# Effect of Drop-out Layers inside Recurrent Neural Networks in Household Load Forcast Application

Mr. Sanaullah Soomro



A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering in Electrical Engineering
Department of Electrical Engineering
FACULTY OF ENGINEERING
Chulalongkorn University
Academic Year 2022

ผลกระทบของชั้นดรอปเอาต์ภายในโครงข่ายประสาทเทียมแบบวนกลับที่ประยุกต์ใช้พยากรณ์
โหลดบ้านเรือน

นายซานาอุลา ซูมโร



วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2565

| | |
|---|---|
| Thesis Title | Effect of Drop-out Layers inside Recurrent Neural Networks in Household Load Forcast Application |
| By | Mr. Sanaullah Soomro |
| Field of Study | Electrical Engineering |
| Thesis Advisor | Associate Professor WANCHALERM PORA |

Accepted by the FACULTY OF ENGINEERING, Chulalongkorn University in Partial Fulfillment of the Requirement for the Master of Engineering

............................................................... Dean of the FACULTY OF ENGINEERING

(Professor SUPOT TEACHAVORASINSKUN, D.Eng.)

THESIS COMMITTEE

............................................................... Chairman

(Assistant Professor SUREE PUMRIN)

............................................................... Thesis Advisor

(Associate Professor WANCHALERM PORA)

............................................................... External Examiner

(Professor Sanya Mitaim)

การคาดการณ์โหลดไฟฟ้าที่แม่นยำมีความสำคัญอย่างยิ่งในการวางแผนการผลิตไฟฟ้าที่มั่นคง ปลอดภัย และคุ้มค่า ข้อมูลการคาดการณ์โหลดไฟฟ้าระยะยาวและระยะสั้นยังสามารถใช้ในการวางแผนกริดและการพิจารณาอื่นๆ ได้อีกด้วย เมื่อเร็ว ๆ นี้ มีการประยุกต์ใช้เทคนิคการเรียนรู้ของเครื่องกันอย่างแพร่หลาย มีการนำไปใช้สำหรับการพยากรณ์โหลดไฟฟ้าทั้งระยะยาวและระยะสั้น ยกตัวอย่างเช่น Long Short-Term Memory (LSTM) ได้รับการออกแบบเพื่อการวิเคราะห์ข้อมูลอนุกรมเวลาเช่น ข้อมูลโหลดไฟฟ้า โดยเฉพาะ งานวิจัยนี้นำเสนอแบบจำลอง LSTM ที่มีการแทรกชั้นดรอปเอาต์เข้าไป สำหรับคาดการณ์โหลดไฟฟ้าของบ้านเดี่ยวในอีก 20 วันข้างหน้า แบบจำลองที่นำเสนอประกอบด้วยชั้นดรอปเอาต์ที่มีอัตราดรอปเอาต์ที่ 0.2 0.3 0.4 0.5 และ 0.6 ได้ทำการทดลองและวิเคราะห์ผลกระทบต่อการคาดการณ์โหลด ผลลัพธ์แสดงให้เห็นว่ามีการเปลี่ยนแปลงที่ดีขึ้นเล็กน้อยเมื่อทำเพิ่มชั้นดรอปเอาต์ โดยความแม่นยำเปลี่ยนแปลงในทางที่ดีขึ้นประมาณ 1% เท่านั้น อย่างไรก็ตามเนื่องจากยิ่งโมเดลที่มีอัตราดรอปเอาต์มากจะใช้ในกรณีทั่วไปได้ดีกว่า (ไม่ยึดติดกับชุดข้อมูลที่ใช้ในการเรียนรู้) ดังนั้นขอแนะนำให้ใช้อัตราดรอปเอาต์ที่ 0.4



| | | | |
|---|---|---|---|
| สาขาวิชา | วิศวกรรมไฟฟ้า | ลายมือชื่อนิสิต | ............................................... |
| ปีการศึกษา | 2565 | ลายมือชื่อ อ.ที่ปรึกษาหลัก | ............................. |

# # 6470020621 : MAJOR ELECTRICAL ENGINEERING
KEYWORD: LSTM Load forecast Neural Network Dropout Layer

*Ensuring precise power load forecasting is highly important in planning the secure, steady, and cost-effective functioning of the power system. Grid planning and decision-making can be based on accurate long- and short-term power load forecasting. Recently, machine learning techniques have gained widespread adoption for both long- and short-term power load forecasting. Specifically, the Long Short-Term Memory (LSTM) is customized for time series data analysis. This research proposes an LSTM model for forecasting the power load of a single house containing electrical appliances over the next 20 days. We conducted a comparative analysis of the impact of dropout layers in load forecasting applications using the LSTM model. The proposed model comprises dropout rates of 0.2, 0.3, 0.4, 0.5, and 0.6, respectively. Their impact on load forecasting is examined. The experimental results demonstrate slight variations in predictions when altering dropout layers. The results show that the effect of dropout layers on the forecast varies the accuracy by only approximately 1%. However, the models with significant dropout rates are more general than those with lower or higher rates. So the model with a dropout rate of 0.4 is suggested.*



| | | |
|---|---|---|
| Field of Study: | Electrical Engineering | Student's Signature ............................... |
| Academic Year: | 2022 | Advisor's Signature ............................. |

# ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my research supervisor, Prof. Wanchalerm Pora, for his invaluable help, encouragement, and supervision throughout the completion of my thesis. Without his guidance and support, it would have been immensely challenging to accomplish this milestone.

Prof. Pora  has been an exceptionally supportive and cooperative mentor, providing me with the necessary guidance and direction at every stage of my research. I am truly grateful for her extensive experience and knowledge in the field of Engineering, which greatly enriched the quality of my thesis. Moreover, I appreciate his patience and encouragement during challenging times. His ability to provide guidance while allowing me the autonomy to explore and learn has been crucial in my personal and academic growth. I am grateful for his open-door policy, which has allowed me to seek his advice and clarification whenever needed.

I am truly fortunate to have had the opportunity to work with such a dedicated and inspiring supervisor. His mentorship has not only contributed to the success of this research project but has also shaped me as a researcher and a professional.

I am also grateful to my colleagues and lab mates at ESID.

My heartfelt thanks go to my family and friends for their unwavering support, love, and encouragement throughout this journey. Their belief in me and their constant encouragement have been my driving force during moments of self-doubt and exhaustion.

Sanaullah  Soomro

## TABLE OF CONTENTS

# LIST OF TABLES

Page

## LIST OF FIGURES

# CHAPTER 1 INTRODUCTION

## 1.1 INTRODUCTION:

Machine learning, a field of study in Artificial Intelligence used for developing models and methods that enable computers to foresee the future or make judgments based on data without having to be explicitly programmed. It relies on statistical techniques and algorithms to enhance a system's performance on specific tasks by recognizing patterns and relationships in data [1]. Machine learning deals with fundamental objectives that is to create systems that can carry tasks with increasing precision and efficiency as system acquire more data. Machine learning usually deals with concept of data that are capable to learn automatically, determining patterns, and coming to conclusions or making predictions without being explicitly told to do so. A large dataset is frequently used to train a model in the machine learning process so that it can recognize relationships and patterns in the dataset [1] . On the basis of fresh data it comes across, the model can then make predictions or choices. In order to produce predictions or judgments, machine learning primarily aims to employ statistical techniques and algorithms to identify patterns and relationships in data. Raising machine learning based products such as the Netflix recommendation engine, automobile engines, and power predictions utilizing various models have become possible because to recent technological advances in storage and computational power. Computer model's creation that enhance intelligent behaviors like to those of humans is the ultimate goal of AI . This concept were given by Boris Katz, a principal research scientist and director of CASIL Info lab [2]. Machines that are capable of recognizing images, comprehending natural language, and performing physical actions represent the kind of "intelligent behaviors" that researchers aim to produce through AI. Machine learning is one approach to AI. Arthur Samuel first described it in the 1950s as the study of enabling computers to learn without being explicitly programmed. An expert from MIT Sloan and Kensho claims that traditional programming requires the machine to be given specific instructions to follow, which can be time- consuming or impossible in some cases, such as recognizing different people in images. Machine learning addresses this issue by allowing computers to learn through experience. The process begins with collecting and preparing data, such as bank transactions, images, or sales reports, this applied during the training of machine learning model. Quality of the resulting program depends with the amount of data available. [1]

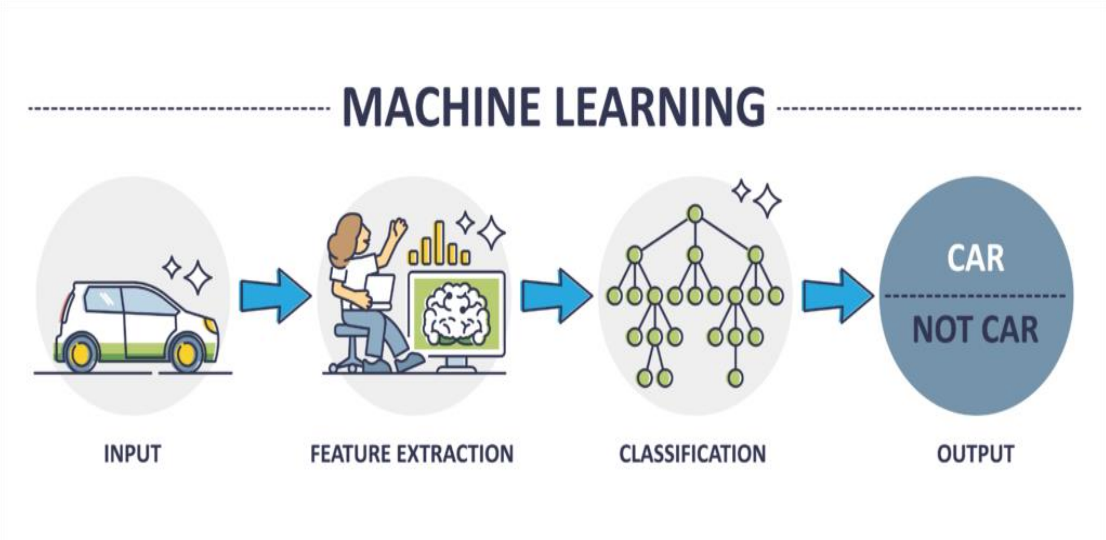



*Figure 1 working structure of machine learning[3]*

Machine Learning is classified into three types:

### 1.1.1 Supervised Machine Learning

For accurate information classification and outcome prediction, supervised machine learning techniques train their algorithms on labeled datasets. The model modifies its weights during cross-validation in order for it to correctly fit the input data. This technique can be employed by organizations to tackle real-world issues like filtering out spam emails and sorting them into a separate folder. To enable the model to generate the required output, a training set comprising input and output data is utilized, the algorithm continues modifying until the error is minimized while monitoring its accuracy through the loss function. The first step in supervised learning is to obtain labeled training data, which serves as feedback for the algorithm, and then dividing it into three sets for validation, testing, and training. [3, 4]

Model's enhancement and lower error, the set of training has been used. For example, a model trained to recognize animal pictures may use a training set containing labeled animal pictures to compare predicted labels with the correct labels. The validation set, a distinct dataset from the training set, is used to evaluate the model's performance to get the effectiveness of learning methods. This evaluation helps to identify the best time to halt training the algorithm to balance model accuracy against overfitting risks. The test set is the final dataset utilized to evaluate the model's ability to handle new, unseen data after optimization using the validation set. This assessment represents a "real-world" scenario that tests the model's ability to generalize. Basically, supervised learning is passing labeled data into a model for training. Input and output pair defines the data labeled that used during the training of model. It can be likened to a teacher who grades a student's response to a question based on its correctness. [3, 4]

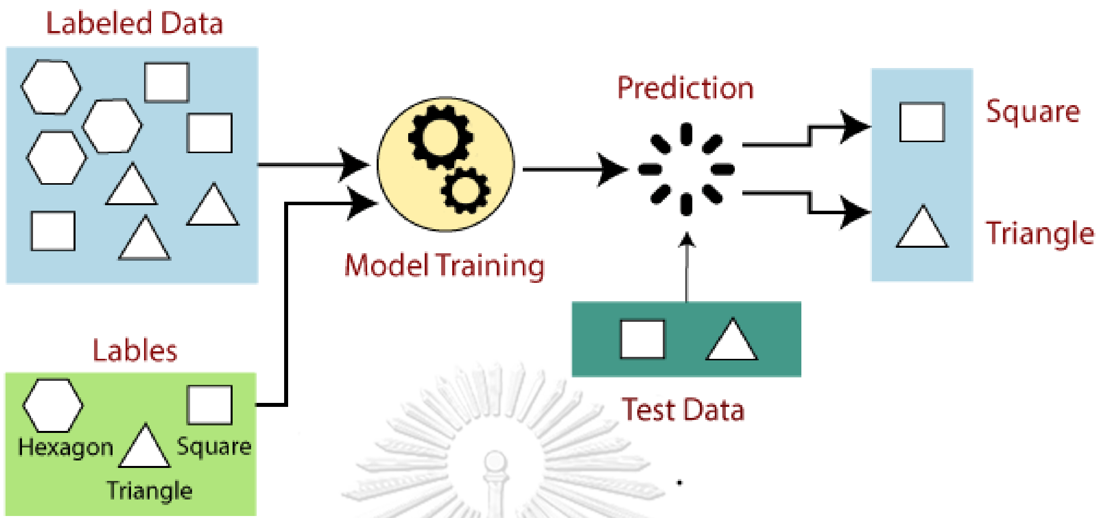



*Figure 2 Structure of supervised machine learning [24]*

### 1.1.2 Unsupervised Machine Learning

An area of artificial intelligence and data science called unsupervised machine learning includes training models using unlabeled data. Unsupervised learning differs from supervised learning in that there are no predetermined target values or class labels. In unsupervised learning, the algorithms aim to discover patterns, structures, and relationships within the data without any specific guidance. The models analyze the input data and identify inherent similarities or groupings, often referred to as clusters. By doing so, they uncover hidden insights or knowledge from the data. Clustering, dimensionality reduction, and association rule mining are common methods for unsupervised learning. Based on their similarities or proximity in the feature space, clustering algorithms group data points together. By translating the data to a lower-dimensional representation while keeping the important information, dimensionality reduction techniques try to make the data less complex. Association rule mining discovers relationships or patterns among items or attributes in the data. Numerous uses for unsupervised learning include picture identification, market basket analysis, anomaly detection, and the customer segmentation. When the data is unstructured or missing analyzed examples, it is especially helpful and can offer valuable insight for subsequent analysis or decision-making. It's crucial to remember that unlike supervised learning, unsupervised learning does not offer clear solutions or forecasts. Instead, it focuses on discovering patterns and structures in the data, allowing humans to interpret and extract meaningful information from the results.[4, 5]

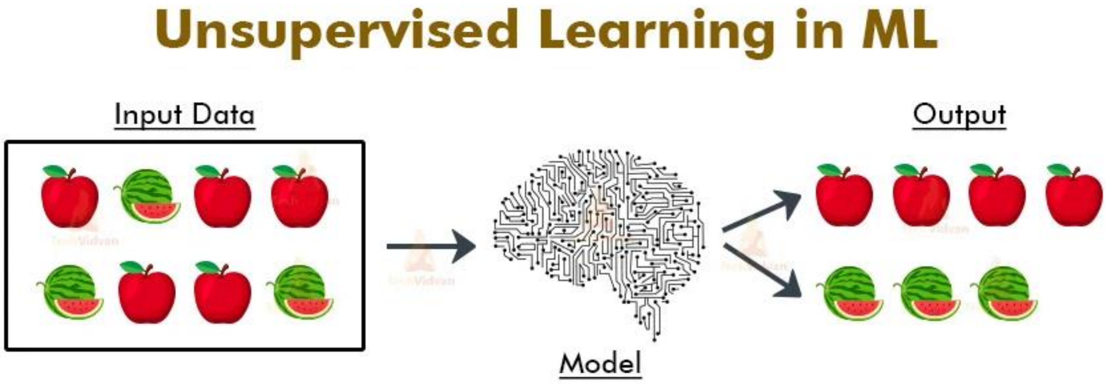



*Figure 3 Unsupervised Machine Learning [25]*

### 1.1.3 Reinforcement Machine Learning

Finding the appropriate course of actions to take in a circumstance in order to maximize reward is the major focus of the machine learning subfield known as reinforcement learning (RL). In contrast to supervised learning which trains the model using labeled data, RL agents have no predefined answer key and must learn through experience. RL is centered around the science of decision making and works by accumulating data from trial-and-error methods. It uses algorithms that learn from results to decide the best set of action to follow next rather than relying on input data like supervised or unsupervised learning. The algorithm receives feedback after each step that allows it to determine if its choice was good, poor, or neutral. For autonomous systems that must make countless small decisions without human interaction, this strategy is especially helpful. With the goal of achieving the greatest results, reinforcement learning is a self-teaching method that learns by doing. [1, 4]

