Short-Term Load Forecasting Using Deep Neural Networks (DNN)

Tareq Hossen, Siby Jose Plathottam, Radha Krishnan Angamuthu, Prakash Ranganathan, Hossein Salehfar
Department of Electrical Engineering
University of North Dakota
Grand forks, ND, USA
tareq.hossen@und.edu

***Abstract*—Load forecasting is an important electric utility task for planning resources in Smart grid. This function also aids in predicting the behavior of energy systems in reducing dynamic uncertainties. The efficiency of the entire grid operation depends on accurate load forecasting. This paper proposes and investigates the application of a multi-layered deep neural network to the Iberian electric market (MIBEL) forecasting task. Ninety days of energy demand data are used to train the proposed model. The ninety-day period is treated as a historical dataset to train and predict the demand for day-ahead markets. The network structure is implemented using Google's machine learning Tensor-flow platform. Various combinations of activation functions were tested to achieve a better Mean Absolute percentage error (MAPE) considering the weekday and weekend variations. The tested functions include Sigmoid, Rectifier linear unit (ReLU), and Exponential linear unit (ELU). The preliminary results are promising. and show significant savings in the MAPE values using the ELU function over the other activation functions.**

***Index Terms*-- Deep learning, Load Forecasting, Neural Network, Tensor flow, Exponential linear activation**

I. INTRODUCTION

Load forecasting plays a vital role in operation and planning of electrical power and energy systems. Load forecasting is classified into following three categories: Long-term (*e.g., 1 to 10 years*)—this is normally used to long-term system planning; medium (*e.g., 1 month to 1 year*)—this type of forecasting is used for efficient operation and maintenance of electric power system., and short-term (*e.g. 1 h to 1 day or 1 week ahead*)— this plays an important role in estimating load flows [1].

Data related to load forecasting are non-linear in nature, making the load prediction task a challenging one. One simple approach, however, to load forecasting is using regression techniques. Regression is a statistical procedure for estimating the relationship between dependent and predictor (independent) variables. It allows one to see how the dependent variable changes with respect to changes in the independent variable. The advantage of this method is that it is easily understandable. The limitation of this approach is there exist a high degree of over fit. The other disadvantage the linear regression is too simple to capture the complex relationship in multi-variate data sets [2]. Logistic regression is the adaptation of linear regression to problem classification (*e.g., yes/no questions, groups etc*.) This method also has high probability and challenges to over-fit the model [2]. In decision trees, a graph based branching method is used to match all the possible outcomes for a decision. It is used normally for a simple problem and not potent enough to solve complex data [1][2]. Other popular method of forecasting is the Random forest. Random forest takes the mean of many decision trees–each of which is made with some random samples. Each tree is weaker than a full-decision tree. When this individual tree is combined with others, it yields a better result. This method is fast to train and can work with high-quality models [1].

Gradient boosting uses weaker trees. In this method, a small change in training set can create radical change in the model. This could be its limitation [2]. The other well-known method of load forecasting is using neural-network. In neural network, the interconnection of neuron passes messages to each other with one or more hidden layers in between them. In deep learning, several hidden layers are placed one after the other. This method can handle extremely complex tasks with high accuracy. The disadvantage of this method is it is very slow to train and require a lot of power. The other disadvantage of this method is it is almost impossible to understand the prediction [3]. A neural network consists of the input layer, hidden layer, and the output layer. These layers are connected by neuron which processes the data. Neural network with a single layer is not capable of understanding the complex relationship between input and output. A neural network with more layers than three layers is known as a deep neural network. A deep neural network has a better capability of feature abstraction of input and output pattern [4]. Recent development in the field of big data and internet of things (IoT) increase the acceptance of deep neural

network (DNN) in multiple research disciplines [4]. In [5], a three-layered neural network based backpropagation technique developed for load forecasting. Multilayer perception neural networks have been used to forecast the demand from domestic users [6].

II. METHODOLOGY

The choice of a neural network structure depends on several factors. In general, neural network modeling is divided into five steps [7]

a. *Select input and output variables.*
b. Build the neural network model.
c. Cluster training, test, and validation data.
d. Train the neural network model with the training data set.
e. Validate the neural network model.

