*A Project Report on*

# Load Forecasting Using Deep Neural Networks

Submitted in partial fulfillment of the requirement of
the degree of

Bachelor Of Technology
Electrical & Electronics Engineering
By

K V S Praneeth(Roll No. : 202224 )

Akash Kumar (Roll No. : 202202)

*Under the esteemed guidance of* :

Dr. Satish Kumar I
Assistant Professor

**DEPARTMENT OF ELECTRICAL ENGINEERING**
**NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL**

WARANGAL - 506004
TELANGANA, INDIA

May, 2024

# Dedication

To our supervisor, **Dr Satish Kumar I**, **Assistant Professor** for his continuous guidance and support throughout the academic year.

We are grateful to **Dr. B.L. Narasimharaju, Professor and Head of Electrical Engineering Department**, for all the facilities that are required to carry out the project work.we are immensely grateful for his moral support during the span of the project.

To all the faculty members of the Department of Electrical Engineering and National Institute of Technology, Warangal who were selfless in imparting their knowledge to us over our undergraduate course. Our learning experiences have helped us immensely in accomplishing this project.

To our friends for their support throughout our times at Warangal. We were never alone although we were hundreds of miles away from home.

To our families who have sacrificed immensely to make sure we had everything we needed.

# Approval Sheet

This Project Work entitled

**Load Forecasting Using Deep Neural Networks**

by

K v s Praneeth (202224) and Akash Kumar (202202)

is approved for the degree of Bachelor of Technology.

Examiners

_____

_____

_____

Supervisor (s)

_____

Dr. Satish Kumar I

Head of the Department

_____

Chairman

_____

Date_____

Place: NIT Warangal

# Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/ data/ fact/ source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

_____

(K V S Praneeth, Roll No.-202224)

_____

(Akash Kumar, Roll No.-202202)

Date:_____

# Certificate

This is to certify that the project work entitled " Load Forecasting Using Deep Neural Networks " is a bonafide record of work carried out by Mr. K V S Praneeth (Roll No. : 202224) and Mr. Akash Kumar (Roll No. : 202202), submitted to the faculty of Department of Electrical Engineering, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Electrical & Electronics Engineering at the National Institute of Technology, Warangal during the academic year 2022-23.

Dr. B L. Narasimharaju
Head of Department,
Department of Electrical Engineering
NIT Warangal

Dr. Satish Kumar I
Assistant Professor,
Department of Electrical Engineering
NIT Warangal

# Abstract

Load forecasting is an essential part for the power system planning and operation. In this project, the focus is on Load Forecasting. The load forecasting is carried out for the power system of Panama. Convolution Neural Network (CNN), Long Short Term Memory Network (LSTM) and Bi-directional Long Short Term Memory Network (Bi-LSTM) have been used for forecasting the loads which have the advantage of learning directly from the historical data. The CNN, LSTM and BI-LSTM here uses data such as past load, weather information like humidity and temperatures. Once the neural network is trained for the past set of data it can give a prediction of future load. This reduces the capital investment reducing the equipment to be installed. The actual data are taken from the Kaggle website. The load data from January 2015 to June 2020 are used to train and test the neural network and to forecast the future. The load forecasting is done for the year 2020 and is validated for accuracy. Finally, there is a comparison drawn between CNN, LSTM and Bi-LSTM on the basis of their Mean Average Percentage Errors.

# Contents

# Abbreviations

| Abbreviation | Full Form |
|---|---|
| STLF | Short Term Load Forecasting |
| CNN | Convolution Neural Network |
| LSTM | Long Short Term Memory |
| MAPE | Mean Absolute Percentage Error |
| RNN | Recurrent Neural Networks |
| SVM | Support Vector Machine |
| ARIMA | Autoregressive Integrated Moving Average |
| ANN | Artificial Neural Network |
| Bi-LSTM | Bi-directional Long Short Term Memory |

# LIST OF FIGURES

# Chapter 1

# Introduction

## 1.1 Load Forecasting

Electric load forecasting is the process used to forecast future electric load of a certain area, given previous data of load and seasonal parameters and current and future weather information.

It provides crucial information to an electric utility which helps its future business and technical operations.

### 1.1.1 Challenges in Load Forecasting

There are many factors helping the electric load that makes correct prediction of electric load a difficult methods.[1] These factors contain "uncertainty" and have no direct relation with the end result. Moreover, the load graph is said to be a nonlinear and nonstationary process that can change very quickly due to climate, seasonal variations.

### 1.1.2 Benefits of Load Forecasting

The industry tries to control the power system such that the electric energy demand and supply are equal. This says that the more precise the forecast, the control and management of the power system is very efficient.

Other important benefits of load forecasting are:

- For accurate release of power from generating stations and hydro thermal plants .
- For minimal cost strategies and efficiency of thermal plants and for a perfect schedule for working .
- This information allows the distributors to prepare the correct necessary actions(load shedding, power purchases).

A 1% reduction in the average forecast error can save hundreds of lakhs or even crores of rupees.

### 1.1.3 Types of Load Forecasting

Load forecasting can be divided into three major categories[2]:

- **Long Term Load Forecasting(LTLF):** Long-term demand forecasts span from eight years ahead up to fifteen years. They play a prominent role in the steps of generation, transmission and distribution network schedule in a power system. The main goal of power system planning is to understand and analyze an economic expansion of the equipment and facilities to produce future electric demand for people.
- **Medium term load forecasting(MTLF):** MTLF forecast for a time span of (1 – 12 months). MTLF forecasting is affected mostly by growth factors, i.e., factors that can change demand such as increase in new loads, climatic changes, demand patterns of large industries , and timely maintenance conditions of large consumers.
- **Short term load forecasting (STLF):** STLF forecast for merely hourly time in total system load. In addition to predicting the hourly values of the system load, an STLF also predicts the daily peak load of the system, the values of system load at whenever we need them, the hourly or half hourly values of energy, the daily and weekly energy.