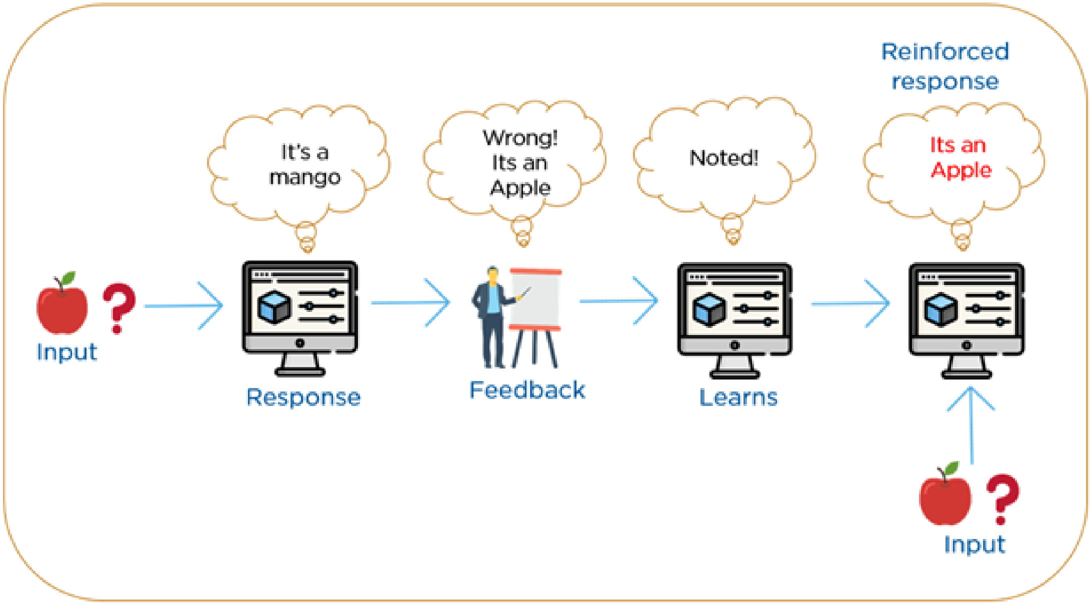



*Figure 4 Reinforcement Machine Learning [22]*

## 1.2 Objectives

1. Develop a deep learning algorithm for household energy prediction.
2. Propose a RNN model for unseen data-points forecasting.
3. Different dropout layers comparison to analyze the effects in forecasted long term unseen values.

## 1.3 Scope of work

1. A dataset containing 47 months of energy usages of household with sampling rate of 1 minute have been used.
2. Comparing the accuracy of different models to get the more accurate results.

### 1.4 Deep Learning

The primary subfield of machine learning known as deep learning (DL) uses methods inspired by the neural network architecture of the human brain. This field has gained significant attention due to its potential for advancing artificial intelligence. While traditional machine learning algorithms use data processing to generate predictions, DL attempts to emulate the way the human brain clusters information to make highly accurate predictions. As a result, DL has generated a lot of buzz about the future of machine learning, as it represents a new frontier in the field. Also called neural organized learning, artificial neural networks are trained for DL using vast quantities of data. [26]. DL algorithms repeat tasks, constantly adjusting and improving their predictions. The effectiveness of these algorithms relies on the massive amounts of data that are generated each day, estimated to be around 1.145 trillion MB [1]. The increase in the amount of data being generated has played a significant role in the development of DL. Although it may sound mysterious, many of us are already encountering DL in our everyday lives. DL networks, also called deep neural networks, comprise numerous hidden layers with millions of interconnected artificial neurons [6]. These neurons are connected via weights, which are numerical values that determine the strength of the connection between nodes. A weight can be either positive or negative, indicating whether one node stimulates or inhibits another. Nodes with higher weight values have a significant influence or weight on other nodes. However, these networks require significantly more training compared to other machine learning techniques. To train effectively, deep neural networks need millions of examples of training data, whereas simpler networks may only require hundreds or thousands. [3, 4]

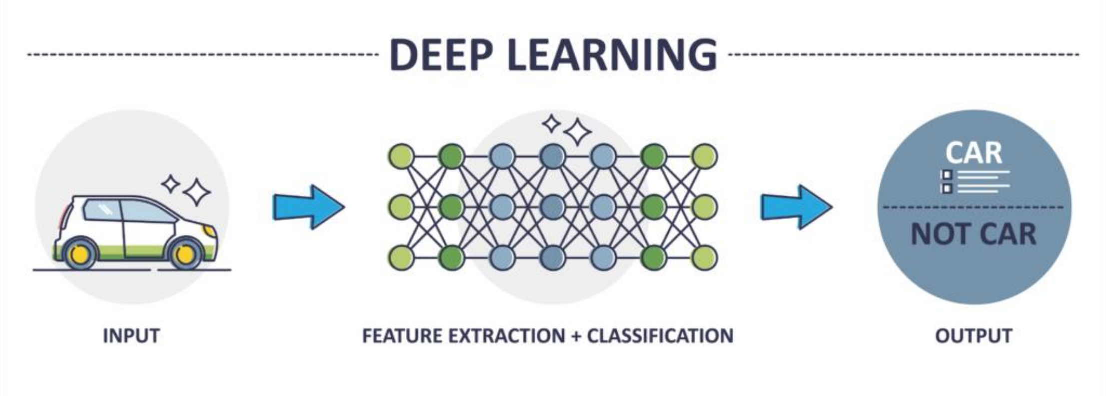



*Figure 5 Deep learning [23]*

## 1.5 Neural Networks

Artificial intelligence composed of neural networks that are created to analyze the data in such a way as human can do. Neural networks are playing a vital role in the field machine learning, in which interconnected neurons are arranged likewise of human brains layered in a structure. Neural networks develop an adaptive system that can solve complicated problems with increased accuracy, such as identifying faces or summarizing texts, by learning from their failures and continuously improving [6]. Neural networks have the potential to enable computers to make intelligent decisions without much human intervention, as neural networks are capable of modeling and learning the complex relationships between output and input data that are linear

[23]. Neural networks take their design features from the human brain, where neurons form a highly interconnected network and exchange electrical signals with one another to process information [7]. Similar to natural neural networks, artificial neural networks contain synthetic neurons that interact to solve issues by utilizing computing systems to carry out complex mathematical calculations [3].

There are several uses for neural networks across different industries, including the ones listed below [7]:

1. Medical image classification for diagnosis.
2. Targeted advertising using behavioral data analysis and social network filtering.
3. Financial forecasts using past financial instrument data.
4. Forecasting energy demand and electrical load.
5. Process and quality control.
6. Identification of chemical compound.

A basic neural network has interconnected artificial neurons in three layers.

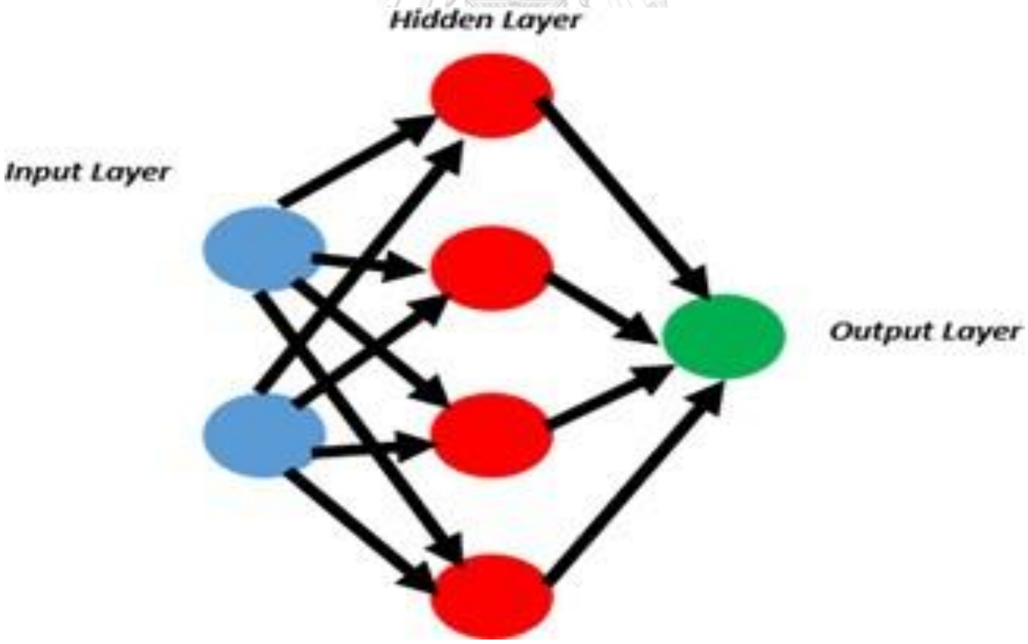



*Figure 5 Structure of layers inside neural network*

### 1.5.1 Input Layer

With its input layer, an artificial neural network takes in data from the external environment. Input nodes process the data by examining and classifying it before sending it to the next layer. [8][7] .

### 1.5.2 Hidden Layer

Data for the input layer is received from other hidden layers or from the input layer itself, which also provides input to other hidden levels. Neural networks could have a lot of hidden layers. Prior to being passed on to the next layer, each hidden layer evaluates and improves the output of the previous layer.[3, 8]

### 1.5.3 Output Layer

An artificial neural network's output layer has the task of delivering the eventual result of the network's data processing. Depending on the problem being solved, output layer's node number is flexible. For instance, the output layer of a binary classification issue would simply contain one node that generates a final value of 1 or 0. In contrast, for a multi-class classification task, a large number of nodes would be required in the output layer.  Figure 6 below depicts the fundamental layout of a neural network [7].

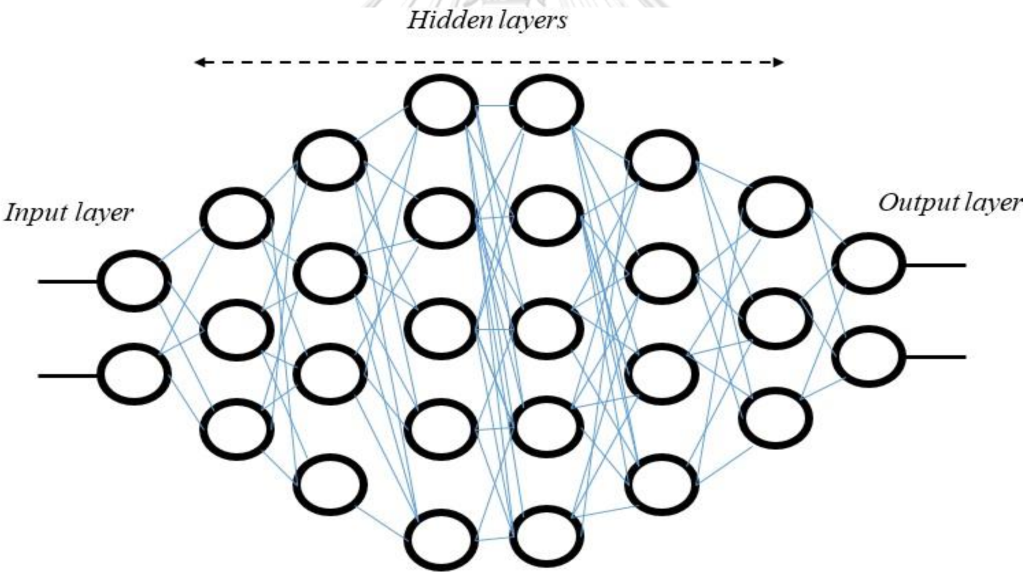



*Figure  6 Basic structure of neural network*

## 1.6 Artificial Neural Network (ANN)

ANN has been motived by the human neural network containing neurons connected together in a way like human brain neurons. It's a network of interconnected artificial neurons, organised in layers and frequently referred to as nodes or units. When using an ANN, data is sent from the input layer, where it is received, to the output layer, where predictions or judgments are formed. The network can learn from the input data and extract representations using hidden layers, that are the input and output layers layers. The network's artificial neurons all take in inputs, process them, and then produce outputs that go to layers below then. By multiplying inputs by corresponding weights and applying an activation function, each neuron performs computations that introduce non-linearity and enable the network to simulate complicated interactions between inputs and outputs. The neural network learns to modify the weights of its connections during training in order to maximize performance on a particular task. Backpropagation is a technique used to accomplish this, in which the network's predictions are compared to the expected outputs, and the mistake is then transmitted backward to adjust the network's weight through. Until network performance's sufficient, this iterative procedure is continued. ANNs are effective for a variety of machine learning applications because of their reputation for learning from enormous amounts of data and generalizing well to new cases. Many different domains, such as time series analysis, systems for recommendation, natural language processing, image and audio recognition, and many more, have effectively utilised them.[7, 9]

Depending on the goal and the data at hand, artificial neural networks' architecture, size, and complexity can change. Convolutional neural networks (CNNs) for image data, feed-forward neural networks (NFNs), and recurrent neural networks (RNNs) for sequential data and more sophisticated architectures like generative adversarial networks (GANs) and transformers are just a few of the various types of neural network architectures that have been developed over time.

