Journal of Multidisciplinary Engineering Science and Technology (JMEST)
ISSN: 2458-9403
Vol. 11 Issue 1, January - 2024

# Recurrent Neural Network (RNN)-Based Injection Substation Load Prediction And Forecasting

**Imoh, Isaiah UDofot**[1]
Department of Electrical/Electronic Engineering,
University of Uyo, Akwa Ibom, Nigeria

**Ekpa, Andikan Kenneth**[2]
Department of Electrical/ Electronic Engineering
Akwa Ibom State Polytechnic Ikot Osurua , Ikot Ekepene

***Abstract*— In this paper, Recurrent Neural Network (RNN)-based injection substation load prediction and forecasting is presented. Mathematical representation of the model is presented and the two key parameters used as the input for the model prediction and forecasting are the feeder and historical injection station load behavior. The long short term memory networks (LSTM). Architectures is used in the RNN model. The algorithm of the RNN Based on LSTM Architecture is also presented. An historical hourly feeder load profile from Udo Udoma substation in Akwa Ibom State Nigeria is used. The model was simulated in python 3 development environment using Pycharm. The load predictions are done on the scaled data which was segmented into 75% training set and 25% test set. The results show a good learning behavior for the training set and the test set as the training loss and the validation loss did not exceed 10 after 50 epochs. Also, the mean squared error for the model predictions is2.04. Since, only three months hourly data was used in the model training and validation, a one month forecasting of the feeder load was conducted using the developed RNN model.**

***Keywords*— Recurrent neural network (RNN), Injection Substation, Load Prediction, Feeder Load Forecasting**

## 1. Introduction

Over the years, various prediction models have been developed to help electrical load planners have insight about the future requirements; hence have some level of precision and accuracy on their proposals [1,2,3]. Nevertheless, there exist some inherent issues which often lead to some drifts from the expectations. For instance, the short term load forecast models has been developed by various researchers who deployed different techniques to tackle the prediction issues identified [4,5]. Each of these methods has their peculiar strengths and weaknesses with respect to parameter sensitivity, precision in prediction, and training difficulties [6,7].

Some researchers have observed that electrical load behavior has recursive and periodic pattern which is subject to the consumers' activities [8]. Such behavioral patterns exhibited by the consumers' load possess real world valued time series nature. The load characteristic data collection mechanism often possess variable dynamics over the period of time under investigation. This variance often stems from the system inherent properties in-terms of superficial intrusion and latency. Consequently, prediction precision and accuracy may significantly vary for different dataset despite the application of the same prediction or forecasting model.

A close observation on a typical electrical load data shows they are random and non-stationary in nature [9]. Their non-stationary attribute is further characterized by level transformation, outliers on time series analysis. Any data point that is located very far from other points within the cluster is considered as an outlier, and they typically result due to errors. An unsophisticated prediction models find it difficult to handle outliers. This is one of the major reasons prediction errors are large in load forecasting. Level transformation, on the other hand denotes sudden and sustained deviation from existing time series data trend. This usually results from policy, which may involve technology amendment. This research proposes application of Long Short Term Memory (LSTM)-based architecture on recurrent neural network (RNN) to solve the electrical load prediction and forecasting problems [10,11,12,13].

## 2. Methodology

### 2.1 System Design Model

The focus in this work is to present details of Recurrent Neural Network (RNN)-based injection station load prediction and forecasting. The two key parameters used as the input for the model prediction and forecasting are the feeder $(x_{fdr})$ and historical injection station load behavior $(x_{lp})$. The interconnection of these input forms nodes and each node has certain weight $w_c$ assigned to it along with a bias factor $b_i$ in the activation function, $f_a$. The system model is presented in Figure 1. The mathematical representation of the model can be expressed as follows:

$$y_p = f_a(\sum_{n=1}^{N} w_{ni} \cdot x_{ni} + b_{ni}) \quad (1)$$

Where, $x_{ni}$ is the $n^{th}$ input vector, $w_{ni}$ is the distributed weight factor for the $n^{th}$ input vector, $b_{ni}$ is the $n^{th}$ bias, $N$ is the total number of inputs to the system.

