute intervals are relatively new area of research. In [2], authors introduced a methodology of short-term functional time-series based forecast to predict household-level electricity demand. A Kalman-filter based forecasting model to predict the residential load is discussed in [1] and forecasting using conditional kernel density estimation is discussed in [3]. A hybrid model approach for forecasting future residential electricity consumption for buildings is developed in [4]. An occupancy model has been developed for residential load forecasting and discussed in [5]. Artificial Intelligence (AI) based forecasting techniques such as Fuzzy logic [6], artificial neural networks [7], support vector machines [8] and wavelets neural networks [9] are investigated under short term load forecasting. The AI methods are most conducive due to their ability to handle non-linear relationships between dependent and independent variables. Recently, DNN has been successfully used in application such as image processing, automatic speech recognition, natural language processing and for time-series modeling tasks such as load forecasting.

There are several review papers on load forecasting focused at aggregated level for commercial users. However, there are limited work on residential level data sets. We think that this is because short term load forecasting at granular level is extremely challenging due to uncertainty and volatility [1], [10-11]. Most short-term load forecasting models focuses on regular pattern that are easily predictable. Residential loads are more uncertain due to erratic and stochastic nature of consumer behavior that is hard to predict. This paper represents a deep learning-based method for the meter-level load forecasting for residential consumers. We have used recurrent neural network for residential load forecasting. We compare our forecasting accuracy by using different recurrent neural network (RNN) models for residential dataset. We also test our dataset with other conventional time-series analysis such as ARIMA, Generalized linear model (GLM), Random Forest (RF) and machine learning approaches Neural network and Support Vector machine (SVM) methods. The rest of the paper is organized as follows: section 2 discusses background information on RNN, LSTM, GRU and some related work, section 3 discusses implementation platforms, and section 4 focuses on preliminary results and discussions.

II. METHODOLOGY

In this paper, the effectiveness of different DNN models are investigated for residential level forecasting. We have trained different RNN models to predict day ahead residential demand using smart meter dataset. In addition, we used different conventional methods such as ARIMA, GLM, Random Forest, neural network and support vector machine for day-ahead forecasting.

A. Structure of RNN

Recurrent neural networks are different from conventional feed forward neural network, RNN are sequenced based model that has the capability of creating temporal correlation from past information and current state [12-14]. Thus, time series forecasting problem the decision made by RNN at time steps *(t-1)* affects the decision make at current time step t [15]. This characteristic of RNN is good choice for single household load forecasting problem as individual house residential consumer natural day to day scheduled energy usage or consumption pattern are the important factor to the energy usages at the later time interval.

In RNN signal can move both in forward and backward direction and can make a loop in the neural network. RNN works with memory as its current computation is derived from earlier input that fed back to the network. It can also be considered as different duplicate of same system that passing the information to the next part [15].

In Fig.1, a groups of neural network A, receiving some input x_t and giving outputs values of h_t. Thus, a loop is formed that allows information to passed from one systems of the network to the next system.

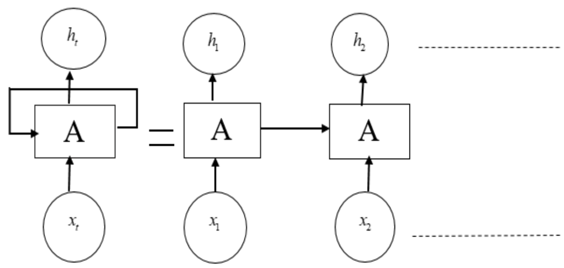

Fig. 1. An unrolled simple recurrent neural network

B. Recurrent Neural Network model for short term load forecasting

We have used three different recurrent model structures for our load forecasting. The structure of these three RNN is given below:

1) Simple RNN:

simple RNN is the basic RNN that accepts input x_t and timely the value is updated by a non linear maping. For a simple RNN the recurrent unit f can be represented as summation of linear transformation and a non linear activation which can be represented by the following equation-

$$h_t = \tanh(w[h_{t-1}, x_t] + b) \quad (1)$$

2) Long short term memory (LSTM)