Overall, artificial neural networks are powerful and flexible models that can learn and generalize from data, enabling them to solve complex problems across various domains. [8 & 37]



### 1.6.1 Input layer

Input layer is the primary component of neural network, which is in charge of gathering input values for each observation's explanatory factors. Typically, the input layer comprises an equal number of input nodes and explanatory variables. Network communicates with one or more hidden layers after receiving patterns from the input layer.[8][9]

The input layer's nodes do not change the data that is put into them, unlike the nodes in the hidden layer that can modify it. The input nodes receive inputs from the external environment, while the hidden nodes have a uniform input that gets replicated across their outputs. Duplicate copies of each input value are sent to all hidden nodes. [8][8][9]

### 1.6.2 Hidden layer

The hidden layers within a neural network are designed to carry out distinct operations on the input data. Each hidden node is linked to incoming connections from input or other hidden nodes, as well as outgoing connections to output nodes or other hidden nodes. Processing that occurs in the hidden layer is controlled by a series of weighted connections. Hidden node contains input values containing connections to determine the output values. Most neural networks feature one or more hidden layers, and each one manipulates the data in a unique way [8].[9]

### 1.6.3 Output layer

The hidden layers transform the input values before connecting them to the "output layer." This layer is responsible for producing an output value that corresponds to the desired response variable, and it is connected to the hidden layers or input layer. Frequently, the output layer for classification problems has simply one node. The output layer's active nodes use a combination of data manipulation techniques to generate the desired output values. A neural network's ability to handle data depends on choosing the right weights, which makes it different from other ways of processing information. [9]

## 1.7 Computational Neural Network

A particular type of neural network used in computer vision and image recognition applications is called a convolutional neural network (CNN). A CNN's architecture is based after how the human visual system functions, and it processes incoming data similarly. A number of interconnected layers of neurons make up the CNN, which uses mathematical operations on the input data to find features and classify it. The first layer of a CNN is a convolutional layer, which employs filters to scan the input image and locate specific features like edges or corners. In order to add non-linearities non-linear activation function used after passing through, such as rectified linear unit (ReLU), convolutional layers output is processed. [3, 10]

Following the initial convolutional layer, CNN usually have additional layers, including pooling layers that decrease the spatial size of the input data and fully connected layers that classify the input based on features identified in earlier layers. To reduce the disparity between the expected and actual output, backpropagation is used during training to adjust the neuron weights in the network. CNNs have proven to be remarkably effective at a variety of image identification tasks, including but not limited to object detection, segmentation, and facial recognition [3, 10]

A special kind of deep learning network have been introduced for processing and analyzing organized grid-like input such as photographs or time series analysis and CNN. Computer vision mainly deals with image classification and picture segmentation, it has demonstrated remarkably good performance. CNNs are distinguished by their capacity to autonomously learn hierarchical data representations by utilizing convolutional and pooling layers. The key components of a CNN are as follows:

1. Convolutional Layers: These layers execute convolutions on the input data by applying a set of learnable filters (also known as kernels). Convolutional operations take the input and extract regional patterns or features that capture spatial relationships. A feature map that depicts the presence of specific features in the input is the output of a convolutional layer.[10]
2. Pooling Layers: By condensing or lowering the feature maps' spatial dimensions, these layers downsample the data. The most popular pooling operation, known as max pooling, pulls the most value possible from a given area. Pooling facilitates the creation of translation-invariant features, controls overfitting, and reduces the dimensionality of the data.[10]
3. Fully Connected Layers: Similar to conventional artificial neural networks, these layers connect all of the neurons from the previous layer to the following layer. Final predictions or outputs are provided by fully linked layers, which aid in the learning of complicated, non-linear correlations in the retrieved characteristics.[10]



Stacks of convolutional layers are frequently used to create CNNs, more than one layers are added. The weights of the network are trained by modifying its parameters based on the disparity between the expected outputs and the true labels. The hierarchical and local receptive field nature of CNNs enables them to automatically learn relevant features at different levels of abstraction. Because of this capability, CNNs are highly suited for jobs involving organized grid-like data, where local patterns and spatial correlations are crucial, including image processing and analysis. Convolutional neural networks have attained cutting-edge performance on numerous computer vision tasks and have been widely adopted in many applications, including image recognition, object detection, image generation, and even natural language processing tasks that involve text representations as images.[1, 3, 10]

# CHAPTER 2 LITERATURE REVIEW

The paper "Application of Long Short-Term Memory (LSTM) Neural Network based on Deep Learning for Forecasting Electricity Energy Consumption" (Application of Long Short-Term Memory (LSTM) Neural Network based on Deep Learning for Forecasting Electricity Energy Consumption) examines LSTM neural networks. The introduction to the paper emphasizes the value of precise forecasting of power use as well as the shortcomings of conventional forecasting techniques. The summary of deep learning and LSTM networks that follows highlights their capacity to recognize intricate patterns in time series data. The methods taken to conduct the literature review is described in the methodology section, along with the search strategy and selection criteria. The review of LSTM-based electricity energy consumption forecasting studies covers topics such as datasets, preprocessing techniques, network architectures, and evaluation metrics used in the reviewed studies. The strengths and limitations of the reviewed studies are discussed. The LSTM-based approach is compared with other traditional forecasting techniques commonly used in electricity consumption forecasting, such as ARIMA, exponential smoothing, Support Vector Regression, and Random Forest. The explanations and summaries of the evaluation measures used to rate the performance of LSTM models, such as MAE, RMSE, and MAPE. The challenges of LSTM-based electricity energy consumption forecasting are discussed, including hyperparameter selection, handling missing data, and model interpretability. The paper makes suggestions for potential future research areas to increase the precision of energy consumption predictions, including ensemble approaches, attention mechanisms, and transfer learning. The paper defines the literature review that emphasizes the usefulness of deep learning models based on LSTM for forecasting electricity energy usage. The advantages of LSTM networks in capturing temporal dependencies and handling nonlinear patterns are emphasized. The review calls for further research and development in this area to address the challenges and enhance the accuracy of energy consumption forecasting.[3]



The application of deep learning and ensemble methods for forecasting household energy demand in smart homes is the primary subject of the study "Deep Learning based Ensemble Method for Household Energy Demand Forecasting of Smart Home" by Saidur Rahman, Md. Golam Rabiul Alam, and Md. Mahbubur Rahman. The article discusses the shortcomings of conventional forecasting techniques and suggests an ensemble strategy based on deep learning to increase accuracy. The preface to the review emphasizes the significance of precise energy demand forecasting in smart homes. It analyzes how complex patterns and linkages in data on energy use could be captured using deep learning approaches, such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), and long short-term memory (LSTM) networks.. The authors explore ensemble methods, which combine multiple models to enhance predictions, and their advantages in improving forecasting accuracy and reducing prediction errors. They review relevant studies that have employed ensemble methods for energy demand forecasting and discuss the performance improvements achieved through these techniques. Examined are the difficulties and factors to be taken into account when using group deep learning techniques to predict household energy consumption. The authors discuss topics such model choice, training and optimization techniques, data preparation, and model assessment. The study's conclusion looked at the possibility of deep learning-based ensemble approaches for precise forecasting of household energy demand in smart homes. The authors advise more study to concentrate on creating hybrid ensemble models that mix several deep learning architectures to take use of their advantages. In order to improve forecasting performance, they also suggest looking at attention and transfer learning mechanisms. The work provides insights into the use of deep learning and ensemble methodologies for improving energy demand forecasting in smart homes and emphasizes the need for additional research in this area. In their work titled "Limitation of Deep-Learning Algorithm for Prediction of Power Consumption," Majdi Frikha, Khaled Taouil, Ahmed Fakhfakh, and Faouzi Derbel examine the drawbacks of deep-learning algorithms. The article focuses on a number of difficulties and limitations deep-learning models encounter in this field. The article starts out by highlighting the value of precise power consumption forecasting and the growing application of deep-learning algorithms for this purpose. It demonstrates how deep learning may be used to identify intricate patterns and nonlinear correlations in data on electricity use. The authors then delve into the limitations of deep-learning algorithms. They discuss issues related to data availability, including the requirement for large and diverse datasets, data quality, and addressing missing values. The review also addresses the challenges associated with model complexity, such as overfitting, and the difficulty of interpreting the learned representations of deep-learning models. Furthermore, the review explores the computational requirements of deep-learning algorithms, including the need for powerful hardware and significant computational resources. The authors discuss potential scalability issues and the trade-off between model complexity and computational efficiency. Additionally, the review highlights challenges related to hyperparameter tuning and model selection in deep-learning algorithms. The authors discuss the sensitivity of deep-learning models to hyperparameter settings and the potential risks of overfitting or underfitting if not appropriately addressed. In conclusion, the paper underscores the limitations of deep-learning algorithms for power consumption prediction. The authors suggest the need for further research to address these limitations, including the development of more interpretable models, improved handling of missing data, enhanced scalability, and refined hyperparameter optimization techniques.[3]

Deep recurrent neural networks (RNNs) are used by the authors of the study "Predicting Electricity Consumption for Commercial and Residential Buildings Using Deep Recurrent Neural Networks" to predict power consumption in commercial and residential buildings. The article investigates how collecting temporal correlations in data on electricity use can help deep learning algorithms increase forecast accuracy. The review begins by highlighting the importance of accurate electricity consumption prediction for energy management and sustainability. It discusses the limitations of traditional forecasting methods and introduces deep RNNs as a promising approach for handling sequential data and capturing complex patterns. The authors examine many research that used deep RNNs to forecast electricity usage in office and residential buildings. The datasets used, preprocessing methods, model architectures, and evaluation criteria applied in these studies are all examined by the authors. The study emphasizes how deep RNNs are useful for modeling nonlinear interactions and identifying long-term dependencies in data on power consumption. Additionally, the review discusses the challenges associated with applying deep RNNs for electricity consumption prediction. The challenges examined include model selection, hyperparameter tuning, managing missing data, and the interpretability of deep RNNs. Three performance assessment metrics—mean absolute error (MAE), root mean square error (RMSE), and mean absolute percentage error (MAPE)—are used to examine the deep RNN models' accuracy in predicting power usage. The use of these metrics in the papers under consideration is outlined. The literature analysis underlines the efficiency of deep RNNs for predicting power usage in both commercial and residential buildings in its conclusion. The authors suggest further research to improve model interpretability, address challenges related to missing data, and explore ensemble methods to enhance forecasting accuracy.[3]

Ana Karen Apolo Pealoza, Alexandre Balbinot, and Roberto Chouhy Leborgne's study "Review of Deep Learning Application for Short-Term Household Load Forecasting" analyzes the use of deep learning methods for precise short-term load forecasting in household settings. Review provides an overview of existing literature and discusses the advantages and challenges associated with applying deep learning models in load forecasting. The review begins by emphasizing the significance of short-term load forecasting in optimizing energy distribution and resource management. It draws attention to the drawbacks of conventional forecasting techniques and proposes deep learning as a potentially effective way for identifying intricate patterns and temporal relationships in home load data. Recurrent neural networks (RNNs), convolutional neural networks (CNNs), and long short-term memory (LSTM) networks are the three types of neural networks used in short-term load forecasting that are explored by the authors. Authors also talk about these models' advantages and disadvantages as well as their applicability for simulating household load data. The paper discusses a number of deep learning applications in load forecasting, including feature selection, model training, and assessment measures. The authors discuss the challenges involved in implementing deep learning models for accurate load forecasting, such as hyperparameter tuning and model selection. Common performance evaluation approaches, such as mean absolute error (MAE), root mean square error (RMSE), and mean absolute percentage error (MAPE), are frequently used to evaluate the accuracy of load forecasting models. It is described how these measures were used in the studies under consideration. In order to illustrate the potential of deep learning techniques for forecasting short-term home loads, the research report concludes. The authors suggest further research to address challenges related to data availability, model interpretability, and scalability in deep learning-based load forecasting models. They propose exploring hybrid models and incorporating external factors, such as weather data, to improve forecasting accuracy.[3]

RNN-LSTM and ARIMA models are used in a paper by M. M. Sachin et al. titled "Analysis of Energy Consumption Using RNN-LSTM and ARIMA Model" to analyze energy consumption trends. The study explores the need for accurate energy consumption analysis and compares the efficacy of several models at forecasting and understanding energy use. The review begins by emphasizing the importance of energy consumption analysis for various purposes, such as energy management and conservation. It highlights the role of accurate energy consumption prediction in optimizing resource allocation and promoting sustainability. The authors discuss the RNN-LSTM and ARIMA models as popular approaches for energy consumption analysis. They provided more light on each model's benefits and drawbacks, particularly in terms of their ability to spot temporal correlations and trends in data on energy use. Comparisons of the performance of RNN-LSTM and ARIMA models in various studies are discussed in the review. The authors examine the datasets used, model training techniques, and evaluation metrics employed to assess the accuracy of energy consumption predictions. The advantages of deep learning models, specifically RNN-LSTM, in capturing complex patterns and nonlinear relationships are highlighted. Various aspects of energy consumption analysis, including data preprocessing, feature selection, and model parameter optimization, are covered. The review discusses the challenges associated with these tasks and provides recommendations for effective solutions. In conclusion, the literature review underscores the effectiveness of RNN-LSTM and ARIMA models in analyzing energy consumption. The authors suggest further research to enhance the interpretability of deep learning models, improve forecasting accuracy, and explore hybrid models that combine the strengths of both approaches.[11]

# CHAPTER 3 METHODOLOGY

## 3.1 Recurrent Neural Network

Recurrent neural networks (RNNs), a more advanced type of neural network, have connections between nodes that are organized in a temporal order. RNNs are ideal for processing sequential input, including time-series data, voice, and speech, due to the network's capacity to exhibit dynamic temporal behavior as a result of its distinctive structure [21]. In contrast to typical neural networks, an RNN's output from each node depends on both the current input and the result from the previous node, forming a kind of "memory" within the network. This feature allows the network to consider the context of previous inputs when processing current inputs. [10][12]

Speech recognition, natural language processing, and image captioning are famous of the many applications where RNNs have succeeded. RNN's are suitable for tasks such as classification, regression, and sequence prediction. RNNs work by processing each input in a sequence and passing its output as a hidden state to the next input. This way, the network can retain the context of the previous inputs and utilize it to generate outputs or make predictions. When the input consists of a list of words or phonemes, RNNs are particularly effective at tasks like language modeling, speech recognition, and machine translation [10] [12].

### 3.1.1 Recurrent Neural Network using Long-Short Term Memory (LSTM) cells

To address the limitations of conventional RNNs in capturing long-term dependencies in sequential input, the recurrent neural network (RNN) architecture known as LSTM was developed. Positions involving time series data, speech recognition, and natural language processing fall under its area of expertise. Key characteristic of LSTM is its ability to remember and selectively retain or forget information over long time intervals. It achieves this by using memory cells, which are responsible for maintaining and updating information over time. To regulate the information flow, these memory cells have gates installed, including input, forget, and output gates. In contrast to the forget gate, which determines whether information needs to be deleted, the input gate determines how much fresh data should be stored in the memory cell. The output gate controls the amount of information that should be sent from the memory cell to the layer or time step below. These memory cells and gates enable LSTM networks to learn and recognize dependencies in the input data across a wide time range. They are therefore useful for tasks requiring the modeling of long-term dependencies or working with a range of length sequences. LSTM networks have found wide applications in various domains, including natural language processing (e.g., language translation and sentiment analysis), speech recognition, time series analysis, and more. They have proven to be highly successful in capturing and leveraging temporal information, making them a valuable tool in deep learning. [3, 12]

The well-known LSTM algorithm for estimating household energy use calls for an RNN with a self-loop based on sequence models. LSTM are frequently used when predicting power using time series data. The sequences' order is crucial since the current step could change the previous one, which could change the series forecast further. The time-stamp movement's input, output, and hidden units are depicted in Fig. 8. The timestamp serves as a representation of the position of sequences. The output forecasts the future value of the long-term household prediction.

