Effective Electrical Load Balancing Using RNNs

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Prithish Samanta

19BCE2261

Akshat Rastogi

19BCE2220

Under the guidance of

Dr. Vishnu Srinivasa Murthy Y

School of Computer Science and Engineering VIT, Vellore.

May, 2023

DECLARATION

I hereby declare that the thesis entitled "Effective Electrical Load Balancing Using RNNs" submitted by us, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering* to VIT is a record of bonafide work carried out by me under the supervision of Vishnu Srinivasa Murthy Y.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore

Date : 12.05.2023

Signature of the Candidate

CERTIFICATE

This is to certify that the thesis entitled “Effective Electrical Load Balancing Using RNNs” submitted by Prithish Samanta 19BCE2261 and Akshat Rastogi 19BCE2220, School Of Computer Science and Engineering, VIT, for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a record of bonafide work carried out by him under my supervision during the period, 01.07.2022 to 30.04.2023, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date : 12.05.2023

Signature of the Guide

Internal Examiner

External Examiner

Head of the Department

Programme

ACKNOWLEDGEMENTS

We would like to extend our heartfelt appreciation to Dr. G. Viswanathan, the Chancellor of VIT, and our project guide Dr. Vishnu Srinivasa Murthy Y for providing us with the invaluable opportunity to explore Recurrent Neural Networks in depth. Their unwavering support and guidance throughout this project were instrumental in its success. We would also like to thank the members of our research team for their collaboration and dedication to this project. We had faced a few challenges in this journey, but we were able to overcome them and finish the project on time. This experience has deepened our knowledge of this complex topic and has equipped us with invaluable research skills that will serve us well in our future endeavors.

Prithish Samanta, Akshat Rastogi
Student Name

Executive Summary

The demand for energy has sharply increased recently all around the world. In recent years, there has been a sharp rise in electricity use. The amount of electricity used is increasing as technology advances on a daily basis. The amount of electricity utilized in agricultural areas is rising along with it, making it difficult for farmers to keep up with the rising demand for electricity. The amount of electricity needed varies depending on the weather on any given day, making it difficult to forecast the amount that would be needed in the near future. In this study, we sought to use recurrent neural networks to overcome this issue. Over the past few months we have tried to build an AI model that can perform load balancing on a time series data. The time series data contains 5 years worth of data on the electricity demands of the farms. This data, LSTM, and GRU have all been used to try and address this issue. Deep learning algorithms that forecast future load requirements have been created. Additionally, we evaluated the effectiveness of the new models in comparison to the previous study's outdated models.

CONTENTS

List of Figures

List of Tables

List of Abbreviations

RNN	Recurrent Neural Networks
LSTM	Long Short Term Memory
GRU	Gated Recurrent Unit
MAPE	Mean Absolute Percentage Error
MSE	Mean Squared Error
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
ERCOT	Electrical Reliability Council of Texas
RTE	Réseau de transport d'électricité

Symbols and Notations

σ	Sigmoid Function
X(t)	Input at time t
h(t)	hidden state (LSTM), memory (GRU)
tanh	tanh function
W	Weights

1. INTRODUCTION

1.1. Theoretical Background

In recent years, we have seen a sharp rise in energy demands across the world. Electricity usage has increased rapidly in the past few years. As technology is improving day by day, the amount of electricity being consumed is also rising. Many industries around the world from different sectors are also utilizing these new technologies in their everyday usage, as a result, their energy consumption is also going up. There are various examples of this, such as the use of sensors to track the functioning of a factory, the use of electric vehicles as an alternative to fuel ones, etc. These trends have also brought various changes in the agriculture sector around the world, where farmers are using new machinery and tools to simplify their tasks. These new machinery can be seen in the form of sprinklers that are installed all over the farms to automate the process of watering the crops, or in the form of sensors that are being used in the farms to monitor and optimize the process of cultivation. The majority of this equipment runs on electricity, and the quantity of electricity it uses might vary greatly depending on the surrounding factors. These factors include soil moisture, dew, atmospheric humidity, and others. These factors fluctuate daily or even hourly and are not consistent. The weather has a significant impact on these factors.

1.2. Motivation

Farmers play an important role in the development of any country irrespective of whether it's a large country or a small one. They are the key to our survival on this planet as they try their level best to feed everybody. A major problem that is faced by the farmers because of the advancements made in predicting the amount of electricity that would be required in the future. Depending on the environmental factors the amount of electricity required would change. For example in the monsoon season, there will be a high amount of rainfall therefore we would require less electricity to run the sprinklers. The opposite could be said about the summer season. Due to the hot climate, more water would be required for irrigation, therefore more electricity would be required. Predicting the optimum amount of electricity will help the farmers in many ways, such as, it can help in

cutting costs, it can save water, and it also reduces the wastage of electricity. This would in turn help the farmers improve and increase their yield.

1.3. Aim of the Proposed Work

To tackle the above problem, our project focuses on building an AI model that helps in predicting the future electrical load. The concept of load balancing would help in accurately predicting the amount of electricity that could be required in the future. Our project explores the concept of RNNs (Recurrent Neural Networks) and utilizes them to solve this problem. Recurrent neural networks (RNN) are a type of artificial neural network that uses sequential data or time series data. Like the feedforward, Recurrent Neural Networks utilize the training data to learn, but unlike but they are distinguished by their “memory”. RNNs memory unit which helps it to remember the necessary information from the prior inputs and outputs and influence the present output. These features of the RNNs make them suitable for performing tasks like time series analysis, speech recognition, and natural language processing.

1.4. Objective(s) of the Proposed Work

In this project, we want to build an AI model that can help us predict the future electrical load using a dataset that takes the above aspects into account. Two different datasets are used by us to build these models. The first one is provided by the Electrical Reliability Council of Texas (ERCOT) while the second one is from the Réseau de transport d'électricité (RTE) France.

We are going to perform time series analysis for this project using the LSTM model and the GRU model. Time series analysis has already been performed on the above mentioned datasets using ARMA and the ARIMA models. Our goal for this project is to create LSTM and GRU models for training the datasets and achieve high accuracies and low error rates.

2. LITERATURE SURVEY

2.1. Survey of the Existing Models/Work

Table. 2.1 Literature Survey

Paper S.No.	Approach	Remarks	Accuracy
1.	This paper tackles the problem of load balancing for a smart grid setup and tries to minimize the error rates while achieving higher accuracy. The LSTM model was used by the author.	According to the authors, LSTM is the best technique to use for time series data as it provides the minimum root mean square value (RMSE).	RMSE - 3.35% MAPE - 5.21% (A big improvement compared to previous algorithms)
2.	Previously manual and trigger-based algorithms were being used to perform load balancing in cloud platforms. This paper tries to use the concept of RNNs and LSTM to make this process easier and to prevent long downtimes.	According to the authors, RNN is a better technique compared to the ones already being used. The current techniques can be slow or might even involve some manual work, in turn increasing the downtime.	Although they haven't included any data, the authors have supplied a graph comparing the two algorithms' actual and projected observations. The predictions made by the LSTM model are more accurate.
3.	In order to achieve load balancing, the author of this study used Deep Peephole LSTM, a variant of the LSTM model. The author discusses the benefits of Peephole LSTM for time series analysis and contrasts it with other algorithms like ARMA and	On performing the mentioned test, the author finds that Deep Peephole LSTM gives better accuracy than a few other methods like ARMA, ARIMA, Single RNN and Deep RNN..	RMSE - 1.3%