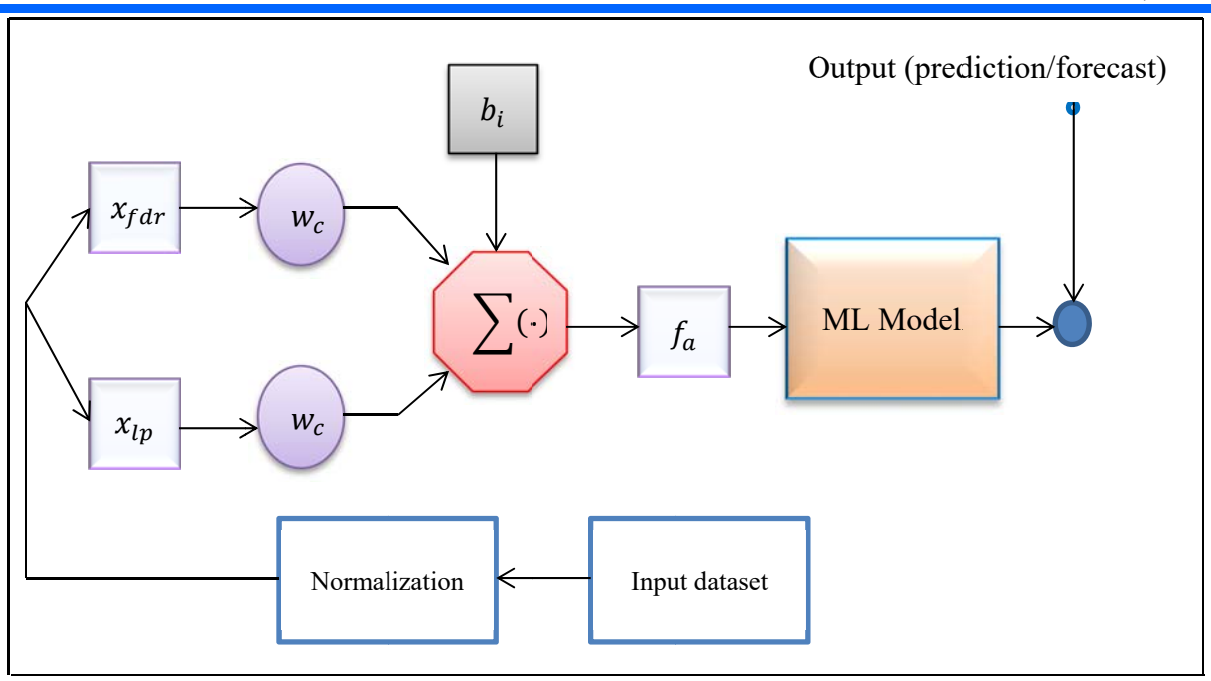


Figure 1: The system model

### 2.2 The Recurrent Neural Network Architecture

There are various architectures that exist for RNN model. However, this work adopts the long short term memory networks (LSTM). The LSTM is an architecture with various gates which regulates error flow through the LSTM cells. The architecture is trained through the back-propagation time technique. With the application of the LSTM model, the gradient issues are mitigated by rerouting the invariant error back through time. Notably, LSTM has five various gates which are non-linear functions and are interrelated in unique manner. The intrinsic status of the cells is linearly updated by the LSTM. This characteristic allows data to be easily transmitted backward across epochs. The LSTM model used in this work is presented in Figure 2.

According to the LSTM model in Figure 2, the content of $x_t$ and $y_t$ are swapped with the input and output of the model whereas, $h_{t-1}$, $h_t$, $y_{t-1}$, and $y_t$ represents the intramural state variables and their content are transmitted between the cells and the virtual layer.Furthermore, $g_1$ and $g_2$ serve as the nonlinear operators which are executed as hyperbolic tangent. Finally the sigmoid operators $(\sigma_f)$ represents the forget gate, the symbol $\sigma_u$ represents the update gate and $\sigma_o$ represents the output gate. The models for these gates are presented mathematically as follows:

Forget gate:

$$\sigma_f[t] = \sigma\left(W_f x[t] + R_f y[t-1] + b_f\right) \quad (2)$$

Candidate state:

$$\tilde{h}[t] = g_1(W_h x[t] + R_h y[[t-1] + b_h) \quad (3)$$

Update gate:

$$\sigma_u[t] = \sigma(W_u x[t] + R_u y[[t-1] + b_u) \quad (4)$$

Cell state:

$$h[t] = \sigma_u[t] \odot \tilde{h}[t] + \sigma_f[t] \odot h[t-1] \quad (5)$$

Output gate:

$$\sigma_o[t] = \sigma(W_o x[t] + R_o y[t-1] + b_o) \quad (6)$$

Output:

$$y[t] = \sigma_o[t] \odot g_2(h[t]) \quad (7)$$

Where, $x[t]$ denotes the input vector, $W_o$, $W_u$, $W_h$, and $W_f$ denote the weight matrix meted on the corresponding inputs of the cell, $R_o$, $R_u$, $R_h$, and $R_f$ states the recurrent connection weights. Each gate defined in Equation 2, Equation 4, and Equation 6 has its unique application. The $\sigma_f$ determines the dataset that should be rejected from past cell state $h[t-1]$. The surviving set from forget gate are parsed to the update gate $\sigma_u$ which determines the quantity of the most recent state $h[t]$ which should be updated with the most recent member $\tilde{h}[t]$. The present state of the LSTM cells is sieved out using $g_2(\cdot)$ nonlinear function and feed through the output gate which filters the actual output $y[t]$. Equation 2, Equation 3, Equation 4, and Equation 6 show that each state relies on the present input $x[t]$ and the past output $y[t-1]$. The diagram in Figure 2 shows that when the forget gate is 1, and the update gate is 0, then the present state is transmitted to the next interval without any modification.

