ENERGY PREDICTION IN BUILDINGS WITH DEEP LEARNING METHODS

A PROJECT REPORT

submitted by

KANISHKA JOSE

TKM21EEPS08

to

the APJ Abdul Kalam Technological University

in partial fulfillment of the requirements for the award of the Degree

of

Master of Technology

In

Power Systems

Department of Electrical & Electronics Engineering

T.K.M COLLEGE OF ENGINEERING
KOLLAM-5

JULY 2023

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

THANGAL KUNJU MUSALIAR COLLEGE OF ENGINEERING
KOLLAM

CERTIFICATE

This is to certify that the report entitled '**Energy Prediction In Buildings With Deep Learning Methods'** submitted by **'Kanishka Jose'** to the APJ Abdul Kalam Technological Universityin partial fulfilment of the requirements for the award of the Degree of Master of Technology in Power Systems, Electrical & Electronics Engineering, is a bonafide record of the project carried out by her under our guidance and supervision. This project report in any form has notbeen submitted to any other University or Institute for any purpose.

Prof. VIKI PRASAD
Asst. Professor [Internal Supervisor]
Department of Electrical and Electronics

[EXTERNAL EXAMINER]

Prof. SHANAVAS. T. N
Associate Professor [PG co -ordinator]
Department of Electrical and Electronics

Dr. SABEENA BEEVI.K
Associate Professor, HOD
Department of Electrical and Electronics

ACKNOWLEDGEMENT

First of all, I am indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in my efforts to complete this project on time.

I am extremely grateful to **Dr. T. Shahul Hameed**, Principal, TKM College of Engineering, and **Dr. Sabeena Beevi. K**, Head of the Department, Department of Electrical and Electronics Engineering, for providing all required resources for successful completion of my project.

I am greatly obliged to **Prof. Shanavas T. N**., Associate Professor, PG co-ordinator, Department of Electrical and Electronics Engineering, for his encouragement and support.

My heartfelt gratitude to **Prof**. **Jibi P Mathew**, Asst. Professor, Project co-ordinator, Department of Electrical and Electronics Engineering, for his valuable suggestions and guidance in the preparation of the project report.

I express my thanks to **Prof. Viki Prasad**, Asst. Professor, Internal supervisor, Department of Electrical and Electronics Engineering, and all staff members and friends for all help and co-ordination extended in bringing out this project successfully in time.

I will be failing in duty if I do not acknowledge with grateful thanks to the authors of the references and other literatures referred to in this project.

Last but not the Least, I am very much thankful to my parents who guided me in every steps which I took.

KANISHKA JOSE

ABSTRACT

Energy consumption is rising because of the change in life style of people. Managing energy of buildings and utilisation of renewable energy resources are very much important. Detecting the energy demand from the historical data set helps to manage the energy consumption pattern of a building. Hybridisation of two methods DWT and LSTM, transformer model and 1D convolutional neural network models are used here. Comparison of these four models is conducted in which the obtained results show that transformer model is efficient. DWT is used to get important characteristics from the non-stationary time series data given in the analysis. A long short term memory method (LSTM) used for the management of energy demand in buildings. Transformer model and 1D Convolutional network are also used for prediction of energy. This method enables monitoring and controlling of energy demand pattern of a building. Thus, this forecasting methods represents to predict building energy consumption.

CONTENTS

LIST OF FIGURES

ABBREVIATIONS

AI	Artificial Intelligence
DWT	Discrete Wavelet Transform
GRU	Gated Recurrent Unit
GWO	Gray Wolf Algorithm
IT	Information Technology
LSTM	Long Short Term Memory
PV	Photo Voltaic
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network

CHAPTER 1
INTRODUCTION

1.1 GENERAL BACKGROUND

With the growth of technologies people depends on more and more buildings for different purposes like accommodation, industrial purposes etc. But these buildings have different energy consumptions. The energy consumption in building is rising, so managing the energy consumption of each building is very much important. For managing the energy consumption of building depending on renewable energies helps in the total balance of the energy consumption of buildings. The use of deep learning methods like discrete wavelet transform and long short term memory method helps in the management of energy demand of buildings. This methods analyses the hourly energy demand of buildings. This method helps in the proper balance in the utilization of renewable energy resources and the share of extra energy to the grid . This method also helps in saving of energy that is storing of this energy helps in the proper use of that energy during peak hours . And depending more on renewable energy resources helps to consume the energy for future generations also.

The use of reliable and green energy resources helps in the improvement of building energy consumption. And it also enhances the quality of life. That is smart active buildings helps to minimize the energy consumption of buildings and also reduce depending more on to the grid thus reduces the green house gas emissions. The proper planning and energy management of smart buildings helps in the proper usage of renewable energy resources efficiently. And also the energy dependency of buildings are time dependent and vary seasonally or even hourly. For the increase in energy security , reducing cost and even for easy availability the use on renewable energy resources increased. For example the use of photovoltaic energy generation which is a renewable energy technology has increased its usage from the last decades. Considering from 2017 to 20222 there is a major increase in the use of solar panels. The people are depending more on this renewable resources because they understood the use and importance of renewable energy. They also understood that using this type of energies will also save energy for future generations .

But the implementation of renewable energy technologies are challenging. Because in the case of PV outputs that vary because of the variation in solar irradiance, temperature etc. And in the case of wind energy technologies that depends on the wind speed and direction. Here an artificial intelligence based methods are used to estimate hourly estimate of building energy demand. By using a hybridization of Long Short Term Memory (LSTM) neural network and Discrete Wavelet Decomposition (DWT) methods.

CHAPTER 2
LITERATURE SURVEY

Literature survey that shows the importance of predicting the energy consumption and energy production in buildings which also helps in improving the energy management system of smart buildings effectively. In[1], the forecasting criteria chosen are wind speed, solar irradiance and temperature. There are several methods most popular in building energy modelling . All these methods are used in the estimation of energy supply, energy demands etc. Although these methods are effective and accurate, these methods are complex to model and implement. To develop such models, the used details about buildings and environmental data are needed. The lack of these several details that leads to achieve low efficiency and accuracy in the modelling of building. Thus implementing these methods that need sufficient resources like data, different computational parameters etc. Thus for the proper implementation several machine learning methods can be used. These methods can handle large amount of data. Thus providing an accurate forecasting analysis. Thus this method can be used as a efficient technique in forecasting. These methods also have the potential to apply it in buildings and to find its energy consumption and energy efficiency of different buildings. Using artificial neural networks in the implementation of energy forecasting helps in the forecasting of building energy demand, wind and solar generation.

In[2], the forecasting criteria is based on the short term load forecasting of anomalous days. The methods that included are Ensemble feedforward, Elman feedforward and RBF neural network. These methods are machine learning methods. And these methods are helpful in getting high performances in the forecasting of supply and energy demands. Elman feedforward is a good machine learning technique as it proved having more efficiency and accuracy as compared to other machine learning techniques.

In[3] the forecasting of wind and solar generation are done. But here for the forecasting some methods are used. They are Grid Genetic algorithm searching algorithm. This method is less time consuming. This method is also suitable for short term forecasting of renewable generation.

In[4] the different forecasting methods are developed to predict the building energy consumption and generation . To forecast energy generation and consumption in a building artificial neural network can be implemented. Artificial neural network that consists of collected nodes or units called artificial neurons. Its name and structure is taken by getting inspired by the human brain. Having different synapses in biological brain, here each connections are like the synapses in a biological brain, that can transmit a signal to other neurons. An artificial neuron after receiving the signals, the each signal received gets processed and will give signals to other neurons connected to it. The each connections are called edges. The neurons and edges typically have weights. The signal strength that are adjusted by weights. The neurons are joined into layers. Different layers that perform different functions are based on their inputs.

In[5] hybridization of lower and upper bound estimation, particle swarm optimization and neural networks are used. These hybridized methods that have more accuracies . Because here two or more methods are being hybridized and used. Thus the quality of each technique that can be hybridized. And all these can be easily implemented for forecasting. Thus it will eliminate more limitations and these methods improve the efficiency and accuracy.

CHAPTER 3

METHODOLOGY

3.1 Load analysis in buildings

When properly executed, a load analysis will sounds into deep and proper insights into energy use. This will helps to reduce the energy costs, protect critical assets, increase productivity, reduce over consumption , maintain the overall efficiency and helps to save the energy for future. A load inventory can show the quantified estimates of different facilities like total load and overall load consumption. This will helps to take a huge role in choosing and building appropriate energy supply systems such as PV (Photovoltaic) panels, batteries etc. And also creating the proper energy indices helps in tracking and comparing the energy consumption pattern of that particular building. When taking or estimating the load of a particular equipment in a building, will give about the amount of electricity it requires to operate. This electricity which is given by the energy supply systems like grid, array of PV panels, or a generator etc. Therefore, from all these we can understand that a proper understanding about the equipment loads are very much important. This is very much important to size an energy system properly and correctly. Thus this load information is important in the proper working of all equipment in a building. But the first important step for predicting the load consumption, need the gathering of load information. Proper load information is very much important in the load analysis. Load information that can be captured from different sources. Some equipment have their own standard wattage values. The load values are needed to be estimated properly meanwhile hourly load values can be gathered. Taking the sum of all load values will give the maximum amount of power that need to operate all the equipment that are used in that particular building. This total load figures can be broken down into different load characteristics which will help one to get an overall idea about the total energy pattern of a building. And also about the total lighting load or even no contact load.

As it is already known that the energy consumption patterns of different buildings are different. Sometimes all the equipment in a building will be turned on at same time or sometimes different equipment will operate at different times in a day. Because of this reason creating a load profile will be very useful and important in knowing about the total energy consumption details of a building. The purpose of creating a load profile is that it sums the equipment load based on the time of day that the particular equipment start operating. For this different types of load profiles that can be used .Considering an hourly load profile, an hourly load profile gives about the total of all loads operating each hour of the day. This load profile will also give about the actual demand for electricity throughout that particular day. The load profile can be understood or calculated as low or high based on the total usage of energy at each hours. That is during the normal working or operating hours in a building the load profile will be higher. Because computer , household or other equipment and devices are being used. But during evening or during off hours, loads are reduced because the building is having less activity. Thus a load profile will shows the minimum and peak load. And more and more it also indicates the actual demand given on the energy supply system. The energy consumption is measured in watt hours. The total consumption of a building is determined by multiplying each load by the number of hours it operate. This same information is also used to create the total load profile.

Energy consumption will also indicates the total energy used in a particular given amount of time. For example this can be daily or annually energy consumption. Considering a building , different loads such as lighting, refrigerator, air conditioning units etc operate for different durations throughout the day. Some equipment operate more during the day time than the night and vice versa. Considering the time of use of all loads throughout the day is very much important for developing a healthy load profile. This can also helps in creating the strategies for different power supply units like photovoltaic systems , battery backup systems, generators etc. Energy consumption is also connected directly with the energy costs. Mostly electricity bills are in amount per kilowatt hour, a set of cost per unit of consumption.

An energy index shows the measure of energy intensity, it is calculated by dividing the buildings energy consumption by some other indicator of building performance. Buildings that consume considerable amount of energy.

But in residential buildings most part of the energy is used for heating, ventilation and cooling. For saving the energy resources for future generation , the efficient use of energy is needed . The relation between the release of carbondioxide to the atmosphere and the use of energy is another motive to render a good and efficient usage of energy. So , many types of researchers are being conducted to analyse the performance of building mainly to the residential sector. For building energy efficiency and performance topics are now widespread in all fields even to artificial intelligence field for more computational efficiency and for building an easier model. Residential sector that consume more than one third of electricity generated in world. Simulations based on energy performance helps in assessing the energy consumption in building and analysing the energy performance and all performance details of a building which depend on many factors.

The depending factors that include the HVAC systems, lighting properties, energy usage. The influence of each factors using a simulation tool or program based on an algorithm helps in providing the easier or better computational facilities and analysis in a building. These all significant amount of conducted theoretical statical or analytical experiment focussed on the subjects like saving energy, efficient usage of energy and building or improving energy performance of a building. The main objective is to achieve the energy prediction of a building. Proper energy prediction in a building also helps in saving the energy for future.

The building sector is one of the important sector that have the highest potential for energy efficiency. It is important to find the ways to reduce the load, increase the efficiency and to utilise the renewable energy resources. To optimize the building performance employing of energy modelling programs are also important in the design process. Using the sensors to control the loads based on occupancy and scheduling the availability of natural resources such as sunlight or natural ventilation is also a good step in building energy efficiency of buildings. Evaluating the performance with smart control that convert the building automation systems with information technology (IT) infrastructures. Thus converting a normal building into smart building. Using of proper metering in building to confirm the building energy is also very important.

Proper energy management tool using AI (Artificial intelligence) techniques helps in the proper analysis of energy usage in a building. Thus the use of deep learning techniques in a building helps in the analysis and construction process of building that achieve much better results in energy cost savings. Energy independence can only be achieved by minimising the energy consumption by generating energy using renewable resources. Energy independence and security are important components for achieving the national security and for adopting new strategies.

3.2 Use of Artificial intelligence in prediction

Today the application of Artificial intelligence technique increased worldwide. There are many applications using Artificial intelligence such as self- driven car ,speech recognition, algorithm design for various gaming application etc. During each decades, Artificial intelligence involved many changes and the application of AI also changed over decades. In , Artificial intelligence the first commercial application was considered in power industry, where Artificial Neural Network (AI) was used in forecasting and this started in the year 1990.

So, based on these Artificial intelligence techniques in load forecasting , the researches published more number of papers for the last three decades. The story of electricity began after the inventory of incandescent lamp in the year 1879. At the starting of this past years , the load forecasting was easy. And is much easier to built load profiles. But now increase in energy consumption increased. And formed a large data set for load forecasting. The first Air conditioner was built in 1902, which also increased the energy consumption pattern. Thus Air conditioners, lighting system brought more energy consumption of consumers.

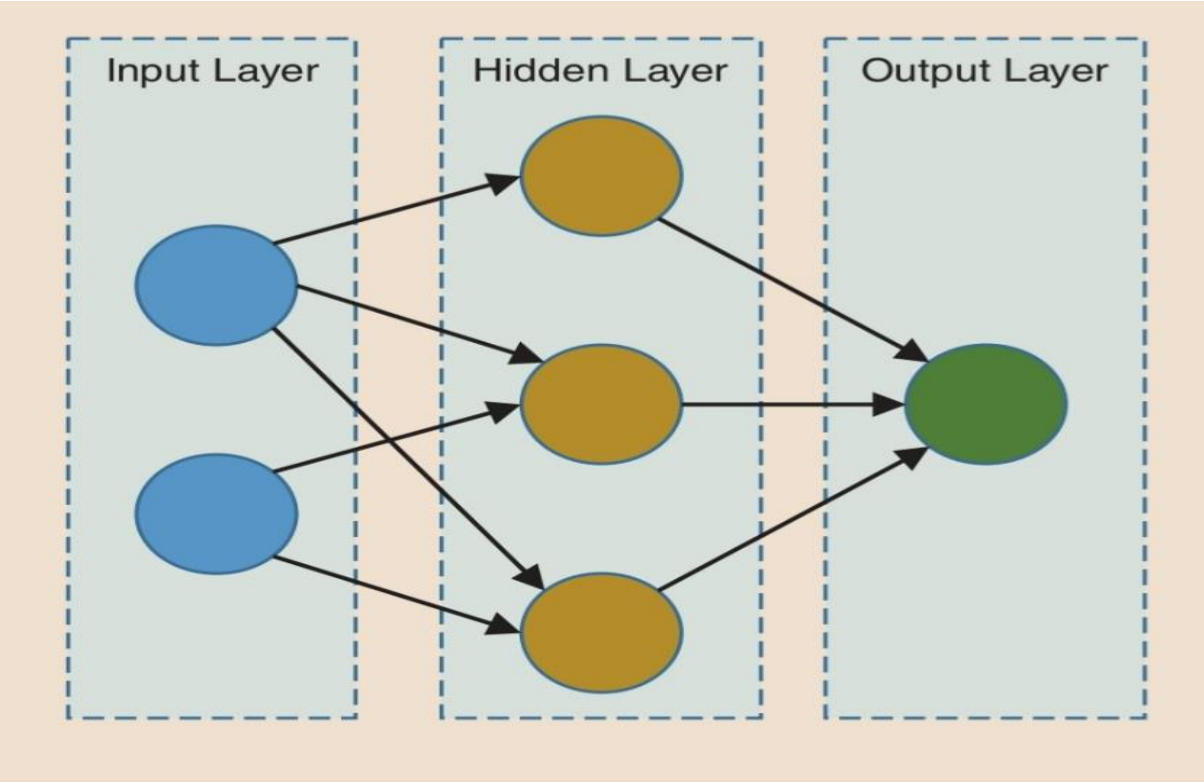

Fig 3.1 A single three layer, feed forward (ANN)

The implementation of biological neural network began in nineteenth century. These neural network have their structure similar to human brain. The collection of these neurons that formed the Artificial Neural Network . The above figure shows a feedforward Artificial neural network with a simple three layer structure.

This structure that has two neurons in the input layer, three hidden layer and an output layer. But each network that were developed using Artificial intelligence had its own draw back . So this provide challenges for existing networks and new networks was established to overcome these challenges of network . Thus in the year of 1970 , that the Artificial intelligence faced more challenges. But each of these challenges was overcame in the coming years. Thus researches found more application of Artificial intelligence during 1970, the AI became as an important topic in power engineering.