The selection of inputs to the neural network is an important aspect of the design. In this work, we consider temperature, wind-speed, solar irradiance, and the prior load data as input. These data sets are normalized to accommodate the application of activation functions. The following expression is used to normalize load:

$$\text{Normalized load} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad (1)$$

Where x is the actual data value. To build the neural network model, the number of hidden layers and neurons are selected by a trial and error method. The datasets are categorized into three sets: training, validation, and testing sets. The test and validation data sets are not used to train the neural network. These datasets are used to evaluate the error by comparing the obtained results with the actual data. Training of the neural network is a process of determining the network weights that provide a minimum error. As an acceptable minimum level of training error does not ensure the same level of performance with all other related input datasets, it is often necessary to validate the network performance once the training process is completed [7].

III. IMPLEMENTATION OF DEEP NEURAL NETWORKS

The neural network model of this work is developed using the TensorFlow deep learning platform [9]. TensorFlow is an open source software library for numerical computation using data flow graphs. The nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicating between the nodes. The flexible architecture of TensorFlow allows one to deploy computations to one or more Central or Graphical Processing Units (CPUs or GPUs) on a desktop, server, or mobile device with a single Application Process Interface (API). TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research. However, the TensorFlow system is general enough to be applicable to other domains as well [9]. The authors developed both shallow and deep neural networks on a Tensor Flow platform. After building the model, it is tested using various combinations of activation functions and neurons.

A. Structure of DNN

The numbers of layers and neurons are key in modeling neural network structures. In shallow neural network (i.e., single hidden layer), we can only vary the number of neurons that are in the single hidden layer, while both the width and the depth of the network can be changed in DNN. For both shallow neural network (SNN) and deep neural network (DNN), the number of input and output neurons is predetermined according to the dimension of the training set and the forecasting period [8]. A heuristics method was used to choose the number of hidden neurons in the shallow neural network (SNN) in [8]. In DNN, it is not possible to do the same as SNN, as one has to consider both the width and the depth of the network [10]. The structure of DNN model used in this work is given below:

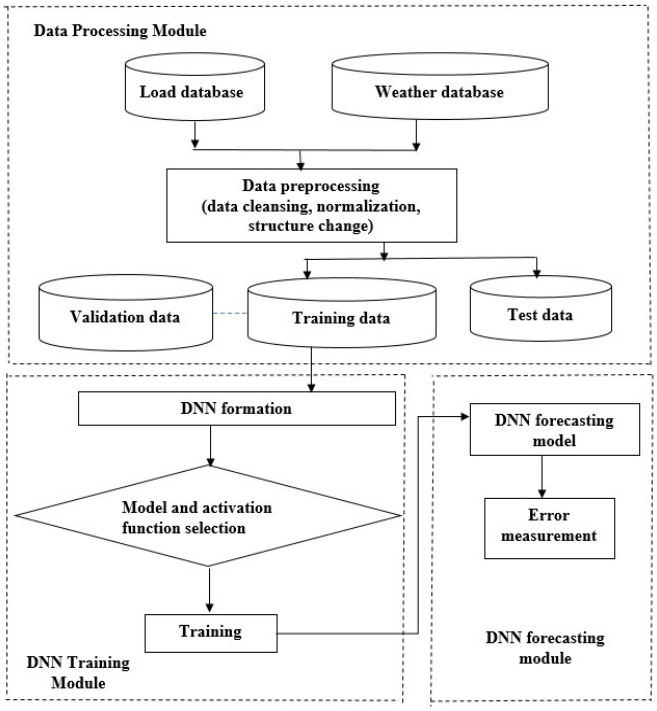

Figure 1. DNN based short term load forecast

B. Dataset

The size of dataset impacts the accuracy, training, and transfer of learning within the deep neural network [11]. For example, the 2016 DNN competition [12] uses 456,567 images for visual recognition applications. The authors of the present work used 90-days hourly data of Iberian Energy Market Operator (MIBEL). We selected a single day (July 31, 2015) to forecast and validate the performance with different deep neural network model.

C. *Activation function*

Activation function is a deciding parameter that evaluates and captures the trends or feature patterns from within the data. If the output value from the activation function is zero, the feature is absent and if the value is one the feature is present in the data. In computational networks, the activation function of a node defines the output from that node given an input or a set of inputs. A standard computer chip circuit can be seen as a digital network of activation functions that can be either "ON" (1) or "OFF" (0), depending on the input. This is like the behavior of the linear perceptron in neural networks. However, the nonlinear activation function allows such networks to compute nontrivial problems using only a small number of nodes. In artificial neural networks, this function is also called the transfer function. In training the multi-layer neural network model, activation function plays an important role in adjusting the weights. In this work, the authors have used a non-linear sigmoid and a rectified linear unit function for hidden layers in the model.