	ARIMA.		
4.	In this paper, the authors use a few kinds of LSTM models for balancing network traffic and generating predictions quickly. They have used Vanilla LSTM, and Delta LSTM.	Due to a lack of computational resources to handle training data at scale and the volatile nature of network data on short-time scales, classical network traffic prediction models can only be utilized for huge time windows.	The Delta LSTM and Vanilla LSTM give similar values for most of the cases.
5	In this study, the authors attempt to enhance the GRU model by minimizing its hidden states and parameters. They have put forth three new models, each slightly different from the GRU model as it is currently known.	The graphs demonstrate that GRU1 and GRU2 almost match GRU's performance on the data, however, GRU3 falls short at this (constant base) learning rate.	The authors state that while GRU1 and GRU2 have indistinguishable performance, GRU3 lags in few areas.
6	This paper uses 6 different types of deep learning techniques to perform time series analysis on a dataset containing 7 years of data. The models used in this paper were Simple Moving Average, Weighted Moving Average, Simple Exponential Smoothing, Holt Linear Trend, Holt Winters, and Centered Moving Average.	4 error measuring techniques were used to compare the algorithm. The techniques were Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE)	The Centered Moving Average algorithm gives the lowest MSE and RMSE, while the Holt Winters algorithm yields the lowest MAE and MAPE.
7	This research builds forecasting models for short- to medium-term aggregate load forecasting using	The LSTM-based model, according to the authors, has demonstrated greater accuracy than a machine learning model that	RMSE = 428.01 MAE = 292.49

	machine learning and long short-term memory-based neural networks in a variety of configurations. The issue of selecting and creating various time series models is resolved in this study.	is hyperparameter tuned for optimisation.	
8	The proposed model uses historical load data, weather data, and statistical features extracted from the historical data, with the purpose of improving the accuracy and consistency of electric load prediction for different time horizons.	According to the test case findings and comparisons, the suggested hybrid model increased the forecasting precision for three different forms of load forecasting.	RMSE for Florida = 50.60, RMSE for northern regions of texas = 119.667
9	In this study, a load forecasting approach based on the LSTM model is given, which takes into account a number of variables, including temperature and wind speed, and prevents gradient disappearance or explosion shortages. This model can timely and accurately depict the electrical grid's load capacity.	The impact of "Replacement of Coal with Electricity" on load forecasting is heavily taken into account in this research. In order to condense load forecasting into the prediction of time series data for the one-dimensional load, an LSTM model was developed in this research.	Load accuracy = 93%.
10	In order to further increase the accuracy of short-term power demand forecasting, this research suggests a new	The authors claim that an enhanced binary coding genetic algorithm was presented based on the genetic algorithm to further	RMSE FOR VMD = 161.51, VLG = 56.73

	hybrid system that combines data preprocessing technology, the most sophisticated deep learning prediction method, and the bionic optimisation algorithm.	increase the model's accuracy.	
11	The future electrical load is predicted in this study using well-defined machine learning techniques like RNN (Recurrent Neural Networks) and LSTM (Long Short Term Memory), which can be used to plan the generation, transmission, and distribution systems of electrical utilities.	In this study, predictions from time series models are contrasted with those made by RNN and LSTM. For large sequence predictions, the scientists found that the machine learning techniques performed better.	RMSE for RNN =41.19 MW, MAPE = 3.67% RMSE for LSTM = 41.56 MW, MAPE = 3.80

2.2. Summary/Gaps identified in the Survey

In paper 2, the authors have implemented Long short-term memory for Load balancing in web servers. Their objective was to better manage the workload or resource allocation for servers in order to increase performance and service quality. This paper's goal was successfully accomplished, but only for linear network architectures. They were unable to accomplish the same goal for random networks and structured networks architectures at the time this research was being written. For implementing the same deep learning model on the other two types of networks the authors would need more training data.

In paper 3, the author discusses the need of effective load balancing and tries to achieve it with the help of the Peephole LSTM technique.Each layer of the multi-layer LSTM used in this study has many neural cells and peephole connections. Deep Peephole LSTM's training time is a concern. The training period is greater due to the complexity of the model.

In paper 4, the authors describe a system for forecasting network traffic that trains a Long Short Term Memory (LSTM) neural network and produces forecasts quickly. The authors offer an analysis for several LSTM variants, including Vanila, Delta, Cluster, and Cluster Data LSTM. In the majority of cases, Delta LSTM is not shown to offer significant advantages over Vanila LSTM. The author claims that there are times when the discrepancy can become extremely large, but the cause of this is unknown.

In paper 5, the authors aim to improve the GRU model by reducing its hidden states and parameters. They have proposed three new models, each slightly different from the existing GRU model. It is discovered that there may be a trade-off between the improved accuracy performance and the reduced number of GRU model parameters. When compared to GRU1, GRU2, and GRU3, GRU3, which computes each gate using only the bias, performs less well.

In paper 6, the authors undertake time series analysis on a dataset of 7 years of data from 2011 to 2017 using 6 different types of deep learning techniques. Simple Moving Average, Weighted Moving Average, Simple Exponential Smoothing, Holt Linear Trend, Holt Winters, and Centered Moving Average were the models employed in this study. The model was used to forecast how much electricity would be used by the college building in 2018. Holt Winters produces the lowest MAPE value, whereas Centered Moving Average produces the lowest MAE value. Simple Moving Average, Weighted Moving Average, Simple Exponential Smoothing, and Holt Linear Trend are the first four models, and their forecasts are not very accurate.

3. OVERVIEW OF PROPOSED SYSTEMS

3.1. Introduction and Related Concepts

In this project, we are trying to perform time series analysis on two datasets, and predict the future electrical loads. Several models have been presented in order to effectively examine the time series. Few of these methods include ARIMA (Auto Regressive Integrated Moving Average), ARMA (Auto Regressive Moving Average), EWMA (Exponentially Weighted Moving Average), etc. Though these models are good in performing time series analysis on datasets, they have many problems which make them inefficient.

The Autoregressive Integrated Moving Average (ARIMA) model is a modification to the traditional ARMA(Autoregressive Moving Average) model. The ARIMA model is useful for predicting time series and is frequently used for non-stationary series, and it is not good in performing multivariate analysis. Since the time-series data of load information is very nonlinear and complicated, many traditional forecasting methods, such as ARMA and ARIMA, which are linear models, are not appropriate for this situation. Exponential Weighted Moving Average (EWMA) is another popular time series prediction method, but the prediction ability of it is very limited. It also faces a problem with its smoothing factor. As the smoothing factor fluctuates significantly over time, it is erroneous to use a constant smoothing factor in the model.

RNN's or Recurrent Neural Networks are a good fit for performing such tasks. RNN is a type of neural network that is good at recognizing sequences that are not recognizable by other types of Neural Networks. RNN's also have the ability to memorize the previous info and apply it to the calculation of the current output. They also provide good results in forecasting by using a limited number of steps. Despite all these benefits of RNNs, it has few big disadvantages which affect its performance too. One of the major problems faced while training a model using RNNs is the Vanishing Gradient and the Exploding Gradient problems.

Gradients are used by RNNs to train and update the neural network's weights. A gradient is the change in loss for a given change in weight. Back propagation is a

technique used by RNNs in each iteration to update these weights and reduce loss. This works as a chain reaction where the gradients closer to the output layers work are multiplied with the gradients of the layers closer to the input layers. If the gradients are too great, the multiplication of the gradients will eventually grow enormous, leading to an Explosive Gradient Problem; if the gradients are too small, the multiplication of the gradients will eventually grow extremely small, leading to a Vanishing Gradient Problem. Another issue that the RNNs run into is that they have poor memories and are unable to account for many elements of the past in order to forecast the future.

To overcome the above mentioned challenges we have used two other RNN based models that have some changes to overcome these problems faced by traditional RNNs. We have used LSTM(Long Short Term Memory) and GRU(Gated Recurrent Unit) for this study.