## 1.2 Short Term Load Forecasting

Short-term load forecasting (STLF) is the process of estimating the future electricity demand for a very short time interval , usually from a few hours to a few days.

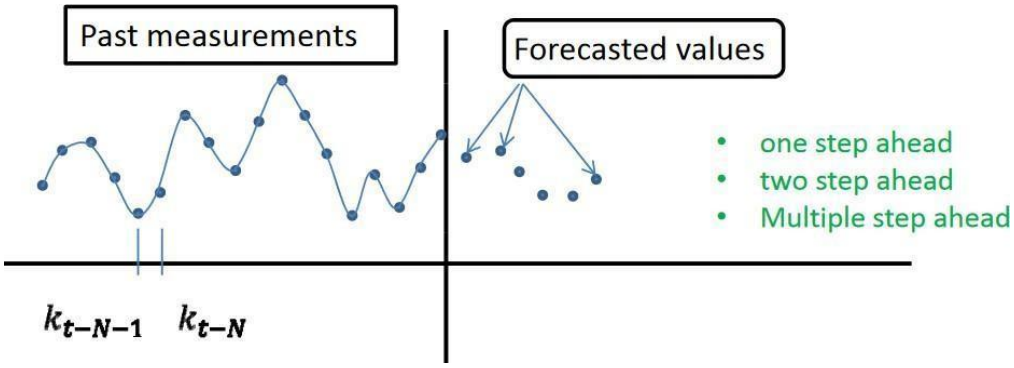



Figure 1.1: load forecasting

### 1.2.1 Need of Short Term Load Forecasting

There are several reasons why short-term load forecasting is necessary:

**Efficient resource planning:** Load forecasting helps utility companies plan for the resources they will need to meet demand. By accurately predicting how much energy will be needed in the near future, they can ensure that they have enough generating capacity and distribution infrastructure to deliver that energy.

**Cost savings:** Accurate load forecasting can help utility companies avoid the cost of running expensive backup generators or buying power on the spot market when demand unexpectedly exceeds supply.

**Grid stability:** Load forecasting is also important for maintaining grid stability. When there is too much demand for electricity and not enough supply, the grid can become unstable, leading to blackouts and other problems. By predicting demand accurately, utility companies can take steps to prevent these situations from occurring.

**Renewable energy integration:** With the increasing use of renewable energy sources like wind and solar, load forecasting has become even more important. Load forecasting can help utility companies plan for fluctuations in renewable energy production and ensure that there is enough backup power available when needed.

Overall, short-term load forecasting is a critical tool for utility companies to ensure that they can meet customer demand, maintain grid stability, and operate efficiently.

### 1.2.2 Factors affecting Short Term Load Forecasting

To construct a promising forecasting model one should have a good knowledge of the features of the system. There are various factors which may change the pattern of the customer load and influence the losses in lines of transmission.

These factors are explained below.

**Time factor:** Time is the most important factor in short term load forecasting because its impact on consumer load is highest. From observing the load curve of several different grid stations, it is found that the load curve has “time of the day” property, also it has “day of week”, “week of month” and “month of season” property.

**Economic Factor:** Price of electricity and the people's buying capability also have an impact on the usage of electricity, hence the more expensive the electricity is, the less it will be used by the domestic consumers. Thus, for load forecasting we must consider the country's economic situation (GDP), and must have a glance at the industrial development of the country for long term load forecasting.

**Weather:** Weather is the most important independent variable for load forecasting. The effect of weather is most prominent for domestic and agricultural consumers, but it can also alter the load profile of industrial consumers. Load forecasting models use weather forecast and other factors to predict the future load, thus, to minimize the operational cost. Weather is often cited as the tipping point that can cause unreliability in the system by decreasing the efficient supply of power. Unpredicted sea breeze, after moon thunderstorms, back door fronts are some of the environmental factors that can decrease the temperature and thus cause overestimated load forecasts. Thus, we are producing more power than required. Temperature can also alter the conductivity of the transmission lines. Thus, temperature can affect the overall carrying capability of the transmission lines. High temperature can increase not only the resistance of the transmission lines, but also it can alter the reactance of the line, due to temperature induced expansion of the length of transmission line.

The weather factor includes the following:

- Temperature
- Humidity
- Precipitation
- Wind speed and wind chill index
- Cloud cover and light intensity

## 1.3 Various Methods Used for Short Term Load Forecasting

### 1.3.1 Multiple regression

Multiple regression evaluation for load forecasting uses the technique of weighted least-squares estimation. Based totally in this analysis, the statistical relationship among total load and climate situations in addition to the day type impacts may be calculated. The regression coefficients are computed by using an similarly or exponentially weighted least- squares estimation using the described amount of previous records[3].

### 1.3.2 Autoregressive Integrated Moving Average (ARIMA)

Autoregressive Integrated Moving Average (ARIMA) is a statistical algorithm used for time series analysis and forecasting. The ARIMA model is based on three components: the autoregressive (AR) component, the integrated (I) variable, and the moving average (MA) variable. The autoregressive variable (AR) uses previous values of the series to guess the upcoming values, assuming that the future value of the series is linearly dependent on its past values. It is the integrated component (I) that explains why the time series is non-stationary by differencing the series to obtain a stationary series. The moving average component (MA) forecasts future values by utilizing historical forecast mistakes assuming that the future value of the series is linearly dependent on past errors. When analyzing time series data, the ARIMA method is usually used to find trends and patterns, and seasonal variations in the data, and to make predictions about future values based on these patterns[4].