A modified version of RNN is LSTM which is developed by Hochreiter & Schmidhuber [14] .The basic RNN has some long term dependency problem which can be bypassed by using LSTM. The general structure of all types of RNN contains repeating module of recurrent unit. In simpleRNNs, this repeating module have a basic structure, such as a single hyperbolic(*tanh)* layer. In LSTM, instead of having a single neural network layer, there are four layers. Fig. 2 represents the structure of LSTM where each block has two parallel lines going in and out. The top line of the structure works as a cell and the bottom line represents the hidden state information. There is another line going in from the lower parts, representing X. In total, LSTM structures contains three inputs and two outputs. x_t would be X_train, h_{t-1} would be h_previous, x_{t+1} would be X_train.next, and h_t would be h_current. The *step by step* structure of LSTM can be represented by the following equation:

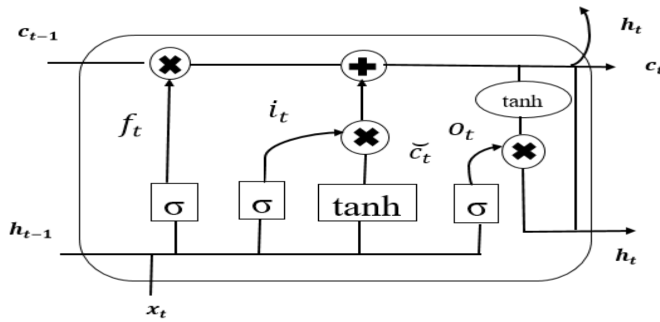

Fig. 2. Long short term memory (LSTM)

$$f_t = \sigma(w_f.[h_{t-1}, x_t] + b_f) \quad (2.a)$$

$$\bar{C}_t = \tanh(w_c).[h_{t-x}.x_t] + b_c \quad (2.b)$$

$$C_t = f_t \times C_{t-1} + i_t \times C_t \quad (2.c)$$

$$O_t = \sigma(w_o.[h_{t-1}, x_t] + b_o) \quad (2.d)$$

$$h_t = O_t \times \tanh(C_t) \quad (2.e)$$

3) Gated recurrent unit(GRU)

Gated Recurrent Unit, or GRU is an improved version of the LSTM . GRU is developed by Cho in 2014 [14] that combines two LSTM gate (the forget and input gates) into a single update gate [14]. Compared to LSTM, a GRU network only has two inputs and one output and no cell layers [15]. GRU unit receieves X_train and h_previous as inputs. They perform certain mathematical calculations and then bypass along h_current. In the next cycle X_train.next and h_current are utilized for more calculations. The structure of GRU unit is given below-

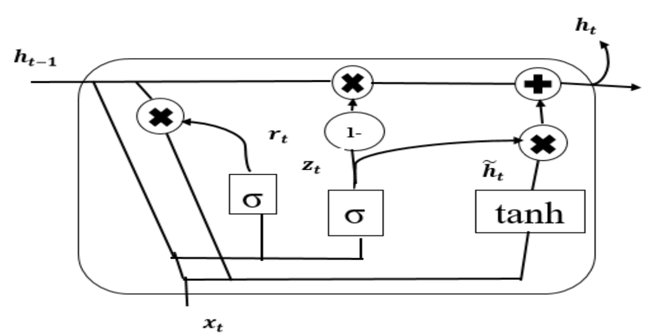

Fig. 3. Gated recurrent unit (GRU)

The step by step operation of GRU structure can be represented by the following equation.

$$z_t = \sigma(w_z.[h_{t-1,} x_t]) \quad (3.a)$$

$$r_t = \sigma(w_r.[h_{t-1,} x_t]) \quad (3.b)$$

$$\tilde{h}_t = \tanh(w.[r_t * h_{t-1,} x_t]) \quad (3.c)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \quad (3.d)$$

III . IMPLEMENTATION OF RECURRENT NEURAL NETWORK (RNN) FOR RESIDENTIAL LOAD FORECASTING

In this paper, we have used *keras* on top of tensorflow to implement the structure of recurrent neural network for load forecasting. Keras is a high-level neural network library written in python and it can run on top of either Theano or tensor flow [16]. Tensorflow platform is one of the most leading machine learning library used in developing deep learning models. However, Tensorflow has some limitations. On the contrary, keras is a highlevel API built on tensorflow that are more user friendly and easy to use.