3.2. Framework, Architecture or Module for the Proposed System

Here we have used two kinds of RNNs for predicting the future electrical load i.e LSTM and GRU. The architecture of both the proposed models are described below.

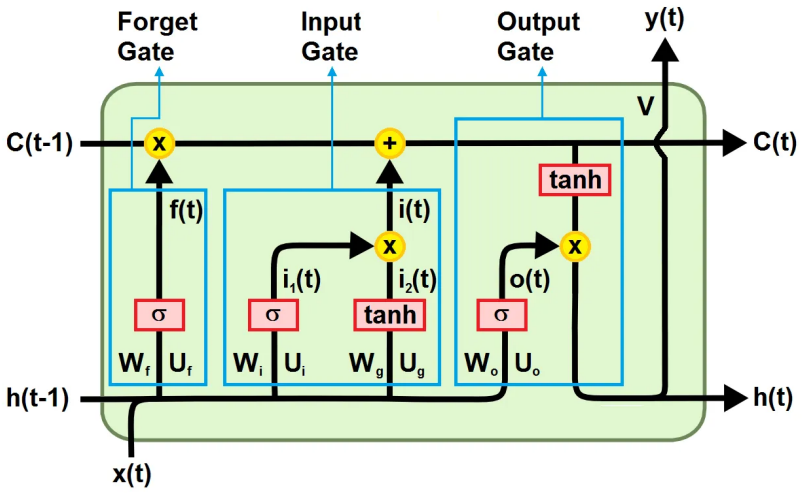

Fig. 3.1 LSTM Architecture

The above diagram represents the architecture of an LSTM cell at time (t). The LSTM was created to address the gradient issues with the conventional RNN. An LSTM unit is composed of:

The Cell State: This memory cell is essential to an LSTM model. It preserves the condition over time. The cell state is represented by Ct at time (t). The cell state facilitates the transmission of useful data from preceding time steps and reduces the impact of short-term memory. The three gates present in the LSTM regulate the cell state information. These gates can either add or remove data.

Forget Gate: The Forget Gate determines, based on the new information received by the cell at time (t), which information in the current cell state must be forgotten. The sigmoid function is applied to the information from the previous hidden state (i.e., ht-1) and the present input (i.e., Xt). If the output is close to 0, it indicates that the information can be neglected, whereas if it is close to 1, it indicates that the information must be saved.

Input Gate: The Input Gate determines what new information will be encoded with the current cell state information. Initially, the sigmoid function is applied to the present input state (i.e. Xt) and the previous hidden state (ht-1). To fine-tune the network, the hidden state and current input are additionally sent to a tanh function, which squeezes values between -1 and 1. Then, the tanh and sigmoid outputs are multiplied element by element. The sigmoid output determines which data should be retained.

Output Gate: The Output Gate determines what information is encoded in the cell state and is transmitted to the network as the next hidden state for the next time step (t+1). In this gate, the previous hidden state and current input are added together and passed to a sigmoid function before the new cell state is transmitted to the tanh function. The tanh output is multiplied with the sigmoid output to determine what information the hidden state should contain. The output is the new condition of hiding. The new cell and hidden states are then carried forward to the subsequent time step (t+1).

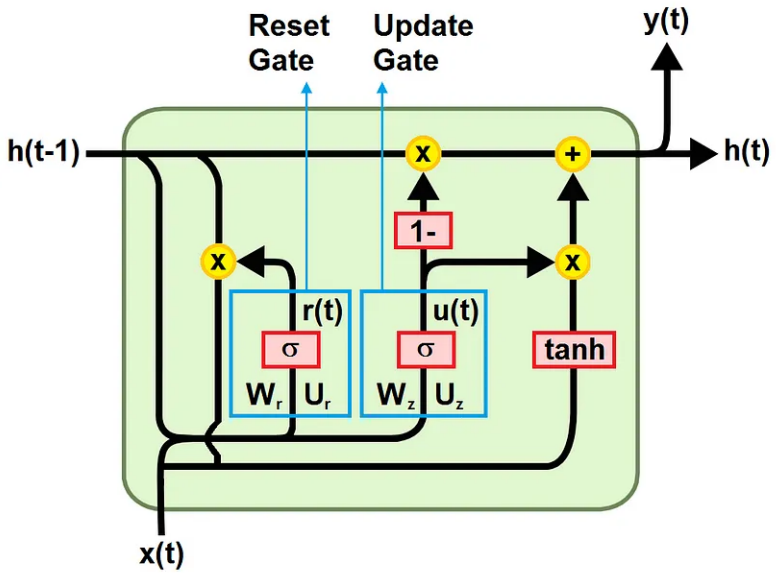

Fig. 3.2 GRU Architecture

The above diagram represents a GRU cell at time (t). GRU is another type of RNN which is similar to LSTMs. Like LSTM, GRUs also use gates to solve the gradient problem. The GRU gates are programmed to retain information from many time steps prior to the actual time step, without washing it through time, or to eliminate information that is extraneous to the prediction. It has been found that the GRU RNN typically performs on par with or better than the LSTM. A GRU unit consists of:

Reset Gate: The Reset Gate determines how much information from previous time stages can be forgotten. It combines incoming data with previously stored information. Initially, the gate computes the weighted sum between the memory (ht-1) and the input (Xt at time t). The preceding result is subjected to a sigmoid function, and its values are compressed between 0 and 1.

Update Gate: The Update Gate assists the model in determining how much information from previous time steps should be conveyed to the future. In order to circumvent the problem of vanishing gradients, the model may duplicate all previous data.

Memory: The memory content utilizes the reset gate to store historical information. The update gate is then used to determine which data should be transmitted to the GRU cell at time (t) (i.e. ht).

We have used Multiple layers for designing our LSTM and GRU models. The original models have one layer, then a typical feedforward output layer. A reliable method for solving difficult sequence issues is the stacking of layers in the LSTM and GRU models. Over time, the input data gains levels of abstraction due to the addition of further hidden layers, which aid in recombining the learned representation from earlier layers. The next layer receives a series of outputs from a layer above rather than a single value. This makes the model efficient.

Using these two types of RNNs we have created models to predict the future electrical load. We then reverse_transfrorm the predicted values and compare the accuracy of the models by using a few algorithms. Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE) are the algorithms we've employed to compare the two models.

3.3. Proposed System Model

As explained above, for this project we have used two special types of recurrent neural networks. These recurrent neural networks differ from conventional RNNs in that they contain gates that regulate the passage of information into the cell. They prevent the cell from gradient problems, like vanishing gradients and exploding gradients. The special types of recurrent neural networks also consist of a special unit called memory or cell state. This unit helps the cell to maintain its state over time. It helps the model get rid of the short memory problem which was faced by regular RNNs. The two types of special RNNs used by us are Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU). Both LSTM and GRU are similar with slight differences. In the upcoming paragraphs we are going to describe the mathematical modeling of the two.

Long Short Term Memory or LSTM has three types of gates. The gates are Forget Gate, Input Gate, and Output Gate. These were discussed in the preceding section. The LSTM cell at time (t) is connected to the LSTM cells at times (t-1) and (t+1),

respectively. The cell at each time step is also associated with an input vector; the input vector associated with the cell at time (t) is denoted as **X(t)**. **C(t-1)** represents the input cell state from the cell at time (t-1) and **C(t+1)** represents the output cell state going to the cell at time (t+1). Each cell state has an additional input known as the hidden state, which is transmitted from the preceding cell. The hidden state of the cell at time (t) is denoted by the symbol **h(t-1)**. The cell at time (t) transmits its hidden state, h(t), to the cell at time (t+1).