### 1.3.3 General Exponential Smoothing

The method uses historical data to calculate the average of past demand values and combines this with a smoothing parameter to produce a forecast for the next period.

The general exponential smoothing method involves three main components: level, trend, and seasonal factors. The demand's average across time is represented by the level component, while its direction and rate of change are represented by the trend component, and the seasonal factors represent regular patterns that repeat over time.

To generate a short-term load forecast using this method, the historical demand data is first decomposed into its level, trend, and seasonal components. The forecast for the next period is then calculated by applying the smoothing parameter to each component, and then recombining the components.

The weight assigned to the most recent data in the forecast is determined by the smoothing value. A more responsive forecast that responds rapidly to shifts in demand is produced by increasing the smoothing parameter, which gives greater weight to recent data. Conversely, a smaller smoothing parameter gives past data more weight, making the forecast less susceptible to short-term volatility and more steady.

Overall, short-term load forecasting using general exponential smoothing provides a simple and effective method for predicting electricity demand in the short-term, with the ability to incorporate various seasonal and trend factors[5].

### 1.3.4 Support vector Machine Regression

One sort of supervised learning technique is the support vector machine (SVM), which divides data into classes or groups by identifying the hyperplane that best divides the data. SVM regression is utilized in load forecasting to identify the hyperplane that most closely matches the historical data and can be used to forecast demand over a specific time period.

To generate a short-term load forecast using SVM regression, the historical data is first preprocessed by cleaning and normalizing it. The data is then split separated training and testing sets, the former being used to train the SVM model and the latter to assess the model's performance.

The demand values that correspond with the historical data are used to train the SVM model. The hyperplane that the model determines best fits the data can be used to forecast the demand for the next period. The model can be adjusted by selecting different kernel functions, regularization parameters, and other hyperparameters to improve the accuracy of the forecast.

The SVM model can be used to forecast demand for a specific time frame after it has been trained The input variables such as temperature, humidity, time of day, day of the week and holidays are fed into the model, and the output is the predicted demand.

Overall, SVM regression-based short-term load forecasting offers a versatile and effective technique for estimating electricity consumption, with the capacity to integrate many factors that influence demand. The method can be adjusted to fit the specific requirements of the energy system and can provide accurate forecasts for short-term planning and operations.

### 1.3.5 Fuzzy Logic

This method uses historical data to identify the relationship between demand and various factors that influence it, such as the day's temperature, humidity, and length. Using membership functions, the input variables are mapped to a collection of linguistic phrases, and a set of rules is created that relates the input variables to the output variable (i.e., demand). The input variables are then fed into the set of rules, and the degree of support for each rule is calculated using fuzzy logic. The result is a set of fuzzy outputs, which are aggregated and defuzzified to produce a single output representing the predicted demand.

The performance of the fuzzy logic-based short-term load forecasting method can be improved by adjusting the membership functions, the rule set, and the aggregation and defuzzification methods. Additionally, the method can be combined with other forecasting methods, such as regression analysis or neural networks, to further improve the accuracy of the forecast. Overall, this method provides a flexible and powerful approach to predicting electricity demand, particularly when there is incomplete or uncertain data. It can be adjusted to fit the specific requirements of the energy system and can provide accurate forecasts for short-term planning and operations. [7].

### 1.3.6 Artificial Neural Network

Short-term load forecasting based on artificial neural networks (ANN) is a method used to predict electricity demand that utilizes a type of machine learning algorithm. This method involves training an ANN using historical load data as input and corresponding demand values as output. The ANN learns to identify patterns and relationships in the data, and is then used to make predictions based on new input data.

Using ANN for short-term load forecasting usually requires multiple phases. To start, past data is gathered and preprocessed to eliminate any errors or outliers. The data is then separated into training and testing sets, where the ANN is trained on the training set and its performance is assessed on the testing set. Next, the number of input and output nodes, as well as the number of hidden layers and nodes, must be determined in order to create the ANN architecture. After the architecture is established, the ANN is trained using an optimization approach, like backpropagation, which modifies the network's weights and biases in order to reduce the error between the demand values that are anticipated and those that actually occur.

Based on fresh input data, the trained ANN can be used to forecast short-term loads. This entails providing the ANN with the necessary input variables, including the temperature and time of day, in order to produce an output that represents the expected demand. Metrics like the root mean squared error and the mean absolute percentage error can be used to assess how accurate the forecast is. All things considered, ANN-based short-term load forecasting offers a strong and adaptable method of estimating electricity demand that can be tailored to the particular needs of the energy system. [8]-[11]

### 1.3.7 Convolutional Neural Networks

Convolutional Neural Networks (CNN) have been applied successfully in load forecasting due to their ability to automatically learn useful features from raw input data. In load forecasting, CNN can be used to model the temporal and spatial dependencies of load data, where the input data includes historical load data, weather data, and other relevant data.

The convolutional layers in CNN can capture the spatial correlations between different regions, while the recurrent layers can model the temporal dependencies over time. By training a CNN model on historical load data, it can learn to identify patterns and trends in the data and make predictions on future load demand. This can help power grid operators to make better decisions on energy generation and distribution, which can lead to more efficient and cost-effective operation of the power grid.[12]-[13]

### 1.3.8 Long Short Term Memory

Neural networks of the Long Short-Term Memory (LSTM) type are frequently employed for time-series forecasting, which includes load forecasting. LSTM networks can be trained on historical data in load forecasting to identify patterns and trends in the evolution of power demand. The network can then use this information to make predictions about future demand based on current and past inputs. LSTMs are particularly useful for load forecasting because they can handle long-term dependencies in the data, which is important for accurately predicting electricity demand over extended periods. They are also capable of learning and adapting to changing patterns in the data over time, making them well-suited for dynamic and unpredictable load forecasting scenarios. Overall, LSTM networks are a powerful tool for load forecasting, and are widely used in the energy industry to help utilities and grid operators plan and manage their operations more effectively [14].