1) Sigmoid function

The sigmoid function is defined as

$$f(x) = \frac{1}{1+e^{-x+\alpha}} \tag{2}$$

where x is input to a neuron. α is the offset parameter and sigmoid function evaluates its value as zero. The sigmoid function is very similar to the step function, which acts similar to threshold. When *x* is a large positive number, the output of the sigmoid function is near to 1.

The difference between the linear and sigmoid functions is that the sigmoid one saturates to a high value of x. The sigmoidal function can make the gradient-based learning difficult. For this reason, use of linear function in deep neural network is discouraged [13].

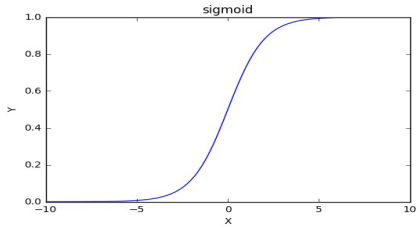

Figure 2. Sigmoid function

2) Rectified linear unit (RELU)

Rectified linear unit is defined as

$$R(x) = \max(0, x) \tag{3}$$

where x is the input to the neuron.

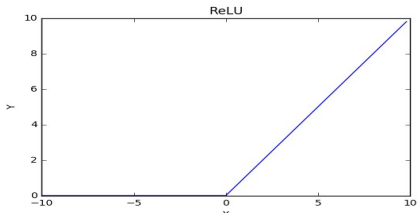

Figure 3. Rectified linear unit

ReLU is similar to linear function with the only change having an output that is zero across half of its domain. Specifically, ReLU has the two following advantages [14]:

i) It is very quick to use and train when compared with other activation functions.

ii) It is not the subject to the vanishing gradient problem. The limitation of ReLU, however, is that its mean output is not zero. For deep neural networks, this function introduces a bias for the next-layer which can slow down the learning process of the network.

3) Exponential linear unit (ELU)

The exponential linear function is defined as

$$f(x) = a(e^{x} - 1), x < 0 \text{, otherwise } f(x) = x \tag{4}$$

where, α is a parameter to be chosen and x is the input.

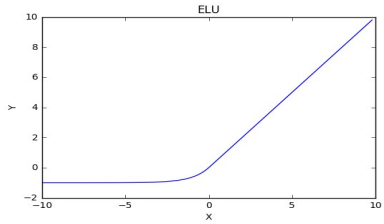

Figure 4. Exponential linear unit

ELU function behaves like the ReLU unit for values of x that are positive, but for all negative values, this function is bounded by a fixed value of α is -1 for. This behavior helps to push the mean activation of neurons closer to zero which is beneficial for learning and is robust to noise [15].

IV. SYSTEM MODEL

For a proper development and implementation of neural networks, a sound training data set is an absolute necessity. This requires several parameters during the design phase such as the availability of historical load data; number of neurons; selection of the number of layers; and the activation functions. For our system model, temperature, wind-speed, solar irradiance, and the prior load data used as input. In this paper, several short term load forecasting cases using various combinations of activation functions were investigated.

A. *Case 1. ReLU Activation Function with Single Hidden Layer*

In this case, five input variables, ReLU activation function, were selected to do the forecast. The number of neurons are varied randomly until a better mean absolute percentage error (MAPE) result is obtained.

B. *Case 2. ReLU Activation Function with two hidden layers*

In this case, ReLU activation function, and two hidden layer were selected to do the forecast. The number of neurons are varied randomly until a better MAPE result is obtained.

C. *Case 3. ReLU-Sigmoid Combination*

In this case, ReLU and Sigmoid activation functions, and two hidden layers were selected to do the forecast. The number of neurons are varied randomly until a better MAPE result is obtained. ReLU is used in layer 1 and sigmoid is used in layer 2.

D. *Case 4. ELU Activation Function with Single Hidden Layer*

In this case, ELU activation function, and one hidden layer is selected to do the forecast. The number of neurons are varied randomly until a better MAPE result is obtained. We used ELU in layer 1.

E. *Case 5. ELU Activation Function in two hidden layers*

In this case, ELU activation function, and two hidden layers were selected to do the forecast. The number of neurons are varied randomly until a better MAPE result is obtained. We used ELU in both layers 1 and layers 2.

F. *Case 6. ReLU-ELU Combination*

In this case, ReLU is used as the activation function in layer 1 and ELU is used for that in layer 2.

G. *Case 7. Sigmoid-ELU Combination*

In this case, sigmoid and ELU activation functions, and two hidden layers were selected to do the forecast. The number of neurons are varied randomly until a better MAPE result is obtained. We used sigmoid in layer 1 and ELU in layer 2.