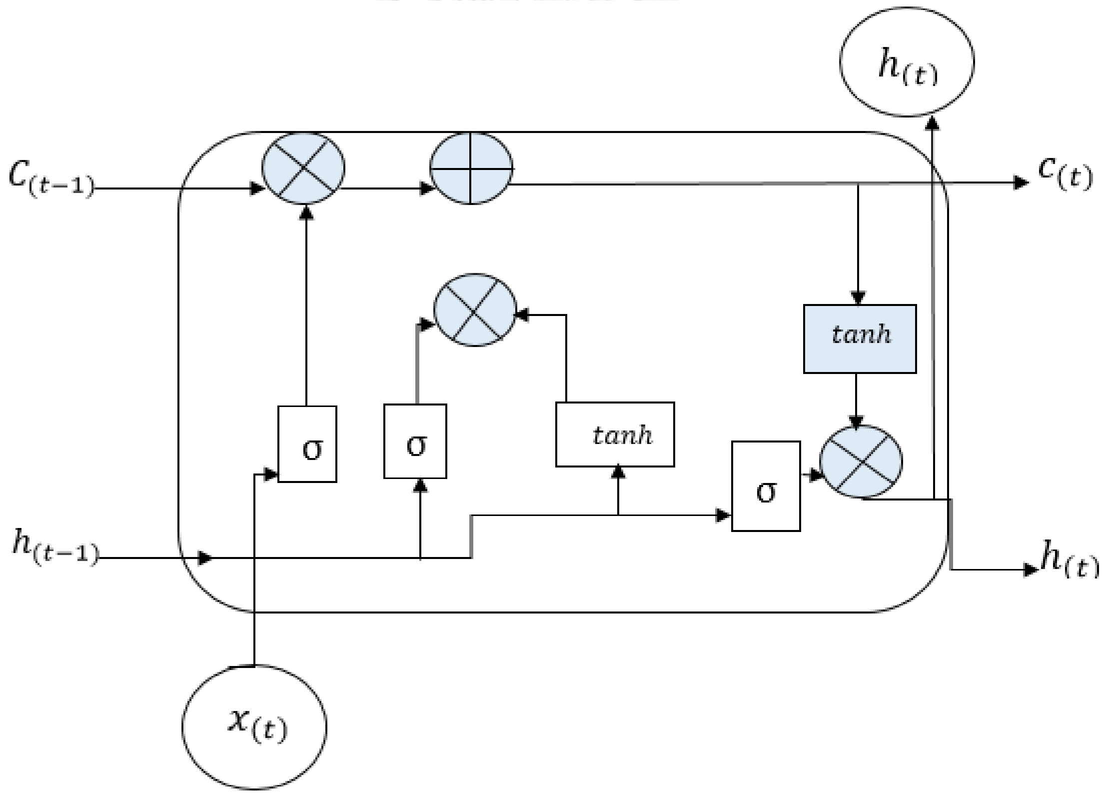



*Figure 7 Structure of LSTM [13]*

A cell state called $C_t$, a form of long-term memory, is the key to LSTM. According to the following equation (1), this cell determines whether elements of input sequences can be left or eliminated by the preceding time steps[13]. In (1), the forget gate ($f_t$) is multiplied by the previous cell state ($C_{t-1}$) to produce the reset, and this cell also determines what data is inserted by multiplying the input gate ($i_t$) by the activation input ($g_t$) [13]. The equation makes use of the hidden state ($h_t$) (2). The output gate ($o_t$) is multiplied by the hyperbolic tangent *of* $c_t$ to update the value of $h_t$. According to (4), (5), and (6), the $f_t$ and $o_t$ are derived using the sigmoid activation function ($\sigma$), bias ($b_f, b_i, b_o$), and matrix of Weights ($W_{fx,fh}$, $W_{ix,ih}$, *and* $W_{ox,oh}$) according to (4),(5),(6). According to (3), the $g_t$ is obtained using the hyperbolic tangent activation function ($tanh$), bias ($b_g$), and matrix of Weights ($W_{gx,gh}$) according to (3). [13]

$$c_t = g_t . i_t + (c_{t-1}^0 . f_t) \qquad (1)$$
$$h_t = tanh(c_t^0) . o_t \qquad (2)$$
$$g_t = tanh\left(W_{g.x}x_t + W_{g.x}h_{t-1} + b_g\right) \qquad (3)$$
$$f_t = \sigma\left(W_{f.x}x_t + W_{f.h}h_{t-1} + b_f\right) \qquad (4)$$
$$i_t = \sigma(W_{i.x}x_t + W_{i.h}h_{t-1} + b_i) \qquad (5)$$
$$o_t = \sigma(W_{o.x}x_t + W_{o.h}h_{t-1} + b_0) \qquad (6)$$

The time-step is represented by an LSTM cell (see Fig. 7). This image also shows the connections between the LSTM cells. The time-step is followed by an update to the $h_t$ *and* $c_t$. Fig. 7 shows the time-step with an LSTM cell [13], and the weights matrix is updated simultaneously but not at each time-step but rather at the end of the mini batch. This graphic also shows the internal connections between the LSTM cells $h_t$ and $c_t$ are altered after the time-step. The weights matrix is altered concurrently, but only at the end of the mini batch; not at each individual time-step [13].

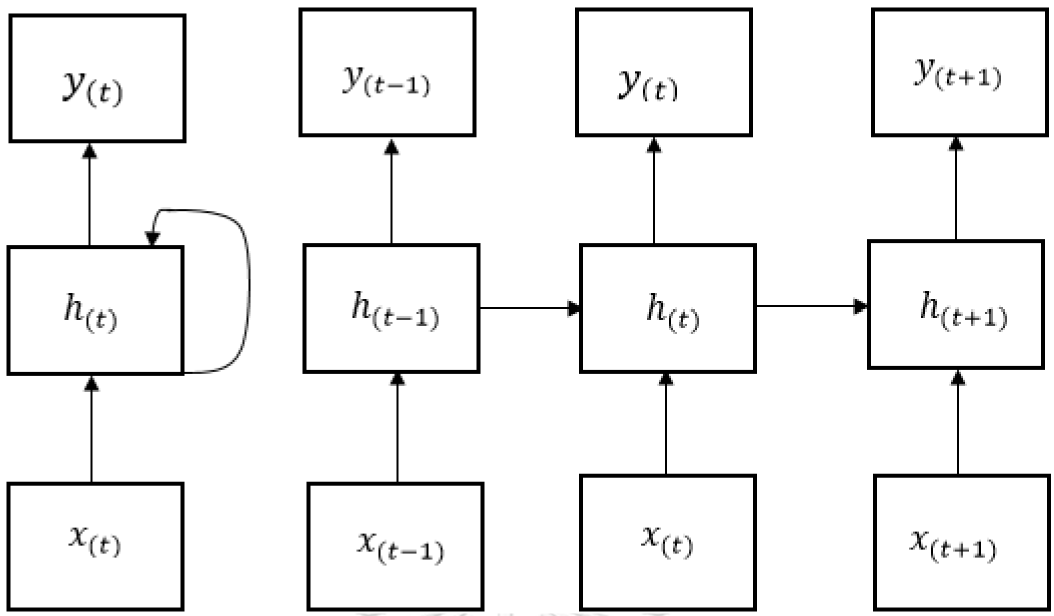



*Figure 8 Structure of RNN[13]*

## 3.2 Dataset

In machine learning, a dataset is a collection of data samples used to teach and test machine learning programs. It is an important part of developing and evaluating machine learning models. Typically, a dataset contains input data points and corresponding output values, which are used to train a model to recognize the underlying patterns or relationships between inputs and outputs. Datasets can be categorized as either structured or unstructured, depending on the nature of the data, and may originate from a variety of sources, including sensors, surveys, or databases. Before being used for machine learning tasks, a dataset may need to be pre-processed or cleaned to eliminate noise, fill in missing values, or standardize the data. A well-designed dataset is crucial for successful machine learning, as it may significantly affect the final models' accuracy and generalizability.[13]

The Individual Household Electric Power use dataset, which contains estimates of an individual household's electric power use, was used in this investigation. The data is gathered at a sampling rate of once every minute and spans a period of time close to four years (47 months). Data from a single household's energy use spanning 47 months, or roughly 4 years, from 2006 to 2010 is included in the dataset. The electricity usage figures for a number of home appliances were gathered. The dataset's columns each indicate a different characteristic of how electricity is used. The total amount of active power consumed by all the households is known as global active power and is measured in kilowatts. Submetering1 displays the overall energy (in watt hours of active energy) utilized by kitchen appliances. The total energy used by laundry appliances is shown in sub-metering 2 (measured in watt hours of active energy). The total energy used by temperature control systems is shown (in watt hours) by sub-metering 3. [14]

We used a variety of preprocessing approaches, which will be deailed in this post, to alter certain dataset portions. First off, certain variables in the dataset were either missing or wrong. For instance, some data instances replaced actual numeric values with question marks (?) instead. These were converted using the Pandas library to NaN (Not a number). The non-index dataset's values were later converted into float32 data. Thirdly, to fill in any gaps in the dataset, we utilized the corresponding quantity from the previous one-day electrical energy usage record. After that, the dataset was re-sampled by "hour," which involved adding up all of the daily minute data with corresponding hourly calculations and using the final number for that day in the Pandas library. [13, 14]

Expalamantory Data Analysis (EDA) is used to study the data, and the most important findings are then gathered. It will cover a variety of topics, including distribution, null values, and other fundamental concepts in data. This will demonstrate some machine learning methods or how to use graphs to explore data. EDA makes advantage of the data distribution and pattern of the required attributes to describe the nature of a value and its contribution to the dataset [15]. Data pattern identification is the main component of EDA; any recurring or ambiguous aspects of the data can be better visualized to comprehend the data. To assess the data pattern, we offer the line graph of "Global Reactive Power," "Global Active Power," "Global Intensity," and "Sub Metering1". But the main focus of this research with “Global Active Power”. [14]



From the given figures, I have extracted the energy consumption data on a daily, weekly, monthly, and yearly basis. The graph indicates the pattern of power usage over time, with the y-axis representing the frequency of power consumption in kW and the x-axis representing the time in minutes. The analysis reveals that power consumption is higher on weekdays, which can be attributed to the usage of appliances such as air conditioners, microwaves, washing machines, refrigerators, and other electrical devices. Additionally, the data indicates that energy consumption is lower on weekends. Furthermore, the energy consumption varies across different months, possibly due to consumer holidays. The yearly power consumption figure provides a comprehensive view of the energy consumption pattern, which can be used for predicting future energy usage. Interestingly, the data also indicates that global active power and global reactive power have the same values, implying a strong correlation between the two variables.[5]

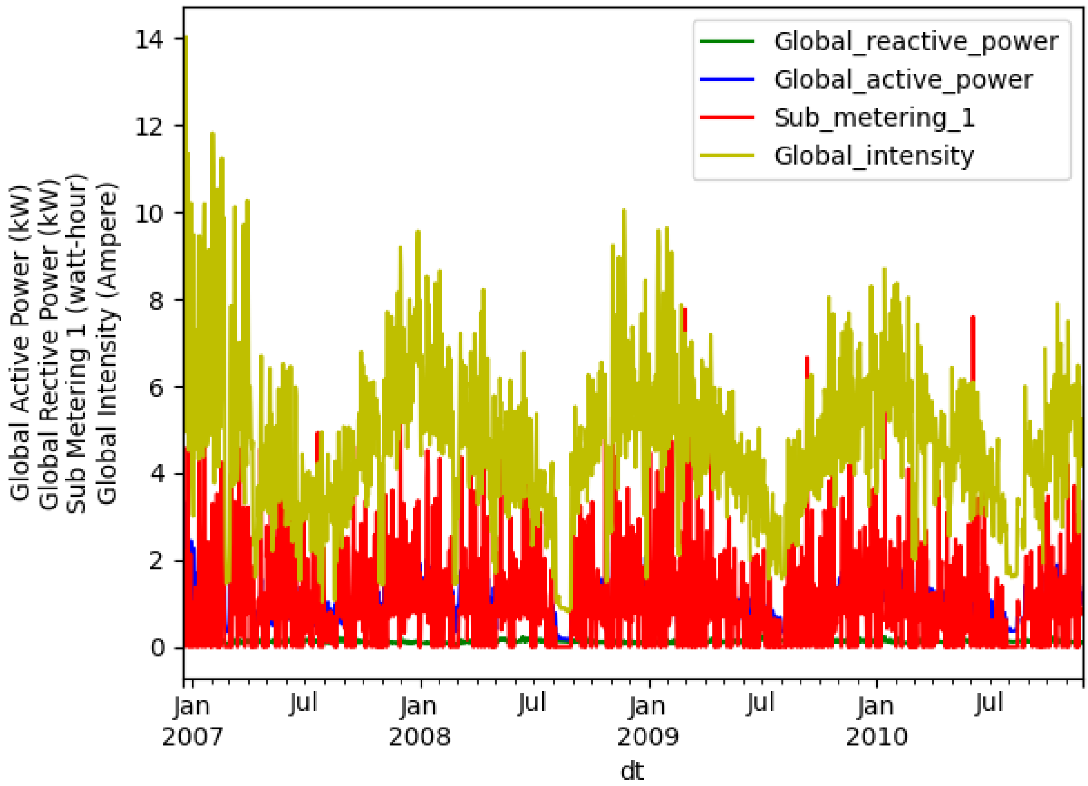



*Figure 9 Daily energy consumption[13]*

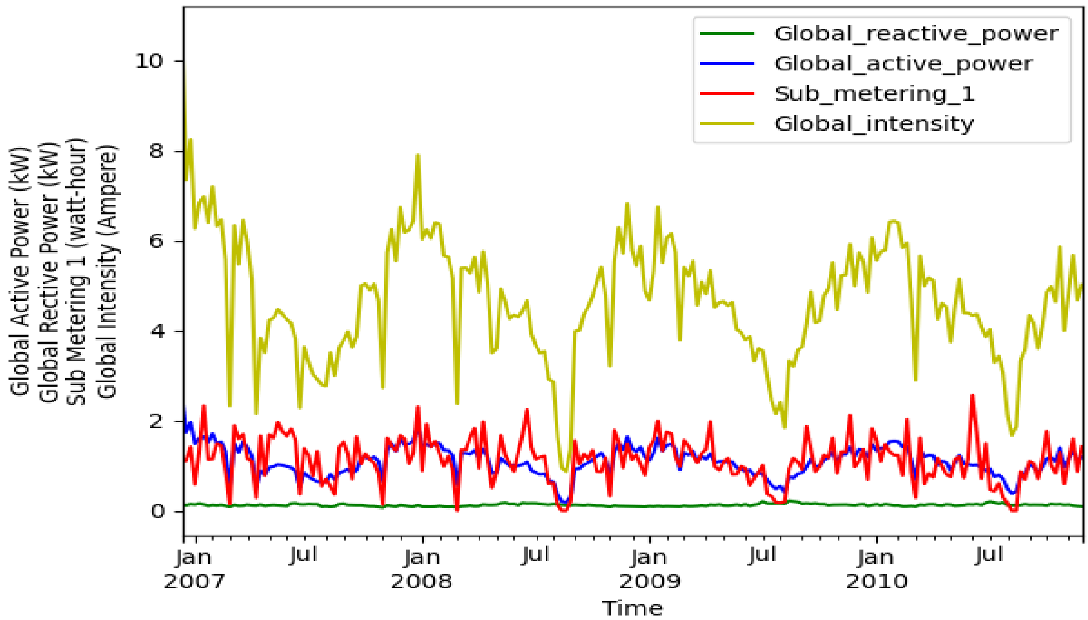



*Figure 10 Weekly energy consumption[13]*

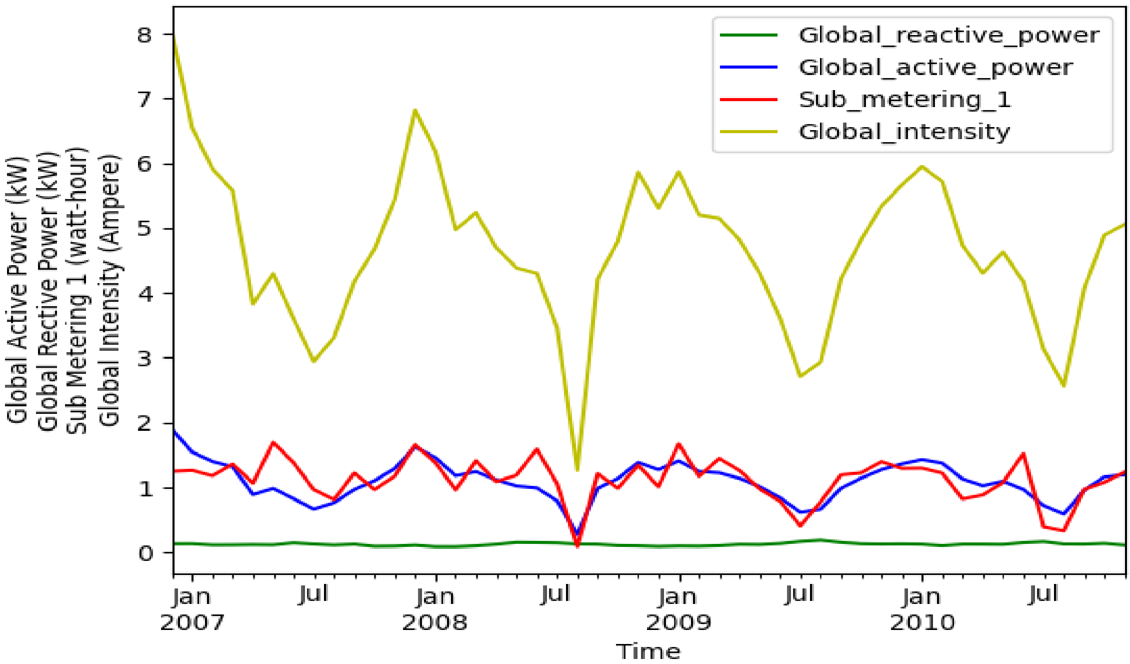



*Figure 11 monthly energy consumption[13]*

Figure 11's x-axis and y-axis clearly show the time in "minutes" and the frequency of energy use, respectively. It is clear that some weeks' days exhibit different patterns than other weeks' days. Consumers may take the next few weeks off due to this. For instance, because fewer people used to go out on Saturday and Sunday, a residence would use less energy during the summer weekends to cool down and even less during the winter months to heat up.[13]

## 3.3 Time Series Analysis



A group of observations compiled throughout time at regular intervals is referred to as a time series. In many different industries, this kind of data is used to identify patterns and trends that develop over time. Hourly weather information, daily stock prices, and monthly sales numbers are a few examples of time series data. A statistical method known as time series analysis can be used to examine time series data and discover important insights including patterns, cyclicality, and seasonality.[3]

Time series analysis can be utilized for two primary purposes: firstly, to forecast future data points and predict future patterns and trends, and secondly, to obtain a deeper understanding of the underlying mechanisms that give rise to the data. In other words, time series analysis enables us to make predictions and produce the data more thoroughly. [3]

There are various types of time series data that can be examined using time series analysis techniques. Some of the most common time series types include:

Trend:

In a time series of data, a trend is a long-term pattern, representing the overall direction or movement of the data over time.