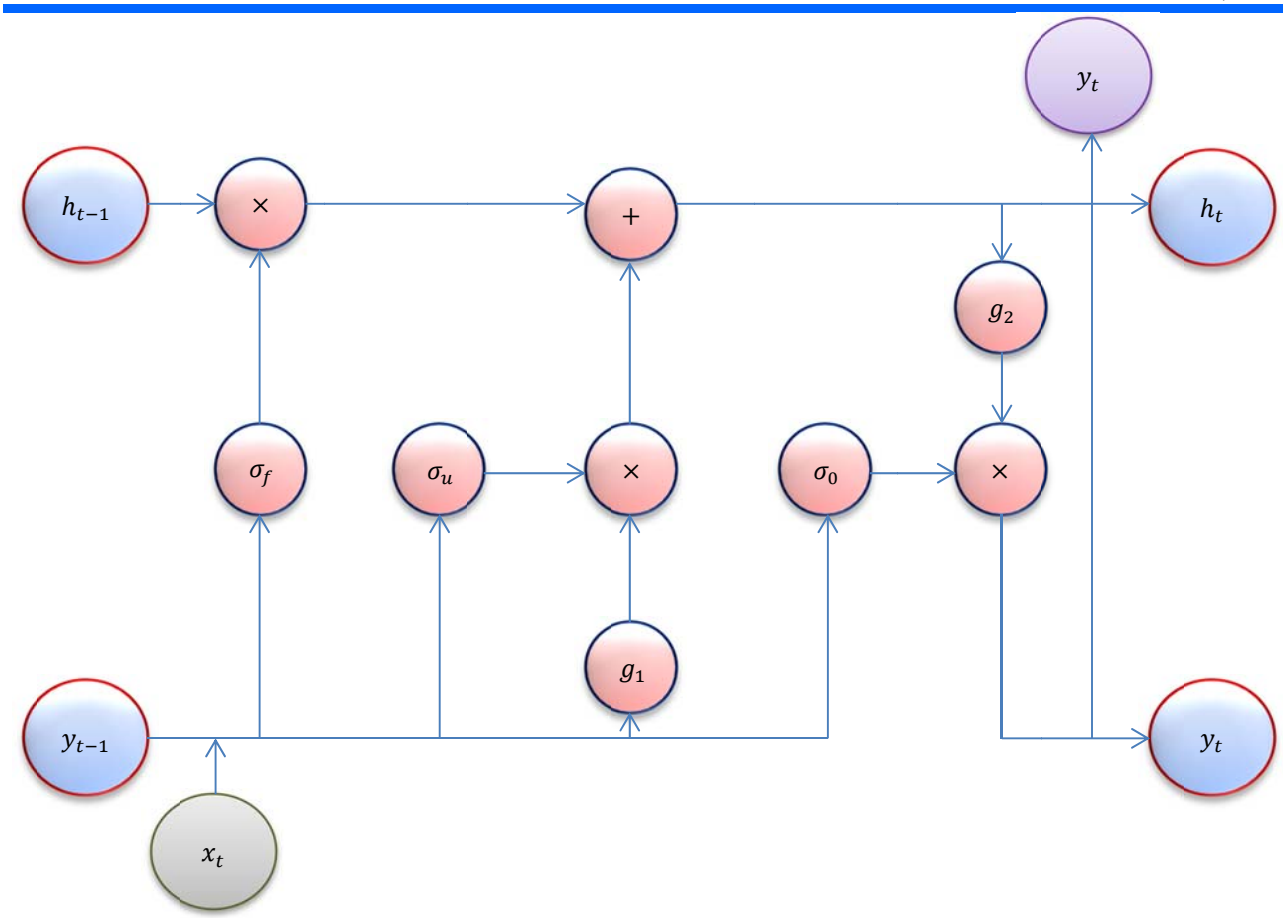


Figure 2: The long short term memory networks (LSTM) model

### 2.4 Simulation of the model

An historical load profile was gathered from Udo udoma substation in Akwa Ibom State Nigeria. The substation contains some feeders but this work utilized the dataset for Udo Udoma feeders. The data entries of Udo Udoma feeder were taken on hourly basis for four months (from May, 2022 to August, 2022). The total row count for valid data is 2808.

The model was simulated in python 3 development environment using Pycharm. First, some relevant python libraries which include numpy, pandas, matplotlib, scipy, sklearn, keras, and seaborn were installed. The collated data were entered in .csv file format before being fed to the model for prediction and forecast. The different feeder dataset were applied independently on the test model to obtain independent results using the RNN model.

The performance of the model is evaluated using Mean Squared Error (MSE) which is computed as:

$$MSE = \frac{\sum(y_i - \hat{y}_i)^2}{n} \quad (8)$$

Where, $y_i$ denotes the $i^{th}$ actual value, $\hat{y}_i$ denotes the $i^{th}$ predicted value, and $n$ denotes the number of samples considered.