### 1.3.9 Bi-Directional Long Short Term Memory

Bi-directional Long Short-Term Memory (Bi-LSTM) networks represent an advanced variant of LSTM networks. These networks excel in time-series forecasting, particularly in load forecasting for the energy sector. Through the examination of past data, Bi-LSTM models are able to detect patterns and trends in the demand for power over time, as well as dependencies in both forward and backward directions. This reciprocal approach allows them to gain a deeper understanding of long-term dependencies, which is crucial for accurate predictions of electricity demand over extended periods. Moreover, Bi-LSTM networks exhibit adaptability to evolving data patterns, making them well-suited for

dynamic and unpredictable load forecasting scenarios. As a result, they have become a powerful tool for load forecasting, leveraging the strengths of bidirectional processing to enhance operational planning and management in the energy industry

## 1.4 Aim of the Project

A short-term load forecasting model based on Deep Learning Networks using Convolutional Neural Network (CNN), Long Short Term Memory (LSTM), and Bi-LSTM networks is proposed to address the issue of low forecast accuracy using traditional methods. The model predicts the peak load forecast for the upcoming hour and uses Mean Absolute Percentage Error (MAPE) to assess the goodness of fit of these algorithms.

# Chapter 2

# Literature Review

In this literature review, we will explore the current state of research on STLF, including the various methods and techniques used for forecasting including statistical methods, artificial intelligence, and machine learning, the challenges and limitations of STLF, and the latest trends and advancements in the field.

***S. Hosein and P. Hosein [15],***

outlines a novel deep neural network (DNN) method for short-term electrical load forecasting (STLF). They presented a thorough examination of various deep learning models and contrasted deep architectures with conventional approaches used to solve the STLF problem. The technology employed was Recurrent Neural Network-Long Short Term Memory (RNN-LSTM), which has a slower training and running duration. 2.36 was the Mean Absolute Percentage Error (MAPE).

***Ming-Guang Zhang [16],***

A new approach for short-term load forecasting in the electric power system is described, based on Support Vector Machines (SVM) regression. Based on Statistical Learning Theory, SVM is a cutting-edge, potent machine learning technique (SLT). The SVM algorithm reduces training time and increases result accuracy; nevertheless, it is not appropriate for a large number of datasets. This approach resulted in a 5.6% MAPE.

***N. Kamel and Z. Baharudin [17],***

suggested a method for STLF that forecasts future loads one week in advance utilizing the signal modeling approach. By solving load projections for load demand, the suggested solution introduced a signal modeling through the use of Burg's method. The technique is a digital signal processing strategy in which the objective is to predict PL(n) over a different interval given a signal (historical data) PS(n) known over a specific interval of time. The prediction was made using the autoregressive approach, and the MAPE was 3.9. The time-series' history is insufficient to forecast the future. To obtain accurate projections, several more features need be taken into account.

***S. H. Rafi, Nahid-Al-Masood, S. R. Deeba and E. Hossain [18],***

Convolutional neural networks (CNNs) and long short-term memory (LSTM) networks were combined to provide a solution for short-term electrical demand forecasting in the power system of Bangladesh. The method showed improved load forecast accuracy and precision. With a Mean Absolute Percentage Error (MAPE) of 2.31, the CNN-LSTM model predicts future load demand over a lengthy period of time with effective handling of long time series data of electric load.

***C. Liu, Z. Jin, J. Gu and C. Qiu [19],***

Describe the structure and training process of a Long Short-Term Memory (LSTM) network designed for short-term load forecasting. The model is designed to predict the load profile for the next 24 hours. The load data used in this study covers a full year, recorded every week from Monday to Sunday at 60-minute intervals, without addressing regular or irregular load fluctuations.

***M. S. Hossain and H. Mahmood [20],***

In order to estimate short-term electrical load, the study created 2 forecasting models employing long short-term memory neural networks . Whereas the second model predicts multi-step intraday rolling horizons, the first model only predicts a single step ahead load. In addition to weather data from the considered geographic area, the load's time series was employed. An extreme learning machine (ELM) and a generalized regression neural network (GRNN) were used to compare the performance of the LSTM NN models.

Overall, this review provides a comprehensive understanding of the current state of short-term load forecasting and offers insights into the future directions of research in those area.

# Chapter 3

# Convolution Neural Network

Short-term load forecasting, or hourly load demands for the following day, is accomplished by using convolution neural networks to predict peak load demands for each hour. This chapter begins with a discussion of the notion of convolution neural networks and ends with a thorough examination of them.

The main steps involved in predicting load forecasting using Convolution Neural Network are Data Preparation, Data Preprocessing, Defining the CNN Architecture, Training the CNN model, evaluating the performance of the model and making load forecasts which are explained in detail.

## 3.1 Brief Historical Review

The history of CNNs dates back to the 1980s, when Yann LeCun and others began working on neural networks that could recognize handwritten digits. These early networks used a convolutional layer to filter images, reducing the number of parameters and improving the network's ability to recognize patterns. However, the computational resources available at the time were limited, and training these networks was a slow and difficult process.

In the 1990s, researchers such as David Hubel and Torsten Wiesel discovered that the visual cortex of the brain uses a hierarchical structure to analyze visual information. This led to the development of more sophisticated neural networks that could mimic this hierarchical structure, with layers of neurons that extract increasingly abstract features from the input data[12].