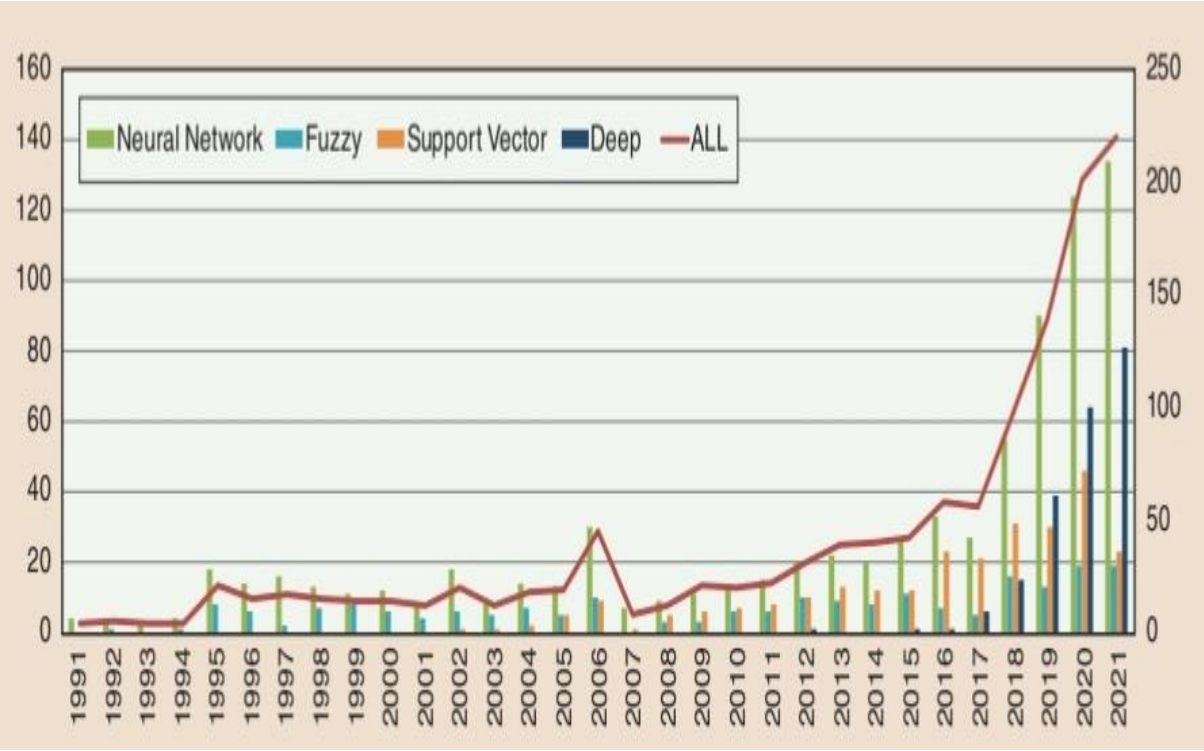

Fig 3.2: The journal papers in the title load forecasting using AI techniques

The first paper about Artifcial neural network for load forecasting was published in 1898. Then from there more and more papers that used AI techniques and machine learning concepts in forecasting. These AI topics include many subtopics such as deep learning , fuzzy systems, support vectors etc. Fuzzy systems are based on the concept of fuzzy logics that included more and more true and false statements and logical statements. Deep learning is based on Artificial neural network and these deep meaning is a network having many hidden layers. The figure 3.2 shows an increase in the number of papers using Artificial intelligence in load forecasting.

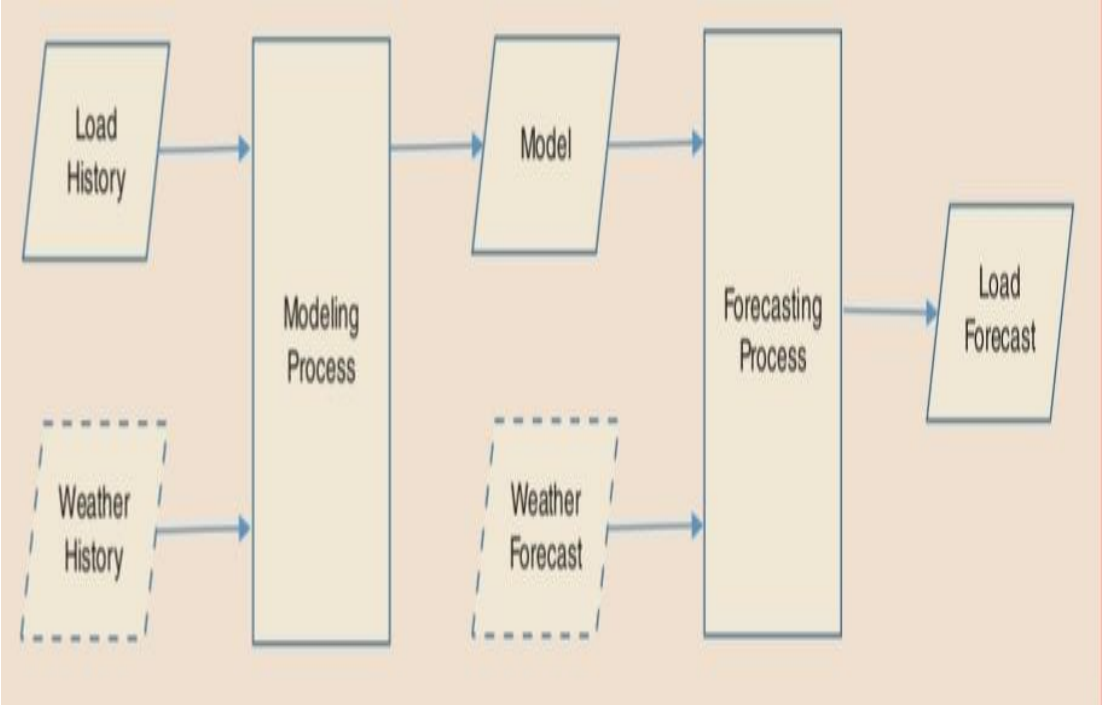

Fig 3.3: A load forecasting process

The load forecasting became successful using Artifical intelligence in 1990. But after that during the later years the load forecasting that shows much progress in terms of innovations, efficiency and accuracy. The figure 3.3 represents a load forecasting process. The first step in load forecasting is to collect historical data. Then add a pre processing step to clear the load and weather history. Or there is also an option to add this pre processing step at the end of all these steps to fine tune that load forecast .

Then the data are sent to a modelling to process. The modelling process include various techniques for load forecasting. The historical data is divided into three periods. First for training. The goodness of fit in this period is called as in sample fit. Accurate model results in good in sample fit. Due to some noises sometimes the sample may overfit. This issue is called overfitting issue.

Then second period which is being set for model selection or validation, this is for to avoid the issue of overfitting. Then from these data sets, a third period which can be used for an out of sample set. The researches will do training and validation steps more number of times on different periods , this is known as cross validation. The load forecasting is also known as time series forecasting and it involve these steps of training and validation.

Artificial neural network and regression analysis are techniques in Artificial intelligence. But they are connected in many ways. By back propagation the Artificial neural network parameters are propagated , while the parameter of regression models are collected by minimizing the sum of squared errors. Many algorithms that are using for different process are nature inspired algorithms. For example , genetic algorithm was developed and it was inspired from the process of natural selection, particle swarm optimization which is being obtained from the motion of bird flocks and schooling fish , then Gray wolf algorithm also known as GWO which is obtained by the behaviour of Gray wolves for hunting. Thus developing of different modelling process lead to different forecasting techniques.

So , from the developed algorithms or techniques some of the algorithms that are developed and used for the process of load forecasting. Thus these forecasting techniques helps in an efficient way to estimate or forecast the load data set. Thus development of algorithms are useful in various fields.

Day to day the research on these field using Artificial intelligence techniques are increasing, thus it also helps to increase or to improve the accuracy in various fields. So not only a single type of ANN, there are different types such as feedforward, deep Artificial neural network and recurrent network. Now there are many availabilities following up with the load forecasting. Thus the development of AI brought huge change in the world . From single layer perceptron to deep learning approaches, huge process has been made in load forecasting . Thus there are many benefits in load forecasting using AI, it also helps to provide power now and to the next generation without any interruptions. It also improve the quality and safety by minimising human errors. Thus it solves different problems in power systems with proper scheduling, calculations and accurate forecasting. Thus it also improves the efficiency, reliability, and sustainability in power production and distribution. Thus it provide a smart power consumption in building thus changing the buildings to smart buildings.

3.3 Load Prediction

The load prediction minimises the utility risk and predict the future consumption of commodities. The prediction or forecasting technique uses customer load data with time series load profiles. Thus this load prediction helps in power system planning , control and operations. The estimation of both demand and energy requirements are very important for effective system planning. The term forecast determined using a systematic process and defining the future load thus improving the efficiency in a building or a system. The prediction that works with historical data, extrapolating the past load data growth patterns into future. The use of renewable energies can also helps in the energy management and saves the energy for future. The emission of carbon and use of fossil fuel can also be reduced using this renewable energy resources and the maximum utilization of power plants. In load prediction, the independent variable is weather. The weather patterns that change seasonally. The consumer behaviour is always affected by weather conditions.

There will be change in energy usage patterns during the cold and hot seasons. That is energy pattern will depend on seasons and usage of electricity on that particular seasons. Temperature, dew factors, and humidity are the different weather factors. These patterns will affect the energy pattern of load profiles.

The detection of load by time frame is also very important. Time is an important key factor in load prediction. The time frame helps in the accurate prediction of current data and future days data. Load forecasting is one of the important process in electricity industries. It has many application infrastructure development , energy correction etc. Time series analysis is based on tuned or seasonal variation. Thus this time series is based on the tuned or seasonal variation. Thus this time series analysis is also used in load forecasting. This prediction technique that involve data mining process, in which the collected volume of data considering for forecasting is gathered and analysed as a first step of prediction process.

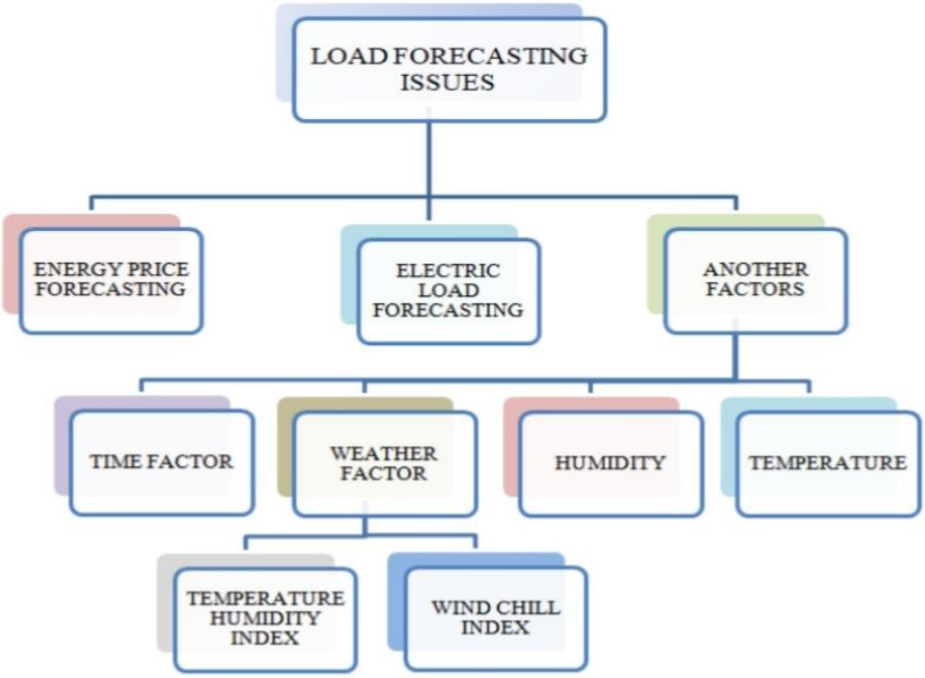

Fig 3.4 : Load forecasting issues.

Thus the load forecasting can be done through different methods. In this for the analysis and prediction in electrical loads regression analysis can be used. Forecasting model cannot be explained in a single step as it involves number of steps and it is a long process where it is started with a model and then by more iterations and reflections get closer to get the best fit model. The prediction or predicting the demand for electricity help the consumer to get reliable supply of power now and also to get it for future ahead.

Load prediction helps to make an analysis and to take decisions on how to operate a bulk electric power system and to do some improvements and developments when needed. The main goal is to provide the proper supply with good services only at reliable costs. Some patterns like collecting load data and predicting future data is possible. Thus the power usage on industry or a building can be balanced.

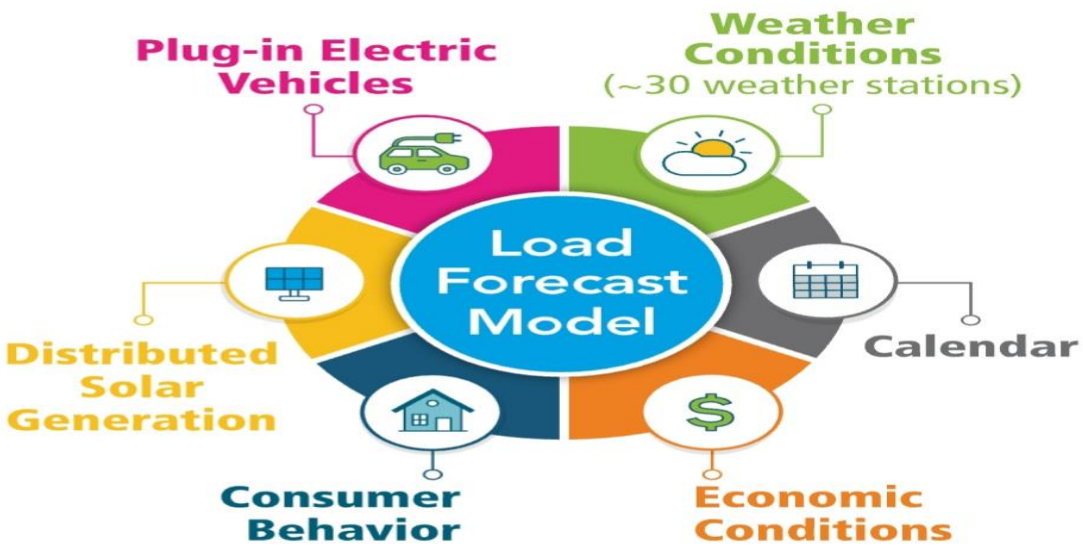

Fig 3.5: Load Forecasting model

By the usage of solar panels and the other types of generation the consumers can reduce the amount of electricity. Sometimes charging of an electric vehicle also require more electricity, equivalent to the power needed for the entire home consumption. Thus the forecasting is complex in the sense that it involves different steps like data collection and using appropriate algorithm models or technique for proper forecasting results. There are different issues in load forecasting which can be time, weather etc. These all issues that can be reduced by incorporating the Artificial Intelligence techniques to the collected data and then processing. Thus the processed output, formed will give an accurate forecasting result.

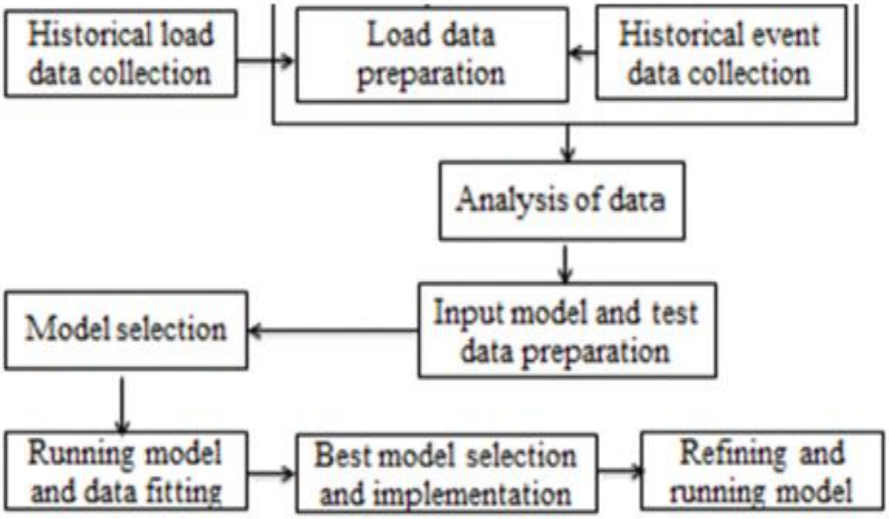

Fig 3.6: Load Forecasting process

3.4 LSTM

LSTM model is also called as long short term memory. It comes under the class of recurrent neural network. An artificial neural network consist of a structure in which each layer is adjusted in such a way that all the neurons are connected. Each neurons in RNN that have a particular cell state or a memory in which according to the internal state the corresponding inputs are working and this is achieved with loops with in that neural network. In RNN, the tan h layers, help to hold the information. But cannot retain for long time, that is why LSTM models are being used.