## 3 Results and Discussions

### 3.1 Data cleaning and preprocessing

The raw data collated from the substation contains 17 columns of data as presented in Figure 3 however, some of the columns of records have data types and formats which are not supported by the machine models adopted in this research. The relevant columns from the raw dataset in Figure 3 include "TIME", "SEC", "FDR", "AKA", "FDR.1", "UU", "FDR.2", "IBB", "FDR.3".

The raw dataset is cleaned and in the cleaned dataset, as presented in Figure 4, all null columns and "Not a Number" (NaN) columns are replaced with zero. This is to ensure that the machine learning model can correctly handle the data.

Each row of the data items are time stamped; that means that the time stamp for each row is unique; hence, the "TIME" column is formatted as the unique index for each data row as presented in Figure 5. Also, data was collated per hour for each day. The time portion of the date time object is appended to the date portion to create the uniqueness.

A segment of the raw load dataset for Udo Udoma is shown in Figure 6 and Figure 7 shows the corresponding scaled dataset.

| TIME | TIME | SEC | FDR | AKA | FDR.1 | KVA | PF | RP | UU | FDR.2 | IBB | FDR.3 | KVA.1 | PF.1 | RP.1 | IBE | Unnamed: 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2022-05-01 01:00:00 | 01/05/2022 01:00 | 3.3 | 1 | 150 | 2.5 | 180 | 3 | 11.1 | 0.9 | 5.5 | LS | LS | NaN | NaN | NaN | NaN | NaN |
| 2022-05-01 02:00:00 | 01/05/2022 02:00 | 3.5 | 2 | 150 | 2.5 | 180 | 3 | 11.1 | 0.9 | 5.5 | LS | LS | NaN | NaN | NaN | NaN | NaN |
| 2022-05-01 03:00:00 | 01/05/2022 03:00 | 3.5 | 3 | 150 | 2.5 | 180 | 3 | 11.1 | 0.9 | 5.5 | LS | LS | NaN | NaN | NaN | NaN | NaN |
| 2022-05-01 04:00:00 | 01/05/2022 04:00 | 4.2 | 4 | 150 | 2.5 | 180 | 3 | 11.1 | 0.9 | 5.5 | LS | LS | NaN | NaN | NaN | NaN | NaN |
| 2022-05-01 05:00:00 | 01/05/2022 05:00 | 4.2 | 5 | 150 | 2.5 | LS | LS | 11.1 | 0.9 | 2.5 | LS | LS | NaN | NaN | NaN | NaN | NaN |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 2022-08-25 20:00:00 | 25/08/2022 20:00 | 20 | 0.3 | L/S | L/S | 10.8 | 0.9 | 0.3 | 150 | 2.5 | 170 | 2.8 | 10.7 | 0.9 | 5.3 | L/S | L/S |
| 2022-08-25 21:00:00 | 25/08/2022 21:00 | 20 | 0.3 | L/S | L/S | 10.8 | 0.9 | 0.3 | 150 | 2.5 | 160 | 2.6 | 10.7 | 0.9 | 5.1 | L/S | L/S |
| 2022-08-25 22:00:00 | 25/08/2022 22:00 | 20 | 0.3 | L/S | L/S | 10.8 | 0.9 | 3.9 | 150 | 2.5 | 160 | 2.6 | 10.7 | 0.9 | 5.1 | L/S | L/S |

Figure 3: The cross section of the raw dataset

| TIME | TIME | SEC | FDR | AKA | FDR.1 | KVA | PF | RP | UU | FDR.2 | IBB | FDR.3 | KVA.1 | PF.1 | RP.1 | IBE | Unnamed: 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2022-05-01 01:00:00 | 01/05/2022 01:00 | 3.3 | 2.5 | 150 | 2.5 | 180 | 3 | 11.1 | 0.9 | 5.5 | LS | LS | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2022-05-01 02:00:00 | 01/05/2022 02:00 | 3.5 | 2.0 | 150 | 2.5 | 180 | 3 | 11.1 | 0.9 | 5.5 | LS | LS | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2022-05-01 03:00:00 | 01/05/2022 03:00 | 3.5 | 3.0 | 150 | 2.5 | 180 | 3 | 11.1 | 0.9 | 5.5 | LS | LS | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2022-05-01 04:00:00 | 01/05/2022 04:00 | 4.2 | 4.0 | 150 | 2.5 | 180 | 3 | 11.1 | 0.9 | 5.5 | LS | LS | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2022-05-01 05:00:00 | 01/05/2022 05:00 | 4.2 | 2.5 | 150 | 2.5 | LS | LS | 11.1 | 0.9 | 2.5 | LS | LS | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 2022-08-25 20:00:00 | 25/08/2022 20:00 | 20 | 2.5 | L/S | L/S | 10.8 | 0.9 | 0.3 | 150 | 2.5 | 170 | 2.8 | 10.7 | 0.9 | 5.3 | 0.0 | 0.0 |
| 2022-08-25 21:00:00 | 25/08/2022 21:00 | 20 | 2.5 | L/S | L/S | 10.8 | 0.9 | 0.3 | 150 | 2.5 | 160 | 2.6 | 10.7 | 0.9 | 5.1 | 0.0 | 0.0 |
| 2022-08-25 22:00:00 | 25/08/2022 22:00 | 20 | 2.5 | L/S | L/S | 10.8 | 0.9 | 3.9 | 150 | 2.5 | 160 | 2.6 | 10.7 | 0.9 | 5.1 | 0.0 | 0.0 |