1. Seasonal: A seasonal time series exhibits a pattern that repeats itself at regular intervals over time, such as daily, weekly, monthly, or yearly cycles. [3, 13]
2. Cyclical: A cyclical time series exhibits a pattern that repeats itself irregularly over time, such as business cycles or economic cycles. [3, 13]
3. Random: A random time series exhibits no identifiable pattern or trend, and the data points appear to be randomly distributed over time. [3, 13]
4. Auto-correlated: An auto-correlated time series is one in which the current value of the variable is correlated with its previous values, meaning that the value at any given point in time depends on the values at previous points in time. [3, 13]
5. Non-stationary: Traditional time series analysis techniques have problems analyzing and forecasting non-stationary time series because their statistical characteristics, such as mean, variance, and autocorrelation, fluctuate over time.[3, 13]

Time series analysis has applications in many fields where data is accumulated over time, including finance, economics, weather forecasting, stock market analysis, and sales forecasting.



### 3.3.1 Single Step Ahead Forecasting



Using historical data, single step forecasting is a technique for estimating the value of the following time step in a time series. This method is often used in a variety of industries, such as stock market analysis, energy demand forecasts, and sales forecasting. Numerous variables, such as the caliber of the time series data, the choice of forecasting models, and the choice of evaluation criteria, all have an impact on its accuracy. Single step forecasting's main goal is to predict the following time step using values from earlier time periods for the independent variables. Single-step ahead forecasting, also known as one-step ahead forecasting, is a method used in time series analysis to predict the following value in a time series based on preceding data. After the prediction is made, the actual value for that time point becomes available, and the process is repeated iteratively for subsequent time points. This approach is called "single-step" or"one-step ahead" because the forecast is made only for the immediate future (i.e., the next time point), without considering any further time steps. The forecast is based solely on the historical information available up to that point. Single-step ahead forecasting is commonly used in time series analysis because it allows for the evaluation and refinement of the forecast at each time step as new data becomes available. It provides a dynamic and adaptive approach to forecasting, where the model can be updated and adjusted based on the actual outcomes observed over time. This technique is particularly useful when dealing with time series data that exhibits changing patterns or dynamics, as it allows the model to adapt to these changes and make accurate short-term predictions. However, it does not provide information about longer-term trends or forecasts beyond the next time point, which may require alternative forecasting methods, such as multi-step ahead forecasting.[10, 13]

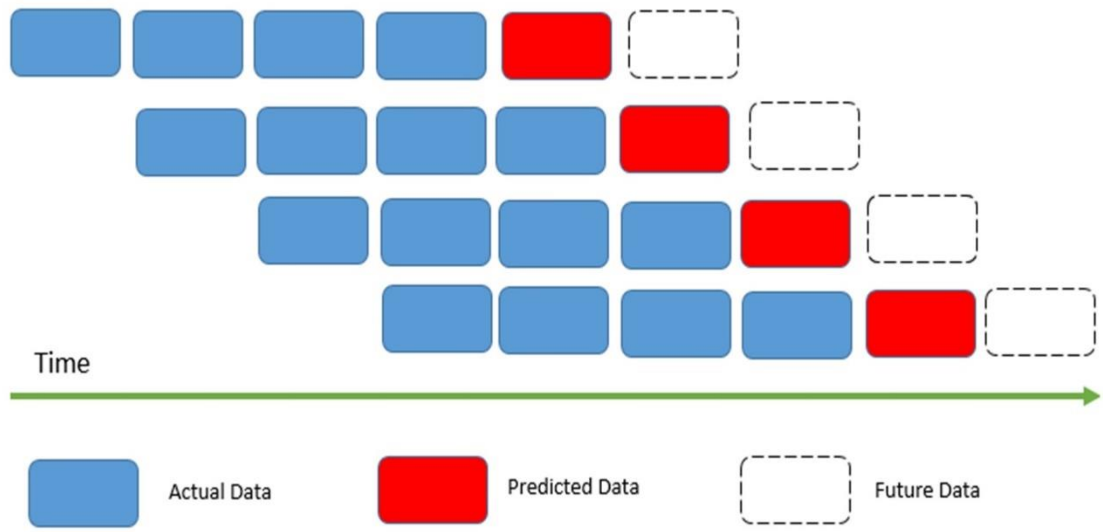



*Figure 12 single step forecasting[13]*

### 3.3.2 Multiple Step Ahead Forecasting

The term "multiple step ahead" refers to the process of predicting or forecasting values for several future time steps using time series data. Forecasting makes an effort to estimate values for a few of future time steps, in contrast to standard time series analysis, which focuses on estimating the next value in a sequence, many steps ahead, based on prior data [24]. This technique is important in various fields such as stock market prediction, weather forecasting, and energy demand forecasting, where it is necessary to forecast values for multiple time horizons to facilitate decision-making processes. Time series data's quality defines the model of forecasting accuracy, and the choice of evaluation metrics are some of the variables that affect how accurate multiple step forward forecasting is [11][13].

The multi-step time series forecasting technique allows for the prediction of many future values in a time series beyond the next time point. Unlike single-step ahead forecasting, which predicts only the next value, multi-step forecasting involves predicting a sequence of future values over a specified horizon. In multi-step time series forecasting, the model is trained on historical data up to a certain point and then used to generate a sequence of future values. The length of this sequence, known as the forecast horizon, is determined by the specific problem and context.

There are different approaches to perform multi-step time series forecasting:

Direct Forecasting: In this approach, the model directly predicts all the future values in the forecast horizon. The model is trained to output a sequence of values, each representing the value at a specific future time point.

Recursive Forecasting: This strategy involves forecasting the next step iteratively using the value that was anticipated as input. Until the required forecast horizon is achieved, the process is repeated. Recursively creating new predictions from the model's previous ones.

Encoder-Decoder Models: These models, also called sequence-to-sequence models, have two major parts: an encoder and a decoder. The historical data is processed by the encoder, who then condenses it into a context vector or fixed-length representation. The decoder then creates a series of future values using this context vector as input. [13]



The temporal dependencies and patterns contained in the data must be carefully taken into account while performing multi-step time series forecasting. This task can be accomplished using a variety of machine learning and statistical techniques, such as transformers, recurrent neural networks (RNNs), autoregressive models, and long short-term memory (LSTM) networks. This technique is valuable when the forecasting problem requires predictions beyond the next time point, such as predicting future sales, stock prices, or energy demand over several time periods. By providing a sequence of future values, multi-step time series forecasting offers a broader and more comprehensive view of the future behavior of the time series. [36][15]

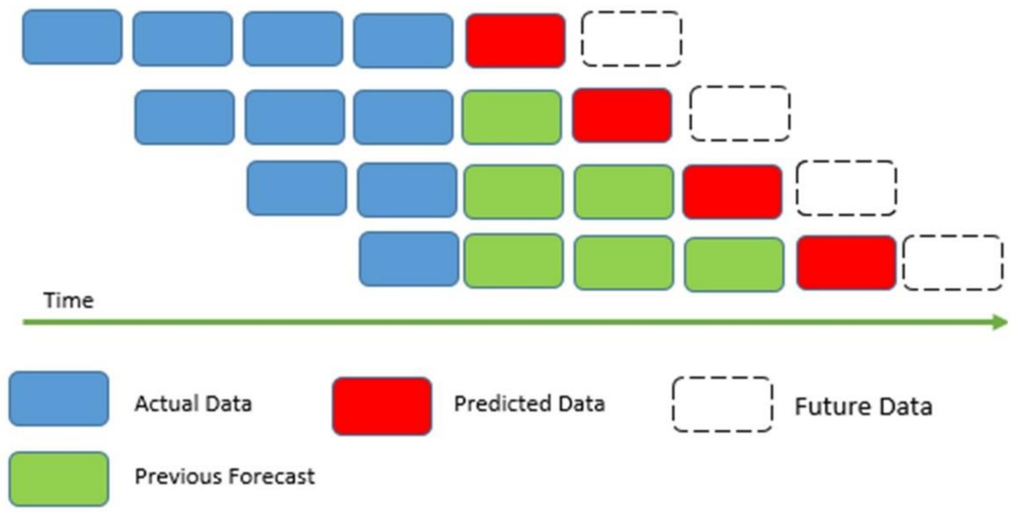



*Figure 13 Multistep forecasting [13]*

A specific method for examining data points that are gathered over time at predictable intervals rather than at random or erroneous intervals is called time series analysis. It entails more than just gathering data gradually; in order to guarantee the accuracy and consistency of the analysis, a significant amount of data is frequently needed. This makes it possible to identify trends, patterns, and seasonal variations as well as to filter out inaccurate data. We used information from the last 47 months to create this article. There are two basic types of forecasting data: single-step forecasting and multi-step forecasting.[13] Single-step forecasting entails training the data series, moving forward one step at a time, and forecasting the subsequent step. In contrast, multi-step forecasting involves training the data with previous data points and moving forward more than one step ahead in time. This approach uses a series of time windows to predict future data points. In this study, we used single- step forecasting for global active power.

## 3.4 Dropout

In deep learning, dropout is a popular regularization method, notably for Recurrent Neural Networks (RNNs). Its primary goal is to randomly deactivate some neurons during training so that the network must learn more reliable and flexible data representations. In each training epoch, dropout chooses a subset of neurons to ignore, reducing the network's dependency on any single set of neurons [16]. By using this technique, the model's performance in improving ability of model for further unseen datapoints and also focusing against overfitting. In RNNs, dropout is applied to the connections between the hidden states to prevent overfitting to the temporal structure of the data[3]. The strength of the regularization can be controlled by adjusting the dropout rate, which specifies the proportion of neurons to be deactivated during training. Typically, the dropout rate is set between 0.2 and 0.5, depending on the model complexity and dataset size [19]. Dropout is effective in forcing the network to learn robust and generalizable features, preventing the model from memorizing noise or irrelevant data, thus improving its performance on unseen data. In general, dropout is an effective technique for improving the performance and generalization of deep learning models, including RNNs. [3]

Regularization methods like dropout layers are frequently employed in neural network architectures like Keras. By randomly "dropping out" a portion of the input units during training, it helps avoid overfitting and enhances the model's capacity to generalize.

A dropout layer randomly becomes a portion of the input units to zero during the forward pass of training, thereby "dropping them out" and eliminating them from the calculations of the following layers. The fraction of units that are dropped out is determined by a hyperparameter called the dropout rate, which specifies the probability of dropping a unit. By randomly dropping out units, dropout layers encourage the neural network to learn more robust and generalizable representations. It reduces the interdependencies between neurons and aids in preventing the model from depending too much on a few input units. Dropout can therefore lessen overfitting and increase how effectively the model generalizes to new data.[16]

During inference or when making predictions, dropout is typically turned off, and the full network with all units intact is used. However, the learned weights are scaled down by the dropout rate to account for the higher activation during training.



In Keras, you can add a dropout layer using the Dropout class. You specify the dropout rate as an argument when creating the layer.

For example:

*from tensorflow.keras.layers import Dropout*

*model.add(Dropout(rate=0.2))*

This code adds a dropout layer with a dropout rate of 0.2, meaning that 20% of the input units will be randomly dropped out during training.

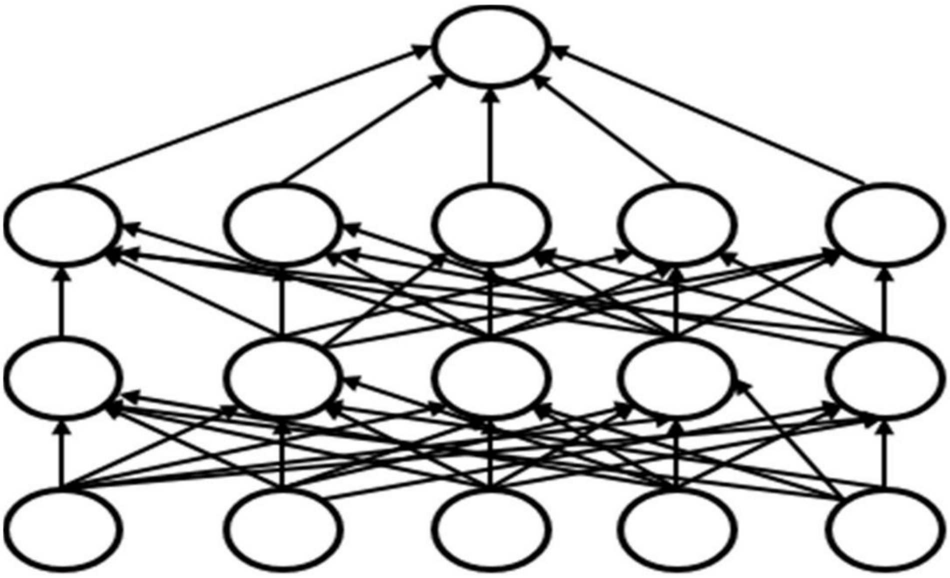


*Figure 14 before applying dropout {14]*

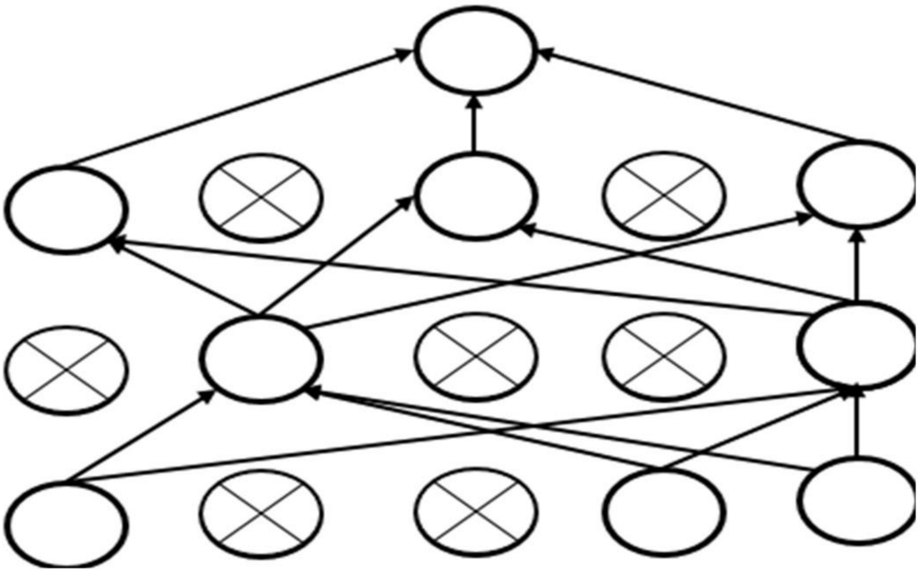


*Figure 15 after applying dropout [14]*

# CHAPTER 4 ANALYSIS OF MODEL AND RESULTS

## 4.1 Model Evaluation

Model evaluation is the process of determining how well a deep learning or machine learning model performs given a particular dataset. It entails evaluating different metrics and examining the model's predictions to learn more about the model's precision, generalizability, and possible areas for improvement. When evaluating a deep learning model, several commonly used evaluation techniques and metrics can be employed. Here are some key concepts:[13, 17]

1. Training and Testing Split: The dataset is often divided into training and testing subsets. The model is trained on the training set, and the testing set is used to evaluate the model's performance. This distinction makes it easier to predict how well the model will work with data that is not real.
2. Loss Function: Using the loss function, the difference between the model's predictions and the actual values is calculated during training. It demonstrates how well the model is assimilating the training set of data. Typical loss functions include categorical cross-entropy for classification problems and mean square error (MSE) for regression problems.
3. Accuracy and Error Metrics: These metrics provide a thorough assessment of the model's performance. While accuracy measures the proportion of accurate predictions, error metrics (such as mean absolute error or mean squared error) quantify the quantity of prediction errors.
4. Confusion Matrix: A confusion matrix, which shows true positives, true negatives, false positives, and false negatives, provides a detailed analysis of the model's predictions for each class for classification tasks. The confusion matrix can be used to calculate additional metrics, such as precision, recall, and F1 score.
5. Overfitting and Validation Set: When a model performs well on training data but struggles to generalize to fresh, untried data, overfitting has taken place. A validation set is frequently utilized during training to identify and reduce overfitting. It is possible to tell if a model is overfitting by how it performs on the validation set and whether or not changes to the model's architecture or hyperparameters are necessary.