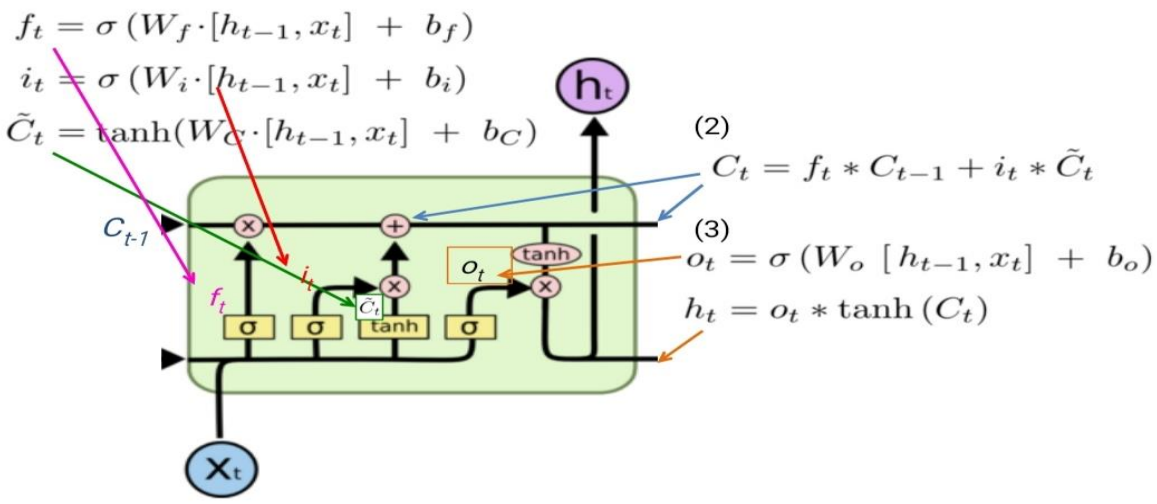

Fig 3.7: Architecture of ANN

LSTM is also a kind of RNN. LSTM modules consists of cell states, three gates that helps to understand , remove data and hold the data from each of the units. This model is capable of understanding long term dependencies. Cell state helps in the transfer of data through different units, without any altering and allow only some linear connections.

Each unit that has a forget gate, input and output. The forget gate decides which information to be used, that is it decides which data to be forgotten from previous cell states. The input gate that control the flow of information to the current cell state using tanh and sigmoid function respectively. Output state decides which data needed to be passed to the next hidden state.

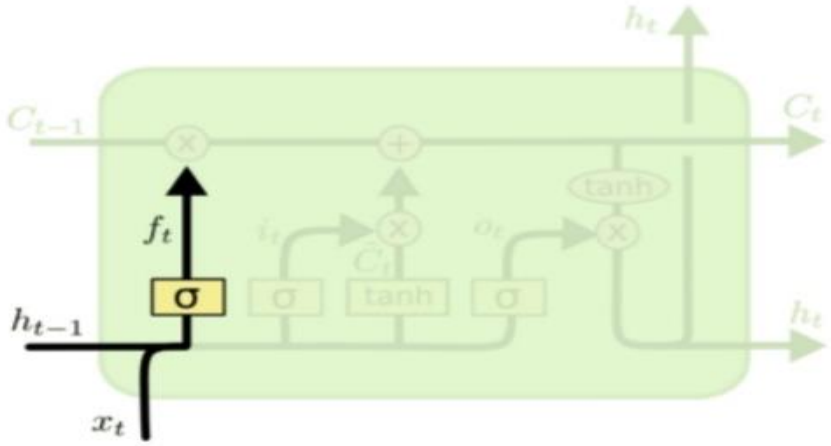

Fig 3.8: Forget gate

LSTM that has memory cells, that are used to remember that information . It also has three gates input gate, forget gate, and output gate which control the information in and output of the memory cells. The term Long Short Term which can be analysed from day to day life of humans. That is human have the ability to remember memories from distant past.

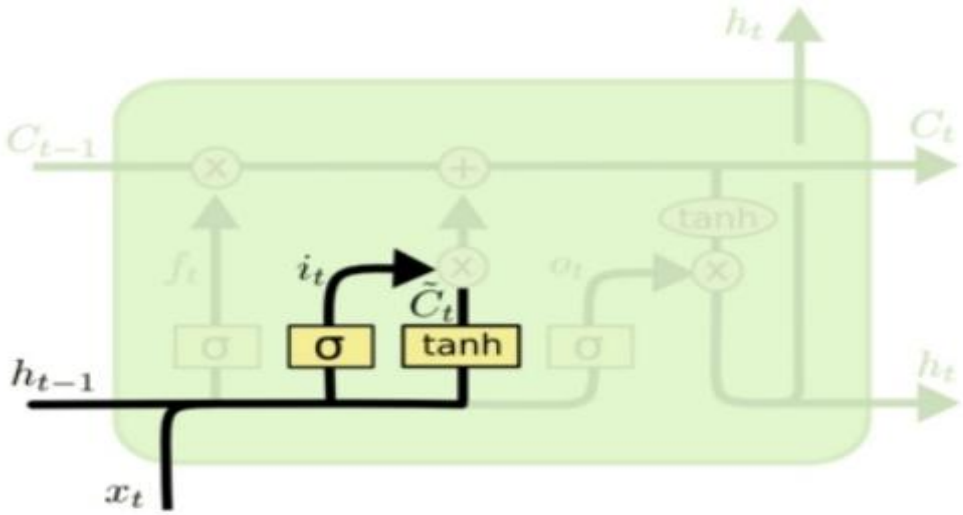

Fig 3.9: Input gate

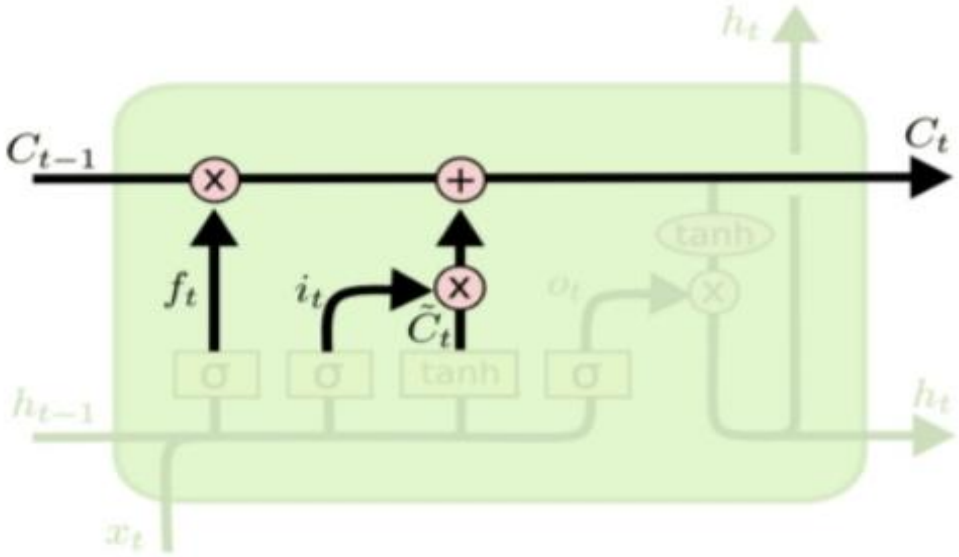

Fig 3.10 Cell State

Not only from the distant past , they can also remember the recent events and can also remember about the events in a sequence order that happened in the life of human . Not only in predictions but the LSTM that also have different advantages and that the LSTM can also be used for image captioning , hand writing recognition , machine translation etc.

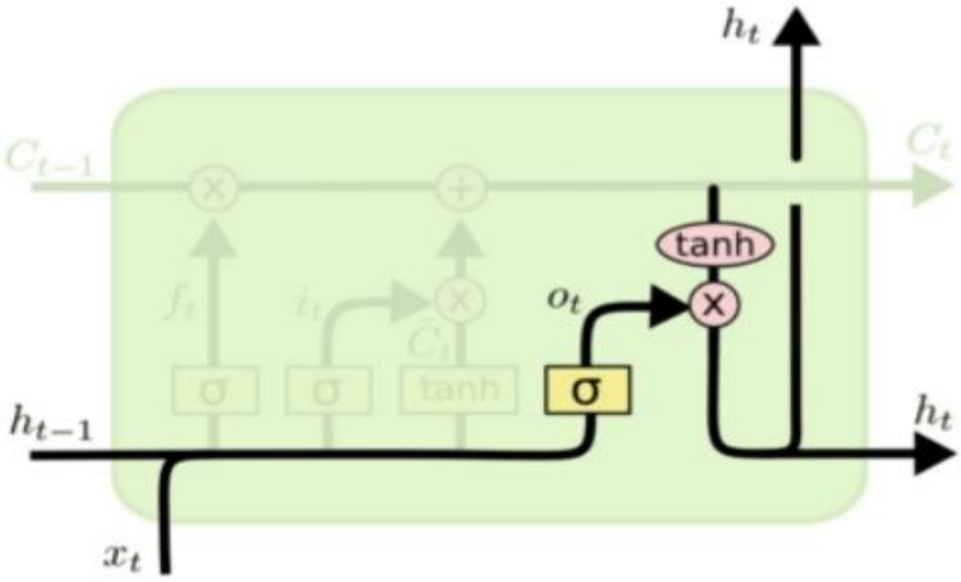

Fig 3.11 Output gate

At earlier times at first before the modelling of LSTM , several other neural networks are being used. But in the case of neural network , they process the information in a feed forward format. In this feedforward way input at one step which I used to produce output at other step. But with the modelling of LSTM , this condition is changed.

Compared to the traditional neural network LSTM is having more advantages than traditional neural network. LSTM that learn not from a single step, but from the sequences of data. They process the information in a recurrent way. They can take the input at one time step and process their output at future time steps.

Cell state is a vector representation , it contain the memory of an LSTM network. It also contain information from the previous and current time step series. Traditional RNN that are having limitations like vanishing gradient problem. LSTM that allow long term dependencies of data. It also have a GRU (Gated Recurrent units). The GRU that contains the forget gate. This helps to selectively forget information or data . And to update all the information in the present step. So, the information are controlled in the current state and it is passed to the next step in an efficient and sequence manner. LSTM can be trained by backpropagation method. Not only by backpropagation method but also by time and reinforcement method.

Advantages of LSTM:

1.Long term dependencies.
2.Remember information for extended or long period of time.
3. Less affected by vanishing gradient problem.
4.They are efficient in working with complex sequential data.

Despite this they are also affected by some disadvantages like requiring data to work properly. They are highly complicated. Not useful for prediction or classification when the obtained data are not in a particular sequence. LSTM will not work with non linear data. That is when the data have a lot of noises. Thus it is not a proper modelling for all type of data sets. Thus LSTM is suitable for a sequence of a data. Bidirectional LSTM are capable of capturing information from both past and future.

Not only in a single direction bidirectional is capable of acquiring information in both the directions.

3.5 DWT

DWT is called as Discrete Wavelet Transform. A DWT is used to decompose a given set of data into number of set . In which each set that are the time series of coefficient which will describe about some particular time evolution in signal corresponding to that particular frequencies in that time evolution. DWT that have the function to reduce the noises from the data sequence. Not only this function it also can be used for signal and image processing. DWT which is used to extract features from time series data and it is used to construct a classification mode.

Single level 2 D- DWT of an image, that will display the vertical level coefficient and approximate coefficients. Using a single level 2 D-DWT using filters that gives an horizontal, vertical, diagonal and approximation of that image. That means somewhat a clear picture is obtained. Thus the noises or unwanted contents are reduced and a much more clear picture which is obtained. Thus using DWT with LSTM in time series data for prediction helps in reducing much more noises and helps to get an accurate picture. Thus the accuracy and efficiency of load data used in prediction that can be increased. Thus instead of using LSTM , LSTM -DWT helps in an accurate prediction of model.

Thus the proposed DWT helps in the compression , denoising , segmentation , and classification of data. Therefore this DWT, when used with LSTM helps in the proper classification of data with denoising and provide compressed data graph or plot with less noises.

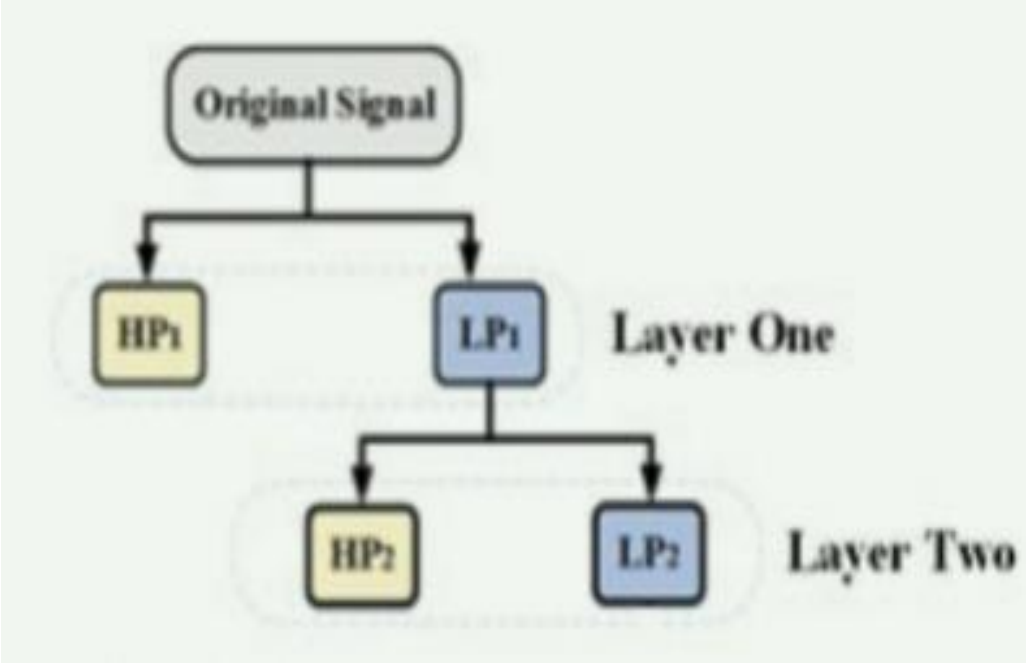

Fig 3.12 : Two layer DWT structure.

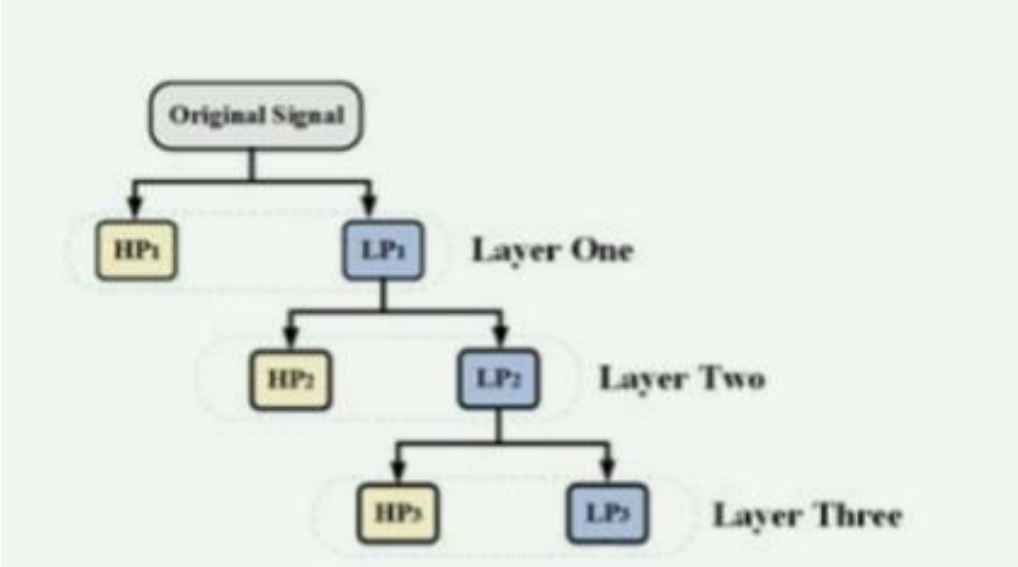

Fig 3.13: Three layer DWT structure.

In this process of DWT, consider a sample signal x, which is decomposed using low pass filter of impulse represented as (g) in figure and high pass filter of impulse response represented as (h) in figure. The block diagram having a filter is given in figure.