A. Load forecasting model using keras

We experimented several parameters to train our model with the keras deep learning packages. The following Fig 4. shows a set-up carried for residential short term load forecasting cases in keras:

```
Case1: Simple RNN forecasting model based on keras
model = Sequential()
Input (length {S})
layers = [1, 50, 100, 1]
model.add(SimpleRNN(layers[1],input_shape=(None,
layers[0]),return sequences=True))
model.add(Dropout(0.3))
model.add(SimpleRNN(layers[2],return sequences=False))
model.add(Dropout(0.3))
model.add(Dense(layers[3]))
model.add(Activation("linear"))
Case2: GRU forecasting model based on keras
model = Sequential()
Input (length {S})
layers = [1, 50, 100, 1]
model.add(GRU(layers[1],input_shape=(None,
layers[0]),return sequences=True))
model.add(Dropout(0.3))
model.add(GRU(layers[2],return sequences=False))
model.add(Dropout(0.3))
model.add(Dense(layers[3]))
model.add(Activation("linear"))
Case3: LSTM forecasting model based on keras
model = Sequential()
Input (length {S})
layers = [1, 50, 100, 1]
model.add(LSTM(layers[1],input_shape=(None,
layers[0]),return sequences=True))
model.add(Dropout(0.3))
model.add(LSTM(layers[2],return sequences=False))
model.add(Dropout(0.3))
model.add(Dense(layers[3]))
```

Fig. 4. Keras code structures for three different RNN scenario

B. Dataset

The paper uses publicly available AMPds dataset that contains one year of data with 11 measurement parameters at one minute intervals for 21 sub-meters [17]. This datasets contains 5,25,600 readings for per year per smart meter[17]. Along with electrical consumtion data,AMPds also includes natural gas and water consumption data. We convert energy consumption data to KWh to mimic the market available smart meter data.

IV. RESULTS AND DISCUSSIONS

Deep neural network contains a millions of parameters, so choosing a proper optimization technique is challenging tasks. In this work, we choose RMSProp optimization. RMSProp is a biased estimation techniques proposed in neural networks for machine learning [18]. It is an gradient based optimization. For RNN modeling RMSProp is most commonly used optimization technique. In our work, we utilize the RMSProp as our parameters update optimizer , with parameter update rule according to the following formulae.

$$E[g^2]_{t-1} = \eta E[g^2]_{t-1} + (1-\eta) g_t^{\,2} \quad (4.a)$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \varepsilon}} g_t \quad (4.b)$$

where, $t \varepsilon R$ is the iteration times . The θ is the parameters of neural network. E is the weighted sum operation. g^2 is vector of gradient square. The $\eta \epsilon (0,1)$ is the learning rate. The ε is a smoothing term that avoids division by zero[18].

A. Error Metrics for Evaluation

To assess the accuracy of the forecasting model, Mean absolute error (MAE) and Mean absolute percentage error (MAPE) were used [7].

$$MAE = \frac{1}{T} \sum_{t=1}^{t=T} \frac{|A_t - F_t|}{A_t} \quad (5)$$

$$MAPE = \frac{1}{T} \sum_{t=1}^{t=T} \frac{|A_t - F_t|}{A_t} \times 100\% \quad (6)$$

where,
F_t = forecasted load , A_t = actual load and T= test set size.

The performance different RNN models with varied sequences are summarized in Table 1. From Table 1, LSTM based RNN with sequence 40 performed better compared to other forecasting models.

Table 1: MAPE summaries for Residential Load Forecasts

Scenario	Sequence (S)	MAPE	Train time (s)	Prediction time (s)	Validation loss	Training loss
LSTM	30	35%	1116.702	0.0190	3.1197e-05	4.2183e-05
LSTM	40	24%	1505	0.0210	2.0085e-05	4.2721e-05
LSTM	50	29%	2145	0.0200	2.5080e-05	4.3218e-05
Simple RNN	30	59%	213	0.019049	2.6471e-05	1.1500e-04
Simple RNN	40	37.7%	294	0.019050	2.4830e-05	2.7659e-04
Simple RNN	50	45.7%	398.01	0.021054	2.6893e-05	1.7445e-04
GRU	30	34.3%	1095	0.0210	2.4373e-05	4.2037e-05
GRU	40	24.7%	1491.87	0.0210	2.1842e-05	3.9982e-05
GRU	50	39.7%	1993.13	0.0200	2.6099e-05	3.9602e-05

Training time and prediction time of the simulation also showed in table 1. Training and validation losses also compared in this table. In order to visualize how each method perform, the first 100 time steps for each forecasting model is plotted Figs.5-7.

