Predicting Electric Power Energy, Using Recurrent Neural Network Forecasting model

Nawzad M. Ahmed [1,2] and Ayad O. Hamdeen[1]

1Department of Statistics & Informatics, College of Administration & Economy, University of Sulaimani, Kurdistan Region – F.R. Iraq

2Department of Finance & Banking, College of Administration and Economics, University of Human Development Sulaimani, Kurdistan Region – F.R. Iraq

Abstract—Electricity is counted as a one of the most important energy sources in the world. It has played a main role in developing several sectors. In this study two types of electricity variables have been used, the first was the demand on power energy, and the second was the consumption or energy load in Sulaimani city. The main goal of the study was to construct an analytic model of the recurrent neural network (RNN) for both variables. This model has a great ability in detecting the complex patterns for the data of a time series, which is most suitable for the data under consideration. This model is also more sensitive and reliable than the other artificial neutral network (ANN), so it can deal with more complex data that might be chaotic, seismic….etc. this model can also deal with nonlinear data which are mostly found in time series, and it deals with them differently compared to the other models. This research determined and defined the best model of RNN for electricity demand and consumption to be run in two levels. The first level is to predict the complexity of the suggested model (1-5-10-1) with the error function as (MSE: mean square error, AIC, and R^2: coefficient of determination). The second level uses the suggested model to forecast the demand on electric power energy and the value of each unit. Another result of this study is to determine the suitable algorithm that can deal with such complex data. The algorithm (Levenberg-Marquardt) was found to be the most reliable and has the most optimal time to give accurate and reliable results in this study.Keywords: recurrent neural network (RNN), artificial neutral network (ANN), Levenberg-Marquardt Algorithm.(LMA).Mean square error (MSE).Akaike information criteria(AIC).Coefficient of Determination (R2). Feed forward Neural Network (FFNN**).**

I. Introduction [4], [8]

This Artificial Neural Networks are comparatively crude electronic models based on the neural network structure of the human brain. The human brain essentially learns from experience. It is natural proof that some issues are beyond the domain of current computers are indeed solvable by small energy efficient packages. Human brain modeling also promises less technical way to develop machine solutions. This new approach to computing also provides more nimble degradation during system overload than its more traditional counterparts(1), it differs from conventional (digital or analog) computing machines that serve to replace, develop or speed-up human brain computation without regard to organization of the computing elements and of their networking. Still, it can be accentuated that the afforded of simulation by neural networks is very great(2), furthermore, Recurrent Neural Networks (RNNs) are sensitive type of artificial neural network models that are well suitable for pattern classification functions whose inputs and outputs are sequences.. RNNs are very expressive and can implement arbitrary memory-bounded computation, and as a result, they can likely be configured to achieve nontrivial performance on difficult sequence functions. However, RNNs have turned out to be difficult to train, especially on problems with complicated long-range temporal structure – precisely the setting where RNNs ought to be most useful. Since their potential has not been realized, methods that address the difficulty of training RNNs are of great importance (3). (RNNs) are subclass ANNs connectionist models that capture the dynamics of sequences via cycles in the network of nodes. Unlike standard feed forward neural networks, recurrent neural networks keep a state that can represent information from an arbitrarily long context window. Although RNN have been difficult to train, and often contain millions of parameters, recent advances in network architectures, optimization techniques, and parallel computation have enabled successful large-scale learning with them. In recent years, systems based on short-term memory (STM) and bidirectional (BRNN) architectures have demonstrated ground-breaking performance on tasks as varied as image captioning, language translation, and handwriting recognition.

II. Theory

2.Recurrent Neural Networks (RNNs)

2.1 Mathematical model for (RNN) [5], [6], [7]*:*

The Recurrent neural networks formally define the standard which forms the focus of the work, Given a sequence of input the nets (X_1, X_2, ,…….., X_T), the network computes a sequence of hidden state (H_1,H_2 ,…..,H_T) , and a sequence of prediction or estimation ($\hat{Y}_1$,$\hat{Y}_2$,…….,$\hat{Y}_T$), by iterating the equations.

$$T_i = W_{hx}X_i + W_{hh}H_{t-1} + B_h \quad (1)$$

$$H_i = E\,(T_i) \quad (2)$$

Journal of University of Human Development
Volume 4 No. 2(2018); DOI: 10.21928/juhd.v4n2y2018.pp53-60
Regular research paper: Received 15 March 2018; Accepted 2 May 2018; Published 30 June 2018
Corresponding author's e-mail: nawzad.mahmud@univsul.edu.iq

$Z_i= W_{yh}H_i+ B_y$ (3)

$\hat{Y}_i= f(Z_i)$ (4)

W_{hx}: The weight matrices between input layer and hidden layer.

W_{yh}: The weight matrices between hidden layer and output layer.

W_{hh} : The matrix of recurrent weights between the hidden layer nodes at adjacent time steps. And f : The activation functions. B_h: Bias of hidden layer.