We also tested the sigmoid-sigmoid combination for layer 1 and 2 respectively, but this combination yielded very poor MAPE values. So, we disregarded the result for brevity of this paper.Tensorflow is used to training and test the designed model. In case of designing and training deep neural network it becomes complex and confusing.Tensorflow programs provide tensorflow graph (*Tensorboard*) to make easier to understand, debug and optimize.Tensor board graph in tensor board of case 2 and case 3 is given below.

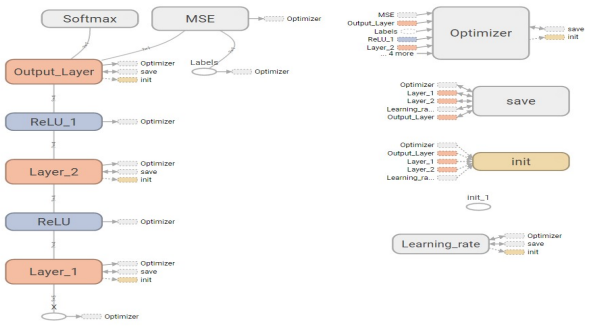

Figure 5. Tensor board for case 2

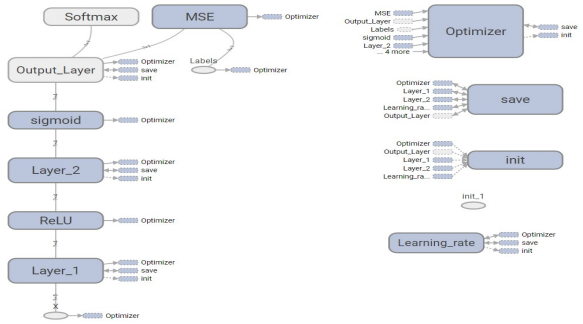

Figure 6. Tensor board for case 3

V. RESULTS AND DISCUSSIONS

An ADAM optimizer inside Tensorflow framework was used to train our model. This optimizer uses a gradient-descent algorithm. This method has a faster convergence rate compared to the stochastic gradient descent (SGD) approach [16]. The simulation settings, and parameters used in this work are all listed in Table 1.

A. *Error Metrics for Evaluation*

To assess the accuracy of the forecasting model, three metrics are used: mean absolute percentage error (MAPE), mean square error (MSE), and root mean square error (RMSE).

$$MAPE = \frac{\sum_{t=1}^{t=N} |y' - y| / y}{N} * 100\% \quad (5)$$

$$MSE = \frac{\sum_{t=1}^{t=N} (y' - y)^2}{N} \quad (6)$$

$$RMSE = \sqrt{\frac{\sum_{t=1}^{t=N} (y' - y)^2}{N}} \quad (7)$$

where,

y' = forecasted load , y= actual load and N= test set size.

B. *Simulation settings*

The simulation settings, and parameters used in this work are listed in Table 1.

Table 1. TensorFlow Parameters

Parameters	***Values***
Total number of samples	*2208*
Training samples	*2184*
Test samples	*24*
Minimum load data at same hour on previous day, P.DD-1	*3640.1 KW*
Minimum load at same hour on previous week, P.DD-6	*3640.1 KW*
Minimum temperature	*3 degrees Celsius*
Maximum load at same hour on previous day, P.DD-1	*7177.2 KW*
Maximum load at same hour on previous day, P.DD-6	*7177.2 KW*
Maximum temperature	*35.7 degrees Celsius*
Maximum irradiance	*975.9 kw*
Maximum wind-speed	*8.1 m/s*
Learning rate	*0.001*
Training epochs	*500000*

From Table 2, it is obvious that ELU function outperformed the other functions. This is due to the fact that ELU does not saturates for the larger values of input x, and the mean of the ELU activation function is closer to zero. This is because of the negative portion of the function characteristics that ensures faster and accurate learning of DNNs.