A weight matrix U links the input vector at time t to the LSTM cell at time t, whereas a weight matrix W links the LSTM cell to the LSTM cells at time t-1 and time t+1. The submatrices for the matrices W and U are, respectively, **(Wf, Wi, Wg, Wo: Uf, Ui, Ug, Uo)**. In this section, we will describe the operation of the gates using mathematical equations.

1. **Forget Gate:** The forget gate determines what information from the previous time phase is significant, what information should be retained, and what information should be discarded.

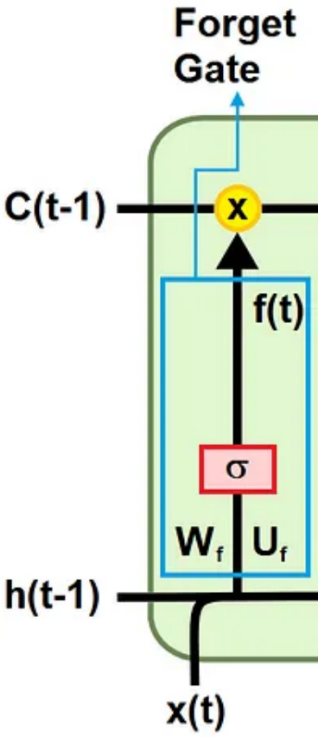

Fig. 3.3 Forget Gate

The sigmoid function collects data from the previous hidden state [h(t-1)] and the present input state **[X(t)]**. This function returns a value between 0 and 1. If the result is close to zero, we can discard the information, but if it is close to one, we must save or transmit it.

$$\mathbf{f(t) = \sigma(X(t)Uf + h(t-1)Wf)}$$

The above equation depicts the function of the forget gate. **f(t)** is the output of the forget gate.

2. **Input Gate:** The state of the cell is changed by the input gate. It chooses what information from the current time step needs to be added.

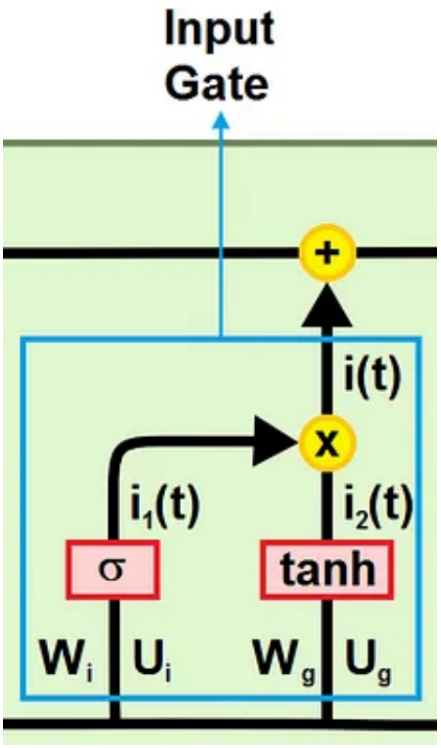

Fig. 3.4 Input Gate

At the start, a sigmoid function is given the previous hidden state **[h(t-1)]** and the current input **[X(t)]**, and it produces an output between 0 and 1.

$$\mathbf{I1(t) = \sigma(X(t)Ui + h(t-1)Wi)}$$

It also sends the hidden state **[h(t-1)]** and the current input **[X(t)]** to a tanh function, which squeezes values between -1 and 1 to make the network's tuning better.

$$\mathbf{I2(t) = tanh(X(t)Ug + h(t-1)Wg)}$$

Then, the tanh **[I2(t)]** and sigmoid **[I1(t)]** outputs are multiplied element by element.

$$\mathbf{i(t) = I1(t) * I2(t)}$$

The above equations depict the function of the input gate. **i(t)** is the output of the input gate.

3. **Output Gate:** This gate determines the output value for the current time phase. This gate determines the next hidden state, which stores information regarding the previous input.

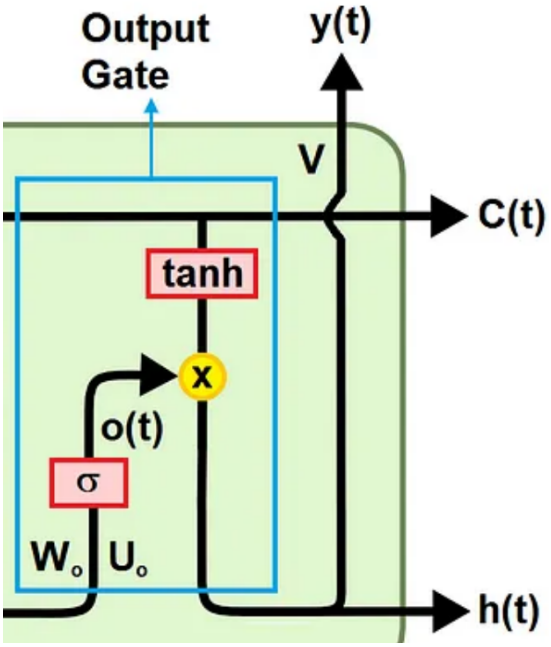

Fig. 3.5 Output Gate

First, the hidden state **[h(t-1)]** from before and the present input **[X(t)]** are added together and then given to a sigmoid function.

$$\mathbf{O(t) = \sigma(X(t)Uo + h(t\text{-}1)Wo)}$$

The new cell state **[C(t)]** is then transmitted to the tanh function. In order to determine which information should be in the hidden state, the tanh output is multiplied by the sigmoid output **[O(t)]** at the end.

$$\mathbf{h(t) = tan(h) * O(t)}$$

The preceding equations illustrate the operation of the output gate. The output of the output gate is **h(t)**. The output is the new condition of hiding.

The new cell and hidden states are then carried forward to the subsequent time phase.

GRU, or Gated Recurrent Unit, has two distinct categories of gates. Reset Gate and Update Gate are the gates. These were discussed in the preceding section. The GRU cell at time (t) is linked to the GRU cells at times (t-1) and (t+1). The cell at each time step is also associated with an input vector; the input vector associated with the cell at time (t) is denoted as **X(t)**. Each cell state also possesses a memory that is transmitted from the prior cell state as **h(t-1)**. The state of the cell at time (t) transfers the memory h(t) to the state of the cell at time (t+1).

A weight matrix U links the input vector at time t to the GRU cell at time t, whereas a weight matrix W links the GRU cell to the GRU cells at time t-1 and time t+1. **(Wr, Wz: Ur, Uz)** are the submatrices for the matrices W and U, respectively. In this section, we will describe the operation of the gates using mathematical equations.

1. **Reset Gate:** It determines how to integrate the new knowledge with its existing knowledge and how much of its existing knowledge can be forgotten.

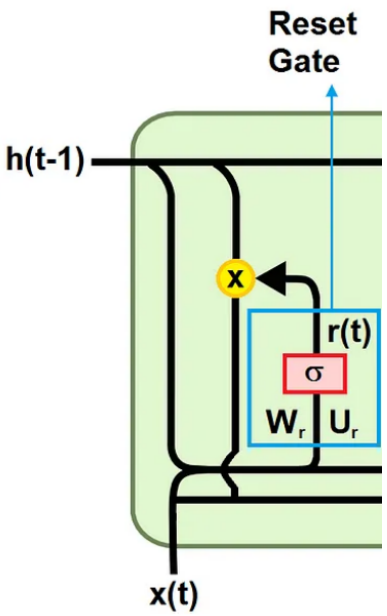

Fig. 3.6 Reset Gate

Initialization involves calculating a weighted sum of the input **[X(t)]** and the memory **[h(t-1)]**. Using a sigmoid activation function, the result is then compressed between 0 and 1.

$$\mathbf{r(t) = \sigma(X(t)Ur + h(t-1)Wr)}$$

The above equations depict the function of the reset gate. **r(t)** is the output of the input gate.