The breakthrough for CNNs occurred in 2012 when a group under the direction of Alex Krizhevsky used a deep convolutional neural network named AlexNet to win the ImageNet Large Scale Visual Recognition Challenge. On the job, AlexNet was able to attain state-of-the-art performance.

of image classification, with an error rate that was significantly lower than the previous best result.

Since then, CNNs have become a standard tool in computer vision and other areas of machine learning. They have been used for a wide range of applications, including object recognition, image segmentation, face recognition, and more.

## 3.1 Convolution Neural Network Architecture

Fully connected layers, pooling layers, and convolutional layers make up CNN's fundamental architecture. The purpose of these layers is to extract characteristics from the input data and make predictions based on these features. Here is a brief description of each layer:

**Convolutional layers:** The convolutional layer applies filters to the input data to extract features. The filters are learned during training and slide over the input data, applying a dot product operation at each location. The output of this operation is a feature map that highlights certain aspects of the input data.

**Pooling layers**: By combining nearby data, the pooling layer lowers the dimensionality of the feature maps. Max pooling is the most used pooling procedure; it takes the maximum value within a window of neighboring values. This operation reduces the size of the feature maps and helps prevent overfitting.

**Fully connected layers**: The fully connected layer takes the flattened output from the previous layers and makes the final prediction based on these features.

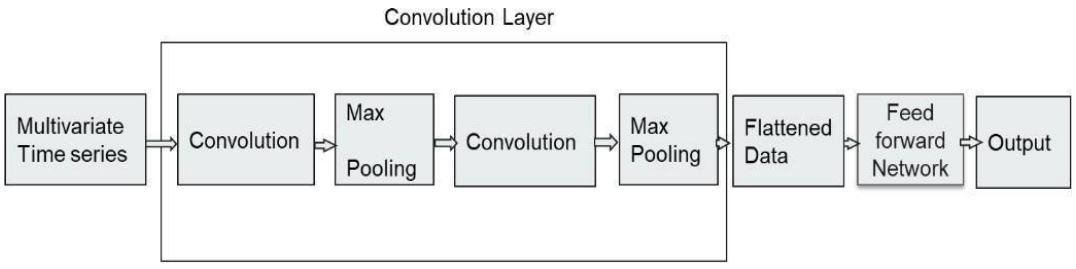



Figure 3.1: CNN Architecture

## 3.2 CNN Network Hyperparameters

**Filter Hyperparameters:**

**Kernel:** The kernel, also known as a filter or a weight, is a learnable matrix used in the convolutional layer to extract features from the input data. The size of the kernel is typically small, such as 3x3 or 5x5, and it slides over the input data to produce a feature map. The values of the kernel are learned during the training process, and the same kernel is applied to all regions of the input data. By stacking multiple convolutional layers with different kernels, the network can learn increasingly complex features from the input data.

**Padding:** Before the input data is sent through a convolutional layer, padding is the process of adding more rows and columns of zeros**.** Padding helps preserve the spatial dimensions of the input data after convolution, as the filters in the convolutional layer can shrink the dimensions of the output feature map. The most common type of padding is zero- padding, where zeros are added around the borders of the input data

**Stride:** The convolutional filter's stride is the number of pixels that the window traverses across the input data. A higher stride value corresponds to a lower output feature map and a reduced computational cost, while a smaller stride value preserves more spatial information in the output feature map. Stride is typically set to 1 for convolutional layers and pooling layers, but can be adjusted to optimize the performance of the network.

**Pooling layer parameters**

**Pooling:** The method of pooling involves lowering the spatial dimensions of the feature map in order to downsample the output of a convolutional layer. Max pooling and average pooling are the two pooling operations that are most frequently utilized. While the average value is preserved in average pooling, the maximum value within each window is retained in max pooling, which involves sliding a window over the feature map. Pooling helps to reduce the computational cost of the network, prevent overfitting, and capture the most important features of the input data.

**Activation Function used in CNN**

**ReLU:**

An activation function called the Rectified Linear Unit (ReLU) is used to add non-linearity to a CNN's hidden layers. The formula for ReLU is f(x) = max(0,x), where x is the function's input. If the input value is positive, the ReLU function simply returns that value; if not, it returns 0. Accordingly, ReLU passes through all positive values unchanged and sets all negative values to zero.

Because ReLU is computationally inexpensive and helps avoid the vanishing gradient issue that can arise in deep networks, it is favored over alternative activation functions. When the gradient of the loss function gets too tiny as it propagates backward through several network layers, it is known as the "vanishing gradient problem," which makes it challenging to train the network efficiently.

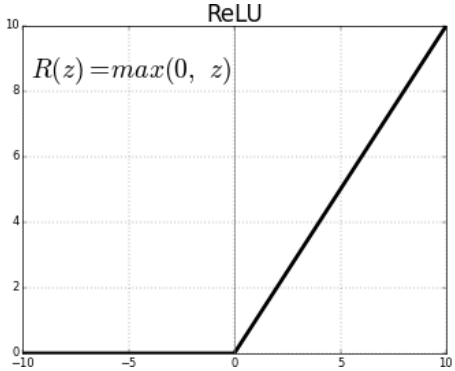



Figure 3.2: ReLU Activation Function

**Dropout:**

In order to avoid overfitting in Convolutional Neural Networks (CNNs), dropout is a regularization technique that is frequently employed. Dropout can be applied to the output of any layer in a CNN, including the fully connected, pooling, and convolutional layers..

Dropout works by randomly eliminating (setting to zero) a specific percentage of the network's neurons during training. This drives the network to acquire more resilient features that are helpful for generating predictions, even in the event that some neurons are absent.. During testing, all neurons are used, but their outputs are scaled by the dropout rate to ensure that the expected value of the output remains the same as during training.