Table 2. MAPE evaluations for entire 90-day data sets

Activation function	Number of neurons	MAPE(%)	RMSE
Case 1: Single layer ReLU	4	1.641	111.02
	6	1.66	112.88
	8	1.77	118.17
	10	1.62	110.86
	12	1.628	109.86
Case 2: Double layer ReLU- ReLU	3	1.52	115.92
	4	1.73	104.4
	5	1.83	120.14
	7	1.6	109.9
	9	1.48	104.29
Case 3: Double layer ReLU- Sigmoid	3	3.7	297.3
	4	2.2	172.8
	6	1.9	154.2
	8	2.7	205.8
	10	3.9	254.5
Case 4: Single layer ELU	3	1.79	119.46
	4	1.5	103.54
	6	1.71	113.96
	8	1.56	107.74
	10	1.49	103.90
Case 5: Double layer ELU- ELU	3	1.008	73.42
	4	1.192	85.74
	6	1.52	107.9
	8	1.36	98.06
Case 6: Double layer RELU- ELU	3	1.86	121.16
	4	1.58	106.52
	6	1.39	97.67
	8	1.77	117.49
	10	1.93	127.11
Case 7: Double layer Sigmoid- ELU	3	2.34	189.75
	4	2.63	182.63
	6	2.6	185.01
	8	3.05	213.62
	10	3.29	211.667

Table 3. Weekend MAPE evaluations

Activation function	Number of neurons	MAPE(%)	RMSE
Case 1: Single layer ReLU	3	1.29	73.22
	4	1.3	74.53
	6	1.31	74.95
	8	1.32	75.29
Case 2: Double layer ReLU- ReLU	4	1.25	70.94
	6	1.22	63.81
	8	1.16	64.88
	10	1.20	67.66
Case 3: Double layer Sigmoid -ReLU	4	1.39	82.31
	6	1.51	100.52
	8	2.03	119.65
	10	1.59	94
Case 4: Single layer ELU	4	1.34	76.1
	6	1.27	72.29
	8	1.36	77.53
	10	1.34	76.35
Case 5: Double layer ELU- ELU	4	1.53	79.2
	6	1.52	86.3
	8	1.31	75.09
	10	1.27	72.34
Case 6: Double layer RELU- ELU	4	1.53	86.47
	6	1.21	68.43
	8	1.41	80.24
	10	1.39	78.47
Case 7: Double layer Sigmoid- ELU	4	1.28	73.17
	6	1.69	106
	8	2.37	133.6
	10	1.96	126.22

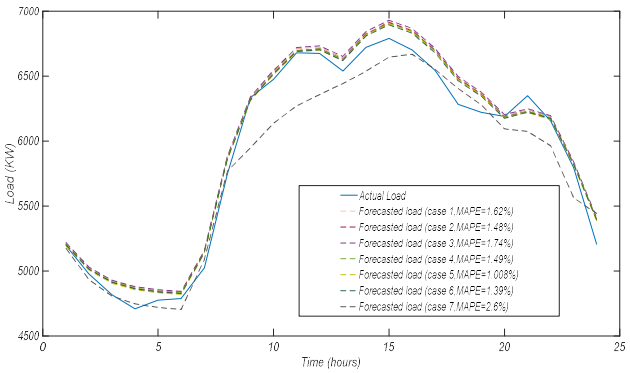

Figure 7. Comparison of different cases load forecast with actual data versus MAPE values

Fig. 7 shows the forecasted graph of all the 7 cases with actual demand data. It is observed that case 5 yields better results, and case 7 yields poor results. The ELU-ELU combination with two layers performs better with the 90-day data set. The authors believe this is due to the closer correlation of the 90-day pattern profile with the ELU activation function.

Table 4. Weekday MAPE Evaluations

Activation function	Number of neurons	MAPE(%)	RMSE
Case 1: Single layer ReLU	4	2.19	150.21
	6	2.20	150.02
	8	2.29	157.66
	10	2.31	159.67
Case 2: Double layer ReLU- ReLU	4	2.31	158.5
	6	2.34	161.65
	8	2.36	162.63
	10	2.30	158.748
Case 3: Double layer Sigmoid - ReLU	4	3.27	241.92
	6	3.44	249.11
	8	3.13	206.62
	10	3.24	206.56
Case 4: Single layer ELU	4	2.25	154.44
	6	2.24	154.27
	8	2.19	150.3
	10	2.20	151.39
Case 5: Double layer ELU- ELU	4	2.29	158.67
	6	2.03	142.15
	8	2.43	169.89
	10	2.42	167.86

Case 6: Double layer RELU- ELU	4	2.31	159.2
	6	2.12	143.24
	8	2.38	165.08
	10	2.27	155.84
Case 7: Double layer Sigmoid-ELU	4	2.75	182.01
	6	3.97	250.13
	8	3.28	216.19
	10	3.6	234.53

In Table 4, weekdays and weekends are treated separately, In this case ELU and ELU combination has a MAPE of 2.03% outperformed other cases.