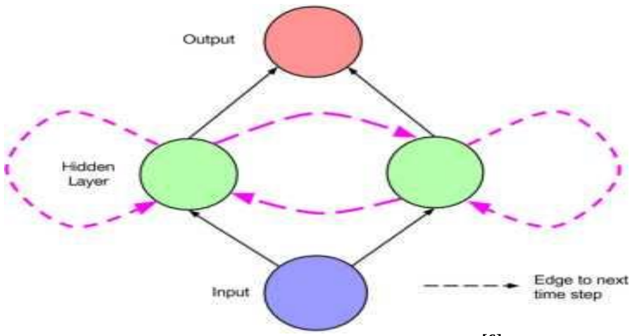

Fig.1: Simple Recurrent Neural Network [6]

Similar to supervised learning-instead of being provided with the correct output value for each given input; In reinforcement learning, the learning of an input and output mapping are performed through continued interaction with environment due to minimize a scalar index of performance.

2.2 Weights [8]

Each neuron has a specific weight and it directly effects on our input. Compared to a biological neurons quantity weight which is corresponding to strength of synaptic connection; weight values are associated with each vector and node in the network, and these values constrain how input data are related to output data. Weight values associated with individual nodes are also known as biases. These values are determined by the iterative flow of training data through the network, i.e., these are established during a training phase in which the network learns how to identify particular classes by their typical input data characteristics. Relative to biological neurons, weight values (parameters) are corresponding to the strength of synaptic connections; this is explained in figure (2), so the effect of (Xi) inputs on (Y) can be possibly determined by using its weight values. For example, cases like impressed giving in the banks, the importance of salary and age of the person who takes the impress can be determined by component weight then compared with output.

$Net=\sum_{j=1}^{n} x_j w_{ij}$, $y_{j}= \sum_{j=1}^{n} x_j w_{ij}$ or $\hat{y}=xw$ (5)

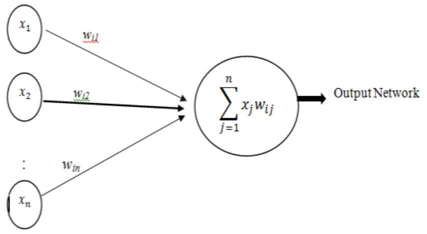

Fig.2: the weight values corresponding to the strength of synaptic connections

2.3 Bias [8]

Another parameter will be added to the net function and the bias improves the performance of the neural networks. This neuron lies in one layer, which is connected to all the neurons in the next layer, but none of the previous layers. Since the bias neuron emits 1 the weights, connected to the bias neuron, are added directly to the combined sum of the other weights. If the bias is present then the net is calculated as:

$Net= \sum_{j=1}^{n} x_j w_{ij} +b$, $y_{j}= \sum_{j=1}^{n} x_j w_{ij} +b$ or $\hat{y}=xw +b$ (6)

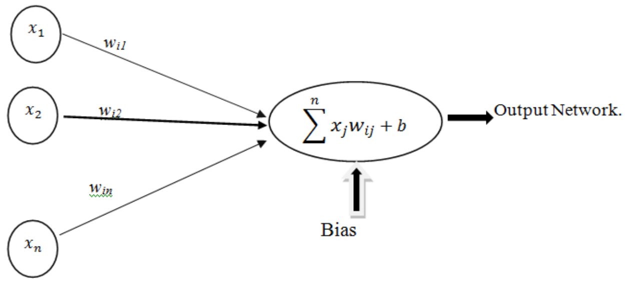

Fig.3: the network with bias [6]

2.4 Difference between RNNs and FFNNs [9]

As described in the neural networks, they can be classified into two types; feed-forward neural networks (FFNNs) and Recurrent Neural Networks (RNNs). FFNNs differ from RNNs regarding to feedback connection between the neurons in the latter. In FFNNs there are not any feedback connections between its neurons. In contrast RNNs allow feedback connections among its neurons at least once, in which the network topology can be very general; each neuron can be connected to each other, even to itself. It is allowing the presence of feedback connections between neurons, which has an advantage; it leads naturally to an analysis of the networks as dynamic systems. A recurrent neural network is another artificial neural network (ANN), while connections between units are from a directed cycle with having loops in the networks as shown below.

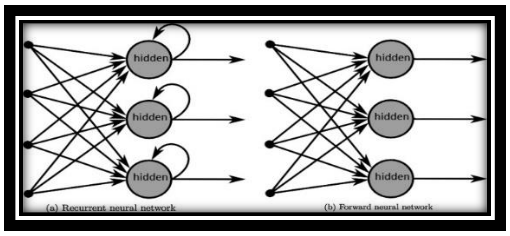

Fig.4: shown the differencing between RNNs and FFNNs

2.5 Levenberg-Marquardt algorithms (LMA) [10]

The perceptron can be trained by adjusting the weights of the inputs with Supervised Learning. In this learning technique, the patterns to be recognized are known in advance, and a training set of input values are already classified with the desired output. Before commencing, the weights are initialized with random values. Each training set is then presented for the perceptron in turn. For every input set the output from the perceptron is compared to the desired output. If the output is correct, no weights are altered. However, if the output is wrong, we have to distinguish which of the patterns we would like the result to be, and adjust the weights on the currently active inputs towards the desired result. The LMA, also known as the damped least-squares (DLS) method, is used to solve non-linear least squares problems. These minimization problems arise especially in least squares curve fitting. While back-propagation is a steepest descent algorithm, the LMA is a variation of Newton's method. The advantage of Gauss–Newton over the standard Newton's method is that it does not require calculation of second-order derivatives. The Levenberg-Marquardt algorithm trains an ANN faster (10–100 times) than the usual back-propagation algorithms. This algorithm is used in many software applications for solving generic curve-fitting problems. However, as for many fitting algorithms, the LMA finds only a local minimum, which is not necessarily the global minimum. The LMA interpolates between the Gauss–Newton algorithm (GNA) and the method of gradient descent. The LMA is more robust than the GNA, which means that in many cases it finds a solution even if it starts very far off the final minimum.