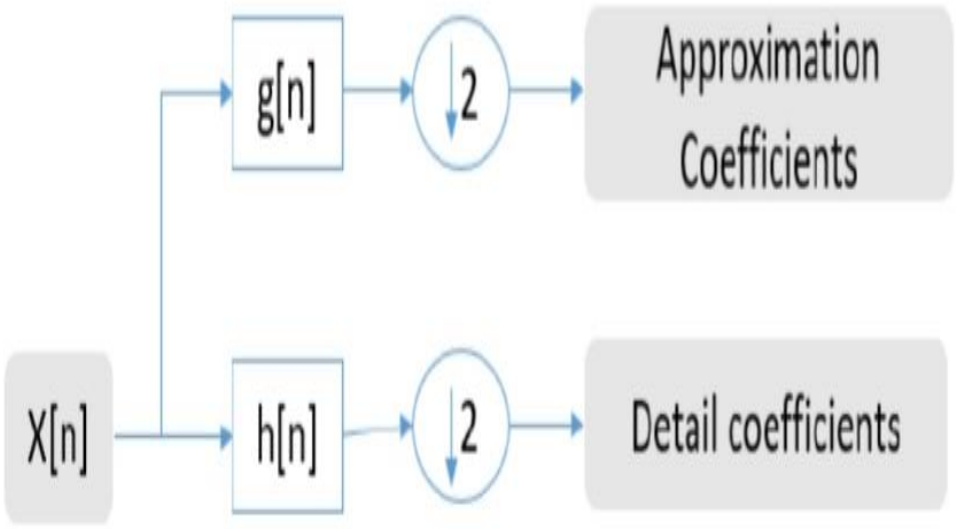

Fig 3.14: Filter analysis block diagram

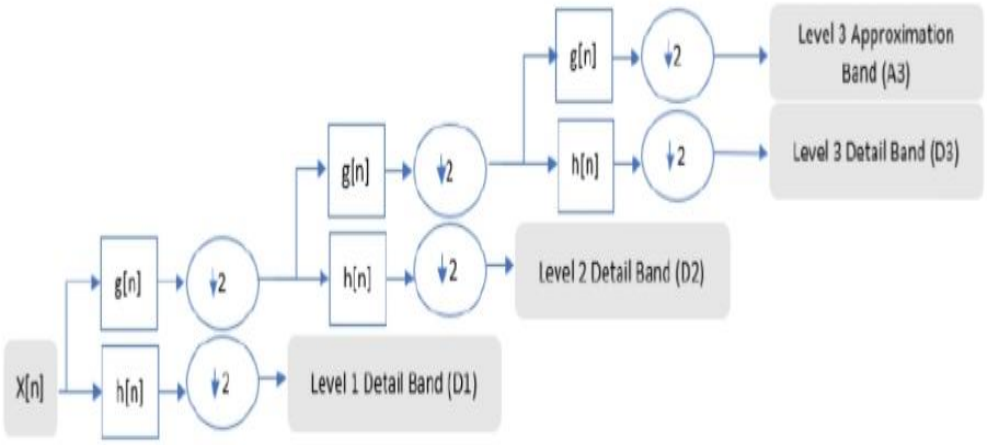

Fig 3.15: Three level decomposition of DWT

After conducting the process of filter half of the frequencies are removed. Thus in this process of sampling half of unwanted frequencies are removed. This process is known as one level decomposition. Again conducting these process in three steps having high pass and low pass filters, in which this decomposition will give highly desired frequency spectrum of the input signal.

3.6 Transformer Network

As the LSTM model that represents data in a sequential order and it depend on the recurrence of data set. But compared to LSTM , this transformer model not need a recurrence of data set. Only the need in transformer model is that position of tokens of sequence of data are needed to be added. The feature of transformer model is that an encoder decoder architecture is provided which will use an attention mechanism. And thus avoiding the recurrence as in LSTM model.

Consider an example, if a base transformer has word embedding of 500 dimensions (it also mean as 500 elements), then the positional encoding also has 500 elements or dimensions. Then step that comes under this model is the summing up of word embedding and positional encoding vector by each element wise. Here in this model the key feature is attention. Thus transformer network is more efficient than LSTM in data or load prediction of time series data. LSTM also suffer from a defect or disadvantage that, it has a short term memory over long sequences. Transformer network that outperforms the previous LSTM load prediction in the same data and will also provide good accuracy in load forecasting with less errors.

In LSTM there is a hidden layer, this layer will be updating with every single input data at each input units. Theoretically important information will also need to propagate into long sequences before giving the corresponding output. As this process is complex and takes more time to work properly. And also due to the vanishing gradient problem the LSTM have the tendency to forget the previous information.

But the transformer model retain direct connections to all previous steps without any hidden state between the units. Here this model has the advantage of direct connection with each units. Transformer also uses a feature called self attention which will helps to filter the unimportant information from important information. As mentioned above the summing up of word embedding and positional encoding is called concatenation. The characteristics of this process is that it is not necessary to have two vectors of equal dimension. Thus this process also help to take only less processing time at the time of training using this model. As in the program code for transformer model pos represents position and i represents dimension in program.

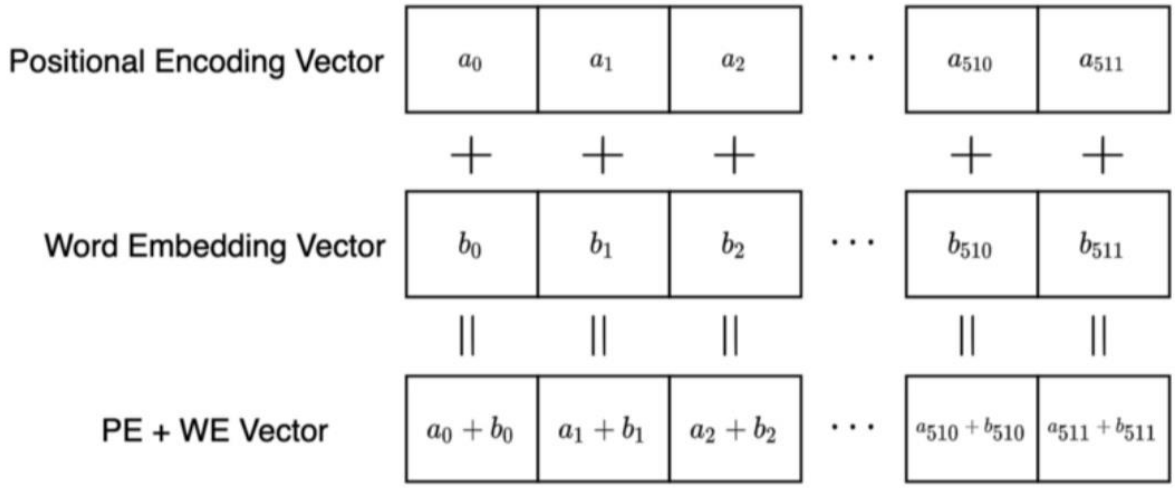

Fig 3.16 : Transformer model position encoding and word embedding

As mentioned above transformer model if represents 500 dimensions for positional encoding or word embedding. Then it can be represented as a vector of 500 dimensions having elements from 0 to 499. Thus this transformer model is a deep forward artificial neural network. Having an advantage of self attention mechanism. Thus it helps in modelling sequential data. This model also helps in sequence to sequence prediction. There are different process that takes place in transformer modelling. One such process is chunking. In this process when trained data exceeded the limit that the transformer model can handle accurately, then that information is divided into small chunks. Thus capable in efficient training of load data. The positional encoding helps in transformer model to make a capability of word ordering. In this way the transformer model helps to generate features and to solve the sequential problem in the transformer from input to output side.

3.7 1 D Convolutional Neural Network

The 1D Convolutional neural network consists of convolutional filters followed by pooling operation. Mostly convolutional network that consists of multiple convolutional stages, and it have a feed forward network to perform the feature extraction and to predict the corresponding output needed. The function of 1 D Convolutional filters is to extract high level features. The features that include edges and curves if considering an image. The extraction of high level features are from the supplied input by using a set of weights and applying non linear attenuation function. The output of this network is given to a pooling operation which will reduce the size of features extracted from filters. Thus this network will learn about the problems and will helps in the processing of features at each stages of convolutional network.

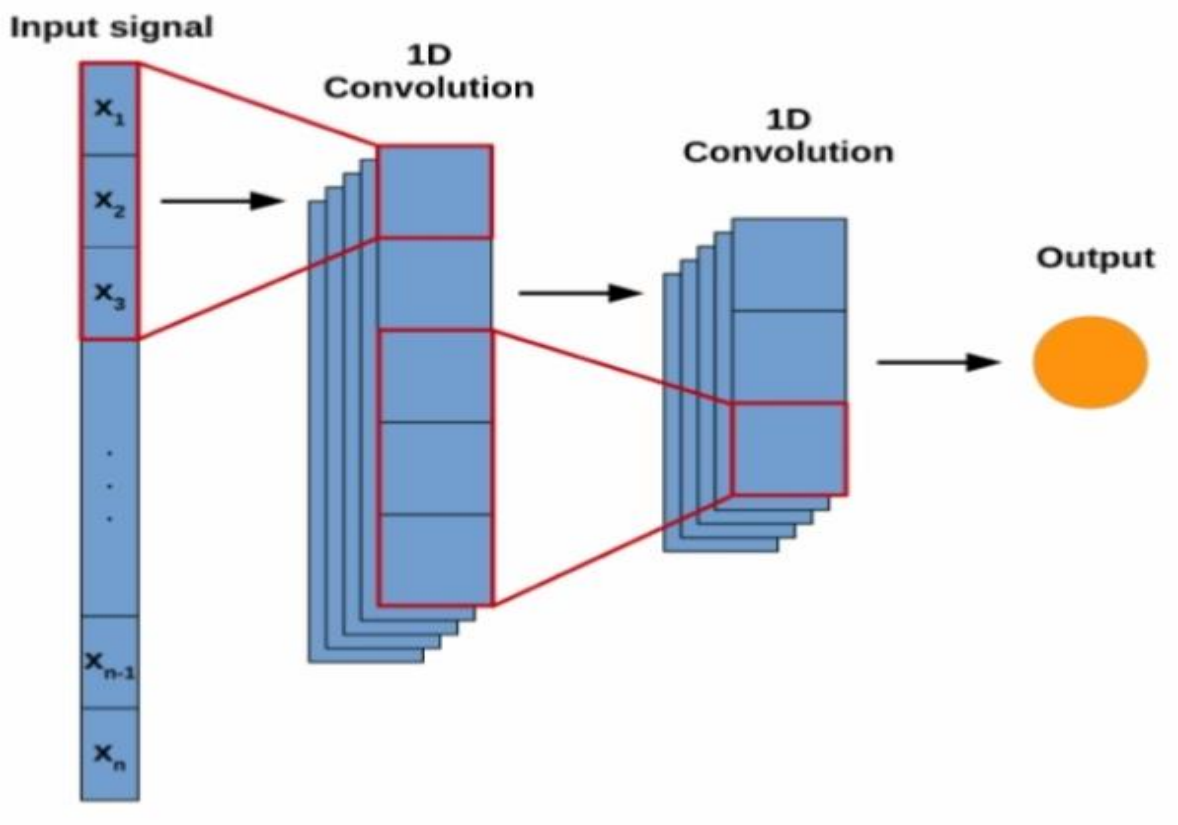

Fig 3.17: 1D Convolutional Network

CHAPTER 4

WORKING

4.1 Working of data using python

LSTM: Consider a set of data which is being gathered and split them for the process of testing and training. Now convert the time series data into series of data according to the supervised learning data form. This collected data under the process in LSTM will have a back look period. The back look period will have some lag during the process which is to predict the value at time t. For example consider a series of data as time variable x, then the corresponding series will be in the following order.

T1 T2…….T

X1 X2……XT

In LSTM , there is a process of lookback ,and when the lookback period is equal to 1, then the corresponding series, described above will changes to

X1,X2…….XT-1

X2 X3…….XT

For all these process the corresponding program is needed to in LSTM , having the condition with lookback. Considering with the program consisting for data for testing and training. Now as the next step train the data in the model. So, under the process of training in this data set the error is calculated at each iteration or steps. Thus these corresponding epochs are needed to be trained or needed to run till the time at which the error is reducing.

DWT: In DWT the program code that consists of dwt with corresponding filers at n levels. This is used as a function to compute discrete wavelet transform coefficients. Consider the term X , this is a time series which can be univariate or multivariate series. Filter that used in DWT to filter out the noises. The n levels in program that shows the corresponding levels of decomposition. The program code also uses wavelet coefficients in each processing of DWT layers. If the inverse of this DWT layer is needed then idwt() is used in the program code. Thus the whole program code built with corresponding functions will create the appropriate result of that time series data. To work with the signal effects that are being produced by the convolutional based algorithm 1 D and 2 D are used.

Transformer network : At transformer network a self attention and a position encoding is provided in the program code. Here the program runs by taking the data directly from the input encoder. No separate functions are needed for encoding. Thus this network using artificial intelligence is more efficient compared to other methods. Thus all these techniques that helps to analysed the data and predicted the outcome.

1 D Convolutional network: For a computer the data set that can be any set of data. There can be any set of data like numerical, categorical or ordinal. The numerical data can be discrete data or continuous data. The continuous data can be numbers of infinite value. The discrete data are numbers limited to integers. In 1 D Convolutional network also the program code which is being used to extract the high level features.

CHAPTER 5

RESULTS

The whole programming is done in the python using the program code. At first for a simple testing and training process, at first here collected 2019 load data and it is being trained and tested. Here its corresponding root mean square error is calculated and its training efficiency is plotted in a graph. The actual vs predicted result using LSTM is also plotted. Both these results are given in the below diagrams.

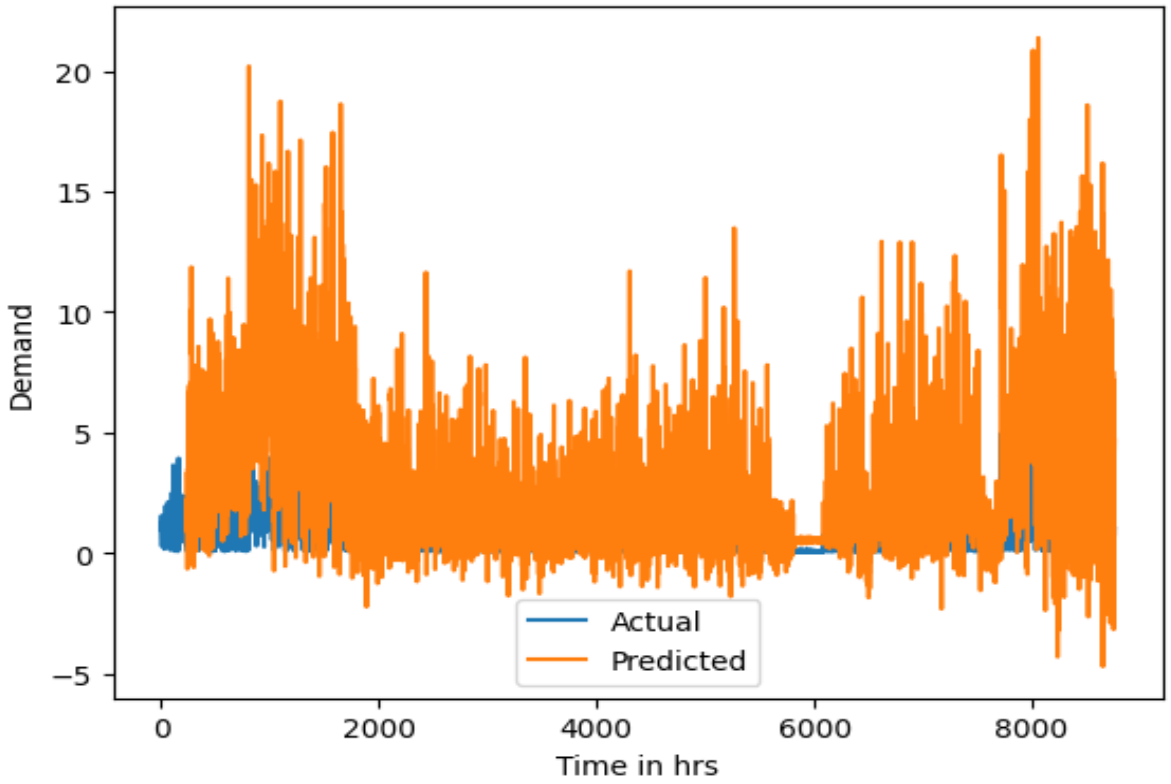

Fig 5. 1: Actual vs predicted plot

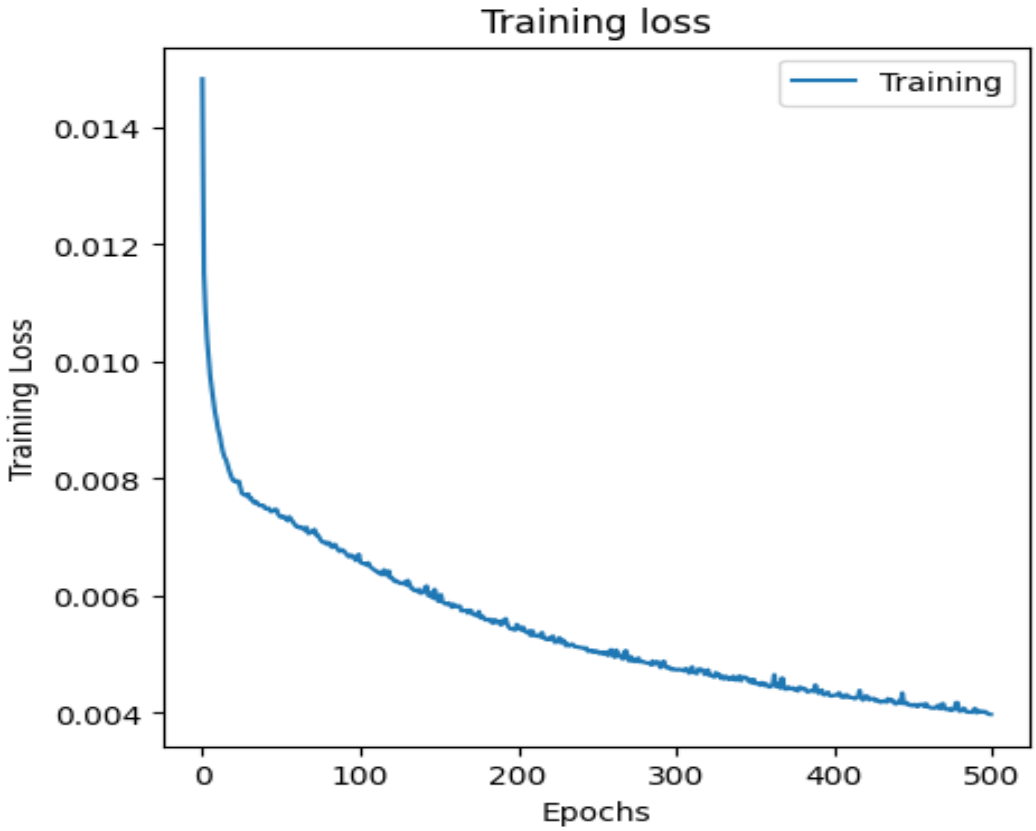

Fig 5.2 Training loss plot

By performing this using LSTM , noticed that the data collected having some noises . Then to remove that noises from the obtained result ,then the hybridisation of two techniques was used . That is then the testing and training is done by using a hybridisation of LSTM and DWT . The below figures will represent the training loss and plot using two layer and three layer dwt and its corresponding noise graphs. Here also the corresponding root mean score error is calculated during the process of training and testing.