2. **Update Gate:** This gate helps the model determine how much information from previous time steps must be transmitted to the future. This is extremely useful because the model can decide to copy all prior knowledge to avoid the vanishing gradient problem.

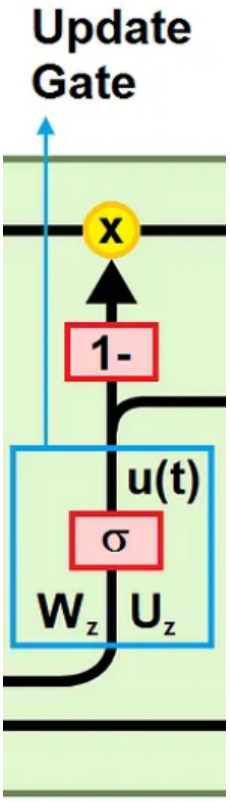

Fig. 3.7 Update Gate

Similar to the reset gate, a weighted sum of the input **[X(t)]** and memory **[h(t-1)]** is calculated. Using a sigmoid activation function, the result is then compressed between 0 and 1.

$$\mathbf{z(t) = \sigma(X(t)Uz + h(t-1)Wz)}$$

These equations illustrate the operation of the update gate. The output of the input gate is **z(t)**.

3. **Current Memory and Final Memory:** The memory stores the important information from the past in the reset gate.

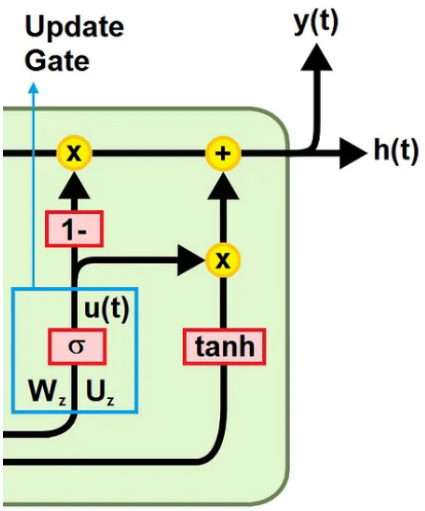

Fig. 3.8 Final GRU Output

First, a multiplication is done between the output of the reset gate, **[r(t)]**, and the final memory at the previous time step, **[h(t-1)]**.

$$\mathbf{h1(t) = (r(t) * h(t-1))}$$

Then a weighted sum is performed between the result and the input, **[X(t)]**. This step is followed by applying tanh function on the whole.

$$\mathbf{h'(t) = tanh(X(t)Uh + h1(t)Wh)}$$

To calculate **[h(t)]**, the vector that transmits information to the subsequent phase, the following operations are carried out. It is computed by performing element-by-element multiplication between the update gate **[z(t)]** and **[h'(t)]** and between **[1 - z(t)]** and **[h(t-1)]**, followed by weighted addition of the two outcomes.

$$\mathbf{h(t) = (1 - z(t)) * h(t-1) + z(t) * h'(t)}$$

The above equations depict the function of the final output of the gru cell. **h(t)** is the vector which holds information for the current unit and passes it to the next time step.

4. PROPOSED SYSTEM ANALYSIS AND DESIGN

4.1. Introduction

In this study, we attempted to use recurrent neural networks, or RNNs, to solve the given challenge. RNNs are an excellent choice for carrying out the aforementioned task since they are adept at identifying sequences that other forms of neural networks find difficult to identify. RNNs can also remember information from the past and use it to compute the present result. They also deliver effective forecasting outcomes with a minimal number of processes. RNNs have two significant disadvantages in spite of these benefits. These are:

1. Vanishing Gradient and Exploding Gradient Problem
2. Short Memory

We have used two special kinds of RNNs that address the concerns outlined above to get around the aforementioned limitations. Long Short Term Memory (LSTM) and Gated Recurrent Units (GRU) are the models that are employed. The gradient difficulties are resolved using LSTM and GRU with the use of gates. The three gates that make up an LSTM—the Input Gate, Output Gate, and Forget Gate—help each neural network keep its cell state constant. On the other hand, GRU features two gates that also aid in solving its gradient-related issues. The Reset Gate and the Update Gate are the GRU's two gates.

The short memory problem of the RNNs is also solved with the help of a memory unit. In LSTM, this memory unit is called Cell State and it maintains the state of the cell over time. The cell state aids in minimizing the impact of short-term memory by communicating the pertinent data from earlier time steps. The three gates in the LSTM control the information about the cell state. This performs a similar function in GRU and is known as memory. The reset gate is used by the memory content to store the pertinent historical data. The information that should be transmitted to the following GRU cells is then determined using the update gate.

4.2. Requirement Analysis

In order to successfully complete this project, a few functional and non-functional requirements must be met. Functional requirements are the features or operations that product developers must provide in order for users to carry out their tasks. However, non-functional requirements describe how the system should function.

Functional Requirements

The demand for relevant datasets must be satisfied in order for the models to function successfully. Two datasets that meet these criteria were used in this project. The Electrical Reliability Council of Texas (ERCOT) contributed the datasets we used, while the other dataset comes from the Réseau de transport d'électricité (RTE) France. The aforementioned data sets satisfy our requirements since they contain information that can be used to forecast future demand for electricity. The dataset includes statistics on dew, humidity, soil moisture, dryness, and wetness, as well as the accompanying electrical demand. The quantity of dew, humidity, dryness, and wetness will all affect how much electricity is used each day.

	Date	DryBulb	DewPnt	WetBulb	Humidity	ElecPrice	Day	Month	Year	Minutes	SYSLoad
0	01/01/2006 0:30	23.9	21.65	22.40	87.5	19.67	1	1	2006	30	8013.27833
1	01/01/2006 1:00	23.9	21.70	22.40	88.0	18.56	1	1	2006	60	7726.89167
2	01/01/2006 1:30	23.8	21.65	22.35	88.0	19.09	1	1	2006	90	7372.85833
3	01/01/2006 2:00	23.7	21.60	22.30	88.0	17.40	1	1	2006	120	7071.83333
4	01/01/2006 2:30	23.7	21.60	22.30	88.0	17.00	1	1	2006	150	6865.44000
5	01/01/2006 3:00	23.7	21.60	22.30	88.0	17.00	1	1	2006	180	6685.92667
6	01/01/2006 3:30	23.6	21.65	22.30	89.0	17.00	1	1	2006	210	6548.62833
7	01/01/2006 4:00	23.5	21.70	22.30	90.0	16.92	1	1	2006	240	6487.83667
8	01/01/2006 4:30	23.5	21.70	22.30	90.0	15.20	1	1	2006	270	6449.17833
9	01/01/2006 5:00	23.5	21.70	22.30	90.0	14.99	1	1	2006	300	6388.27833

Fig. 4.1 ERCOT Data

	Date	Day	Month	Year	Hours	Temperature	Demand	Forcast J-1	Forcast J
0	01-01-13	1	1	2013	0	6.800	61194	60300	60200
1	01-01-13	1	1	2013	1800	6.710	59674	59500	58700
2	01-01-13	1	1	2013	3600	6.620	57877	57500	56700
3	01-01-13	1	1	2013	5400	6.530	57755	57800	57400
4	01-01-13	1	1	2013	7200	6.440	57243	57600	57200
5	01-01-13	1	1	2013	9000	6.390	56660	56400	55600
6	01-01-13	1	1	2013	10800	6.340	54376	54900	53400
7	01-01-13	1	1	2013	12600	6.305	52640	53200	51600
8	01-01-13	1	1	2013	14400	6.270	50796	51800	49900
9	01-01-13	1	1	2013	16200	6.270	49860	50700	48900

Fig. 4.2 RTE Data

In order to make this data usable we need to perform certain tasks on the data. The first task is feature selection. To eliminate any duplicate variables, feature selection must be conducted on the data, which is a crucial operation. The generalization capacity of the model as well as the overall accuracy of a classifier may both be hampered by these duplicate variables. Additionally, it might complicate the model. We have performed feature selection and reduced the datasets to the data below.