2.6: Some types of Performance measure to choose the best network [11] *:*

1. Akaike Information Criterion (AIC)

The statistical measure named by (Akaike Information Criterion), which is one frequently used criterion for nonlinear model identification. AIC formula is given by:

$$\text{AIC} = -2\log L + 2m \tag{7}$$

Where: m: is the number of weights (parameters) used in RNN, and also:

$$-2\log L = -2\left[\sum_{t=1}^{n} [\log(2\pi) + \log \sigma_e^2 + \frac{(yt-\hat{y}_t)^{\wedge}2}{\sigma_e^2}\right] \tag{8}$$

σ_e^2 : The error of variance. y_t: Desired output to the network. $\hat{y}_t$: The network output at time (t), n: The number of input observation to train the network.

Or

$$\text{AIC} = n*\ln(\text{SSE} / n) + 2m \tag{9}$$

Such that:

n: The number of training cases.

m: Denotes number of parameters of weights in suggested (RNN).

m = n (nh +1) +2nh +1

nh: The Number of nodes in hidden layer(s).

The measure of Fitness model is given by:

$$\text{Fitness} = 1 / \text{Testing set (MSE)},\ 0 \le Fitness < \infty \tag{10}$$

The decision of disability for these (RNNs) was made with respect to the accuracy measure values (Fitness and AIC) for each design, maximum fitness corresponding minimum AIC value indicates the best RNN architecture performance. These measures of accuracy were used for all (RNN), which candidate during this study is (R^2).

$$R^2 = 1 - \frac{SS_{residual}}{SS_{total}} \tag{11}$$

2.7 Non-linear autoregressive moving average model (NARMA) [12]

In this case focus on nonlinear of ARMA model for recurrent neural network and how to apply (NARMA) model in RNN. Let us have a simple non-linear generalization of ARMA (p,q) model:

$$x_t = \delta\left(x_{t-1}, x_{t-2}, \dots, x_{t-p}, \epsilon_{t-1}, \epsilon_{t-2}, \dots, \epsilon_{t-q}\right) + \epsilon_t \tag{12}$$

x_t: denoted the set of observation depend on time (t).

ϵ_t: denotes random noise, independent of past (x_t).

δ: is an unknown smooth function with the assumption the best (MSE).

The prediction equation is:

$$\hat{x}_t = \hat{\delta}(x_{t-1}, x_{t-2}, \dots, x_{t-p}, \hat{\epsilon}_{t-1}, \hat{\epsilon}_{t-2}, \dots, \hat{\epsilon}_{t-q}) \tag{13}$$

w_{ij}'': denotes the coefficients of a full matrix weights.

f: denotes the activation function.

If the model $\hat{\delta}(x_{t-1}, x_{t-2}, \dots, x_{t-p}, \hat{\epsilon}_{t-1}, \hat{\epsilon}_{t-2}, \dots, \hat{\epsilon}_{t-q})$is chosen, then the RNN approximate it as.

$$\hat{x}_t = \delta\left(x_{t-1}, x_{t-2}, \dots, x_{t-p}\right) = \sum_{i=1}^{I} W_i\, f\left(\sum_{j=1}^{p} w_{ij} x_{t-j} + \sum_{j=1}^{q} w_{ij}' \left(x_{t-j} - \hat{x}_{t-j}\right)\right) \tag{14}$$

This model is a special case of the fully interconnected RNN

$$\hat{x}_t = \sum_{i=1}^{I} W_i\, f\left(\sum_{j=1}^{p} w_{ij}'' x_{t-j}\right) \tag{15}$$

2.8: Recurrent Neural Networks versus Feed-forward Models [13]

When using neural Networks in a dynamical system context it is important to decide about the model structure. In the context of Input/ Output models it's important to make a distinction between NARX (Non-linear Autoregressive with exogenous) and NOE (Non-linear Output Error) models. In NARX models one has.

$$\hat{O}_t = f\left(O_{t-1}, O_{t-2}, \dots, O_{t-q}, I_{t-1}, I_{t-2}, \dots, I_{t-q}\right) \tag{16}$$

O_t: denotes the true output at discrete time instant (t).

I_t: denotes the input at time (t).

$\hat{O}_t$: denotes the estimated output at time (t).

q: denotes the number corresponds to the order of the system.