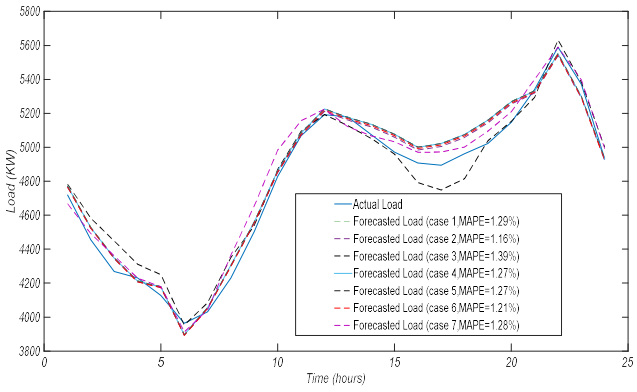

Figure 8. Weekend Forecasts

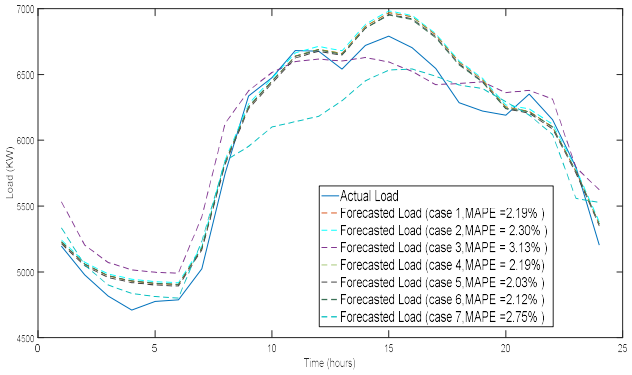

Figure 9. Weekday Forecasts

ELU function performed, because it has improved learning characteristics compared to other activation functions. In contrast to the Sigmoid and ReLU activation functions, the ELU values are negative in x-axis which allows to make the mean value closer to zero, and this speed up the learning process, and enable the gradient closer to the unit's natural gradient.

VI. Conclusion

The paper investigated the application of DNN's in electric power system analysis. This paper uses a 90-day Iberian market dataset to predict the day-ahead loads. Multiple combinations of activation functions were trained, and tested with both single and double-layer neural networks. The results indicate that the combination of ELU with ELU performs better than other combinations when evaluated against MAPE values. On weekend data sets, the ReLU-ReLU combination outperform the other combination. The scalability of the model on its accuracies are yet to be evaluated for larger data sets.

References

[1] Srivastava, A. K., Ajay Shekhar Pandey, and Devender Singh. "Short-term load forecasting methods: A review." *Emerging Trends in Electrical Electronics & Sustainable Energy Systems (ICETEESES), International Conference on*. IEEE, 2016.

[2] http://www.datasciencecentral.com/profiles/blogs/prediction-algorithms-in-one-picture.

[3] Raza, Muhammad Qamar, and Abbas Khosravi. "A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings." Renewable and Sustainable Energy Reviews 50 (2015): 1352-1372.

[4] Hosein, Stefan, and Patrick Hosein. "Load Forecasting using Deep Neural Networks."

[5] Ryu, Seunghyoung, Jaekoo Noh, and Hongseok Kim. "Deep neural network based demand side short term load forecasting." Energies 10.1 (2016): 3.

[6] Hernández, Luis, et al. "A multi-agent system architecture for smart grid management and forecasting of energy demand in 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] Charytoniuk, W., and M. S. Chen. "Neural network design for short-term load forecasting." Electric Utility Deregulation and Restructuring and Power Technologies, 2000. Proceedings. DRPT 2000. International Conference on. IEEE, 2000.

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

[10] Shi, Heng, Minghao Xu, and Ran Li. "Deep Learning for Household Load Forecasting–A Novel Pooling Deep RNN." IEEE Transactions on Smart Grid (2017).

[11] Soekhoe, Deepak, Peter van der Putten, and Aske Plaat. "On the Impact of Data Set Size in Transfer Learning Using Deep Neural Networks." International Symposium on Intelligent Data Analysis. Springer International Publishing, 2016.

[12] http://image-net.org/challenges/LSVRC/2016/index

[13] https://algorithmsdatascience.quora.com/ReLu-compared-against-Sigmoid-Softmax-Tanh

[14] http://cs231n.github.io/neural-networks-1/

[15] Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter. "Fast and accuate deep network learning by exponential linear units (elus)." *arXiv preprint arXiv:1511.07289* (2015).

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