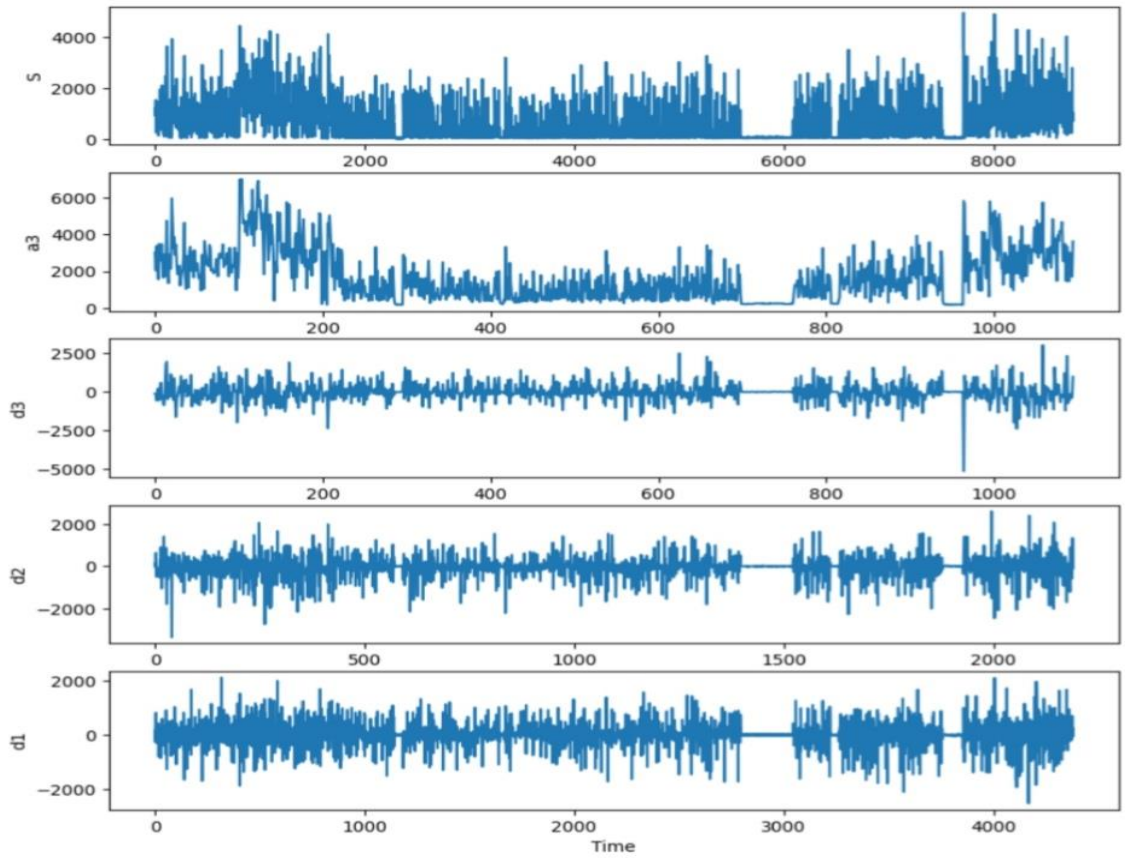

Fig 5.3: plot for three level dwt decomposition.

The next figure that will show the plot of two level dwt decomposition. From both these plots the a2 in two level decomposition has higher level of short term fluctuations compared to a3 in the first lot. Thus it is clear from these plots that increasing the layers will helps in extracting the more clear long term patterns.

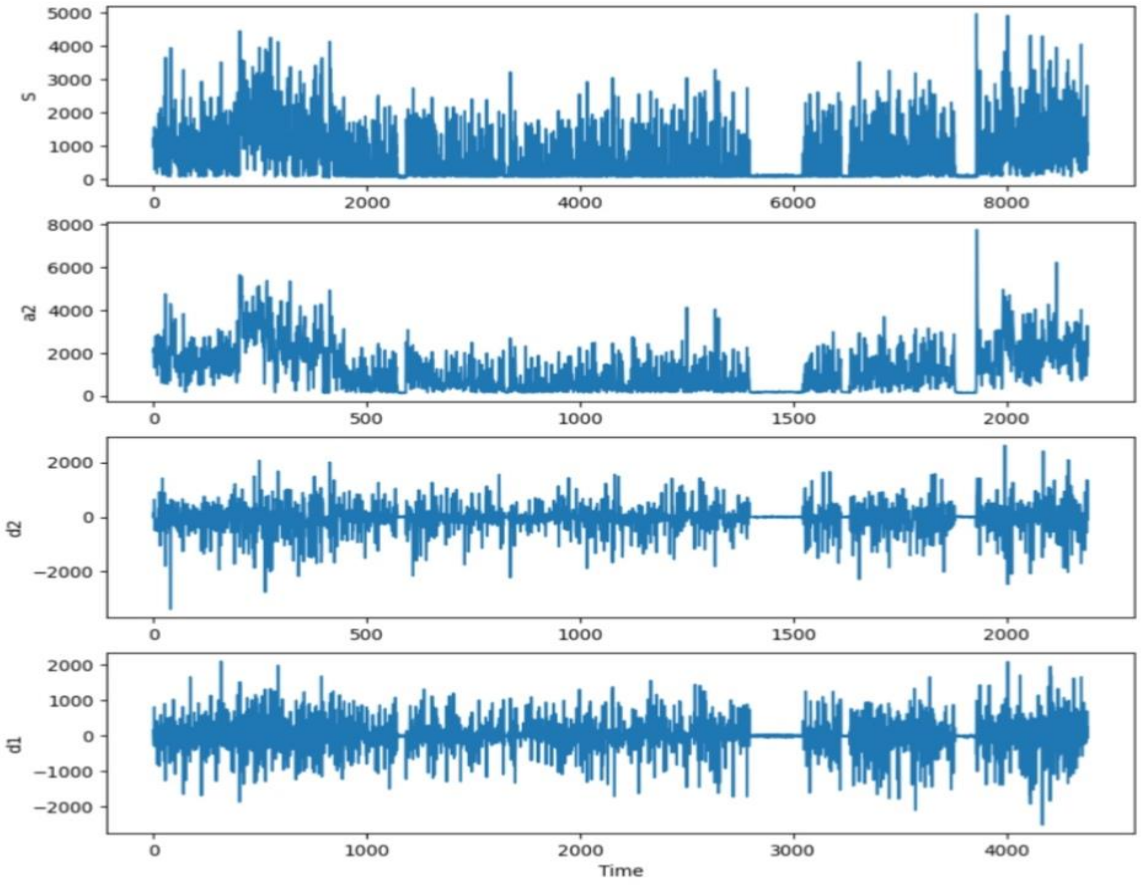

Fig 5.4: Plot for two level dwt decomposition.

The next figure plotted below that compares the two layer and three layer and denoising pattens of a building based on its energy level and the corresponding obtained noise level is also plotted below. This combined plot that shows a picture in which all the corresponding noise levels are removed and its hourly prediction can be analysed easily.

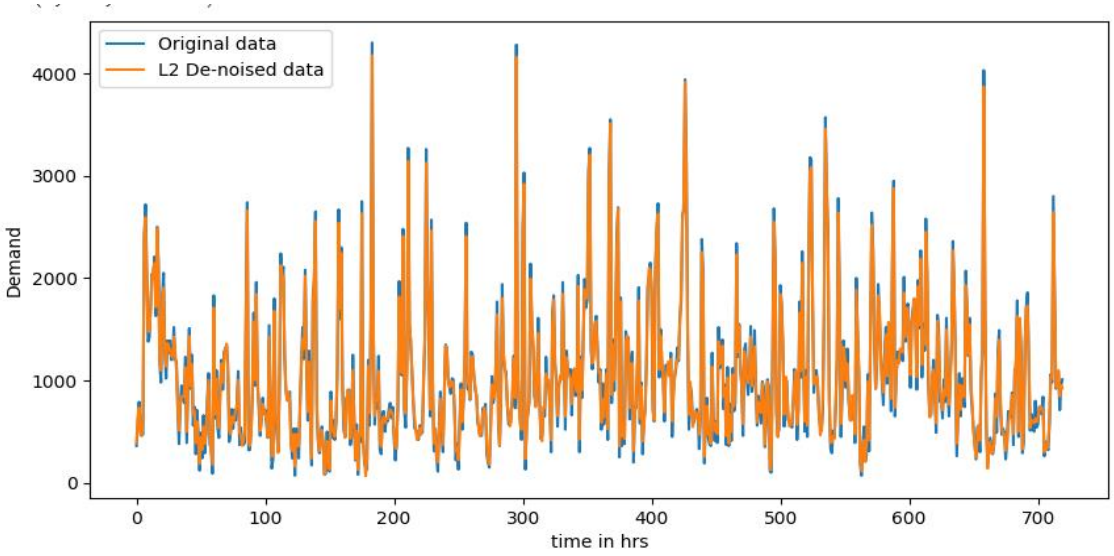

Fig 5.5: Three layer denoised energy demand of a building

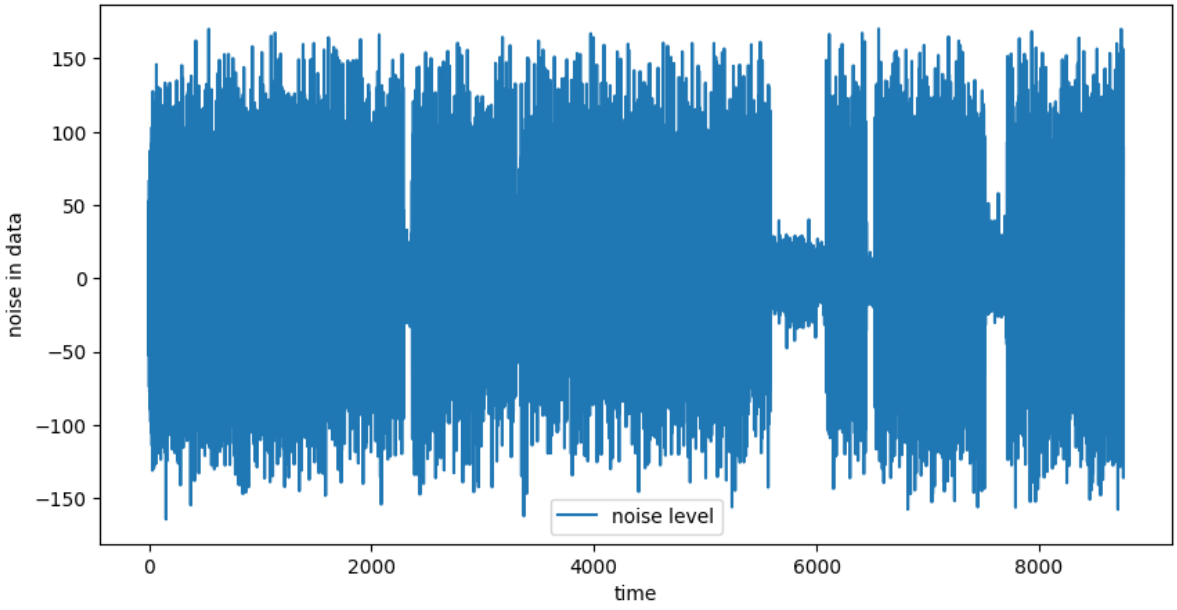

Fig 5.6: Noise data

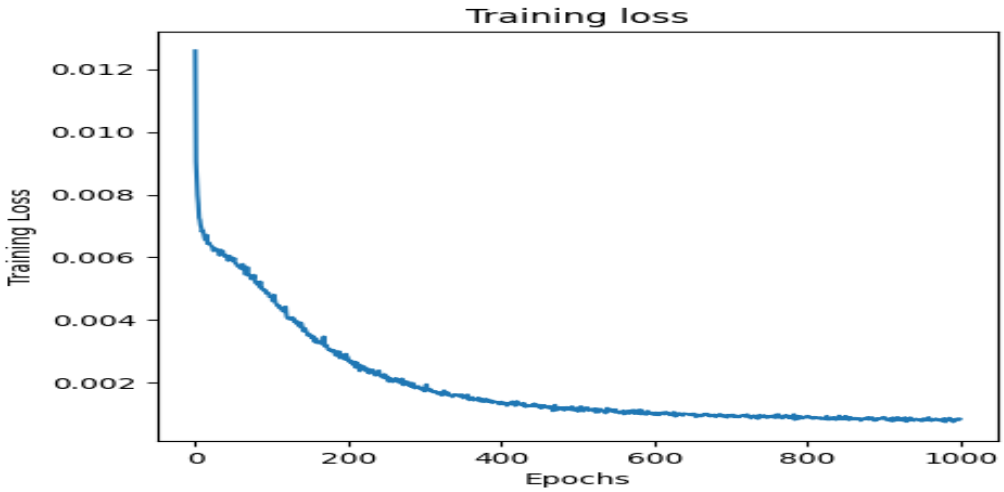

Fig 5.7: Training loss

The above figure that shows the training loss obtained during the hybridisation of LSTM-DWT.

Then next in this project a comparison is made between the different techniques like LSTM- DWT, Transformer algorithm and 1D Convolutional network. For this considered the data from 2015 to 2018, with 24564 hourly data are being used for prediction . Then its corresponding actual vs predicted graph is plotted . And its training loss and validation graph is also considered . Then the corresponding RMSE is calculated . And that RMSE value is compared to find which one works efficiently . And here obtained that transformer algorithm works efficiently.