In NOE models one has.

$$\hat{O}_t = f(\hat{O}_{t-1}, \hat{O}_{t-2}, \dots, \hat{O}_{t-q}, I_{t-1}, I_{t-2}, \dots, I_{t-q}) \tag{17}$$

Note that one has a recursion now on the variable ($\hat{O}_t$) in constraint with the NARX model. From a neural networks perspective, the NARX model may be considered as a FFNN model, while the (NOE) model is Recurrent Neural Network. Then, models for time series prediction are closely related to these models by omitting the input variable (I), one obtains then:

$$f(O_t, O_{t-1}, \dots, O_{t-q}) \tag{18}$$

Which is parameterized by a (Multi-Layer-Perceptron, MLP), given by:

$$\hat{O}_{t+1} = W' \tanh\left(V[O_t, O_{t-1}, \dots, O_{t-q}] + \beta\right) \tag{19}$$

V: Nonlinear function, or activation function (squash function).

It is not necessary that the past values$(O_t,\ O_{t-1}, \dots, O_{t-q})$ are subsequent in time certain values could be omitted or values as different time scales could be taken. In order to generate prediction, the true values $(O_t)$are replaced then by the estimated values ($\hat{O}_t$) and the iterative is generated by the RNN.

$$\hat{O}_{t+1} = W' \tanh\left(V[\hat{O}_t, \hat{O}_{t-1}, \dots, \hat{O}_{t-q}] + \beta\right) \tag{20}$$

For a given initial condition Instead of Input/ Output models one may also take discrete time non-linear state space descriptions.

$$\begin{cases} \hat{X}_{t+1} = f\ (\hat{X}_t, I_t) \\ \hat{O}_t = g\ (\hat{X}_t) \end{cases} \tag{21}$$

Recurrent Neural Network models are used in control application, where one first identifies a model and then design a controller based upon the identified model and applies it to the real system, either in a non-adaptive or adaptive setting. When using neural networks in a dynamical systems context, one should be aware that even very simple Recurrent Neural Networks can lead to complex behavior such as chaos. In this sense stability issues of multi-layer RNNs are important e.g. towards applications in signal processing and control. The training also more complicated than for FFNNs. In the RNN case a cost function is defined on a dynamical system (iterative system) which leads to more complicated analytic expressions for the gradient of the cost function.

III. Application

In order to designate the best architecture for an (RNN) model that gives a best performance for data under consideration, then the following steps must be applied:

3.1Recurrent Neural Network Designing: Data Description

The data were collected from the province of Sulaimani / Directorate of control and communication for electricity during the January of 2013 to July of 2015 in the average of daily power energy (load and demand) as (940 consecutive Obs.) as at time series (t=1,2,….,940). The data is measured by Ampere (A).

3-2 The Application Steps of Recurrent neural Networks

This part includes the application for creating Recurrent Neural Networks (RNN) for time series prediction. The Matlab software (R2014a V.8.3.0.532) has been used to apply (RNN) for data that described above. The application of Recurrent Neural Network for time series prediction in this thesis was done with the following steps:

First step

In this case, we identify data within MATLAB software, because we need two types of data in (RNN) (input data is demand on power energy(Xt; t= 1,2,3,4,…..,940)), and target data is load power energy (Yt; t= 1,2,3,4,…..,940)), the data that we have contains two types (Demand power and Load power) in this thesis, the input data as the Demand power (Xt) and target data as a Load power (Yt).

Second step: Normalization of data

The goal of this step is to normalize the data and make them bounded between (-1, 1), or (0, 1) this coding depend on the behavior of the type of ANN or RNN that we make use of, especially if the data under consideration contains complex patterns as we have in our data under consideration it may be more flexible to use the first type of normalization above. This process is only for coding the observations of time series data to make them understandable inputs for all layers to the recurrent neural network layers. The equation below is a type of normalization for input time series data in the range (-1, 1).

$$Z = \left(\frac{xi - \min(xi)}{\max(xi) - \min(xi)} * 2\right) - 1 \tag{22}$$

Zi: normalized observation. (i=1,2,3,…,n) , Xi= origin obs. Max (xi): Maximum value of series (xi), Min (xi): Minimum value of series (xi).

Note: (data files were taken from center of electric power energy distribution in Sulaimani).

Third step: Data Partitioning

In artificial neural networks generally two types of partitioning can be used the first type is named by sequential partition that divided the data sequentially from first observation to the last with the order themselves using the proportion that the researcher suggested it as (%70) for training set, and (%15) for each testing and validation set respectively the second type is randomly partitioning as the researcher used it in this thesis. The random partition is better than the other because the only random partition can recognize the complex patterns in prediction or forecasting models. But the only disadvantage of the random partition is the researcher can't return to the steps of application during the process that is because of the random choose of sets which are made by the software in the partition. Then numerically the partitions for the ratios are as follow. The first set for training the network with input data is equal to (658) observations, the second set for testing the network with input data is equal (141) observations and the third set for validation chosen with input data is equal (141) observations.

Forth step: Create the Network architecture

In this case, the structure of the recurrent neural network that contains three layers (Input layer, Hidden layer(s) and output layer). At this stage we choose the best way for the performance of the network, two important measures, the first is the (AIC), and the second is the fitness coefficient value for choosing best recurrent neural network having the best performance that recognized all complex patterns exists, which depends on minimum (AIC) and maximum (fitness), in this study the best RNN chosen to analyze data, two hidden layers network is used, which is explained in table (2) and figure (5) below.