In practice, dropout is implemented by introducing a dropout layer after each layer that we want to apply dropout to. The dropout layer randomly sets a fraction of the inputs to zero with probability p, where p is a hyperparameter that controls the dropout rate. The remaining inputs are scaled by a factor of 1/(1-p) to ensure that the expected value of the output remains the same.

**Optimization Method used in CNN**

**Adam:**

Adam is a well-liked optimization technique that is employed in the training of many deep learning models, including CNNs. Since it is an adaptive learning rate optimization approach that calculates adaptive learning rates for each parameter, the term "Adam" stands for Adaptive Moment Estimation.

The Adam optimizer combines two techniques: adaptive learning rates and momentum. Adaptive learning rate means that the learning rate is adjusted automatically for each weight in the network based on the past gradients. This helps the optimizer converge faster and more efficiently, as it adapts the learning rate to the specific parameter.

Momentum, on the other hand, helps to smooth out the optimization process by taking into account the previous gradient updates. It introduces a "memory" effect that helps the optimizer to overcome local minima and reach the global minimum faster.

The Adam optimizer measures the squared gradients (second moment) and the exponentially decaying average of previous gradients (first moment). With each iteration, the network's parameters are updated using these estimations.

In summary, the Adam optimizer is a powerful algorithm that combines the benefits of adaptive learning rates and momentum to optimize the weights and biases of a CNN in an efficient manner. It is widely used in deep learning because it is robust and converges quickly, even when dealing with high-dimensional data.

## 3.3 Learning Process

The learning process of a CNN for load forecasting typically involves the following steps:

**Data Collection**: The first step is to collect historical electricity consumption data for the area of interest. This data can include information about the time of day, the moment in a week, the season, and other relevant factors that may affect electricity demand.

Data Preprocessing:After the data is gathered, preprocessing is required in order to get it ready for CNN training. This may entail dividing the data into training and validation sets, normalizing it to a specified range, and transforming it into a time-series format.

**CNN Architecture:** The CNN architecture needs to be designed and defined based on the task that is specified and data at hand. This can involve selecting the number and type of convolutional layers, pooling layers, and other layers such as dropout or batch normalization.

**Training the Model:** The next step is to train the CNN model using the training data set. During training, the model adjusts its weights and biases to minimize the error between the predicted and actual electricity demand values. During the training phase, the model is fed training data iteratively, and the weights and biases are updated in accordance with the gradients of the loss function.

**Model Evaluation:** After the model has been trained, it needs to be evaluated using the validation data set. This stage aids in evaluating the model's performance and locating any possible problems, such as under- or overfitting.

**Fine-tuning:** Once the model has been evaluated, it may be necessary to fine-tune the CNN architecture or adjust hyperparameters to further improve its performance. This can involve tweaking the number of layers, changing the learning rate, or adjusting other model parameters.

**Deployment:** Once the model has been trained and evaluated, it can be deployed to make load forecasts for new data. This involves feeding the new data through the trained model and obtaining predictions for the future electricity demand.

In summary, the learning process of a CNN for load forecasting involves collecting and preprocessing data, designing and training the model, evaluating its performance, fine-tuning the model, and deploying it to make forecasts for new data[13].

## 3.4 Advantages and Disadvantages of Using CNN in Short Term load forecasting

**Advantages:**

- In order to capture the intricate temporal patterns in the load data, CNNs can automatically extract pertinent features from the input data.
- CNNs can be trained on big datasets and are very scalable, making them suitable for dealing with the vast amount of data in short-term load forecasting.
- CNNs can handle multiple input variables and capture the non-linear relationships between them, which is essential for accurate load forecasting.
- CNNs are highly parallelizable, which makes them well-suited for implementation on modern GPUs, resulting in faster training and inference times.

**Disadvantages:**

- CNNs require a large amount of data to achieve good performance, which can be challenging to obtain in the case of short-term load forecasting.
- CNNs can be computationally expensive to train, requiring significant computational resources, which can be expensive to acquire and maintain.
- CNNs are susceptible to overfitting, especially when the dataset is small, which can result in poor performance on unseen data.
- CNNs require careful hyperparameter tuning,such as the filter size, learning rate, and number of layers, all of which can take a long time. and challenging to get right.

# Chapter 4

# Long Short Term Memory Neural Network

## 4.1 Introduction

Recurrent neural networks (RNNs) with Long Short-Term Memory (LSTM) are specifically engineered to manage long-term dependencies in sequential data, such time series or natural language phrases. It is challenging for traditional RNNs to learn long-term dependencies due to the phenomenon known as "vanishing gradients," in which the gradients that are back propagated during training eventually get exponentially small.

In order to solve this issue, LSTM was developed. It uses gating mechanisms to regulate the flow of information into and out of the memory cell and a memory cell to store information for extended periods of time. The network can learn and recall long-term dependencies by using the memory cell's ability to selectively forget or remember data from earlier time steps.

## 4.2 Brief History Review

The development of recurrent neural networks (RNNs) for sequential data processing tasks in the late 1980s and early 1990s is credited with giving rise to LSTM (Long Short-Term Memory). Because RNNs could retain a memory of past inputs and use this information to predict future inputs, they were especially well-suited for these tasks.

But shortly, scientists identified a flaw in conventional RNNs: the vanishing gradient issue. This happens when backpropagating gradients in the network get very small over time, which makes it hard for the network to figure out long-term dependencies. Consequently, the capacity of RNNs to identify intricate patterns within sequential data was constrained.