A deep learning model's quality and effectiveness must be evaluated in order to be determined. Practitioners can decide on model enhancements, fine-tuning, or choosing the optimal model for deployment in real-world scenarios by assessing numerous metrics and comprehending the model's performance.

The model was only trained on the first year of data for the demonstration, which sped up the training process. The prepared dataset was divided into train and test sets. The model was then evaluated using data from the three years that's followed [29]. To speed up the computation, the values were resampled hourly. The model contains 365 days as a training set while other 3 year data as a testing. The model used an RNN with one neuron in the output layer and 100 neurons in the first visible layer for predicting global active power. One hour was selected as the step size. The model was trained using a 70-piece batch across 100 epochs. Additionally, the model's training and testing losses were examined.

### 4.1.1 Learning Curve

An illustration of the link between a model's performance and the volume of training data it has been exposed to is called a learning curve in machine learning. It demonstrates how, as the amount of the training dataset rises, the model's performance either increases or stabilizes. A machine learning model's behavior and performance during the learning process can be evaluated and analyzed using learning curves. They shed light on the model's learning process, its degree of underfitting or overfitting, and how the model's performance evolves with the availability of new data. A typical learning curve consists of two axes: [13]



1. Performance Metric: A performance statistic, such as accuracy, error rate, mean squared error, or F1 score, is represented on the y-axis. It measures how effectively the model is carrying out the current task. The type of problem being tackled determines the precise performance statistic that is employed.
2. Training Dataset Size: The size of the training dataset is shown on the x-axis and is commonly expressed as the quantity of samples or observations. It displays the volume of information used to train the model.

By repeatedly training the model with different subsets of the training data, the learning curve is produced. A different validation or test set is then used to evaluate the model's performance. The process is repeated for different training set sizes, incrementally increasing the amount of data used for training. The learning curve shows how the performance metric changes as the training dataset size increases. It may reveal the following patterns:[13]

1. Underfitting: Even with a big training dataset, if the performance metric is poor, it may indicate that the model is underfitting. This indicates that a more complicated or expressive model is needed because the current model is unable to represent the underlying patterns in the data.
2. Overfitting: When the performance metric is high on the training set but significantly lower on the validation set, overfitting is evident. The model cannot generalize to new data because it has memorized the training data too thoroughly. Increasing the size of the training dataset can reduce overfitting.
3. Convergence: As the amount of the training dataset grows, the performance metric may plateau, according to the learning curve. This indicates that the model has converged and further increasing the dataset size may not lead to significant improvements in performance.



Learning curves provide insights into the trade-off between model complexity and dataset size. They help guide decisions on model selection, data collection, and whether additional data would be beneficial. By analyzing learning curves, practitioners can make informed decisions about model performance and optimize the learning process. Figures below contains the dropout values from 0.2, 0.3, 0.4, 0.5 and 0.6. The different values of dropout shows the different predictions. In comparison 0.4 dropout rate shows the better predicted values than other dropout rates. For the purpose of prediction, it can be visible that 0.4 dropout rate was better for our RNN model [15] [13].

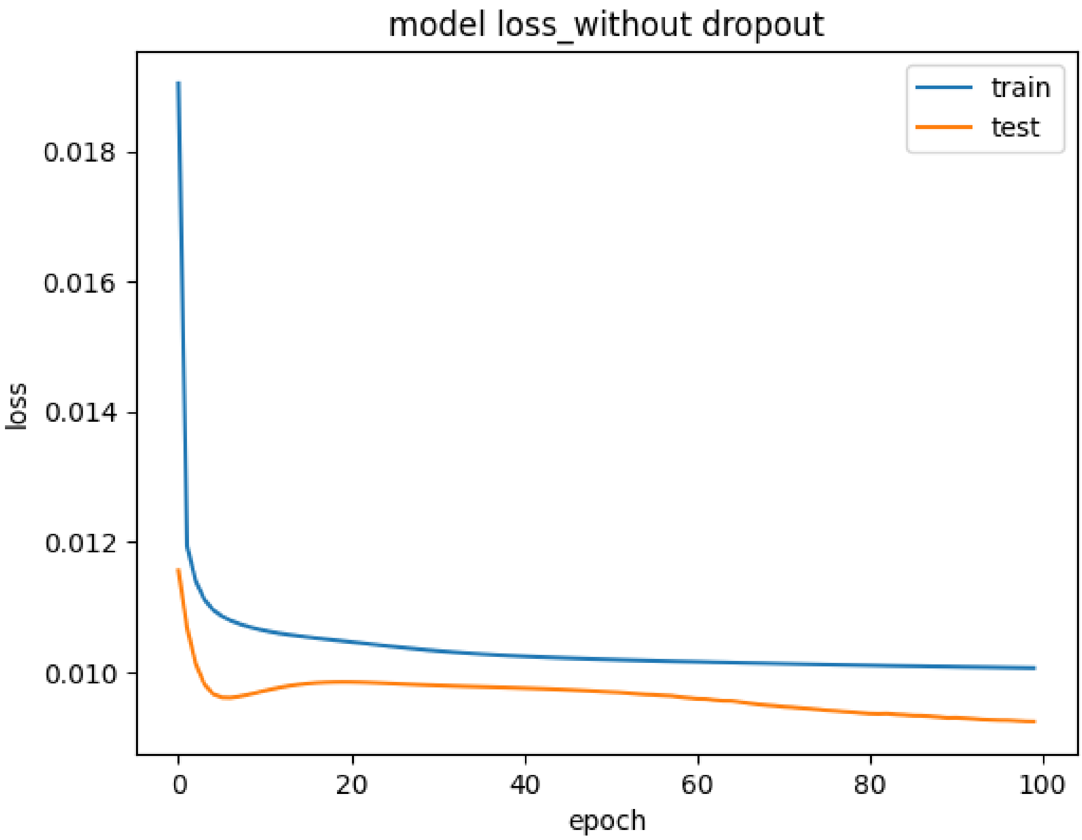
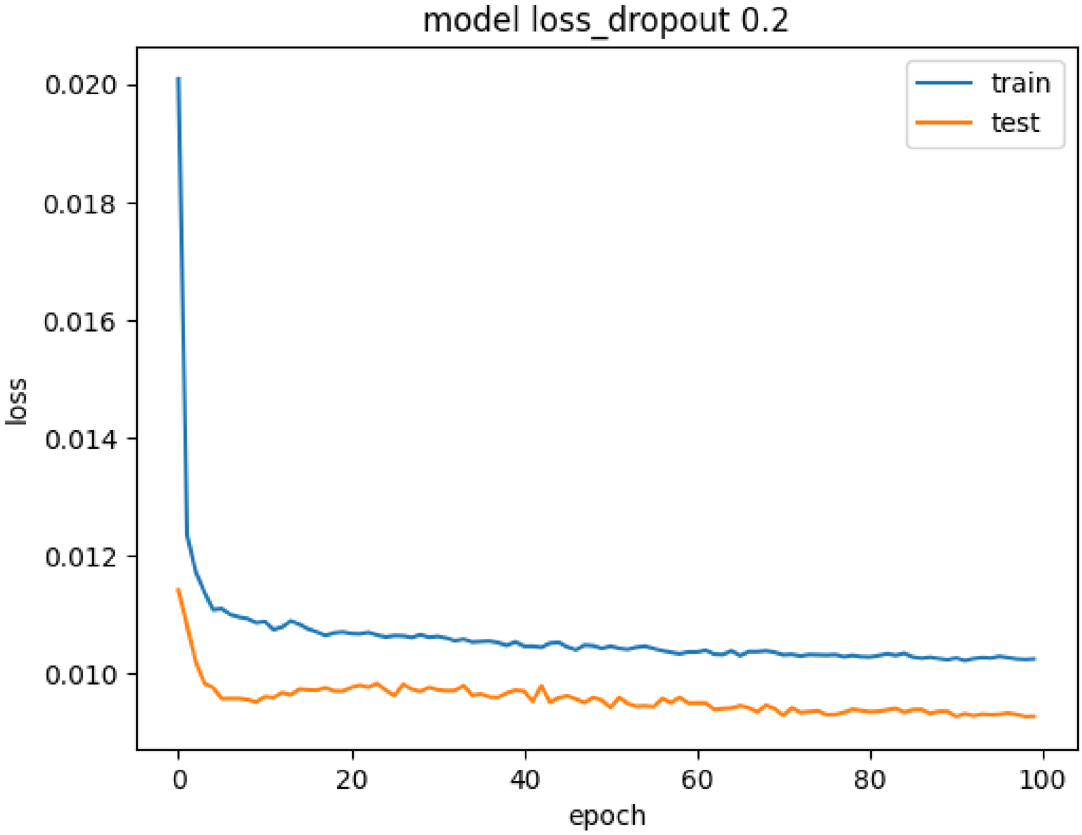
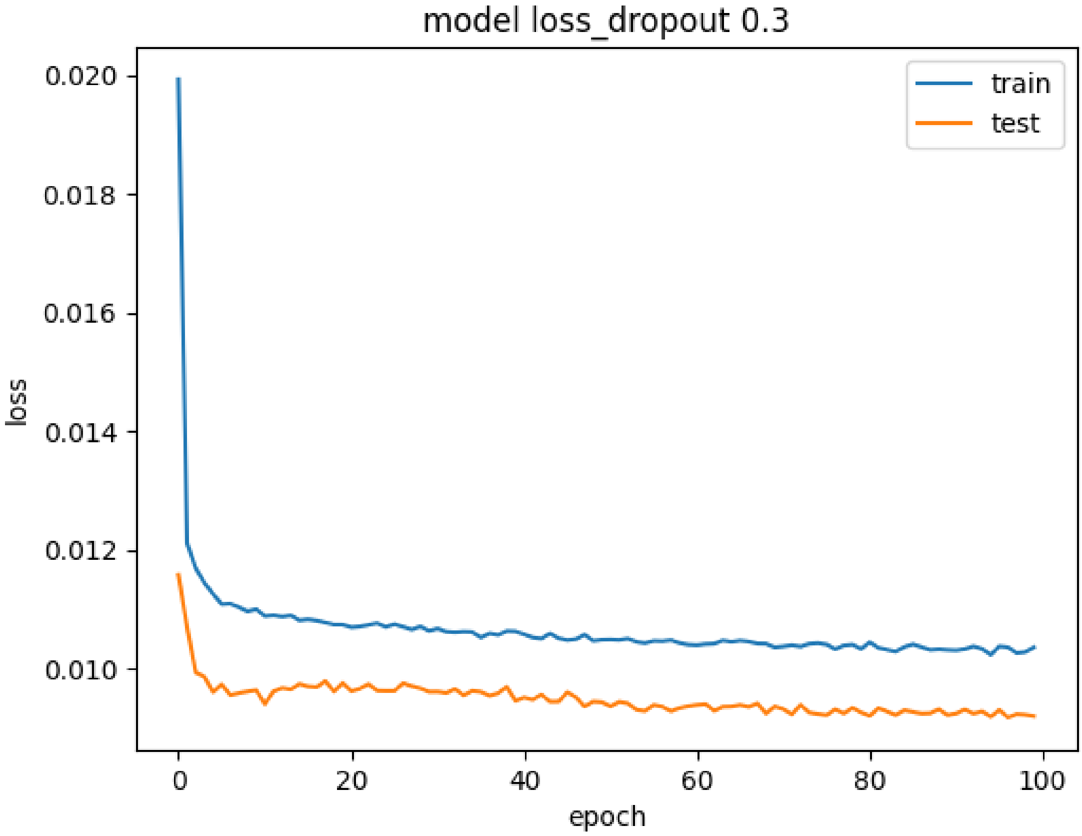
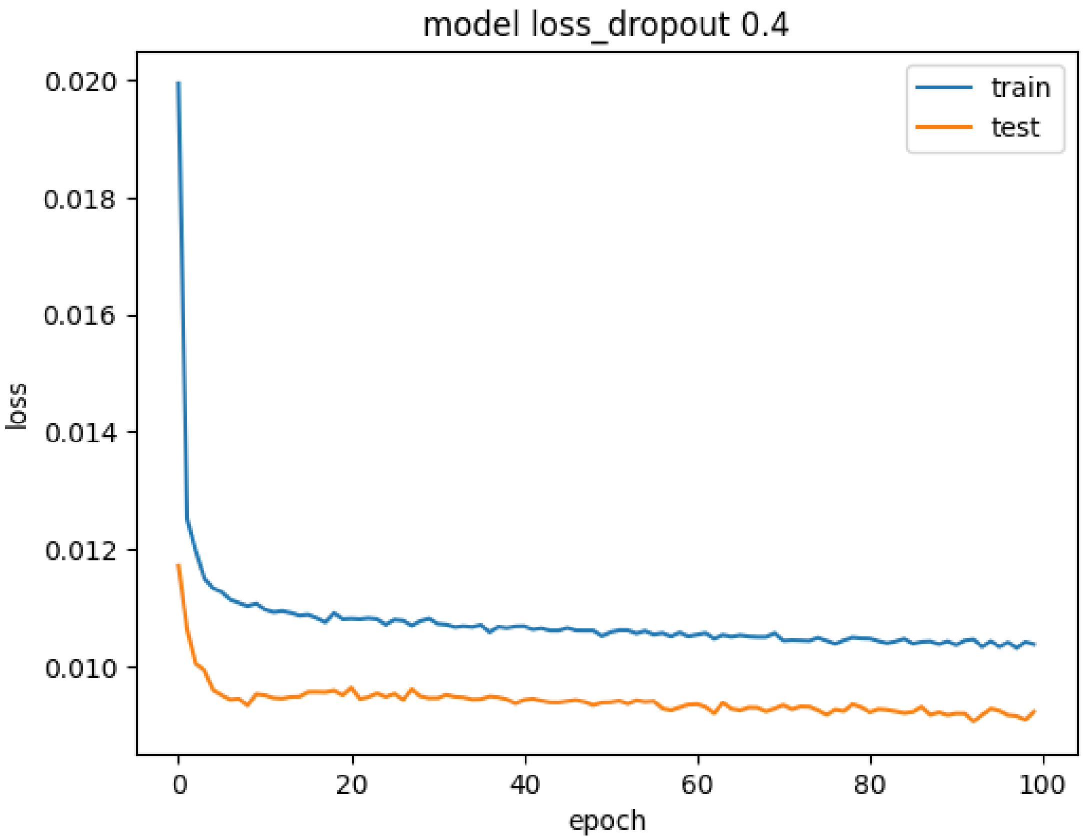
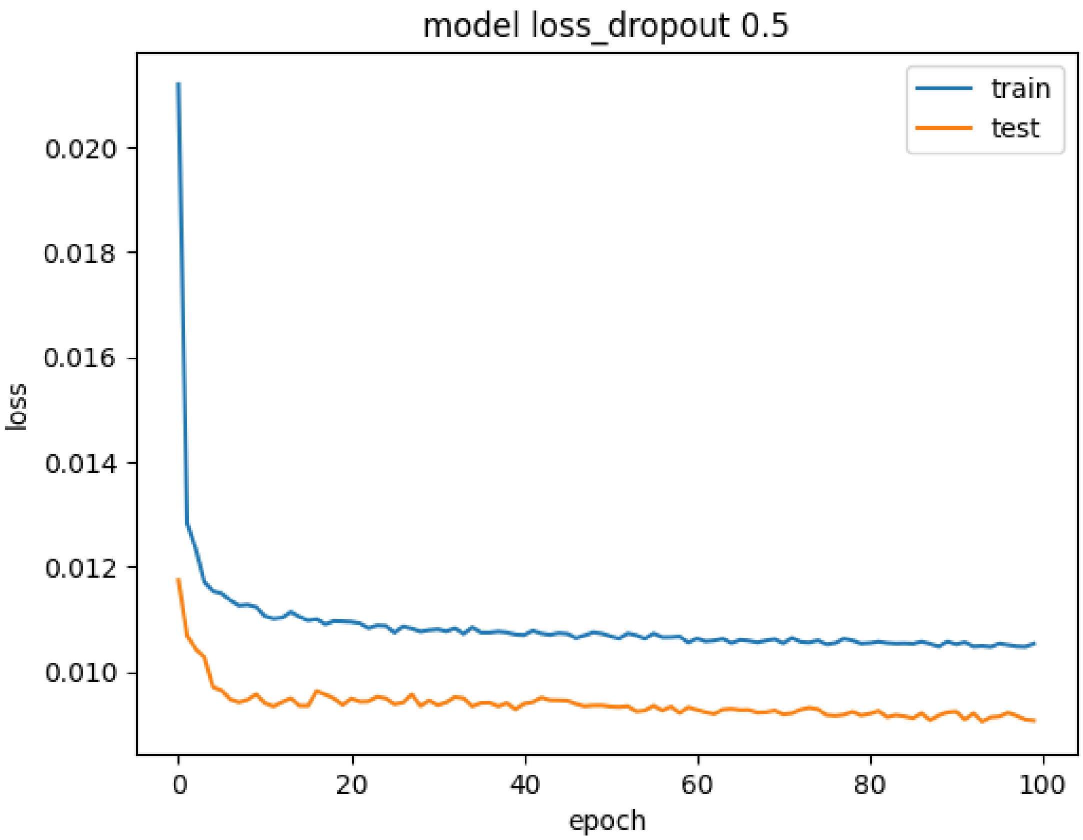