Date	SYSLoad	DryBulb	DewPnt	WetBulb	Humidity	ElecPrice
01/01/2006 0:30	8013.27833	23.9	21.65	22.40	87.5	19.67
01/01/2006 1:00	7726.89167	23.9	21.70	22.40	88.0	18.56
01/01/2006 1:30	7372.85833	23.8	21.65	22.35	88.0	19.09
01/01/2006 2:00	7071.83333	23.7	21.60	22.30	88.0	17.40
01/01/2006 2:30	6865.44000	23.7	21.60	22.30	88.0	17.00

Fig. 4.3 ERCOT after feature selection

Date	Demand	Temperature	Forcast J-1	Forcast J
01-01-13	61194	6.80	60300	60200
01-01-13	59674	6.71	59500	58700
01-01-13	57877	6.62	57500	56700
01-01-13	57755	6.53	57800	57400
01-01-13	57243	6.44	57600	57200

Fig. 4.4 RTE after feature selection

Next we have converted all the values of both the datasets to type float32 to maintain some consistency in them. The essential step after that is to convert values from 0 to 1. In order to properly analyze trends and patterns in the data, we have to transform the above values of the columns because they are at significantly different scales. The data was scaled using the MINMAX Scaling Algorithm. All data are scaled between [0,1] in a MinMax normalization, where [0] represents the least value and [1] represents the maximum value. The Min-Max Scaling Algorithm's formula is as follows:

Xnorm = (X - Xmin) / (Xmax - Xmin)

The MinMaxScaler is imported from the sklearn module. After scaling the data, the values looks like this:

	0	1	2	3	4	5
0	0.286575	0.503741	0.921779	0.836134	0.865591	0.027667
1	0.253941	0.503741	0.923313	0.836134	0.870968	0.027559
2	0.213599	0.501247	0.921779	0.834034	0.870968	0.027610
3	0.179297	0.498753	0.920245	0.831933	0.870968	0.027446

Fig. 4.5 ERCOT After Scaling

	0	1	2	3
0	0.502233	0.283751	0.492260	0.645923
1	0.478164	0.281326	0.479876	0.629828
2	0.449709	0.278901	0.448916	0.608369
3	0.447777	0.276475	0.453560	0.615880

Fig. 4.6 RTE After Scaling

After scaling the data, we have performed another important operation on the given data. We have created a function called create_timeseries_data() to transform the data. The function returns a variable which concatenates the data of time (t-1) with the future value to be predicted at time (t).

id	var1(t-1)	var2(t-1)	var3(t-1)	var4(t-1)	var5(t-1)	var6(t-1)	var1(t)
1	0.286575	0.503741	0.921779	0.836134	0.865591	0.027667	0.253941
2	0.253941	0.503741	0.923313	0.836134	0.870968	0.027559	0.213599
3	0.213599	0.501247	0.921779	0.834034	0.870968	0.027610	0.179297
4	0.179297	0.498753	0.920245	0.831933	0.870968	0.027446	0.155779

Fig. 4.7 Transformed ERCOT

id	var1(t-1)	var2(t-1)	var3(t-1)	var4(t-1)	var1(t)
1	0.502233	0.283751	0.492260	0.645923	0.478164
2	0.478164	0.281326	0.479876	0.629828	0.449709
3	0.449709	0.278901	0.448916	0.608369	0.447777
4	0.447777	0.276475	0.453560	0.615880	0.439669

Fig. 4.8 Transformed RTE

The data was divided into three parts: Train, Test, and Validate. This was the last thing that was done with the data. About 80% of the data for ERCOT (i.e., 70118 rows of the dataset) are present in the training set. The remaining 10% of the data, or the remaining 8764 rows of data, are in the testing set and the remaining 10% are in the validation set. Similar divisions are made in the RTE dataset.

Using the training data and the validation data, the LSTM and GRU models will both be trained. The projected SYSLoad values will then be compared to the test_Y values, which are the actual values, after the model has been evaluated using the test_X.

```
train_X = train_X.reshape((train_X.shape[0], 1, train_X.shape[1]))
val_X = val_X.reshape((val_X.shape[0], 1, val_X.shape[1]))
test_X = test_X.reshape((test_X.shape[0], 1, test_X.shape[1]))
print(train_X.shape, train_y.shape, val_X.shape, val_y.shape, test_X.shape, test_y.shape)
```

```
(70118, 1, 6) (70118,) (8764, 1, 6) (8764,) (8765, 1, 6) (8765,)
```

Fig. 4.9 ERCOT Train, Validate, and Test

```
train_X = train_X.reshape((train_X.shape[0], 1, train_X.shape[1]))
val_X = val_X.reshape((val_X.shape[0], 1, val_X.shape[1]))
test_X = test_X.reshape((test_X.shape[0], 1, test_X.shape[1]))
print(train_X.shape, train_y.shape, val_X.shape, val_y.shape, test_X.shape, test_y.shape)
```

```
(56102, 1, 4) (56102,) (7012, 1, 4) (7012,) (7013, 1, 4) (7013,)
```

Fig. 4.10 RTE Train, Validate, and Test

Now the data was ready to be implemented. Our next requirement is a functioning LSTM and GRU model which can use the data. For creating the AI models we used the Keras python library to implement LSTM and GRU. We built sequential, multi-layered LSTM and GRU modes. The model consists of three LSTM layers, followed by a dense layer that forecasts the future Electricity Load/ Price. Inorder to build an LSTM model, we start by adding a sequential layer. The add method is used to layer additional layers on top of the Sequential Layer after that. The first LSTM layer parameter is the number of neurons or nodes (75 in our case) that we wish to include in the layer. Return_sequences, the second argument, is set to true because the model will eventually include more layers. The number of time steps and the number of indications are the first and last parameters for the input_shape, respectively. We add a Dense Layer last. Since we can only predict one output value, the thick layer's neuron count is set to 1.

Before we can train our LSTM on the training data, we must lastly assemble it. We employed the mean absolute error (MAE) as our loss function, and the Adam optimizer to lower the loss or improve the algorithm.

Our last requirement in this project is to use the models we have created and predict the electric load of the test values. We then have to scale the predicted values back to its original form using the inverse_tranform() function. We can then predict the efficiency of the models.

4.2.1.1. Product Perspective

The final result consists of two AI models that use Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) special types of recurrent neural networks. On the ERCOT and RTE datasets, these models are constructed to carry out time series analysis.

The models are designed to forecast the future electrical demand that farms

would require. The model must also be able to forecast the load with the least degree of inaccuracy.

4.2.1.2. Product Features

The LSTM and GRU models, which were developed using the ERCOT and RTE datasets, have decent accuracy. The project's goal is to deliver results with more precision than what was achieved in the earlier research, ARMA, ARIMA, Simple LSTM, and Complex LSTM have all been evaluated on the same datasets. To surpass that accuracy would be a highlight of our product.

4.2.1.3. User Characteristics

The project’s major goal is to help the farmers in predicting the future electrical load which majorly depends upon the environmental factors like temperature, humidity etc. So, the major users of this project are the farmers. They are the backbone of our society and therefore this project intends to help those who help the world.

4.2.1.4. Assumption & Dependencies

This project is dependent on a few libraries that need to be imported to achieve the goal. We have import libraries like numpy, pandas, matplotlib, sklearn, keras, and tensorflow. Another dependency of the project is the preprocessing of the data. To preprocess the data we first converted all the values to float32 and then using the min-max scaling algorithm, we scaled the values between 0 and 1. Then data is then further processed using a few functions.