Figure 4 : Cross section of cleaned data

```
DatetimeIndex(['2022-05-01 01:00:00', '2022-05-01 02:00:00',
               '2022-05-01 03:00:00', '2022-05-01 04:00:00',
               '2022-05-01 05:00:00', '2022-05-01 06:00:00',
               '2022-05-01 07:00:00', '2022-05-01 08:00:00',
               '2022-05-01 09:00:00', '2022-05-01 10:00:00',
               ...
               '2022-08-25 15:00:00', '2022-08-25 16:00:00',
               '2022-08-25 17:00:00', '2022-08-25 18:00:00',
               '2022-08-25 19:00:00', '2022-08-25 20:00:00',
               '2022-08-25 21:00:00', '2022-08-25 22:00:00',
               '2022-08-25 23:00:00', '2022-08-26 00:00:00'],
              dtype='datetime64[ns]', name='TIME', length=2808, freq=None)
```

Figure 5: Cross section of the training data index

| TIME | | |
|---|---|---|
| 2022-05-01 01:00:00 | 0.9 | 5.5 |
| 2022-05-01 02:00:00 | 0.9 | 5.5 |
| 2022-05-01 03:00:00 | 0.9 | 5.5 |
| 2022-05-01 04:00:00 | 0.9 | 5.5 |
| 2022-05-01 05:00:00 | 0.9 | 2.5 |
| ... | ... | ... |
| 2022-08-25 20:00:00 | 150.0 | 2.5 |
| 2022-08-25 21:00:00 | 150.0 | 2.5 |
| 2022-08-25 22:00:00 | 150.0 | 2.5 |
| 2022-08-25 23:00:00 | 150.0 | 2.5 |
| 2022-08-26 00:00:00 | 150.0 | 2.5 |

2808 rows × 2 columns

Figure 6: Un-scaled data slice for Udo Udoma dataset

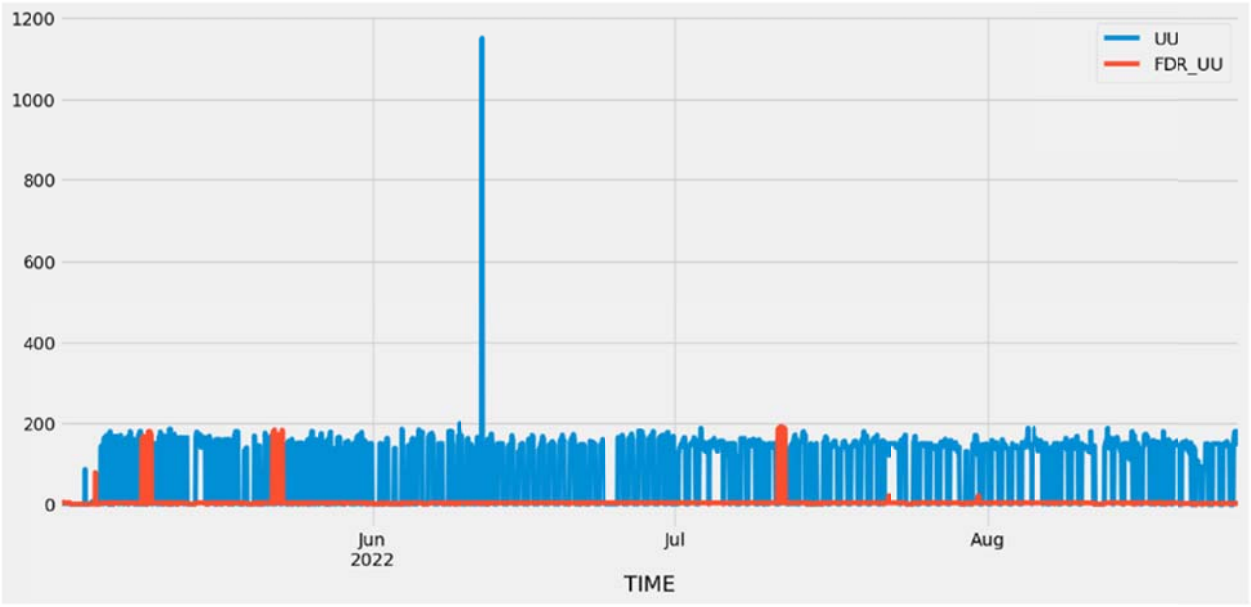


Figure 7: Visualization of raw data showing the historical load and feeder state for Udo Udoma

### 3.2 Result on the load prediction based on the RNN Model

The load predictions are done on the scaled data. Furthermore, in order to make predictions, the scaled dataset was segmented into 75% training set and 25% test set. The model prediction output is presented in Figure 8 for the Udo Udoma injection substation load prediction.