In 1997, Jürgen Schmidhuber and Sepp Hochreiter introduced the LSTM architecture in order to address the vanishing gradient issue. The addition of a memory cell that could selectively remember or forget information from earlier time steps was the main breakthrough of LSTM. The network was able to selectively recall or forget information as needed thanks to gating mechanisms that managed the flow of information into and out of the memory cell.

The original LSTM architecture had three gating mechanisms: three gates: an output, an input, and a forget. Information entering the memory cell was managed by the input gate, while information leaving the cell was managed by the forget gate. Information sent from the memory cell to the network's output was managed by the output gate. With the use of these gating mechanisms, the LSTM was able to retain or forget information according to the network's prior state and current input.

Over the years, researchers have made various modifications and improvements to the original LSTM architecture, leading to several different variants such as Peephole LSTM, Gated Recurrent Unit (GRU), and Variational LSTM. These variants have different gating mechanisms, activation functions, and memory cell structures, but all share the same goal of addressing the vanishing gradient problem and improving the performance of RNNs in sequential data tasks.

## 4.3 Long Short Term Memory Neural Network Architecture

The essential shape of an LSTM, as shown in fig .An LSTM resembles an RNN in its repeated structure, but its modules feature unique internal components that are evident in determination. The cell state, which transmits data alongside the chain, is an essential component of an LSTM. Several components known as gates are used to alter or remove information from the cell state. The forget gate, input gate, and output gate are the three gates that make up an LSTM module.

A sigmoid layer in the forget gate uses the current input (xt) and the prior hidden state (ht−1) to produce an output between 0 and 1.

This layer makes the final decision on which statistics should be retained or deleted. A value of zero indicates disregard for the prior data, while a value of one indicates retention of the prior records. The output of the forget gate (ft) is expressed as

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \quad \text{where}$$

$W_f$ = weight matrix between forget gate and input gate , $b_f$ =connection bias at t , t=timestamp

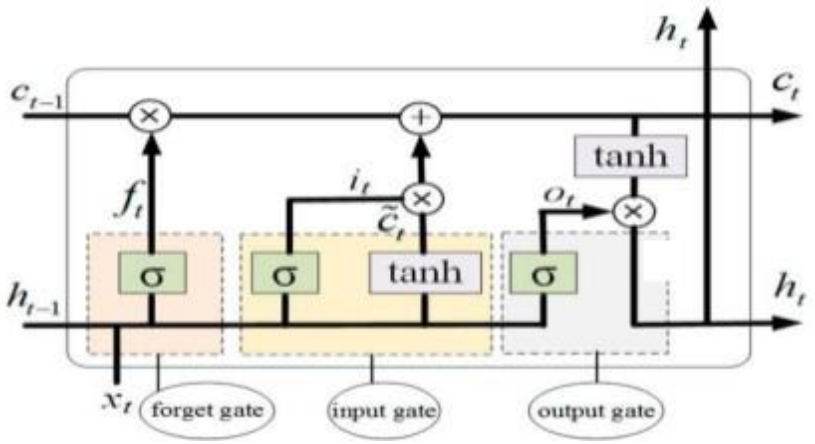



Figure 4.1: Long short-term memory (LSTM) architecture

Afterwards, the forget gate decides what data will be added to the cell state by using a tanh function and a sigmoid function. The inputs for both functions are xt and ht−1. The tanh function controls the network by squashing values between -1 and +1, while the sigmoid's output establishes the significance of the present data. The two outputs are then multiplied.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\widetilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad \text{Where,}$$

$i_t$=input gate at t, t= timestamp, $W_i$= weight matrix of sigmoid operator between input gate and output gate, $b_i$= bias vector at t, $C`_t$ =value generated by tanh, $W_c$=Weight matrix of tanh operator between cell state information and network output ,$b_c$= bias vector at t

The data in the cell state is updated using the output from the input and forget gates.

The forget gate output and the current cell state are multiplied pointwise to accomplish this. The multiplication will also return zero if ft is zero, indicating complete dropout of the preceding value. If ft is 1, on the other hand, it is kept. Afterwards, the cell state is updated by the pointwise addition as

$$C_t = f_t \times C_{t-1} + i_t \times \widetilde{C}_t. \quad \text{Where,}$$

$f_t$=forget gate at t, $i_t$=input gate at t, $C_{t-1}$= Previous cell state

The output gate chooses the final output at the last stage. The following hidden state, ht, is also represented by this output. In this gate, the current cell state Ct is sent through a tanh function, and a sigmoid function receives ht−1 and xt as input. The information that the hidden layer will include is then ascertained by multiplying the sigmoid and tanh outputs.

$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right) \quad \text{Where,}$$

$o_t$= output gate, $W_o$= Weight matrix of output gate $b_o$= bias , $h_t$=LSTM output

## 4.4 LSTM Network Hyperparameters

**Number of LSTM units**: The number of LSTM units, also known as the hidden size, determines the size of the LSTM cell.The ability of the model to identify intricate patterns in the data can be improved by increasing the number of LSTM units. A balance between model capacity and generalization is crucial since larger models may be more prone to overfitting.

**Number of LSTM layers**: The depth of the network is determined by the number of LSTM layers. The ability of the model to recognize long-term dependencies in the data can be improved by adding more layers. Deeper models, however, could be more difficult to train and need additional data to avoid overfitting.

**Dropout rate**: A regularization method called dropout removes a certain percentage of LSTM units at random during training. By pushing the model in this way, overfitting can be avoided. to learn more robust features. The dropout rate determines the fraction of units to drop outA greater dropout rate may cause regularization to increase, but it may also cause the model's training to lag.