In order to predict the accuracy of the models, we need to use the reverse_tranform() function to scale the predicted values to its original form.

This project has no particular assumptions. We have implemented the

models using the data that was provided in the datasets, and trained the models by preprocessing that data.

4.2.1.5. Domain Requirements

Domain requirements are basically characteristic of a defined or say a particular category. In this project we have performed time series analysis using the LSTM model and the GRU model. Time series analysis has already been performed on the above mentioned datasets using ARMA, ARIMA, and LSTM models. The major goal for this project is to create LSTM and GRU models for training the datasets and achieve better accuracies and low error rates. This project's domain basically refers to the agriculture side with its major users being farmers. So with the help of this project users will be able to predict electrical load with great accuracy.

4.2.1.6. User Requirements

This project is intended to solve the difficulties of farmers in predicting the future electrical load. So the major users of this project are the farmers. Since this project majorly focuses on building an AI model so there are no such user requirements required. The project focuses majorly on improving the accuracy of the models by comparing them with the previous ones.

4.2.2. Non Functional Requirements

4.2.2.1. Product Requirements

The product has been created with the help of python libraries and Jupyter Notebook. These two would be required to run the model. We could also run the models as python scripts on our local devices but then we need to make sure that our devices have enough computational power to train the models.

4.2.2.1.1. Efficiency

Two well-known techniques that are used to gauge a forecasting model's accuracy were employed to examine the effectiveness of the models. We employ Mean Absolute Percentage Error (MAPE) and Mean Absolute Error (MAE), respectively.

4.2.2.1.2. Reliability

While testing both LSTM and GRU models, satisfactory results were obtained. This proves that the models created for this study are reliable.

4.2.2.1.3. Portability

The models created by us are portable and they could be used by anyone if they run the python scripts on their device. One could also provide data of similar kind to predict the output.

4.2.2.1.4. Usability

LSTM and GRU models have provided satisfactory accuracy and therefore they are usable.

4.2.2.2. Organization Requirements

4.2.2.2.1. Implementation Requirements

The product has been created with the help of python libraries and Jupyter Notebook. In order to deploy the models created we would require either a jupyter notebook or a google collab notebook with the necessary libraries or we can also deploy the code as python scripts. Libraries like Numpy, Pandas, Keras, Tensorflow and SkLearn are required.

4.2.2.2.2. Engineering Standard Requirements

The engineering standard requirements of the project is to achieve

good efficiency. We aimed to develop a model that is superior to the existing ones. The model's efficiency needs to be higher than the earlier models. To assess the effectiveness of the models, we have utilized Mean Absolute Percentage Error (MAPE) and Mean Absolute Error (MAE) as the two techniques to determine efficiency.

4.2.2.3. Operational Requirements (Describe how the following operational requirement(s) apply to your work.)

Economic: This project is based on predicting the future electrical load. Various environmental factors led to the change in the amount of electricity.For example when there is a high amount of rainfall, then they would require less electricity to run the sprinklers. Similarly in the summer season due to the hot climate, more water would be required for irrigation, therefore more electricity would be required. So this project helps in cutting down the costs of the farmers which can be used for better purposes elsewhere.

Environmental: This project is totally environmental based as there is accurate prediction of the amount of electricity required then it will help the farmers in many ways, such as, it can help in cutting down the costs, it can also save water, and majorly it reduces the wastage of electricity. This would in turn help the farmers improve and increase their yield.

Social: The motivation behind this project is itself a very big social cause as this project intends to help those who are the backbone of our society. Farmers play a vital role in the development of any country irrespective of whether it’s a large country or a small one. They are the key to our survival on this planet as they try their level best to feed everybody. So this project has a very big social impact.

4.2.3. System Requirements

4.2.3.1. H/W Requirements

The following project does not require any special hardware requirements. The whole project was conducted with the help of the Google Collab and Jupyter Notebook applications. The only hardware requirement was a functioning laptop capable of running the above mentioned applications.

4.2.3.2. S/W Requirements

This project had two basic software requirements in order to run the models that were created. These applications were Google Collab Notebook and Jupyter Notebook. Apart from these we used various Python libraries to write the code. Pandas, Numpy, Tensorflow, SKLearn, and Keras libraries were used for this purpose. The Python Pandas package was utilised to work with data sets. It also included a range of tools for viewing, arranging, organizing, and modifying data.

To work with arrays, the Python module NumPy was used. The ndarray NumPy array object provides a number of supporting methods that make using ndarray fairly straightforward.

A potent high-level neural network Application Programming Interface (API), Keras is built on Python.This makes it possible to test deep neural networks quickly, and it helped us successfully implement the LSTM and GRU models.

5. RESULT AND DISCUSSION

After training and testing the models with the datasets we arrived at the following results. Before analyzing the result we had to perform an important step. We had to convert the values which were scaled between 0 to 1, back to its original form. After converting the values back to their original form, we predicted the loads by using both the LSTM and the GRU models for both the datasets. The SYSLoad for the ERCOT dataset is almost similar.

Few of the predicted values for the ERCOT and RTE datasets are shown in the table below.

Table. 5.1 RESULTS FOR ERCOT DATASET

LSTM		**GRU**	
Predicted SYSLoad	**Actual SysLoad**	**Predicted SysLoad**	**Actual SysLoad**
11852.76	11852.76	11852.76	11852.76
11828.277	11818.86	11896.432	11818.86
11798.684	11786.23	11863.876]	11786.23
.	.	.	.
.	.	.	.
8525.036	8413.14	8521.961	8413.14
8427.8	8173.79	8427.353	8173.79
8183.515	8063.36	8189.484	8063.36

Table. 5.2 RESULTS FOR RTE DATASET

LSTM		GRU	
PREDICTED	**ACTUAL**	**PREDICTED**	**ACTUAL**
38790.	38790.	38790.	38790.
40786.035	39908	40961.7	39908
41711.066	40904.	41887.312	40904.
.	.	.	.
.	.	.	.
70693.766	71924.	70890.336	71924.
72164.84	75424.	72312.22	75424.
75483.94	75504.	75688.74	75504.

Using the above predicted values we have plotted a graph for both the datasets. The first graph displays the predicted LSTM values, predicted GRU values and the actual values for the ERCOT dataset. The second graph plots the same values for the RTE dataset. In the graphs below, the predicted LSTM values are displayed with dotted red lines, the predicted GRU values are displayed with dotted green lines, and the actual values are represented with dotted blue lines.

Note: We have printed the graph of the first 100 predicted values

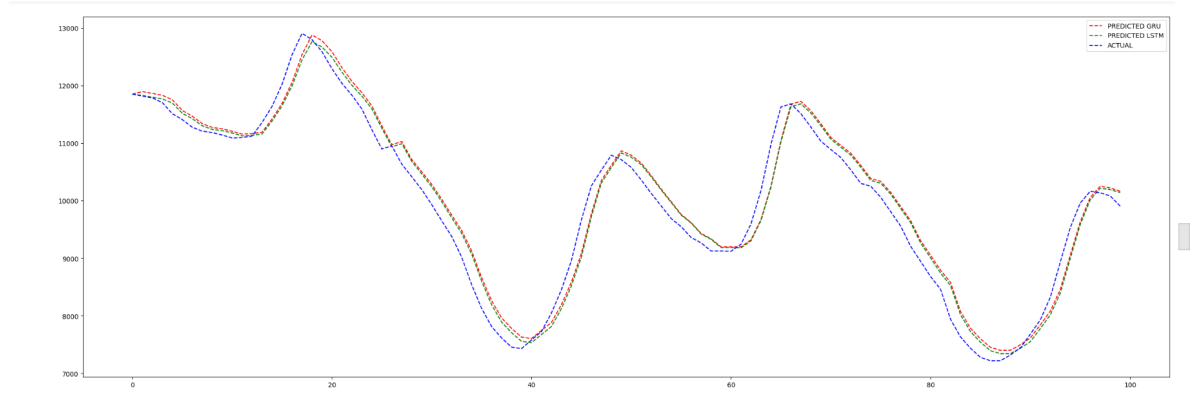

Fig. 5.1 Graph Showing predicted GRU, predicted LSTM along with actual values for ERCOT dataset