Also, graphical representation of the raw dataset for Udo Udoma feeder is shown in Figure 9 while the graphical representation of the 75% training dataset, 25% test dataset (prediction set), and forecast for the next 30 days for Udo Udoma feeder is shown in Figure 10. The results presented show a good learning behavior for the training set and the

test set as the training loss and the validation loss did not exceed 10 after 50 epochs.

| | TrainX | TrainY |
|---|---|---|
| 0 | [[-1.0202008579150579, 0.05054877894658822], [... | [-1.0316221398626126] |
| 1 | [[-1.0202008579150579, 0.05054877894658822], [... | [-1.0316221398626126] |
| 2 | [[-1.0202008579150579, 0.05054877894658822], [... | [-1.0316221398626126] |
| 3 | [[-1.0202008579150579, 0.05054877894658822], [... | [-1.0316221398626126] |
| 4 | [[-1.0202008579150579, -0.10095727451034596], ... | [-1.0316221398626126] |
| ... | ... | ... |
| 2779 | [[0.8719248513964959, -0.10095727451034596], [... | [0.8719248513964959] |
| 2780 | [[0.8084732850211923, -0.10600747629224377], [... | [0.8719248513964959] |
| 2781 | [[0.8084732850211923, -0.10600747629224377], [... | [0.8719248513964959] |
| 2782 | [[0.8084732850211923, -0.10600747629224377], [... | [0.8719248513964959] |
| 2783 | [[0.8084732850211923, -0.10600747629224377], [... | [0.8719248513964959] |