*Figure 16 model loss without dropout*


model loss_dropout 0.2
train
test
loss
0.020
0.018
0.016
0.014
0.012
0.010
0
20
40
60
80
100
epoch


model loss_dropout 0.3
train
test
loss
0.020
0.018
0.016
0.014
0.012
0.010
0
20
40
60
80
100
epoch


model loss_dropout 0.4
train
test
0.020
0.018
0.016
0.014
0.012
0.010
loss
0
20
40
60
80
100
epoch


model loss_dropout 0.5
train
test
0.020
0.018
0.016
0.014
0.012
0.010
loss
0
20
40
60
80
100
epoch

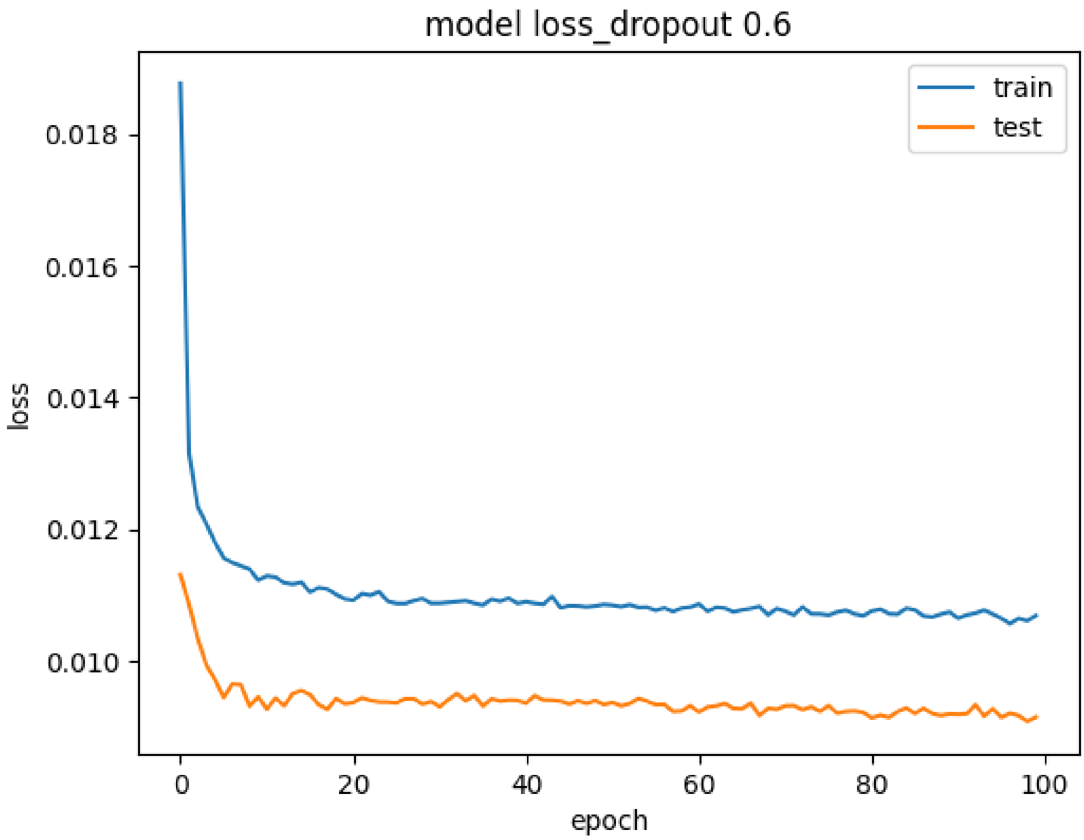



*Figure 17 model loss at dropout 0.2, 0.3, 0.4, 0.5 and 0.6*

We can see that the training loss began to decrease more rapidly than the testing loss in comparison at a specific period. Testing likewise has a decreasing ratio, while training continues to decline. In contrast, the rate of the testing loss declines more slowly than the rate of the training loss. The model loss were found with different drop-out rates. The dropout rates 0.2 0.3, 0.4, 0.5 and 0.6 were analyzed by the model loss. [13, 18]

As we can see dropout 0.2 and 0.3 are almost same in shape. While drop 0.4 and 0.5 have different variation in the model loss. The loss for 0.4 and 0.5 is greater than dropout of 0.2 and 0.3. But, if we compare them with the prediction table shown below, it will be clear that the prediction and actual values have differences for drop 0.4. However, dropout 0.2 and 0.3 have almost same difference values. Dropout 0.5 have highest difference of values. [19]

Dropout is the regularization techniques used to overcome the overfitting problems. The dropout rates have been applied to the model during training. Dropout layers randomly sets a fraction of the inputs to zero at each update, which effectively drops them out of the network for that particular forward and backward pass. Dropout layer added to encourage the neurons to be more self-reliant on a few dominant features. Dropout layers are more power tool in reducing overfitting, improving generalization, and increasing the robustness of the model.

### 4.1.2 Mean Square Error (MSE)

Mean Squared Error is referred to as MSE. It is a typical statistic used to assess how well a regression or machine learning model is working. MSE calculates the average squared difference between a dataset's true values and anticipated values.

The following is the MSE calculation formula:

*MSE* = *(1/n)* $\sum_{i=1}^{n} Y_i - Y_i^{\wedge})^2$

Where:

$n$ is the number of data points in the dataset.

$y_i$ represents the predicted value for the $i^{th}$ data point.

$\bar{y}$ represents the true value for the $i^{th}$ data point.

The difference between each predicted value and its matching true value is taken, squared, added, and then divided by the total number of data points to produce the MSE. MSE has several useful properties:

1. It is always a non-negative value.
2. A lower MSE indicates better model performance, as it means the predicted values are closer to the true values.
3. Squaring the differences penalizes large errors more than smaller errors, which can be useful depending on the problem.



The accuracy of models that predict continuous numerical variables, such as home prices, stock prices, or temperature values, is frequently assessed in regression tasks using MSE. It offers a numerical assessment of how well the model matches the data, which is useful for contrasting models or fine-tuning model parameters. Using NumPy or scikit-learn libraries, you may compute MSE in Python. For this, people frequently utilize the scikit-learn function mean_squared_error().

### 4.1.3 Root Mean Square Error (RMSE)

RMSE stands for Root Mean Squared Error. In regression projects, the average size of the errors between the predicted and true values is a common evaluation metric. Calculating RMSE involves taking the square root of the mean of the squared differences between the projected values and the actual values. The following is the formula to calculate RMSE:

$MSE = \frac{1}{n}\sum_{n=1}^{n}(Y_i - Y_i^{\hat{}})^2$

Where:

$n$ is the number of data points in the dataset.

$y_i$ represents the predicted value for the i-th data point.

$\bar{y}$ represents the true value for the i-th data point.

RMSE is similar to Mean Squared Error (MSE), but the RMSE provides an interpretable value in the same unit as the target variable, unlike MSE, which is in squared units. RMSE returns the error measure to the data's original scale by taking the square root of the MSE. When errors are huge in scale or when outliers significantly affect how well a model performs, RMSE is helpful too. In our case we have used MSE. It is frequently employed in many different fields, including machine learning, finance, and economics.[13, 20]

The model contains 100 nodes in the architecture and one fully connected output node. The weight of fully connected was dropout during the training by inserting a drop-out layer. Fully connected layers allow neural networks to learn complex non-linear relationships between the input and output data. The number of neurons in a fully connected layer determines the dimensionality of the output. For example, if you have a fully connected layer with 100 neurons, the output of that layer will be a vector of size 100. The number of neurons in the input layer of the network should match the dimensionality of the input data. A fully connected layer contains the weights that is connected with the output of the previous layer.

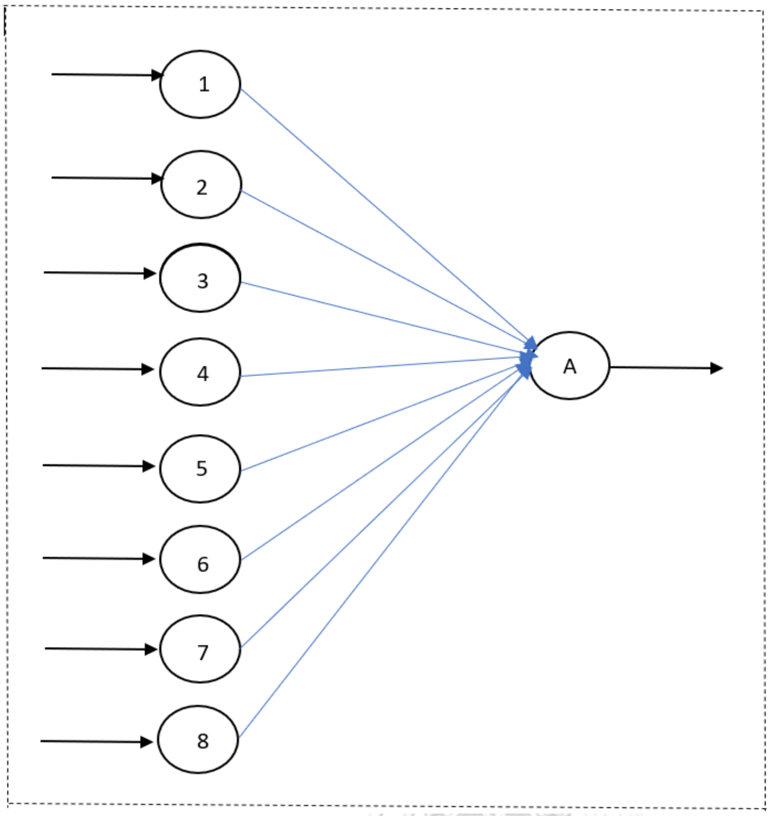



*Figure 18 Fully connected layer*

In Python, you can calculate MSE using libraries such as NumPy or scikit-learn. The numpy library provides a convenient *sqrt()* function to compute the square root, and you can combine it with the *mean_squared_error()* function from scikit-learn.

*Table 1 MSE at different dropouts*

| Model MSE ($kW^2$) | |
|---|---|
| Without Dropout | 0.521 |
| Dropout 0.2 | 0.517 |
| Dropout 0.3 | 0.515 |
| Dropout 0.4 | 0.511 |
| Dropout 0.5 | 0.514 |
| Dropout 0.6 | 0.513 |
| Dropout 0.7 | 0.618 |
| Dropout 0.8 | 0.623 |
| Dropout 0.9 | 0.625 |

*Table 2 Model result difference at 0.4 dropout*

In the training process, a technique called dropout is applied where certain layer outputs are randomly disregarded. This results in the layer appearing and functioning as if it had a different number of nodes and connections to the preceding layer. Essentially, each update to a layer during training is carried out with a varied perspective of the configured layer. The model shows the better accuracy at dropout rate of 0.4.

| **Model Result Difference at 0.4 Dropout** | | |
|---|---|---|
| Actual Values (Global Active Power kW) | Predicted Values (Global Active Power kW) | Difference (Global Active power kW) |
| 2.22339 | 2.17727 | 0.0461201 |
| 2.34973 | 2.30528 | 0.0444522 |
| 2.52264 | 2.77573 | -0.253087 |

Lower the MSE lower the difference. We found that at 0.4 dropout rate over difference is lower than the different dropout rates.

## 4.2 Model Prediction

The model has been implemented time series prediction using the Keras library, which is a high-level deep learning framework. The steps involved in the tutorial are as follows:

1. Import the necessary libraries, including Keras, NumPy, and Matplotlib.
2. Load and preprocess the time series data. This typically involves converting the data into a suitable format for training the LSTM model.
3. Split the data into training and testing sets. The training set is used to train the LSTM model, while the testing set is used to evaluate its performance.
4. Normalize the data to ensure that all values fall within a specific range, usually between 0 and 1. This step helps the LSTM model converge faster during training.
5. Reshape the data to meet the input requirements of the LSTM model.
6. Build the LSTM model architecture using the Keras Sequential API. The model consists of an LSTM layer and one or more dense layers.
7. Train the LSTM model on the training data, specifying the number of epochs and the batch size.

8. Evaluate the model's performance on the testing data by calculating metrics such as mean squared error (MSE) or mean absolute error (MAE).
9. Make predictions on new, unseen data using the trained LSTM model.
10. Visualize the actual and predicted values using Matplotlib. [21]

```
model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100)               43200

 dropout (Dropout)           (None, 100)               0

 dense (Dense)               (None, 1)                 101

=================================================================
Total params: 43,301
Trainable params: 43,301
Non-trainable params: 0
_________________________________________________________________
```

*Figure 19 model summary*

The summary of the model demonstrates following things:

1. The model has only one layer of LSTM: LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN). In this model, there is only one layer of LSTM, which means that the sequential data is processed by this single LSTM layer.

2. Neurons are the basic processing units in neural networks. In this model, the LSTM layer consists of 100 neurons, which are responsible for learning and capturing patterns in the time series data.

3. After the LSTM layer processes the input data, it is connected to a fully-connected layer. This fully-connected layer has a single neuron, which serves as the output of the model.

4. The weight of the fully-connected neuron was drop-out during training: Dropout is a regularization technique used to prevent overfitting in neural networks. In this model, a dropout layer is inserted after the LSTM layer and before the fully-connected layer. During training, some of the connections between the LSTM layer and the fully-connected layer are randomly set to zero (dropped out), which helps to reduce the model's dependence on specific neurons and improves its generalization capability.

The model architecture includes a single LSTM layer with 100 neurons, followed by a dropout layer, and then a fully-connected layer with a single output neuron.