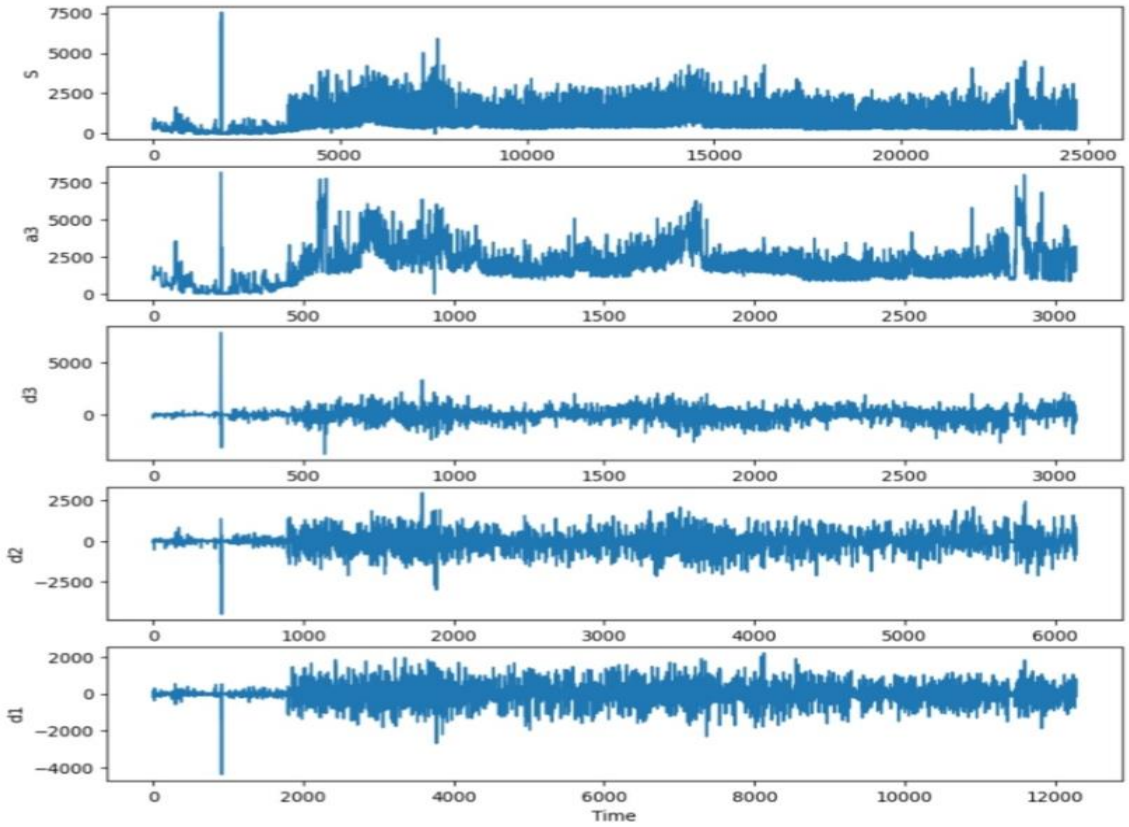

Fig 5.8: Plot of three level dwt decomposition

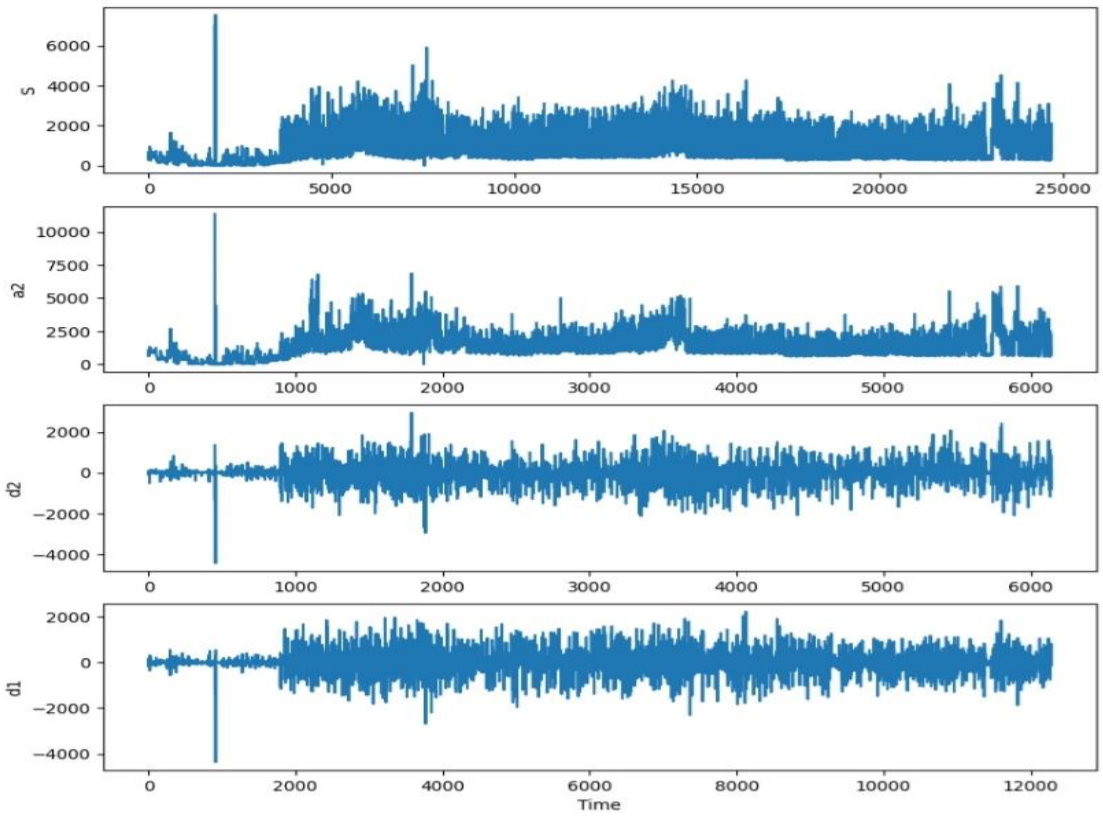

Fig 5.9: Plot of two level dwt decompostion

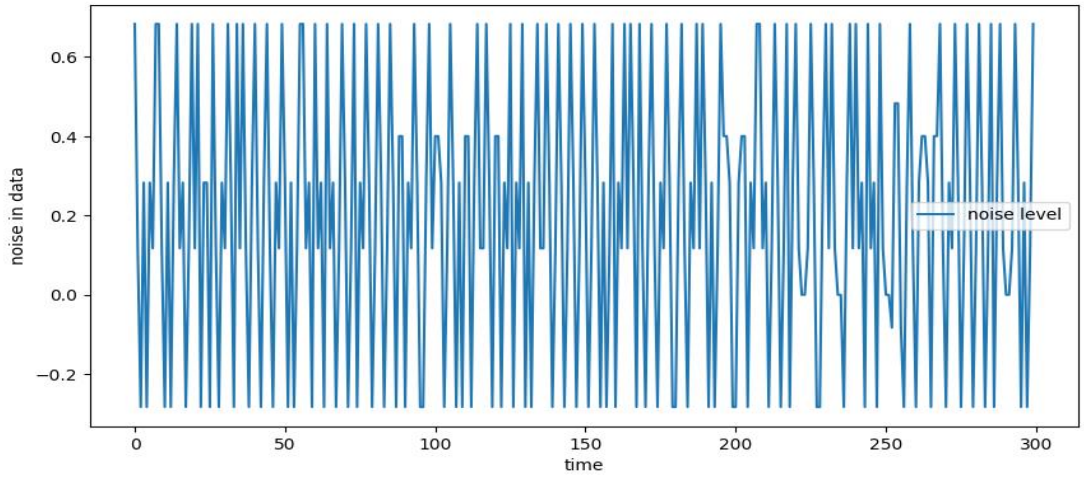

Fig 5.10: Noise level plot

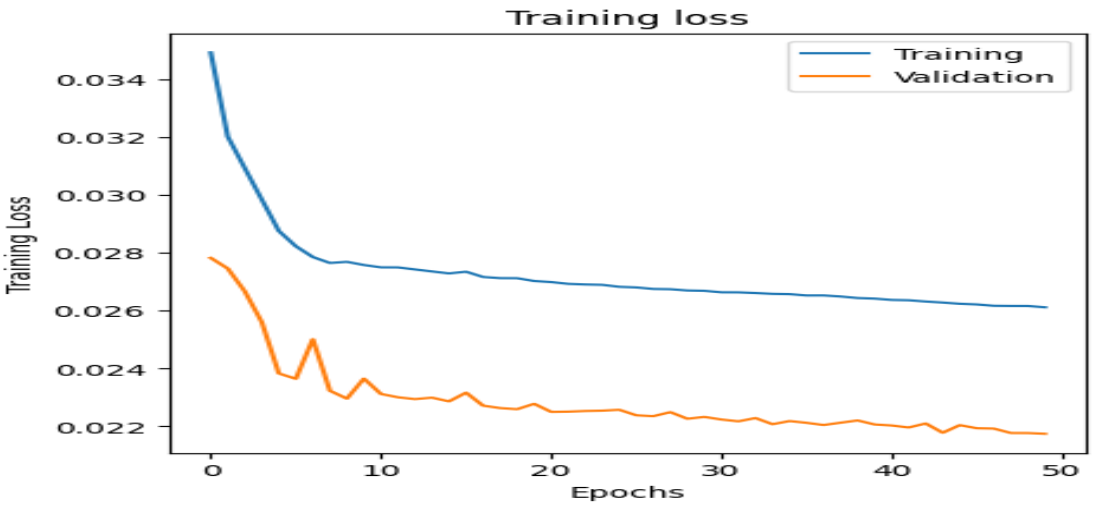

Fig 5.11: Training loss and validation plot

The above figure shows the training and validation plot. Here after a particular value the validation graph shows a saturation level which shows that the LSTM – DWT training is good in prediction of load in buildings . And the below graph that shows its actual vs predicted plot.

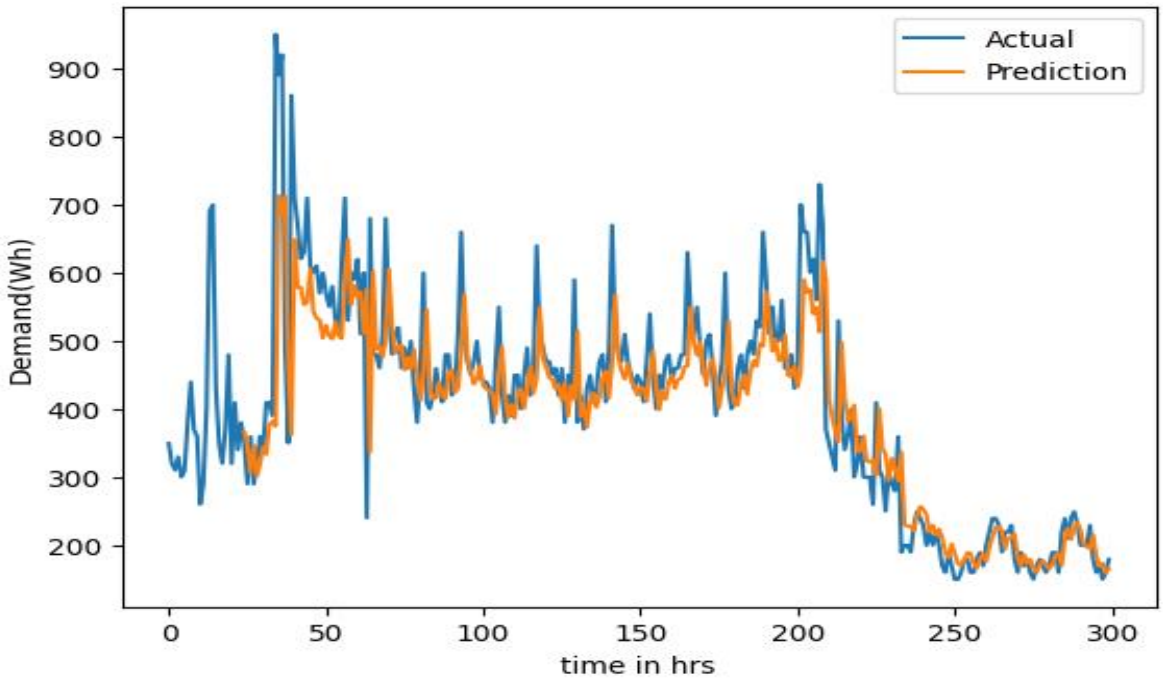

Fig 5.12: Actual vs Predicted plot

Next figure below shows the training loss plot and actual vs predicted plot made by transformer network.

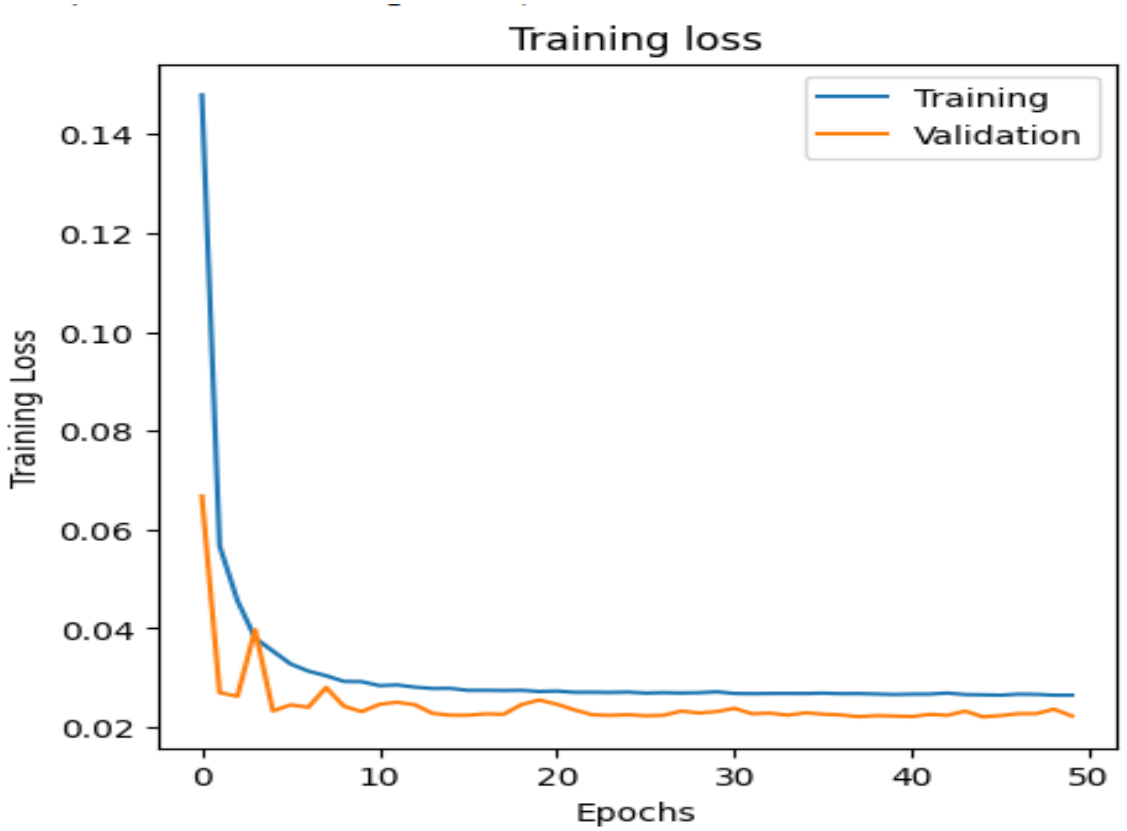

Fig 5.13: Training loss and validation plot

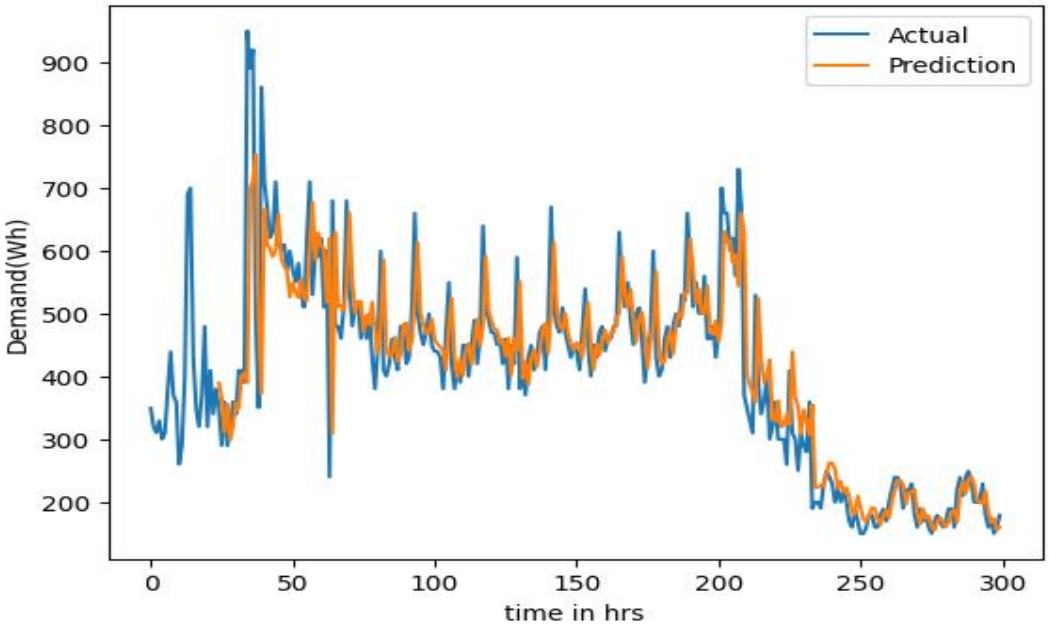

Fig 5.14: Actual vs predicted plot

Next figure shown below shows the training loss and actual vs predicted graph made with 1 D Convolutional neural network.