Table (2) Represent the best architecture of (RNN)

Layers	Nodes	Activation function
Input layer	1	---------------
Hidden layers	5	Tansig (hyperbolic)
	10	Tansig (hyperbolic)
Output layer	1	Purelin (linear)

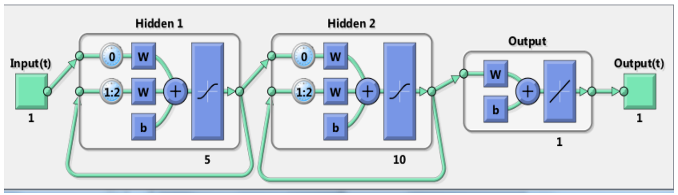

Fig.5: represent the best architecture RNN model (1-5-10-1)

The best Network is [1-5-10-1], depend on Maximum fitness, Minimum AIC and mean square error for (training, testing, validation and overall data set). By using applying the activation function between layers and change the number of nodes between layers we get the best model is [Tan-sig1Tan-sig2Purelineoutput].

MSE (tr)	MSE (ts)	MSE (val)	MSE (all)	R2 (tr)	R2 (ts)	R2 (val)	R2 (all)
0.000 478	0.000 239	0.000 78	0.001 5	0.999 05	0.999 02	0.9994 6	0.999 1

MSE (train), MSE (test), MSE (validation), and MSE (all sets): Mean Square Errors for (Training, Testing, Validation, and all data set).R2tr, R2ts, R2val, and R2all: Coefficient of determinations for (Training, Testing, Validation, and all data set).

Fifth step: Training suggested network

In this case during training suggested the Recurrent Neural Network, the data would be analyzed and change weights among nodes to reflect dependencies and patterns. In this section we made use of training algorithm. Then we choose the best algorithm named by (Levenberg-Marquardt) which is explained in figure (6) that shows the best training state. It is clear that the best efficiency is occurred in epoch 12. The learning function of learning data is shown in repetition 12 in Fig (6) below:

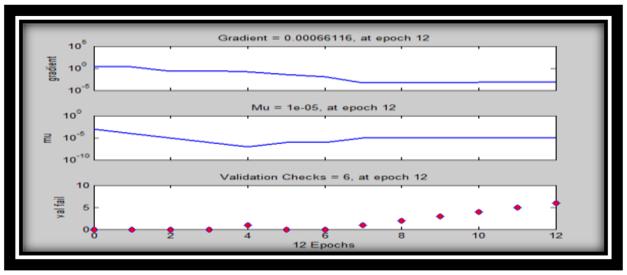

Fig.6: the training set

The variation of the gradient error (0.00066116) and validation checks at epoch (12) equal to (6).The diagram of learning errors, assessment errors and test errors and the best training performance with the best validation performance for (RNN), shown in figure (7).

In this figure below the best of training network performance is (0.00028243) at epoch (6) because the minimum global located at epoch (6).

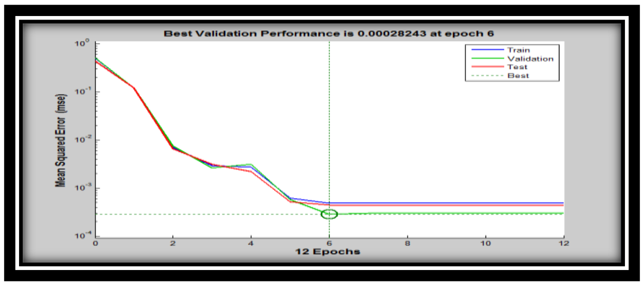

Figure (7): The training performance

The performance for this model equal to MSE= 0.0015 for all model. From the table (3): that represents finding the best of the best stages of architecture model in (RNN) for data under consideration. By using all techniques such as, changes of (number of nodes, hidden layers and activation function)

Regression Plot:

The regression plot in figure (8) consists of (R2 training, R2 testing, R2validation and R2 for all data) with the model output for each cases, (see figure below).

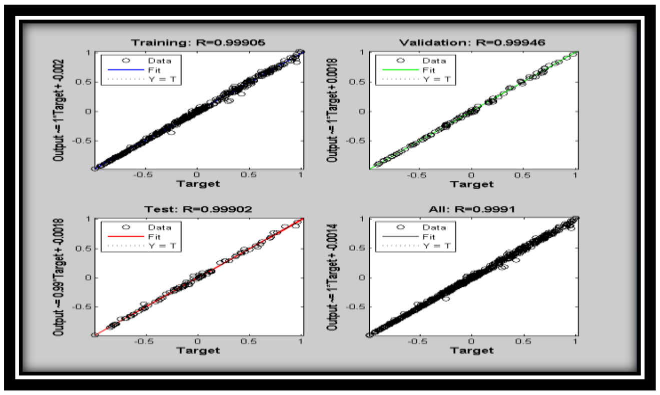

Fig.8: regression plot of (training, testing, validation, and all over data)

From the figure (8) that contains the regression plot of (training, testing, validation, and all data) shows the best performance of the detected recurrent neural network (1-5-10-1) model, also it represents the optimal architecture that represents all sets as describe above this regression plot, also tells as how can the model that we suggested is the best one among several trails for finding the best architecture moreover also the regression plot tells us that the errors that may be produced from this (RNN) is approximately distributed normally, also their weights for all layers in suggested network.