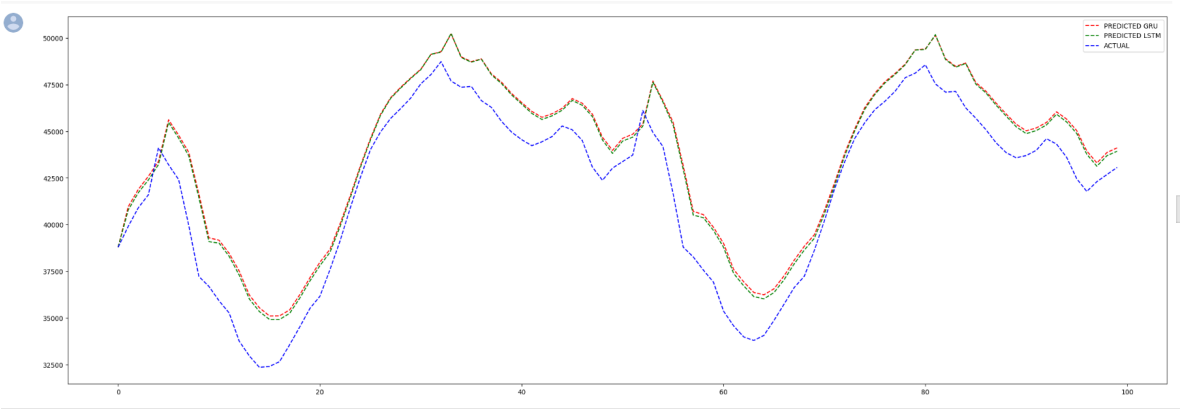

Fig. 5.2 Graph Showing predicted GRU, predicted LSTM along with actual values for RTE dataset.

The Mean Absolute Percentage Error (MAPE) and Mean Absolute Error (MAE) have been used to find the efficiency of the models. Both the models efficiencies have been displayed below along with the efficiencies that were achieved in the previous research with different models.

Note: The efficiency for both the old models, from the previous research and the new models, created by us have been found out by the same algorithms.

Below are the efficiencies for the ERCOT dataset, both, previous research and by us.

Table. 5.3 MAPE and MAE values achieved for the ERCOT DATASET in the previous research

MODEL	MAPE	MAE
ARIMA	9.13	3451
SARIMA	4.36	1638
Simple LSTM	2.63	716.534
Complex LSTM	1.664	229.630

Table. 5.4 MAPE and MAE error values achieved for the ERCOT DATASET by us

MODEL	MAPE	MAE
LSTM	1.511	194.91
GRU	1.535	200.98

The MAPE and MAE values from the above tables indicate that the LSTM and the GRU model have outperformed the previous models.

Among the two, i.e. GRU and LSTM, LSTM has given slightly better efficiency.

Below are the efficiencies for the RTE dataset, both, previous research and by us.

Table. 5.5 MAPE and MAE error values achieved for the RTE DATASET in the previous research

MODEL	MAPE	MAE
Simple LSTM	5.304	660.653
Complex LSTM	5.256	658.651

Table. 5.6 MAPE and MAE error values achieved for the RTE DATASET by us

MODEL	MAPE	MAE
LSTM	1.654	1361.0085
GRU	1.397	1397.5688

The LSTM and GRU models have surpassed the earlier models, according to the above tables' MAPE values, while the earlier research's MAE values for the Simple and Complex LSTM models are superior.

Among the two, LSTM provides slightly better MAE values when compared to GRU, but GRU outperforms the LSTM model when considering the MAPE values.

6. REFERENCES

[1] Devinder Kaur, Rahul Kumar, Neeraj Kumar, and Mohsen Guizani. Smart grid energy management using rnn-lstm: A deep learning-based approach. In 2019 IEEE global communications conference (GLOBECOM), pages 1–6. IEEE, 2019.

[2] G Selvi, S Girirajan, and J Briskilal. Load balancing using lstm network and docker. Int. J. of Aquatic Science, 12(2):2205–2213, 2021.

[3] Lei Fu. Time series-oriented load prediction using deep peephole lstm. In 2020 12th International Conference on Advanced Computational Intelligence (ICACI), pages 86–91. IEEE, 2020.

[4] Aggelos Lazaris and Viktor K Prasanna. An lstm framework for modeling network traffic. In 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pages 19–24. IEEE, 2019.

[5] Rahul Dey and Fathi M Salem. Gate-variants of gated recurrent unit (gru) neural networks. In 2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS), pages 1597–1600. IEEE, 2017.

[6] YW Lee, KG Tay, and YY Choy. Forecasting electricity consumption using time series model. International Journal of Engineering & Technology, 7(4.30):218, 2018

[7] Salah Bouktif, Ali Fiaz, Ali Ouni, and Mohamed Adel Serhani. Optimal deep learning lstm model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches. Energies, 11(7):1636, 2018.

[8] Md Jamal Ahmed Shohan, Md Omar Faruque, and Simon Y Foo. Forecasting of electric load using a hybrid lstm-neural prophet model. Energies, 15(6):2158, 2022.

[9] Zexi Chen, Delong Zhang, Haoran Jiang, Longze Wang, Yongcong Chen, Yang Xiao, Jinxin Liu, Yan Zhang, and Meicheng Li. Load forecasting based on lstm neural network and applicable to loads of “replacement of coal with electricity”. Journal of Electrical Engineering & Technology, 16(5):2333–2342, 2021.

[10] Yu Jin, Honggang Guo, Jianzhou Wang, and Aiyi Song. A hybrid system based on lstm for short-term power load forecasting. Energies, 13(23):6241, 2020.

[11] Gul Muhammad Khan, Atif Rashid Khattak, Faheem Zafari, and Sahibzada Ali Mahmud. Electrical load forecasting using fast learning recurrent neural networks. In The 2013 International Joint Conference on Neural Networks (IJCNN), pages 1–6. IEEE, 2013.

[12] https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[13] https://medium.com/nerd-for-tech/what-is-lstm-peephole-lstm-and-gru-77470d84954b

[14] https://medium.datadriveninvestor.com/multivariate-time-series-using-gated-recurrent-unit-gru-1039099e545a

[15] https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/#:~:text=An%20optimizer%20is%20a%20function,overall%20loss%20and%20improving%20accuracy.

[16] https://stackabuse.com/time-series-analysis-with-lstm-using-pythons-keras-library/

[17] https://www.datarobot.com/blog/introduction-to-loss-functions/#:~:text=Further%20reading-,What's%20a%20loss%20function%3F,ll%20output%20a%20lower%20number.

[18] https://keras.io/guides/

[19] https://machinelearningmastery.com/stacked-long-short-term-memory-networks/#:~:text=LSTMs%20in%20Keras-,Why%20Increase%20Depth%3F,range%20of%20challenging%20prediction%20problems.

[20] https://towardsdatascience.com/time-series-forecasting-with-deep-learning-and-attention-mechanism-2d001fc871fc#:~:text=Advantages%20of%20Recurrent%20Neural%20Network&text=RNNs%20can%20find%20complex%20patterns,be%20dependent%20on%20previous%20ones.