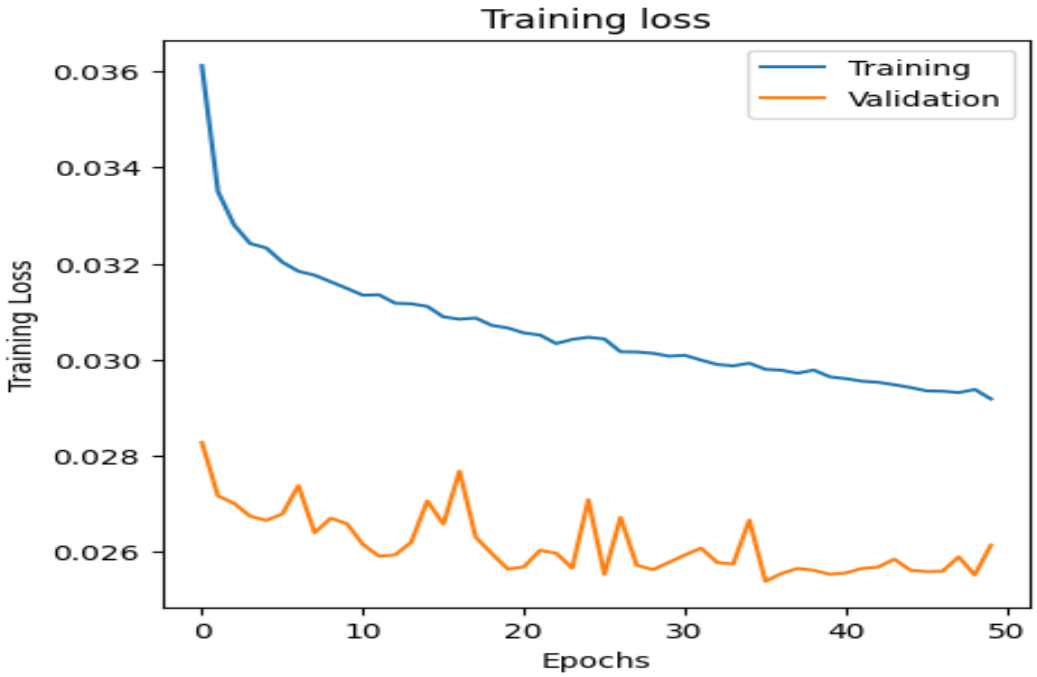

Fig 5.15: Training loss and validation plot

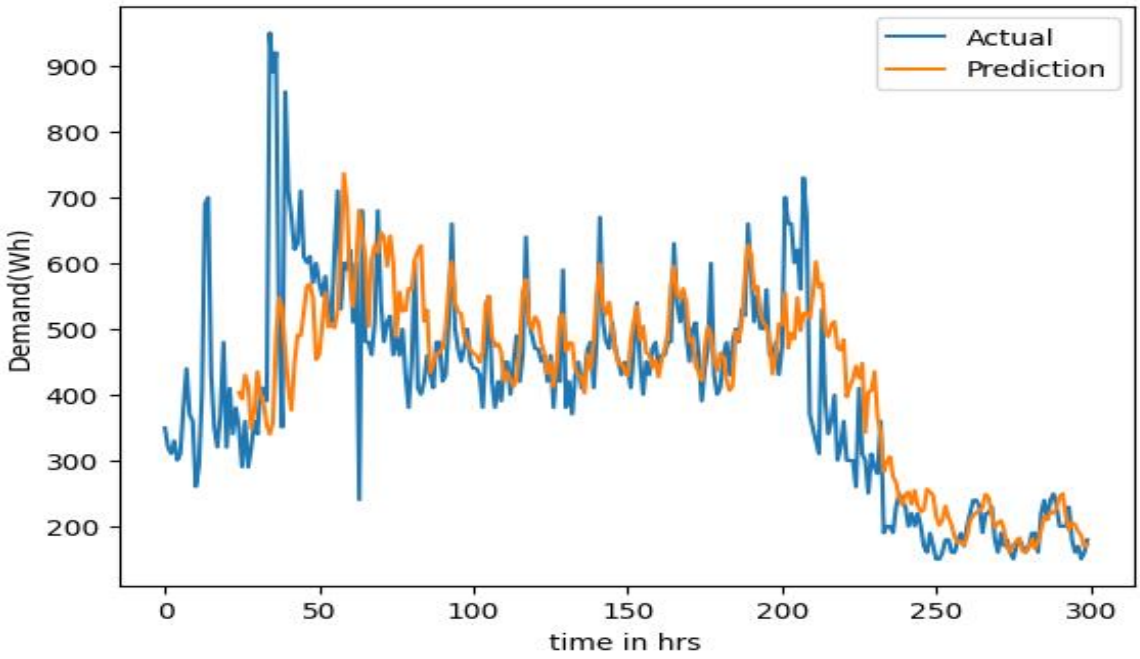

Fig 5.16: Actual vs predicted plot

Next plot shows the comparison plot made between LSTM – DWT, Transformer and 1 D Convolutional plots made between actual and predicted data.

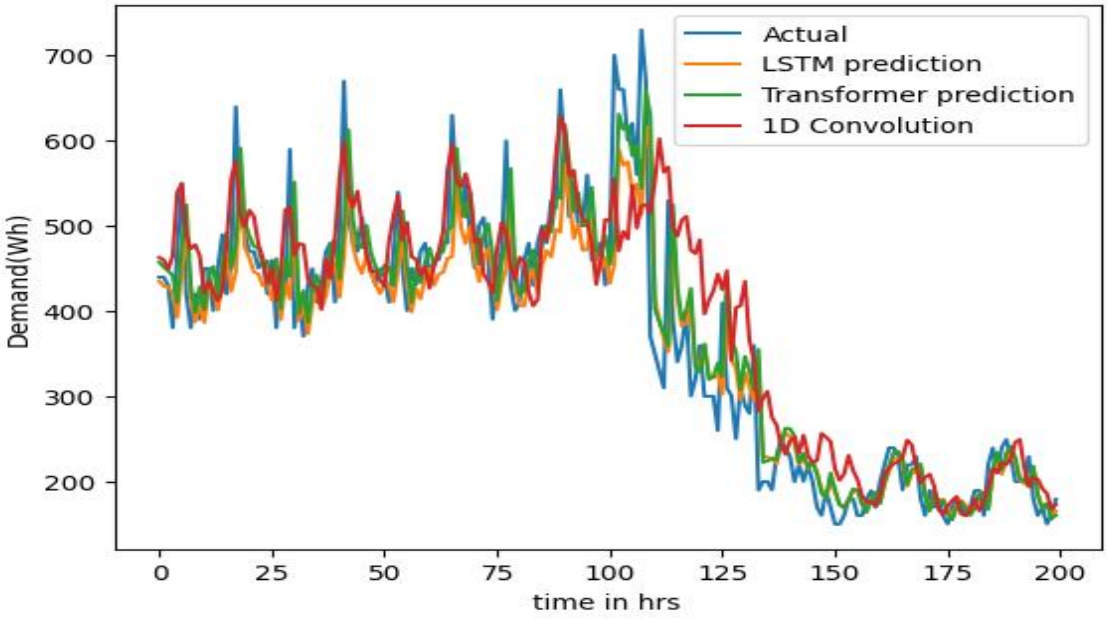

Fig 5.17: Comparison plot

Next a scatter plot between actual and predicted is plotted. And also a residual density plot is given in below figures.

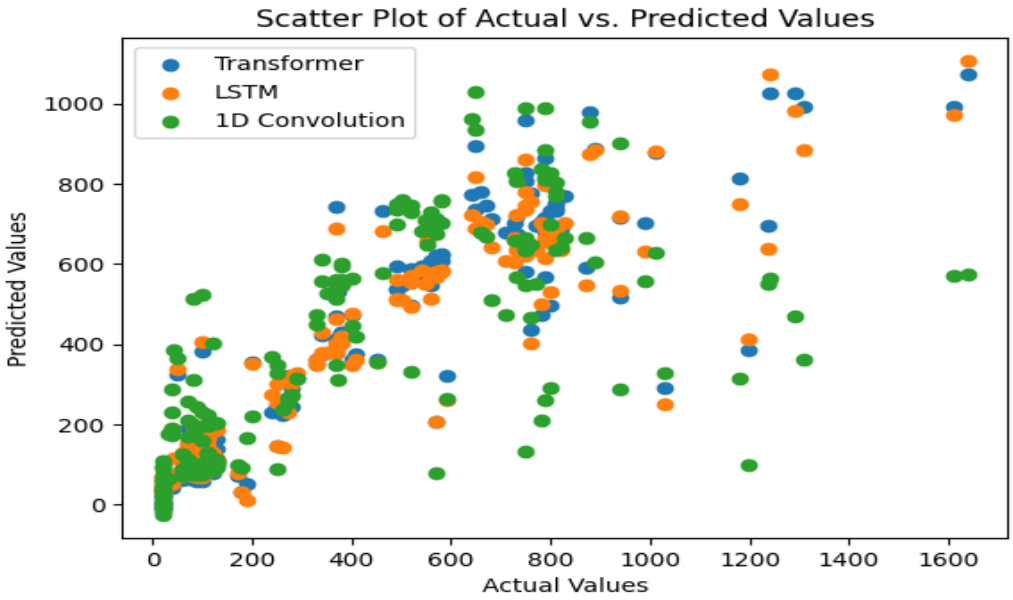

Fig 5.18: Scattered plot of actual vs predicted

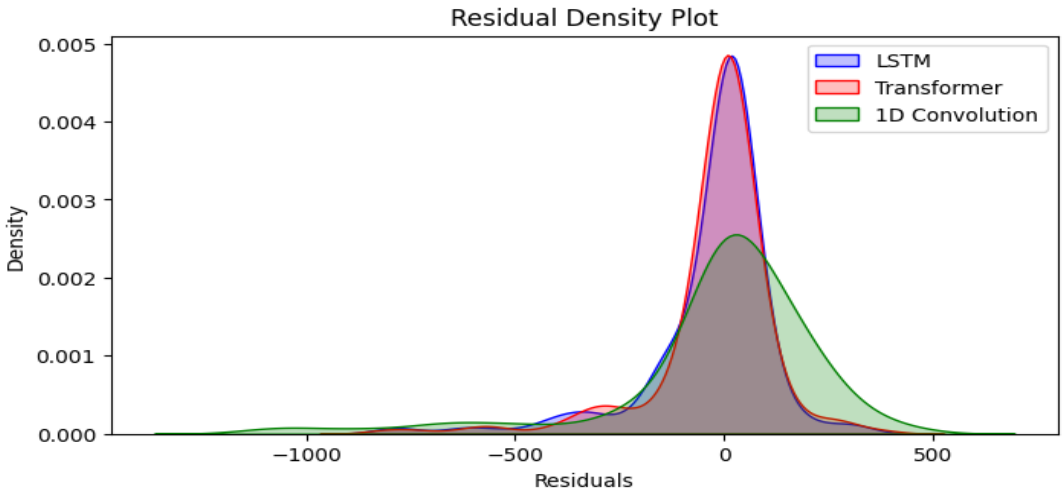

Fig 5.19: Residual Density plot

CHAPTER 6

CONCLUSION

In this project prediction of load data in a building is done by using LSTM – DWT , Transformer network and 1 D Convolutional network . And its corresponding actual vs predicted graph is plotted . And also the validation and training loss is plotted. From all these results the transformer network that shows a better accuracy with only a value of 392.69 RMSE during training. Its RMSE is less compared to other techniques during training. Thus the comparison of these methods and the load prediction is done using the data obtained from the building. And here obtained to a conclusion that use of Artificial intelligence in load forecasting is efficient and it helps to predict the future energy consumption in buildings.