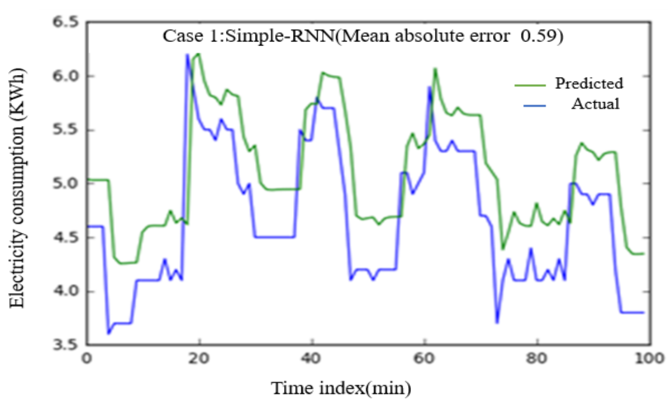

Fig. 5. Residential load forecast using Simple RNN

Table 2: MAPE Summaries for Conventional Methods

Conventional time series model for residential load forecast	
ARIMA	74%
GLM	75%
RF	74%
SVM	50%
FFNN	73.54%

Table 2 show the MAPE values for conventional time-series methods. It is important to note that we were not able to perform the day-ahead forecasting computation with 2-year datasets for conventional methods, due to larger computation run-time. Instead, we used smaller 1-year dataset. For RNN, the computational run-time is less over the conventional methods. Some inferences are: Support vector machine (SVM) perform better than other conventional methods; RNNs are much better than the conventional methods.

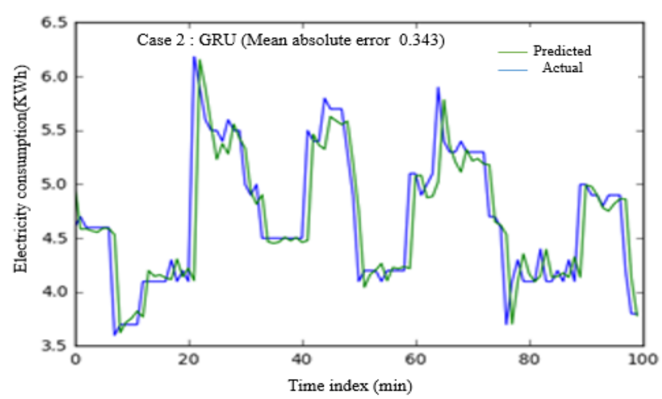

Fig. 6. Residential load forecast using GRU

Figs 8 and 9 show MAPE values for RNN and conventional methods. All frameworks is built on a desktop PC with a 3.4 GHz Intel i7 processor and 16GB of memory using the Keras library [19] with tensor-flow backend [20], [21].