2784 rows × 2 columns

Figure 8: Prediction output for Udo Udoma dataset

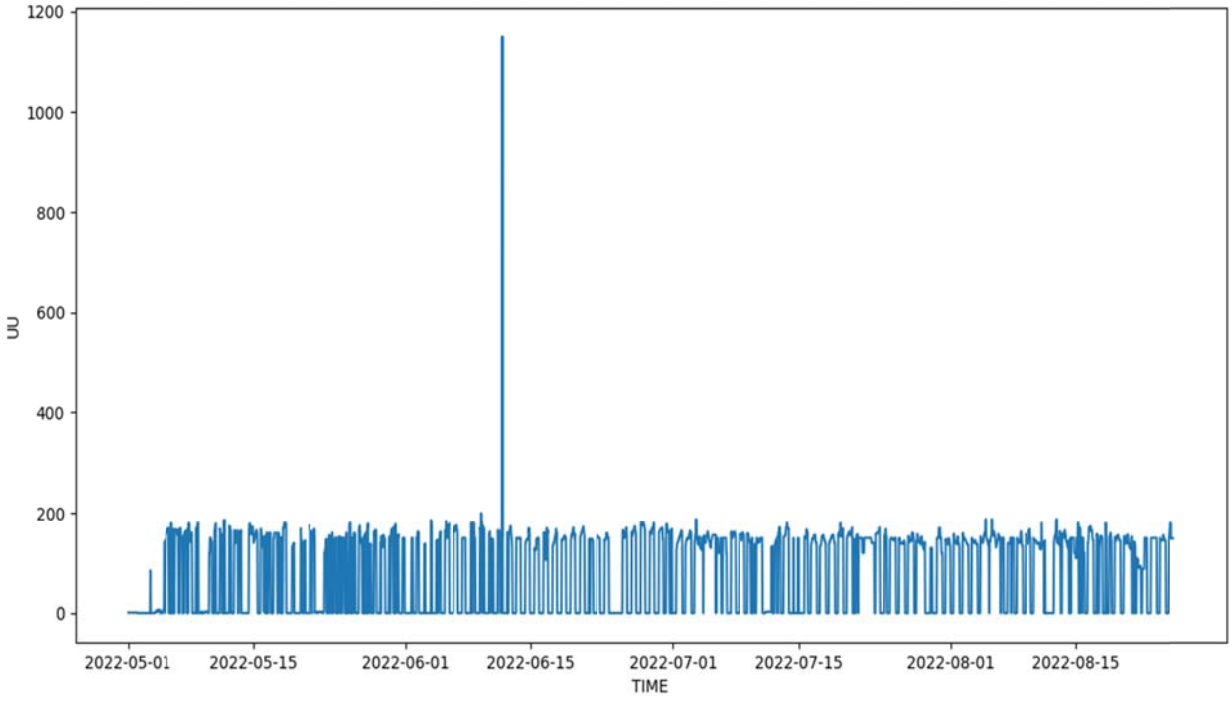


Figure 9: Graphical representation of the raw dataset for Udo Udoma

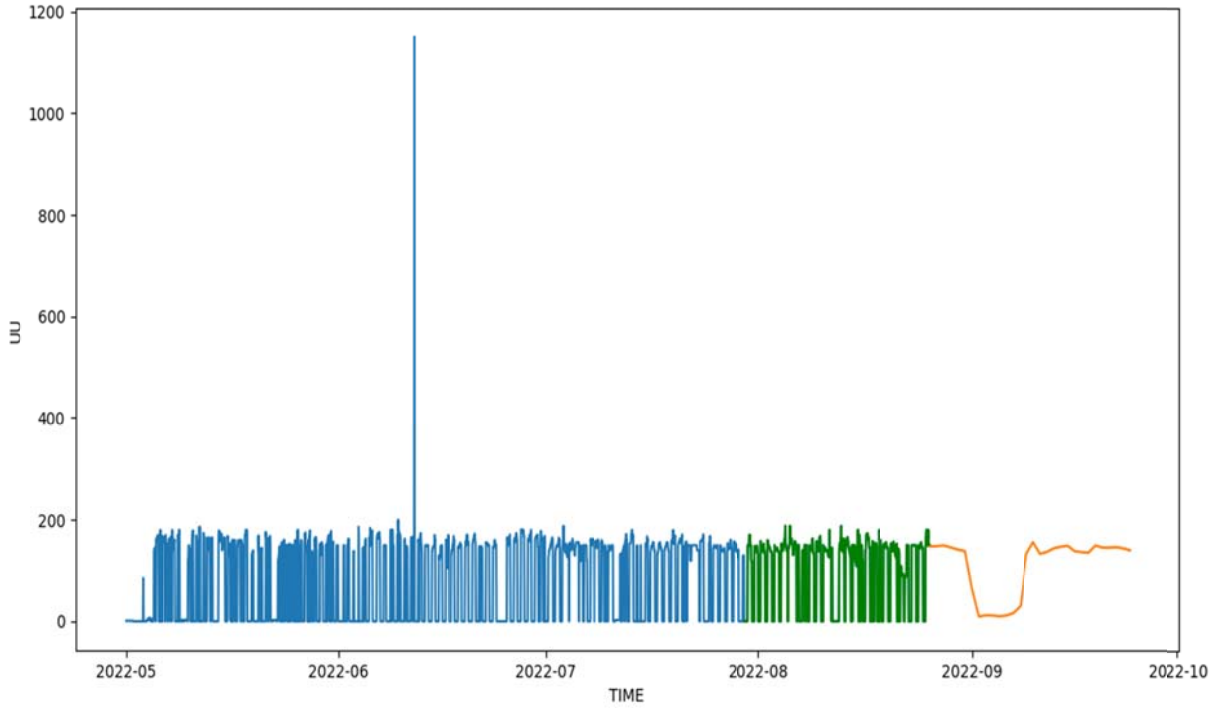


Figure 10: Graphical representation of the 70% training dataset, 30% test dataset (prediction set), and forecast for the next 30 days for Udo Udoma

### 3.3 Performance Evaluation for the RNN/LSTM Architecture

The LSTM model was constructed using the python Sequential function with 64 neurons. The activation function was set to 'relu' while the two target data columns (the historical load data and the feeder state) was fed to the input of the LSTM. The model was compiled using an optimizer called 'Adam' while the $MSE$ was computed for each of the prediction. Analysis of the model performance was performed on two metrics which include the training loss and the validation loss. Training loss shows the efficiency of the model fitness on the training data, whereas validation loss shows the efficiency of the model fitness on the new data. The dataset was trained for 50 epochs. The results in Figure 11 show the training and validation loss for Udo Udoma load data training after 50 epochs. From the results presented, the MSE for the model predictions is 2.04.