PROGRAM

```
import numpy as np
import pandas as pd
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, roc_auc_score, confusion_matrix, accuracy_score, f1_score, roc_curve
from sklearn.preprocessing import MinMaxScaler
from keras.utils import pad_sequences
from keras.models import Sequential
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
from keras.layers import Activation, Dense, Dropout, Embedding, LSTM
import re
from IPython.display import display
import os
import string
import time
import random
import matplotlib.pyplot as plt
from pandas import read_csv
import math
from sklearn.metrics import mean_squared_error
import csv
import io
random.seed(10)
# importing the data file needed for the analysis
from google.colab import files
uploaded = files.upload()
random.seed(10)
input_file=io.BytesIO(uploaded['DIS.csv'])
df = read_csv(input_file, header=None, index_col=None, delimiter=',')
all_y = df[1].values
# normalize the dataset
```

```
dataset = all_y.reshape(-1, 1)
scaler = MinMaxScaler(feature_range=(0, 1))
dataset = scaler.fit_transform(dataset)

def create_dataset(dataset, look_back=1):
  dataX, dataY = [], []
  for i in range(len(dataset)-look_back-1):
    a = dataset[i:(i+look_back), 0]
    dataX.append(a)
    dataY.append(dataset[i + look_back, 0])
  return np.array(dataX), np.array(dataY)
# split into train and test sets, First 334 days test data, december 31 days training data
train_size = 8040
test_size = len(dataset) - train_size
train, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:]

# reshape into X=t and Y=t+1, timestep 240 (10 days)
look_back = 240
trainX, trainY = create_dataset(train, look_back)
testX, testY = create_dataset(dataset, look_back)
# reshape input to be [samples, time steps, features]
trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))
# create and fit the LSTM network, optimizer=adam, 50 neurons, dropout 0.2
model = Sequential()
#model.add(LSTM(15, return_sequences=True, input_shape=(1, look_back)))
# model.add(Dropout(0.2))
model.add(LSTM(5, input_shape=(1, look_back)))
model.add(Dense(1, activation='linear'))
model.summary()
# training the database
model.compile(loss='mse', optimizer='adam')
fitness = model.fit(trainX, trainY, epochs=500, batch_size=look_back, verbose=2)
#Plotting the error at each epoch
```

```
plt.figure(figsize=(5, 5), dpi=100)
plt.plot(fitness.history['loss'])
plt.legend(['Training'])
plt.title('Training loss')
plt.xlabel('Epochs')
plt.ylabel('Training Loss')
# make predictions
trainPredict = model.predict(trainX)
testPredict = model.predict(testX)
 #invert predictions
trainPredict = scaler.inverse_transform(trainPredict)
trainY = scaler.inverse_transform(trainY)
testPredict = scaler.inverse_transform(testPredict)
testY = scaler.inverse_transform(testY)
# calculate root mean squared error
trainScore = math.sqrt(mean_squared_error(trainY[0], trainPredict[:,0]))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = math.sqrt(mean_squared_error(testY[0], testPredict[:,0]))
print('Test Score: %.2f RMSE' % (testScore))
import numpy as np
import pandas as pd
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, roc_auc_score, confusion_matrix,
accuracy_score, f1_score, roc_curve
from sklearn.preprocessing import MinMaxScaler
from keras.utils import pad_sequences
from keras.models import Sequential
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
from keras.layers import Activation, Dense, Dropout, Embedding, LSTM
import re
from IPython.display import display
import os
import string
```

```
import time
import random
import matplotlib.pyplot as plt
from pandas import read_csv
import math
from sklearn.metrics import mean_squared_error
import csv
import io
#for DWT imports
from skimage.restoration import denoise_wavelet
from pywt import wavedec
# importing the data file needed for the analysis
from google.colab import files
uploaded = files.upload()
random.seed(10)
input_file=io.BytesIO(uploaded['DIS.csv'])
df = read_csv(input_file, header=None, index_col=None, delimiter=',')
all_y = df[1].values *1000
coeffs3 = wavedec(all_y, 'db1', level=3)
coeffs2 = wavedec(all_y, 'db1', level=2)
# plot for level 3 DWT decomposition
plt.figure(figsize=(10, 10), dpi=100)
plt.subplot(5, 1, 1)
plt.plot(all_y)
plt.ylabel('S')
plt.subplot(5, 1, 2)
plt.plot(coeffs3[0])
plt.ylabel('a3')
plt.subplot(5, 1, 3)
plt.plot(coeffs3[1])
plt.ylabel('d3')
plt.subplot(5, 1, 4)
plt.plot(coeffs3[2])
plt.ylabel('d2')
```

```
plt.subplot(5, 1, 5)
plt.plot(coeffs3[3])
plt.ylabel('d1')
plt.xlabel('Time')
# plot for level 2 DWT decomposition
plt.figure(figsize=(10, 10), dpi=100)
plt.subplot(4, 1, 1)
plt.plot(all_y)
plt.ylabel('S')
plt.subplot(4, 1, 2)
plt.plot(coeffs2[0])
plt.ylabel('a2')
plt.subplot(4, 1, 3)
plt.plot(coeffs2[1])
plt.ylabel('d2')
plt.subplot(4, 1, 4)
plt.plot(coeffs2[2])
plt.ylabel('d1')
plt.xlabel('Time')
all_y_denoise_3 = denoise_wavelet(all_y, method='BayesShrink', mode='soft',
wavelet_levels=3, wavelet='sym8', rescale_sigma='True')
all_y_denoise_2 = denoise_wavelet(all_y, method='BayesShrink', mode='soft',
wavelet_levels=2, wavelet='sym8', rescale_sigma='True')
# plot for denoise level 2
plt.figure(figsize=(10, 5), dpi=100)
plt.plot(all_y[-720:,])
plt.plot(all_y_denoise_2[-720:,])
plt.legend(['Original data', 'L2 De-noised data'])
plt.xlabel('time in hrs')
plt.ylabel('Demand')
# plot for noise level+
plt.figure(figsize=(10, 5), dpi=100)
plt.plot(all_y - all_y_denoise_2)
plt.legend(['noise level'])
```

```
plt.xlabel('time')
plt.ylabel('noise in data')
# Function defenition to create the dataset in bases of the time step
def create_dataset(dataset, look_back=1):
  dataX, dataY = [], []
  for i in range(len(dataset)-look_back-1):
    a = dataset[i:(i+look_back), 0]
    dataX.append(a)
    dataY.append(dataset[i + look_back, 0])
  return np.array(dataX), np.array(dataY)
# normalize the dataset
dataset=all_y_denoise_3.reshape(-1, 1)
scaler = MinMaxScaler(feature_range=(0, 1))
scalerModel = scaler.fit(dataset)
dataset = scalerModel.transform(dataset)
# split into train and test sets, First 334 days test data, december 31 days training data
train_size = 8040
test_size = len(dataset) - train_size
train_act, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:]
val_size = 7540
train, valid = train_act[0:val_size,:], train_act[val_size:len(train_act),:]

# reshape into X=t and Y=t+1, timestep 24 (1 days)
look_back = 24
trainX, trainY = create_dataset(train, look_back)
valX, valY = create_dataset(valid, look_back)
testX, testY = create_dataset(dataset, look_back)
trainXall, trainIall = create_dataset(train_act, look_back)
# reshape input to be [samples, time steps, features]
trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))
valX = np.reshape(valX,(valX.shape[0], 1, valX.shape[1]))
trainXall = np.reshape(trainXall, (trainXall.shape[0], 1, trainXall.shape[1]))
# create and fit the LSTM network, optimizer=adam, 50 neurons, dropout 0.2
```

```
model = Sequential()
model.add(LSTM(10, return_sequences=True, input_shape=(1, look_back)))
model.add(Dropout(0.2))
model.add(LSTM(5, input_shape=(1, look_back)))
model.add(Dense(1, activation='linear'))
model.summary()
# training the database
model.compile(loss='mae', optimizer='adam')
print('Fitting the data into the DNN')
fitness = model.fit(trainX, trainY, epochs=500, batch_size=look_back, verbose=1, validation_data=(valX, valY))
#print('Training accuracy is', fitness.history['accuracy'][0])
#Plotting the error at each epoch
plt.figure(figsize=(5, 5), dpi=100)
plt.plot(fitness.history['loss'])
plt.plot(fitness.history['val_loss'])
plt.legend(['Training', 'Validation'])
plt.title('Training loss')
plt.xlabel('Epochs')
plt.ylabel('Training Loss')
import numpy as np
import pandas as pd
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, roc_auc_score, confusion_matrix, accuracy_score, f1_score, roc_curve
from sklearn.preprocessing import MinMaxScaler
from keras.utils import pad_sequences
from keras.models import Sequential
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
from keras.layers import Activation, Dense, Dropout, Embedding, LSTM
import re
from IPython.display import display
import os
```

```
import string
import time
import random
import matplotlib.pyplot as plt
from pandas import read_csv
import math
from sklearn.metrics import mean_squared_error
import csv
import io
#for DWT imports
from skimage.restoration import denoise_wavelet
from pywt import wavedec
df = pd.read_csv('/content/Residential_9.csv')
df = df.dropna()
print(df)
all_y = df.iloc[:,2]*1000
coeffs3 = wavedec(all_y, 'db1', level=3)
coeffs2 = wavedec(all_y, 'db1', level=2)
# plot for level 3 DWT decomposition
plt.figure(figsize=(10, 10), dpi=100)
plt.subplot(5, 1, 1)
plt.plot(all_y)
plt.ylabel('S')
plt.subplot(5, 1, 2)
plt.plot(coeffs3[0])
plt.ylabel('a3')
plt.subplot(5, 1, 3)
plt.plot(coeffs3[1])
plt.ylabel('d3')
plt.subplot(5, 1, 4)
plt.plot(coeffs3[2])
plt.ylabel('d2')
plt.subplot(5, 1, 5)
plt.plot(coeffs3[3])
```

```
plt.ylabel('d1')
plt.xlabel('Time')
# plot for level 2 DWT decomposition
plt.figure(figsize=(10, 10), dpi=100)
plt.subplot(4, 1, 1)
plt.plot(all_y)
plt.ylabel('S')
plt.subplot(4, 1, 2)
plt.plot(coeffs2[0])
plt.ylabel('a2')
plt.subplot(4, 1, 3)
plt.plot(coeffs2[1])
plt.ylabel('d2')
plt.subplot(4, 1, 4)
plt.plot(coeffs2[2])
plt.ylabel('d1')
plt.xlabel('Time')
all_y = all_y.to_numpy().reshape(-1)
import pywt
def denoise_wavelet(signal, wavelet='db1', level=2):
    coeffs = pywt.wavedec(signal, wavelet, level=level)
    threshold = 0.4
    denoised_coeffs = [pywt.threshold(c, value=threshold, mode='soft') for c in coeffs]
    denoised_signal = pywt.waverec(denoised_coeffs, wavelet)

    return denoised_signal
denoised_data = denoise_wavelet(all_y)
denoised_data = denoised_data[:len(all_y)]
denoised_data,all_y
plt.figure(figsize=(10, 5), dpi=100)
plt.plot(all_y[:300])
plt.plot(denoised_data[:300])
plt.legend(['Original data', 'L2 De-noised data'])
plt.xlabel('time in hrs')
```

```
plt.ylabel('Demand (Wh)')
# plot for noise level+
plt.figure(figsize=(10, 5), dpi=100)
plt.plot(all_y[:300] - denoised_data[:300])
plt.legend(['noise level'])
plt.xlabel('time')
plt.ylabel('noise in data')
mport numpy as np

def create_dataset(dataset, look_back=1):
    dataX, dataY = [], []
    for i in range(len(dataset) - look_back):
        a = dataset[i:(i + look_back)]
        dataX.append(a)
        dataY.append(dataset[i + look_back])
    return np.array(dataX), np.array(dataY)

 #normalize the dataset
dataset= denoised_data.reshape(-1,1)
scaler = MinMaxScaler(feature_range=(0, 1))
scalerModel = scaler.fit(dataset)
dataset = scalerModel.transform(dataset)
# split into train and test sets, First 334 days test data, december 31 days training data
train_size = 22000
test_size = len(dataset) - train_size
train_act, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:]
train_size = 19000
train, valid = train_act[0:train_size,:], train_act[train_size:len(train_act),:]
print(train.shape,valid.shape,test.shape)
look_back = 24
trainX, trainY = create_dataset(train, look_back)
valX, valY = create_dataset(valid, look_back)
testX, testY = create_dataset(dataset, look_back)
trainXall, trainIall = create_dataset(train_act, look_back)
```

```
trainX = np.reshape(trainX, (trainX.shape[0], trainX.shape[1],1))
testX = np.reshape(testX, (testX.shape[0], testX.shape[1],1))
valX = np.reshape(valX,(valX.shape[0],  valX.shape[1],1))
trainXall = np.reshape(trainXall, (trainXall.shape[0], trainXall.shape[1],1))
trainX.shape,valX.shape,testX.shape
# create and fit the LSTM network, optimizer=adam, 50 neurons, dropout 0.2
model = Sequential()
model.add(LSTM(128,                return_sequences=True,                input_shape=(
look_back,1),name="LSTM_1"))
model.add(LSTM(96, input_shape=(look_back,1),name="LSTM_2"))
model.add(Dense(1, activation='linear',name="Prediction_Layer"))
model.summary()
# training the database
import tensorflow as tf
model.compile(loss='mae', optimizer=  tf.keras.optimizers.Adam(learning_rate=0.0001))
print('Fitting the data into the DNN')
fitness = model.fit(trainX, trainY, epochs=50, validation_data=(valX, valY))
#Plotting the error at each epoch
plt.figure(figsize=(5, 5), dpi=100)
plt.plot(fitness.history['loss'])
plt.plot(fitness.history['val_loss'])
plt.legend(['Training', 'Validation'])
plt.title('Training loss')
plt.xlabel('Epochs')
plt.ylabel('Training Loss')
# make predictions and accuracy
trainPredict = model.predict(trainX)
testPredict = model.predict(testX)
model.evaluate(testX,testY)
trainPredict_orig = scalerModel.inverse_transform(trainPredict.reshape(-1,1))
trainY_orig = scalerModel.inverse_transform(trainY.reshape(-1,1))
testPredict_orig_lstm = scalerModel.inverse_transform(testPredict.reshape(-1,1))
testY_orig = scalerModel.inverse_transform(testY.reshape(-1,1))
# calculate root mean squared error
```

```
trainScore = math.sqrt(mean_squared_error(trainY_orig, trainPredict_orig[:,0]))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = math.sqrt(mean_squared_error(testY_orig, testPredict_orig_lstm[:,0]))
print('Test Score: %.2f RMSE' % (testScore))
# shift test predictions for plotting
testPredictPlot_lstm = np.empty_like(dataset)
testPredictPlot_lstm[:, :] = np.nan
testPre = testPredict_orig_lstm[:,0]
testPredictPlot_lstm[look_back:len(dataset), :] = testPre.reshape(-1,1)
plt.plot(scalerModel.inverse_transform(dataset)[:300],label="Actual")
plt.plot(testPredictPlot_lstm[:300],label="Prediction")
plt.xlabel('time in hrs')
plt.ylabel('Demand(Wh)')
plt.legend()
plt.show()

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import numpy as np

#positional encoding
class PositionalEncoding(layers.Layer):
    def __init__(self, maxlen, embed_dim):
        super(PositionalEncoding, self).__init__()
        self.maxlen = maxlen
        self.embed_dim = embed_dim
        self.pos_encoding = self.positional_encoding()

    def get_angles(self, pos, i):
        angle_rates = 1 / np.power(10000, (2 * (i // 2)) / np.float32(self.embed_dim))
        return pos * angle_rates

    def positional_encoding(self):
```

```
                angle_rads = self.get_angles(np.arange(self.maxlen)[:, np.newaxis],
np.arange(self.embed_dim)[np.newaxis, :])
        angle_rads[:, 0::2] = np.sin(angle_rads[:, 0::2])
        angle_rads[:, 1::2] = np.cos(angle_rads[:, 1::2])
        pos_encoding = angle_rads[np.newaxis, ...]
        return tf.cast(pos_encoding, dtype=tf.float32)

    def call(self, inputs):
        return inputs + self.pos_encoding[:, :tf.shape(inputs)[1], :]


class TransformerBlock(layers.Layer):
    def __init__(self, embed_dim, num_heads, ff_dim, rate=0.1):
        super().__init__()
        self.att = layers.MultiHeadAttention(num_heads=num_heads, key_dim=embed_dim)
        self.ffn = keras.Sequential(
            [layers.Dense(ff_dim, activation="relu"), layers.Dense(embed_dim)]
        )
        self.layernorm1 = layers.LayerNormalization(epsilon=1e-6)
        self.layernorm2 = layers.LayerNormalization(epsilon=1e-6)
        self.dropout1 = layers.Dropout(rate)
        self.dropout2 = layers.Dropout(rate)

    def call(self, inputs, training=True):
        attn_output = self.att(inputs, inputs)
        attn_output = self.dropout1(attn_output, training=training)
        out1 = self.layernorm1(inputs + attn_output)
        ffn_output = self.ffn(out1)
        ffn_output = self.dropout2(ffn_output, training=training)
        return self.layernorm2(out1 + ffn_output)


maxlen = 24
embed_dim = 64
```

```
num_heads = 10
ff_dim = 64
inputs = layers.Input(shape=(maxlen, 1))
pos_encoding = PositionalEncoding(maxlen, embed_dim)(inputs)
transformer_block = TransformerBlock(embed_dim, num_heads, ff_dim)
x = transformer_block(pos_encoding)
x = layers.GlobalAveragePooling1D()(x)
x = layers.Dropout(0.1)(x)
x = layers.Dense(20, activation="linear")(x)
x = layers.Dense(10, activation="linear")(x)
outputs = layers.Dense(1, activation="linear")(x)

model_transformer = keras.Model(inputs=inputs, outputs=outputs)

import tensorflow as tf
model_transformer.compile(loss='mae',
optimizer=  tf.keras.optimizers.Adam(learning_rate=0.0001))
fitness = model_transformer.fit(trainX, trainY, epochs=50, validation_data=(valX, valY))
plt.figure(figsize=(5, 5), dpi=100)
plt.plot(fitness.history['loss'])
plt.plot(fitness.history['val_loss'])
plt.legend(['Training', 'Validation'])
plt.title('Training loss')
plt.xlabel('Epochs')
plt.ylabel('Training Loss')
# make predictions and accuracy
trainPredict = model_transformer.predict(trainX)
testPredict = model_transformer.predict(testX)
model_transformer.evaluate(testX,testY)
trainPredict_orig = scalerModel.inverse_transform(trainPredict.reshape(-1,1))
trainY_orig = scalerModel.inverse_transform(trainY.reshape(-1,1))
testPredict_orig_transformer = scalerModel.inverse_transform(testPredict.reshape(-1,1))
testY_orig = scalerModel.inverse_transform(testY.reshape(-1,1))
# calculate root mean squared error
```

```
trainScore = math.sqrt(mean_squared_error(trainY_orig, trainPredict_orig[:,0]))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = math.sqrt(mean_squared_error(testY_orig, testPredict_orig_transformer[:,0]))
print('Test Score: %.2f RMSE' % (testScore))
# shift test predictions for plotting
testPredictPlot_trans = np.empty_like(dataset)
testPredictPlot_trans[:, :] = np.nan
testPre = testPredict_orig_transformer[:,0]
testPredictPlot_trans[look_back:len(dataset), :] = testPre.reshape(-1,1)
plt.plot(scalerModel.inverse_transform(dataset)[:300],label="Actual")
plt.plot(testPredictPlot_trans[:300],label="Prediction")
plt.xlabel('time in hrs')
plt.ylabel('Demand(Wh)')
plt.legend()
plt.show()

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

maxlen = 24
embed_dim = 64
num_heads = 8
ff_dim = 64

inputs = layers.Input(shape=(maxlen, 1))
x = layers.Conv1D(filters=64, kernel_size=3, activation="relu")(inputs)
x = layers.MaxPooling1D(pool_size=2)(x)
x = layers.Conv1D(filters=128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling1D(pool_size=2)(x)
x = layers.Conv1D(filters=256, kernel_size=3, activation="relu")(x)
x = layers.GlobalAveragePooling1D()(x)
x = layers.Dropout(0.1)(x)
x = layers.Dense(20, activation="linear")(x)
```

```
outputs = layers.Dense(1, activation="linear")(x)

model_conv = keras.Model(inputs=inputs, outputs=outputs)

importtensorflow as tf
model_conv.compile(loss='mae',
optimizer=  tf.keras.optimizers.Adam(learning_rate=0.0001))
fitness = model_conv.fit(trainX, trainY, epochs=50, validation_data=(valX, valY))
plt.figure(figsize=(5, 5), dpi=100)
plt.plot(fitness.history['loss'])
plt.plot(fitness.history['val_loss'])
plt.legend(['Training', 'Validation'])
plt.title('Training loss')
plt.xlabel('Epochs')
plt.ylabel('Training Loss')
# make predictions and accuracy
trainPredict = model_conv.predict(trainX)
testPredict = model_conv.predict(testX)
model_conv.evaluate(testX,testY)
trainPredict_orig = scalerModel.inverse_transform(trainPredict.reshape(-1,1))
trainY_orig = scalerModel.inverse_transform(trainY.reshape(-1,1))
testPredict_orig_conv = scalerModel.inverse_transform(testPredict.reshape(-1,1))
testY_orig = scalerModel.inverse_transform(testY.reshape(-1,1))
# calculate root mean squared error
trainScore = math.sqrt(mean_squared_error(trainY_orig, trainPredict_orig[:,0]))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = math.sqrt(mean_squared_error(testY_orig, testPredict_orig_conv[:,0]))
print('Test Score: %.2f RMSE' % (testScore))
# shift test predictions for plotting
testPredictPlot_conv = np.empty_like(dataset)
testPredictPlot_conv[:, :] = np.nan
testPre = testPredict_orig_conv[:,0]
testPredictPlot_conv[look_back:len(dataset), :] = testPre.reshape(-1,1)
plt.plot(scalerModel.inverse_transform(dataset)[:300],label="Actual")
```

```
plt.plot(testPredictPlot_conv[:300],label= "Prediction")
plt.xlabel('time in hrs')
plt.ylabel('Demand(Wh)')
plt.legend()
plt.show()


plt.plot(scalerModel.inverse_transform(dataset)[100:300],label="Actual")
plt.plot(testPredictPlot_lstm[100:300],label="LSTM prediction")
plt.plot(testPredictPlot_trans[100:300],label="Transformer prediction")
plt.plot(testPredictPlot_conv[100:300],label = "1D Convolution")
plt.legend()
plt.xlabel('time in hrs')
plt.ylabel('Demand(Wh)')
plt.legend()
plt.show()

print("1D Convolution RMSE: ",math.sqrt(mean_squared_error(testY_orig,
testPredict_orig_conv[:,0])))
print("LSTM RMSE: ",math.sqrt(mean_squared_error(testY_orig,
testPredict_orig_lstm[:,0])))
print("Transformer RMSE: ",math.sqrt(mean_squared_error(testY_orig,
testPredict_orig_transformer[:,0])))
1D Convolution RMSE:  427.98112167160343
LSTM RMSE:  389.63005231916287
Transformer RMSE:  385.8739492080216start_index,end_index =600,800

residuals_transformer = testPredictPlot_trans[start_index:end_index] -
scalerModel.inverse_transform(dataset)[start_index:end_index]
residuals_lstm = testPredictPlot_lstm[start_index:end_index] -
scalerModel.inverse_transform(dataset)[start_index:end_index]
residuals_convolution = testPredictPlot_conv[start_index:end_index] -
scalerModel.inverse_transform(dataset)[start_index:end_index]
time_index = np.arange(len(dataset[start_index:end_index]))
plt.scatter(time_index,residuals_lstm, label='LSTM')
```

```
plt.scatter(time_index,residuals_transformer, label='Transformer')
plt.scatter(time_index,residuals_convolution, label='1D Convolution')
plt.axhline(0, color='black', linestyle='--', linewidth=0.5)
plt.xlabel('Time')
plt.ylabel('Residuals')
plt.title('Residual Plot')
plt.legend()
plt.show()
```

REFERENCES

[1] Seyed Azad Nabavi, Naser Hossein Motlagh ***,"Deep learning in energy modelling: Application in smart buildings with distributed energy generation*** **",** Energy, vol.9,pp.3110960,sep.2021

[2] N.Ayoub,F.Musharavathi, S Pokharel and H A Gabbar ***,"ANN model for energy demand and supply forecasting in a hybrid energy supply system,"***in IEEE Int.Conf.Smart Energy Grid Eng,Aug 2018, pp 25-30

[3] N. H. Motlagh, S. H. Khajavi, A. Jaribion, and J. Holmstrom, **'An IoT basedautomation system for older homes: A use case for lighting system,"** in Proc. IEEE 11th Conf. Service-Oriented Comput. Appl. (SOCA) ,pp. 16,,Nov. 2018.

[4] J K Gruber, M. Prodanovic and R.Alonso,**"** ***Estimation and analysis of building energy demand and supply costs",***energy, vol.83,pp216-225, Dec 2015

[5] X. Guan, Z. Xu, and Q.-S. Jia**, "Energy-efcient buildings facilitated.by microgrid,"** IEEE Trans. Smart Grid, vol. 1, no. 3, pp. 243252, Dec. 2010