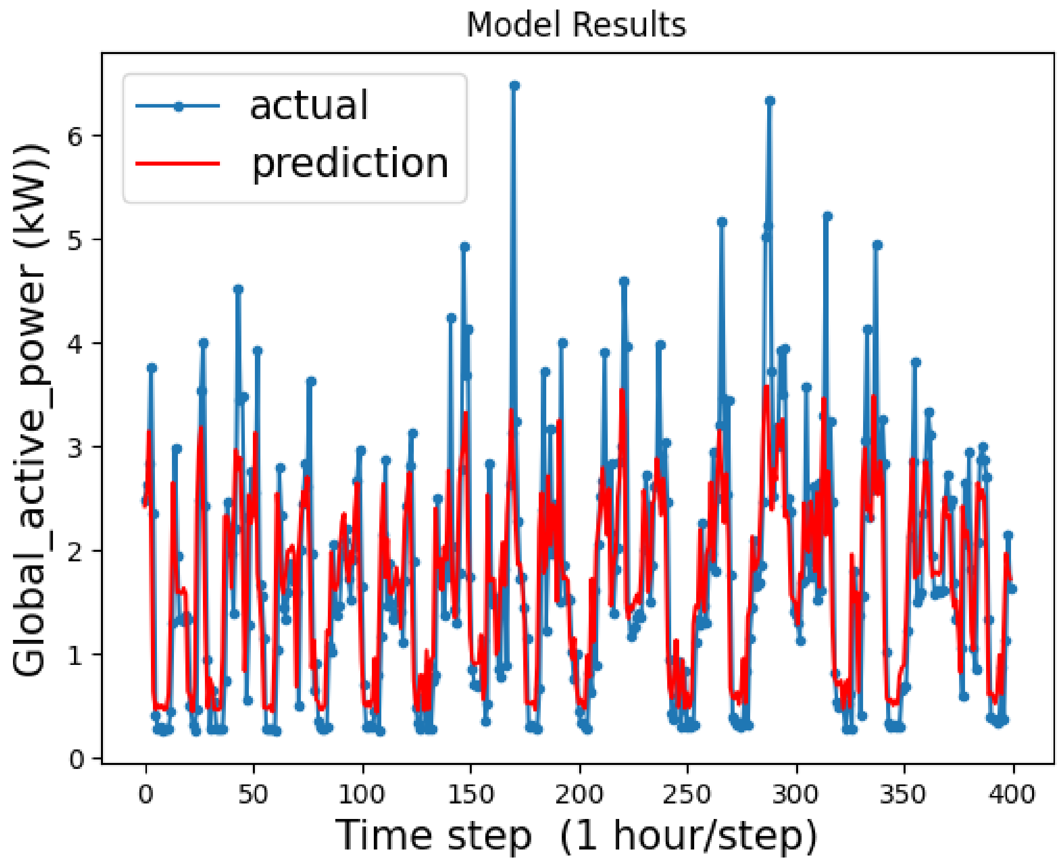



*Figure 20 actual vs predicted values[13]*

The predicted results are presented in Figure 11. This figure likely includes a line graph or some visual representation of the predicted values.

The predicted results in Figure 11 are then compared with the actual values. This means that the model's predictions are checked against the real data points to assess how closely they align. To enhance the accuracy of the comparison, the analysis focuses on the most recent 100 days' worth of data from the "tail" end of the dataset. By doing this, the comparison is conducted with data that is closer in time to the predicted values, which may provide a more relevant assessment.

The error between the predicted and actual values is shown in Figure 12. This figure likely represents the differences between the predicted values and the actual values as a separate graph or chart. The purpose of Figure 19 is to highlight the points where the predictions deviate from the actual values, indicating the instances where the model's accuracy is not on par with the real data.[15] [13].

The figure also defines that the model is not 100% accurate. However, it states that the model demonstrated a higher level of accuracy for forecasting. This implies that while the model may not be entirely precise, it still performed relatively well in predicting future values based on the available data.

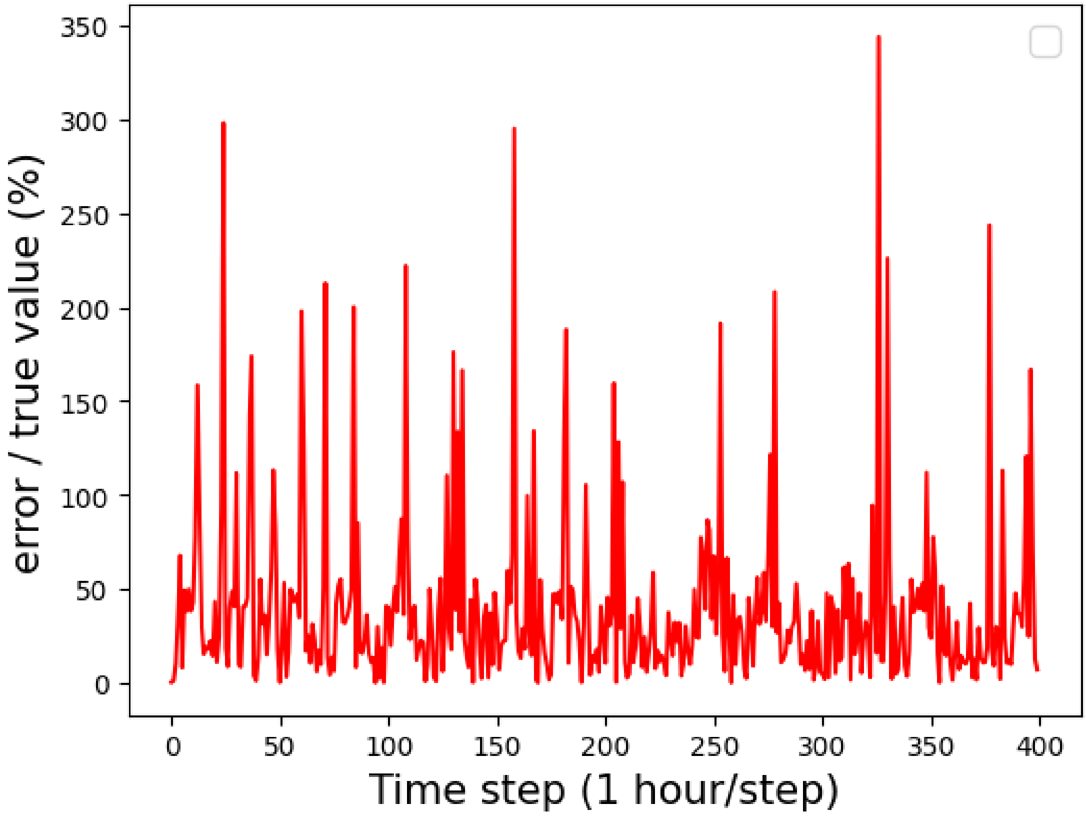



*Figure 21 error over true values[13]*

Different dropouts have been analyzed. The differences can better show the accuracy of model. It can be seen from the table 1 that at dropout 0.4 there are less number of differences founded as compare to dropout 0.2, 0.3 and 0.5. The dropout 0.4 can be better fit for our RNN model.

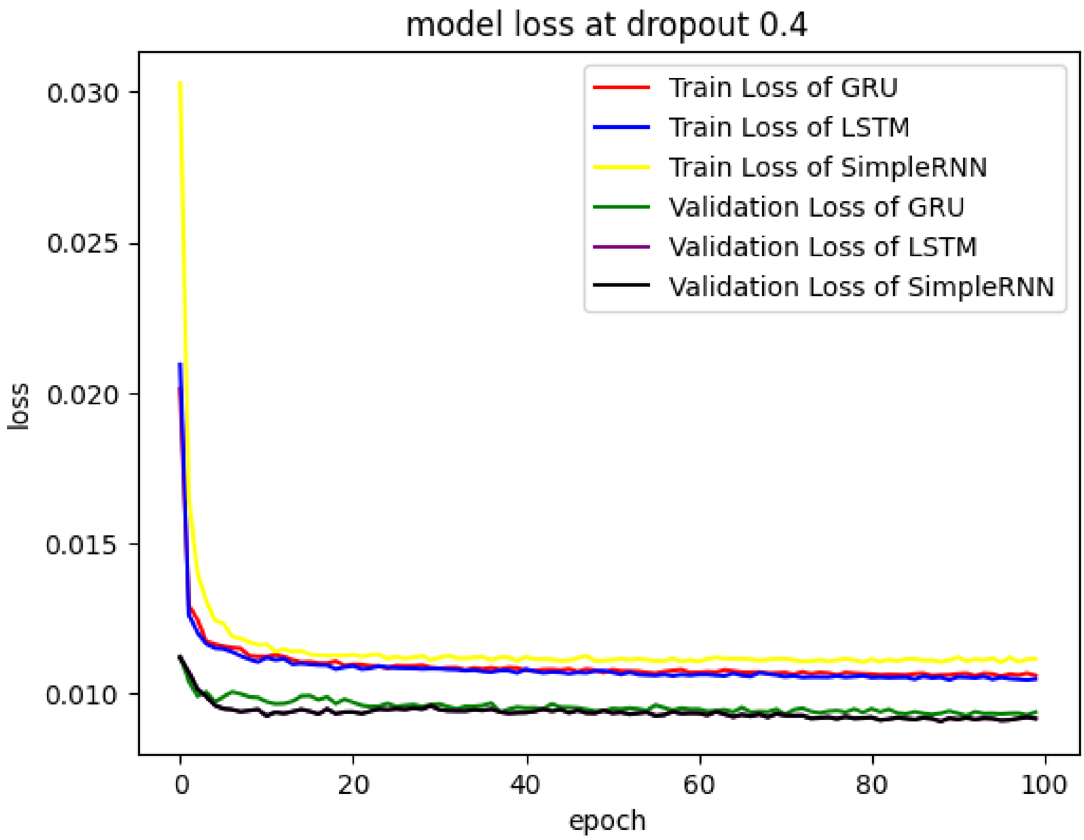



*Figure 22 Model loss for LSTM, GRU and SimpleRNN*

Figure above defines the losses for different models. It shows that LSTM have low loss than simple RNN and GRU.

*Table 3 MSE comparison of models*

| **MSE($kW^2$) at 0.4 dropout** | |
|---|---|
| LSTM | 0.511 |
| GRU | 0.512 |
| SimpleRNN | 0.618 |



LSTM, GRU, and SimpleRNN differ in how they process input sequences. Simple RNN passes information from one time step to the next but struggles with the vanishing gradient problem, limiting its ability to learn long-term dependencies. In contrast, LSTM improves upon Simple RNN with a more complex cell structure and specialized memory units. This allows LSTM to selectively remember or forget information over long sequences using a "cell state" as an information highway, enabling it to capture long-term dependencies effectively. GRU is another variant that addresses the vanishing gradient problem by simplifying the LSTM architecture. It combines the cell state and hidden state into a single "hidden state" vector and introduces "update" and "reset" gates to control information flow. GRU performs similarly to LSTM but with improved computational efficiency.

# Chapter 5 Conclusion

This thesis presents evidence that the model was able to predict the overall trend of the data. This suggests that the model was successful in capturing the general pattern or direction of the energy consumption data. The dataset used in the analysis contains information on energy consumption for approximately 47 months. This implies that the data spans a period of around four years, providing a substantial amount of historical information for the model to learn from and make predictions. The work compares the actual values of the energy consumption with the predicted values generated by the model. This step allows for an evaluation of the model's accuracy in terms of its ability to generate predictions that align with the real data. The model used in the analysis employs Long Short-Term Memory (LSTM), a type of recurrent neural network often used for time series data. The statement suggests that the model's accuracy is influenced by different dropout rates. Dropout is a regularization technique used in neural networks to prevent overfitting. It is applied to the connections between the LSTM units, randomly dropping out a fraction of the units during training to improve generalization. We further analyzed the Root Mean Square Error (RMSE) of the predictions for different dropout rates. RMSE is a common metric used to measure the average difference between predicted and actual values, providing an indication of the model's prediction accuracy. By comparing the RMSE values for various dropout rates, the researchers assessed the impact of different dropout configurations on the model's performance. The model concludes by stating that with a dropout rate of 0.4 the accuracy can be increase as compare to other dropout rates. This also suggests that among the different dropout rates tested, a dropout rate of 0.4 resulted in the most accurate predictions in terms of aligning with the actual energy consumption data.

In future work, we are going to apply dropouts at different layers of neural networks including input, hidden and output layers. Increasing number of neurons can also increase the accuracy of the model. Different models including Transformers can also rise the model accuracy.

# REFERENCES


[1] S. Shah, "Machine Learning: A Review of Learning Types," 11 July 2020.

[2] A. Alanazi, "Using machine learning for healthcare challenges and opportunities," *Informatics in Medicine Unlocked,* vol. 30, p. 100924, 2022.

[3] https://www.ibm.com/topics/supervised-learning (accessed.

[4] P. Cunningham, M. Cord, and S. J. Delany, "Supervised learning," in *Machine learning techniques for multimedia: case studies on organization and retrieval*: Springer, 2008, pp. 21-49.

[5] A. Corbo, "What Is Unsupervised Learning?," Jan. 03, 2023. [Online]. Available: https://builtin.com/machine-learning/unsupervised-learning.

[6] C. C. Aggarwal, "Neural networks and deep learning," *Springer,* vol. 10, no. 978, p. 3, 2018.

[7] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research,* vol. 15, no. 1, pp. 1929-1958, 2014.

[8] "What Is A Neural Network?" https://aws.amazon.com/what-is/neural-network/ (accessed.

[9] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer,* vol. 29, no. 3, pp. 31-44, 1996.

[10] L. J. B. B. Frey, "Adaptive dropout for training deep neural network."

[11] M. W. W. R M Holdaway 1 "Computational neural networks: enhancing supervised learning algorithms via self-organization " https://pubmed.ncbi.nlm.nih.gov/2345046/ (accessed.

[12] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329,* 2014.

[13] S. Soomro and W. Pora, "Effect of Drop-out Layers Inside an Long Short-Term Memory for Household Load Forecast Application," in *2023 5th International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, 2023: IEEE, pp. 1-7.

[14] M. R. Biswas, M. D. Robinson, and N. Fumo, "Prediction of residential building energy

consumption: A neural network approach," *Energy,* vol. 117, pp. 84-92, 2016.

[15] N. a. G. Hinton, "3d object recognition with deep belief nets," Advances in Neural Information Processing Systems, 22:1339–1347, 2009. .

[16] A. G. "A review of Dropout as applied to RNNs." https://adriangcoder.medium.com/a-review-of-dropout-as-applied-to-rnns-72e79ecd5b7b (accessed.

[17] Y. B. I. Goodfellow, and A. Courville, "Deep Learning. Massachusetts," MIT Press, 2016.

[18] J. M. Marina Segura, Adolfo Hernandez, "Machine Learning Prediction of University Student Dropout: Does Preferences Play a Role," Mathematics 2022, 10(18), 3359.

[19] G. H. Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," Journal of Machine Learning Research 15 (2014) 1929-1958, .

[20] M. L. S. Farzana, A. Baldwin, and M. U. Hossain, "Multi-model prediction and simulation of residential building energy in urban areas of Chongqing, South West China," Energy Buildings, vol. 81, pp. 161169,Oct. 2014. doi: 10.1016/j.enbuild.2014.06.007. .

[21] M. S. S. Karatasou, and V. Geros, "Modeling and predicting building’s energy use with artificial neural networks: Methods and results," Energy Buildings, vol. 38, no. 8, pp. 949958, Aug. 2006.doi: 10.1016/j.enbuild.2005.11.005.

[23] https://www.ait.de/en/deep-learning/

[24] https://www.javatpoint.com/supervised-machine-learning

[25] https://techvidvan.com/tutorials/unsupervised-learning/

# VITA

| | |
|---|---|
| **NAME** | Sanaullah Soomro |
| **DATE OF BIRTH** | 01 FEB 1998 |
| **PLACE OF BIRTH** | Karachi, Pakistan |
| **INSTITUTIONS ATTENDED** | Chulalongkorn University<br>Quaid-E-Awam University Of Engineering Science & Technology |
| **HOME ADDRESS** | Karachi, Sindh, Pakistan |
| **PUBLICATION** | Soomro S, Pora W. Effect of Drop-out Layers Inside an Long Short-Term Memory for Household Load Forecast Application. In2023 5th International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA) 2023 Jun 8 (pp. 1-7). IEEE. |
| **AWARD RECEIVED** | N/A |