3.3 Prediction and Forecasting

From the figure above it represents as that the error produced after comparing the actual and the output of the suggested network is distributed normally that makes the result more efficient than any other weight distribution. This makes us to decide that the network in general if the errors are distributed normally it really comes from a normal weight set that was estimated for the suggested network. This also makes us to state that the model (1-5-10-1) is more generalized than the others that are not distributed normally, we can also state that if the errors are random then they tend to be distributed normally. The randomness of error that is necessary for fitting any best model.

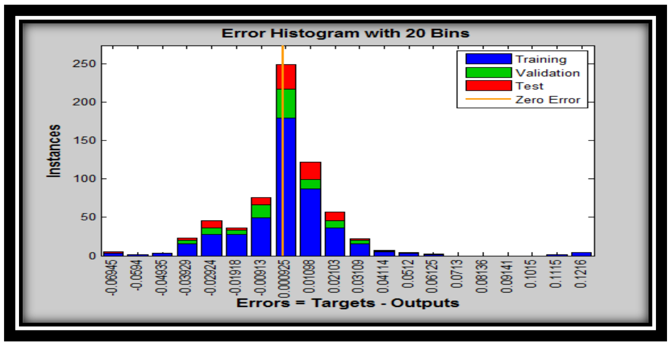

Figure (9) : errors histogram in training step.

Table (3): finding the best architectures of RNN model

Net	R^2tr	R^2ts	R^2val	R^2all	MSEtr	MSEts	MSEval	MSEall	AIC	Fitness	Itera.
1-5-5-1	0.998	0.999	0.999	0.998	0.00081	0.00035	0.00042	0.0016	-4156.04	2844.707	17
	0.998	0.999	0.998	0.998	0.00077	0.00054	0.00037	0.0017	-4116.15	1847.37	10
	0.998	0.999	0.999	0.998	0.00075	0.00049	0.00049	0.0017	-4116.15	2038.944	40
1-10-5-1	0.999	0.999	0.999	0.999	0.00045	0.002	0.00042	0.0028	-3717.81	500	18
	0.999	0.995	0.998	0.998	0.00078	0.00043	0.00043	0.0016	-4086.04	2299.221	14
	0.998	0.999	0.998	0.998	0.00074	0.00061	0.00047	0.0018	-4008.54	1649.811	25
1-5-10-1	0.999	0.999	0.999	0.999	0.00048	0.00024	0.00078	0.0015	-4234.51	4176.063	10
	0.999	0.994	0.999	0.998	0.00075	0.00043	0.00059	0.0018	-4018.54	2334.485	11
	0.999	0.998	0.993	0.998	0.00077	0.00039	0.00047	0.0016	-4096.04	2538.2	166
1-10-10-1	0.998	0.999	0.999	0.998	0.0008	0.00042	0.00038	0.0016	-3976.04	2389.772	11
	0.998	0.999	0.998	0.998	0.00077	0.00038	0.00055	0.0017	-3936.15	2647.113	13
	0.999	0.994	0.999	0.998	0.00075	0.00051	0.00049	0.0018	-3898.54	1970.172	101
1-15-10-1	0.998	0.999	0.999	0.998	0.00076	0.00059	0.00034	0.0017	-3846.15	1704.216	48
	0.998	0.999	0.999	0.998	0.00073	0.00067	0.00043	0.0018	-3808.54	1489.603	13

Table (4): The result (summarized table for some obs.) Recurrent Neural Network for time series prediction where the (R2) and (MSE) for the model (1-5-10-1) R2= 0.9991, MSE= 0.0015 respectively.

No.	Actual Data*	Prediction	No.	Actual Data*	Prediction
1	-0.77366	-0.7661	-	---------	--------
2	-0.93769	-0.9344	-	-----------	----------
3	-0.74725	-0.7392	53	-0.82219	-0.8157
4	-0.58468	-0.5737	54	0.873516	0.892
5	0.582171	0.5848	55	0.035943	0.0547
6	-0.26887	-0.2516	56	-0.01613	0.0032
7	-0.76731	-0.7597	57	0.730319	0.7367
8	-0.44057	-0.4268	58	-0.12256	-0.1033
9	-0.54721	-0.5355	59	0.887074	0.9071
10	0.256599	0.269	60	0.09349	0.1112
11	0.363957	0.3721	-	-----------	----------
12	-0.10777	-0.0884	-	-----------	-----------
13	0.44942	0.4544	138	-0.63548	-0.6254
14	-0.40331	-0.3888	139	-0.81128	-0.8046
15	0.593851	0.5965	140	-0.14145	-0.1223
16	-0.35806	-0.3426	141	-0.14688	-0.1278