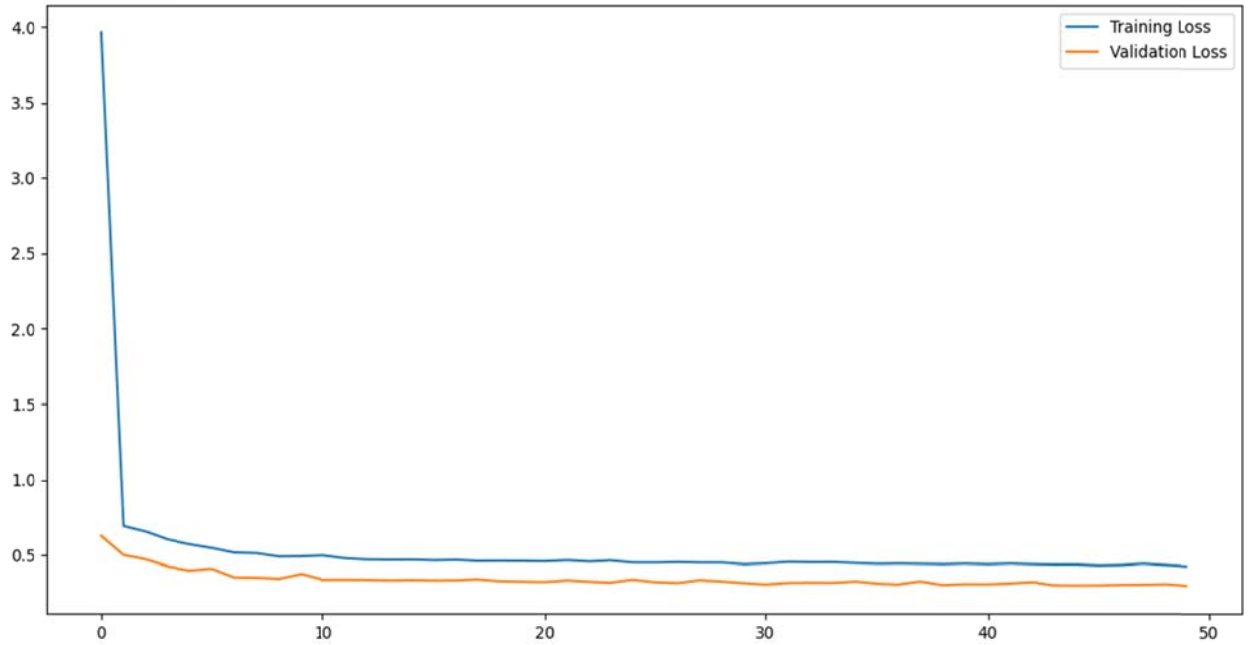


Figure 11: Training loss versus validation loss for dataset training of Udo Udoma data after 50 epochs

## 4. CONCLUSION

Application of Recurrent Neural Network (RNN) in predicting and also forecasting of injection substation load is presented. The study used the RNN to model the injection substation load based on a case study dataset and then evaluated the model performance using Mean Squared Error (MSE). Furthermore, the RNN model was used to forecast the injection load for the given case study substation. In all, the model performance shows that the RNN is suitable for the modeling and prediction of the injection load for the given case study substation.

## REFERENCES

1. Azeem, A., Ismail, I., Jameel, S. M., & Harindran, V. R. (2021). Electrical load forecasting models for different generation modalities: a review. *IEEE Access*, *9*, 142239-142263.
2. Iwendi, C., & Wang, G. G. (2022). Combined power generation and electricity storage device using deep learning and internet of things technologies. *Energy Reports*, *8*, 5016-5025.
3. Wazirali, R., Yaghoubi, E., Abujazar, M. S. S., Ahmad, R., & Vakili, A. H. (2023). State-of-the-art review on energy and load forecasting in microgrids using artificial neural networks, machine learning, and deep learning techniques. *Electric Power Systems Research*, *225*, 109792.
4. Ma, P., Cui, S., Chen, M., Zhou, S., & Wang, K. (2023). Review of family-level short-term load forecasting and its application in household energy management system. *Energies*, *16*(15), 5809.
5. Zhu, J., Dong, H., Zheng, W., Li, S., Huang, Y., & Xi, L. (2022). Review and prospect of data-driven techniques for load forecasting in integrated energy systems. *Applied Energy*, *321*, 119269.
6. Wazirali, R., Yaghoubi, E., Abujazar, M. S. S., Ahmad, R., & Vakili, A. H. (2023). State-of-the-art review on energy and load forecasting in microgrids using artificial neural networks, machine learning, and deep learning techniques. *Electric Power Systems Research*, *225*, 109792.
7. Maleki, A., Nejati, E., Aghsami, A., & Jolai, F. (2023). Developing a supervised learning-based simulation method as a decision support tool for rebalancing problems in bike-sharing systems. *Expert Systems with Applications*, *233*, 120983.
8. Eddine, M. D., & Shen, Y. (2022). A deep learning based approach for predicting the demand of electric vehicle charge. *The Journal of Supercomputing*, *78*(12), 14072-14095.
9. Trapp, A., & Wolfsteiner, P. (2021). Fatigue assessment of non-stationary random loading in the frequency domain by a quasi-stationary Gaussian approximation. *International Journal of Fatigue*, *148*, 106214.
10. Liu, Y., Li, D., Wan, S., Wang, F., Dou, W., Xu, X., ... & Qi, L. (2022). A long short-term memory-based model for greenhouse climate prediction. *International Journal of Intelligent Systems*, *37*(1), 135-151.
11. Peng, L., Wang, L., Xia, D., & Gao, Q. (2022). Effective energy consumption forecasting using empirical wavelet transform and long short-term memory. *Energy*, *238*, 121756.
12. Sak, H., Senior, A. W., & Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling.
13. Tian, Y., & Pan, L. (2015, December). Predicting short-term traffic flow by long short-term memory recurrent neural network. In *2015 IEEE international conference on smart city/SocialCom/SustainCom (SmartCity)* (pp. 153-158). IEEE.