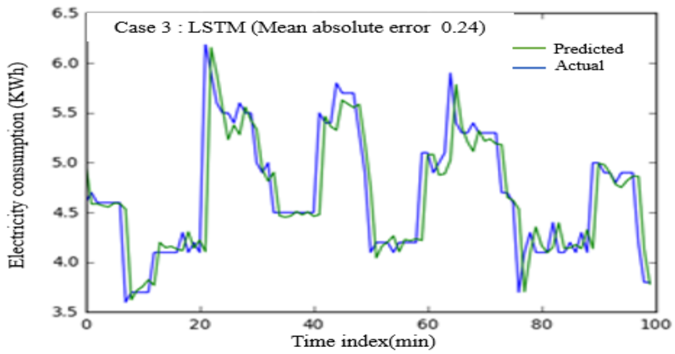

Fig. 7. Residential load forecast using LSTM

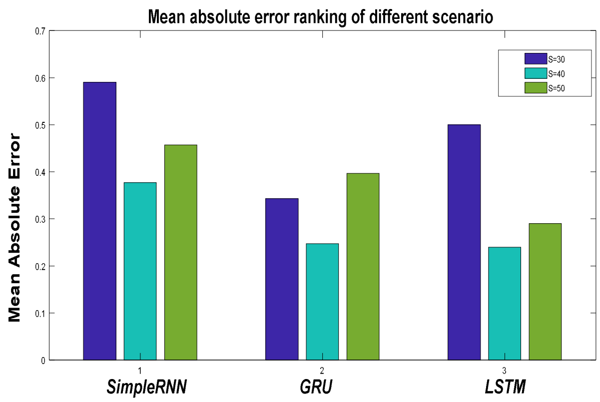

Fig. 8. MAPE ranking for RNN's

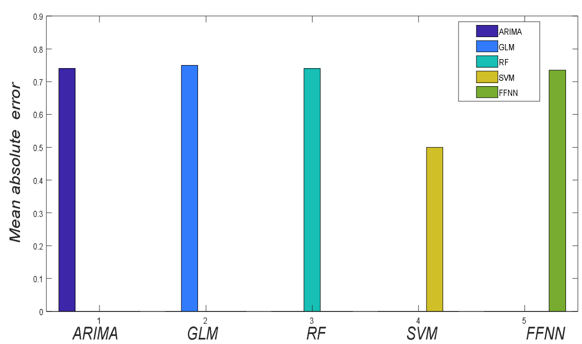

Fig. 9. MAPE ranking for conventional methods

V. Conclusion

Load forecasting for a single residential customer at 1-minute interval is a challenging task due to the uncertainty involved at this granular scale. Yet, we explored the effectiveness of recurrent neural networks for smart-metering data sets over conventional methods. Preliminary MAPE results indicate RNNs are outperforming for these data sets to predict day-ahead energy forecasting for varying sequence length of samples. However, many aspects deserve future exploration. It is important to explore connection between electric load consumption with weather conditions and electricity price. Moreover, learning from larger residential user patterns need to be explored more for scalable better meter level forecasting.

References

[1] Ghofrani, Mahmoud, et al. "Smart meter based short-term load forecasting for residential customers." North American Power Symposium (NAPS), 2011. IEEE, 2011.

[2] Chaouch, Mohamed. "Clustering-based improvement of nonparametric functional time series forecasting: Application to intra-day household-level load curves." IEEE Transactions on Smart Grid 5.1 (2014): 411-419.

[3] Arora, Siddharth, and James W. Taylor. "Forecasting electricity smart meter data using conditional kernel density estimation." Omega 59 (2016): 47-59.

[4] Dong, Bing, et al. "A hybrid model approach for forecasting future residential electricity consumption." Energy and Buildings 117 (2016): 341-351.

[5] Swan, Lukas G., and V. Ismet Ugursal. "Modeling of end-use energy consumption in the residential sector: A review of modeling techniques." Renewable and sustainable energy reviews 13.8 (2009): 1819-1835.

[6] Chow, Mo-yuen, and Hahn Tram. "Application of fuzzy logic technology for spatial load forecasting." Transmission and Distribution Conference, 1996. Proceedings., 1996 IEEE. IEEE, 1996. virtual power plants." IEEE Communications Magazine 51.1 (2013): 106-113.

[7] Shayeghi, H., H. A. Shayanfar, and G. Azimi. "Intelligent neural network based STLF." International Journal of Computer Systems Science and Engineering 4.1 (2009).

[8] Chen, S-T., David C. Yu, and Alireza R. Moghaddamjo. "Weather sensitive short-term load forecasting using nonfully connected artificial neural network." *IEEE Transactions on Power Systems* 7.3 (1992): 1098-1105.

[9] Hossen, Tareq, Siby Jose Plathottam, Radha Krishnan Angamuthu, Prakash Ranganathan, and Hossein Salehfar. "Short-Term Load Forecasting Using Deep Neural Networks (DNN)."

[10] A. S. Nair, P. Ranganathan, H. Salehfar and N. Kaabouch, "Uncertainty quantification of wind penetration and integration into smart grid: A survey," 2017 North American Power Symposium (NAPS), Morgantown, WV, 2017, pp. 1-6.

[11] J. Pagel, M. Campion, A. S. Nair and P. Ranganathan, "Clustering analytics for streaming smart grid datasets," 2016 Clemson University Power Systems Conference (PSC), Clemson, SC, 2016, pp. 1-8.

[12] Kodogiannis, Vassilis S., Mahdi Amina, and Ilias Petrounias. "A clustering-based fuzzy wavelet neural network model for short-term load forecasting." *International journal of neural systems* 23.05 (2013): 1350024.

[13] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[14] Hochreiter, Sepp, et al. "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies." (2001).

[15] Jozefowicz, Rafal, Wojciech Zaremba, and Ilya Sutskever. "An empirical exploration of recurrent network architectures." *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. 2015.

[16] Chollet, François. "Keras: Deep learning library for theano and tensorflow." *URL: https://keras. io/k* (2015).

[17] Makonin, Stephen, et al. "AMPds: A public dataset for load disaggregation and eco-feedback research." *Electrical Power & Energy Conference (EPEC), 2013 IEEE*. IEEE, 2013.

[18] Kingma, Diederik, and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).

[19] F. Chollet. (2015). Keras. Available: https://github.com/fchollet/keras

[20] https://www.tensorflow.org/

[21] https://blog.keras.io/keras-as-a-simplified-interface-to-tensorflow-tutorial.html