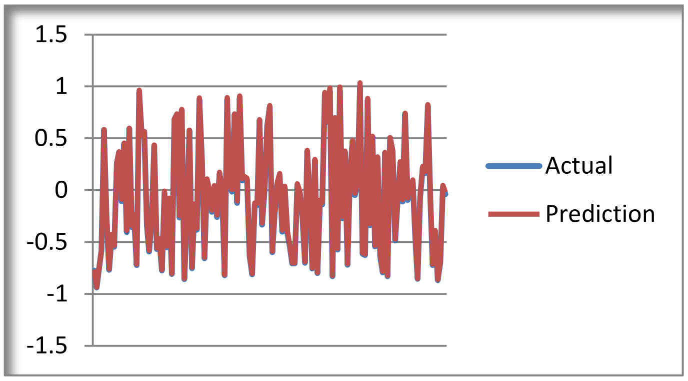

Fig.10: The difference between Actual data and prediction

From the table (5) that represents the prediction of demand on electric power energy to show the performance of suggested model (1-5-10-1) that gives as MSE= 0.0015, R2= 0.9991for over all data (training, testing, and validation set) comparing this result by the others which gives us the best among the epochs used in our data. From the figure (10) that shows the difference between the actual data (demand on electric power energy this data normalized by the equation (22) and prediction for (141 days) in the validation set.

Table (5): Forecasting for two months, the result of applying the (RNN) model (1-5-10-1) for (60) observation after (940), for normalized actual data observations, using equation (22)

No.	Actual Data*	Forecast Data	No.	Actual Data*	Forecast Data
1	0.8709	0.8274	31	-0.163	-0.1961
2	0.9001	0.8598	32	-0.268	-0.2934
3	0.9446	0.9094	33	-0.292	-0.3165
4	0.9994	0.9706	34	-0.197	-0.227
5	1	0.9713	35	-0.257	-0.2834
6	0.955	0.921	36	-0.028	-0.0704
7	0.6705	0.6101	37	-0.037	-0.0785
8	0.7312	0.6749	38	-0.806	-0.8026
9	0.7213	0.6642	39	-0.747	-0.7471
10	0.749	0.694	40	-0.388	-0.406
11	0.8436	0.7973	41	-0.235	-0.2628
12	0.9671	0.9345	42	-0.334	-0.3552
13	0.951	0.9165	43	-0.167	-0.1996
14	0.7881	0.7365	44	-0.042	-0.0833
15	0.8359	0.7889	45	-0.052	-0.0927
16	0.8713	0.8279	46	0.0235	-0.0224
17	0.7894	0.7379	47	-0.07	-0.1091
18	0.6952	0.6364	48	-0.183	-0.2147
19	0.6491	0.5875	49	-0.45	-0.4645
20	0.6071	0.5437	50	-0.48	-0.4934
21	0.3884	0.3234	51	-0.54	-0.5501
22	0.3971	0.3319	52	-0.471	-0.4849
23	0.4974	0.4315	53	-0.465	-0.4786
24	0.3777	0.3129	54	-0.39	-0.4083
25	0.3015	0.2392	55	-0.815	-0.8113
26	0.1884	0.1317	56	-0.951	-0.9401
27	-0.037	-0.0788	57	-1	-0.9855
28	-0.319	-0.3417	58	-0.989	-0.9756
29	-0.129	-0.1638	59	-0.794	-0.7916
30	0.0482	0.0005	60	-0.708	-0.7096

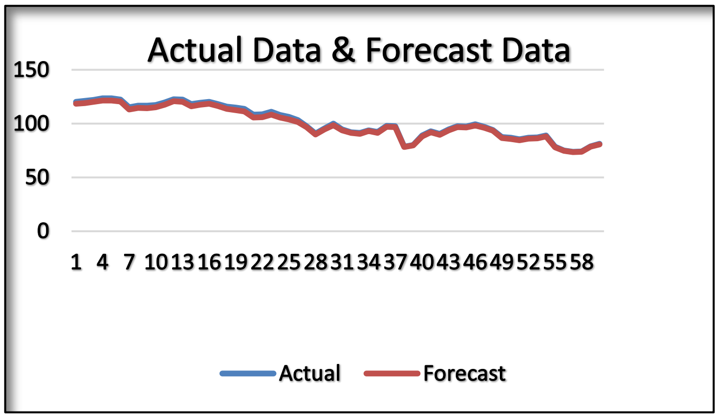

Figure (11) represents the actual data and forecast data for two months.

Table (7): The differences between actual data and forecasting for two months (Diff.= Actual data – Forecast data).after using suggested (RNN)

No.	Diff.	No.	Diff.	No.	Diff.	No.	Diff.
1	-1.886	16	-1.886	31	-0.809	46	-1.1785
2	-1.877	17	-2.007	32	-0.8047	47	-0.9286
3	-1.91	18	-2.249	33	-0.8202	48	-0.7986
4	-2.037	19	-2.372	34	-0.7943	49	-0.9487
5	-2.038	20	-2.471	35	-0.7996	50	-0.9456
6	-1.926	21	-2.532	36	-1.0237	51	-0.8911
7	-2.316	22	-2.547	37	-1.0016	52	-0.9483
8	-2.15	23	-2.616	38	-0.4327	53	-0.9493
9	-2.178	24	-2.511	39	-0.5029	54	-0.9137
10	-2.104	25	-2.301	40	-0.9115	55	-0.4242
11	-1.913	26	-1.852	41	-0.7931	56	-0.3374
12	-1.95	27	-1.0007	42	-0.8569	57	-0.2954
13	-1.92	28	-0.843	43	-0.8067	58	-0.3064

14	-2.01	29	-0.8391	44	-0.9891	59	-0.4443
15	-1.923	30	-1.2652	45	-0.9657	60	-0.5673

IV. RESULTS ANALYSIS

Some important results of application are explained from below:

1-The predicted model RNN (1-5-10-1) and its results give us an idea that the power energy system cannot be expanded in their usages because during the comparison between the load power energy and the demand, there is a very small error of prediction that we can see it in table (7), then we can recommend that if the governorate could not expand and develop the system then they will not able to provide a new or some new service and productive institutions factories because up until present time, there is a balance which can be seen clearly between electric power energy consumption and the actual power energy in use for Sulaimani.

2-The study found the best model of for data under consideration through (RNN) model is (1-5-10-1) where (1) nodes for input layer, (5) nodes for first hidden layer, (10) nodes for second hidden layer and (1) nodes for output layer with the activation functions between layers are [Tansig1 Tansig2 Pure-lineoutput], as shown the table (2), and the figure (5) represents the best architecture of (RNN) model. The suggested recurrent neural network (1-5-10-1) as required for detecting pattern of the data has a performance scale with$R2$ = 0.9991, MSE = 0.0015, Fitness model = 4176.063 and AIC = -4234.51 as shown the table (3).

3- Figure (8) the plot of regression consists of (R2 training, R2testing, R2validation and R2 all data) with the model output for each cases and shows the best performance of the detected recurrent neural network (1-5-10-1) model. Also the regression plot tells us that the errors produced from this (RNN) are approximately distributed normally.

4- Table (4) represents the result of applying the (RNN) model (1-5-10 1) for (141) observation in (validation set) for electric power energy on demand and show the result Recurrent Neural Network for time series prediction for validation set in electric power energy demand and figure (10) the difference between actual data and prediction where the (R2) and (MSE) for the model (1-5-10-1) are the R2= 0.9991, MSE= 0.0015.

5- In partial application of this study, and fitting a prediction model RNN (1-5-10-1) and using it for forecasting for nearly two months (60days) and comparing these forecast values with actual data for the same time period is a guide that the estimated RNN above is as perfect as possible to enable us to state that this model is un optimum and therefore it can be used to forecast the quantity of demand on electric power energy to know how much energy the governorate in Sulaimani needs in the future, this makes the governorate to construct an idea to treat and recover the electric power energy demand for Sulaimani.

V. REFERENCES

Anderson, Dave and McNeill, George, (1992), “ARTIFICIAL NEURAL NETWORKS TECHNOLOG”, Kaman Sciences Corporation, 258 Genesse Street, Utica, New York 13502-4627.

Azoff, E. M., (1994), “Neural Network Time series forecasting of financial markets“, Chi Chester , England , John Wiley @ sons.

Sutskever, Ilya, (2013), “TRAINING RECURRENT NEURAL NETWORKS”, a thesis submitted in conformity with the requirements for the degree of Doctor of Philosophy Graduate Department of Computer Science University of Toronto.

C. Lee Giles , Steve Lawrence, A. C. Tsoi, (2011), “Noisy Time Series Predicting using a Recurrent Neural Network and Grammatical Inference” , Machine Learning , Volume 44, Number 1/2 , July/ August, pp. 161-183 .

Martens, James & Sutskever, Ilya, (2011), “Learning Recurrent Neural Networks with Hessian-Free Optimization “ , Appearing in Proceedings of the 28th International Conference on Machine Learning, Bellevue ,WA , USA, 2011.

Lipton, Zachary C & Berkowitz, John & Elkan, Charles, (2015), “A Critical Review of Recurrent Neural Networks for Sequence Learning”, University of California, San Diego.

Menhaj. Mohammed. B, (2013), “Computational Intelligence (Val.1) FUNDAMENTAL of NEURAL NETWORKS”, Eighth Edition, Amir Kabir Publisher, Iran.

Makridakis. Spyros, Steven C. Wheelwright and Rob Hyndman, (1998), “FORECASTING Methods and Application”, third Edition, (1998), Johan Wiely and Sons, Inc, United States of America.

Alex Graves, (2008), “Supervised Sequence Labelling with Recurrent Neural Networks”, Technischue Universitat Munchen Fakultat Fur Informatik , PHD. Thesis.

K Levenberg. “A Method for the Solution Of Certain Non-linear Problems In Least Squares”, Quarterly Of Applied Mathematics, 2(2):164-168, Jul.1944.

Paulo. C, Miguel. R, Jose. N. (2001), “Evolving Time Series Forecasting Neural Network Models”, Departmento de Informatica Universidad do Minho 4750-057Braga, Portugal.

Jerome Connor, Les E. Atlas, and Douglas R. Martin (1992), “Recurrent Neural Networks and NARMA Modeling”, Interactive Systems Design Laboratory, Dept. of Electrical Engineering and Dept. of Statistics, University of Washington Seattle, Washington 98195.

Johan A. K. Suykens Jone Van Gestel, Jos De Brabanter, Bart De Moor and Joos VandewalleK. U. Leuven, Belgium (first published 2002 Reprinted 2005), “Least Squares Support Vector machines”, British Library Cataloguing – in – Publication Data