**Learning rate**: The learning rate determines how quickly the model adjusts its parameters during training. A higher learning rate can make the model converge faster, but it may also make the training unstable and prevent the model from reaching the optimal solution. A lower learning rate can improve stability but may require more iterations to converge.

**Batch size**:The amount of samples handled during each training iteration is determined by the batch size. Higher batch sizes can improve the effectiveness of training by processing more samples in parallel, but it can also require more memory and may make the model more sensitive to noisy samples. A smaller batch size can reduce memory requirements but may slow down the training process.

**Sequence length**: The sequence length determines the number of time steps the model processes in each iteration. A longer sequence length can increase the model's ability to capture long-term dependencies, but it can also require more memory and computational resources. A shorter sequence length can reduce memory requirements but may limit the model's ability to capture long-term dependencies.

**Activation functions**: The activation functions determine how the LSTM cell processes input and output data. Common activation functions include sigmoid, tanh, and relu. Choosing the right activation functions can help the model learn more effectively and prevent vanishing or exploding gradients.

**Sigmoid Activation Function**

The sigmoid activation function is a type of non-linear function commonly used in neural networks, including in Long Short-Term Memory (LSTM) networks.

Any value entered into the sigmoid function can be converted to an output value between 0 and 1 using its distinctive S-shaped curve. The sigmoid function is defined as:

$$f(x) = \frac{1}{1 + e^{-x}}$$

In the context of LSTM, the sigmoid function is used in two different places: the input gate and the forget gate.

The input gate controls the amount of fresh data that is added to the cell state. The current input and the previously concealed state are its two inputs. After multiplying by each other, the two inputs are run via a sigmoid function. The amount of current input that is contributed to the cell state is then managed by the output that is produced, which is referred to as the "input gate activation".

The forget gate is responsible for determining how much information is discarded from the state of the cell. The current input and the previously concealed state are its two inputs. After multiplying by each other, the two inputs are run via a sigmoid function. The resulting output, called the "forget gate activation", is then used to control how much of the previous cell state is retained.

The sigmoid function is a good choice for the input and forget gates because it outputs values between 0 and 1, which can be interpreted as probabilities. This means that the gates can control how much information is added or discarded in a smooth and continuous way.

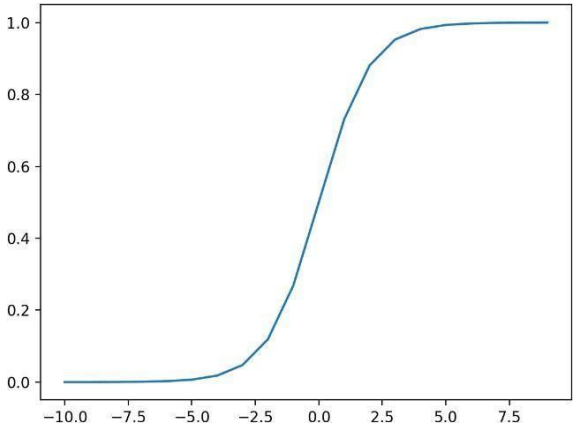


Figure 4.2: Sigmoid Activation Function

In summary, A nonlinear function that is frequently utilized in the input and forget gates of long short-term memory (LSTM) networks is the sigmoid activation function. Its distinctive S-shaped curve, which represents probabilities, links input values to output values between 0 and 1. By utilizing forget gates and the sigmoid function in the input, LSTM networks can control how much information is added or discarded from the cell state in a smooth and continuous way.

**Tanh Activation Function**

A nonlinear function that is frequently utilized in the input and forget gates of long short-term memory (LSTM) networks is the sigmoid activation function. Its distinctive S-shaped curve, which represents probabilities, links input values to output values between 0 and 1. By utilizing forget gates and the sigmoid function in the input,, and is defined as:

$$f(x) = \frac{\left(e^{x} - e^{-x}\right)}{\left(e^{x} + e^{-x}\right)}$$

In an LSTM network, the tanh activation function is used in several places.

Firstly, it is used as the activation function for the gates in the LSTM cell. The gates are responsible for regulatingthe information flowing through the cell and consist of element-wise multiplication and sigmoid activations, which generate values between 0 and 1. The final output of the gate is then obtained by multiplying the output of the sigmoid activation by the output of the tanh activation. The output of the gate is guaranteed to be between -1 and 1 by the tanh activation function, which helps to prevent the gradients from vanishing or exploding during training.

Secondly, The activation function for the cell state itself is the tanh activation function. The input, forget and input gates, the previous cell state, and tanh activations are all used to update the cell state. Tanh activations guarantee that the values are between -1 and 1. This helps to keep the cell state stable and prevent it from growing too large or too small during training.

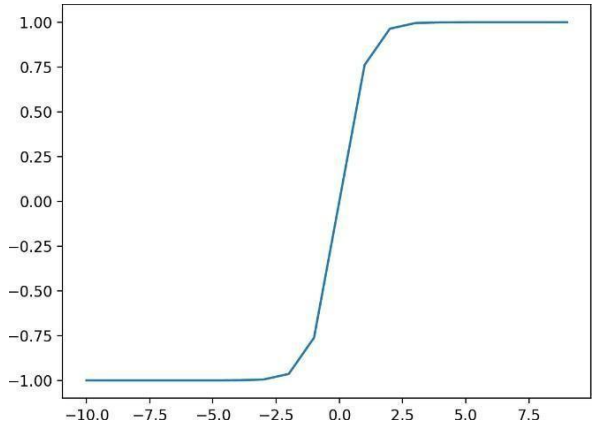



Figure 4.3: tanh Activation Function

Overall, the tanh activation function is an important component of LSTM networks, helping to regulate the flow of information and prevent the gradients from vanishing or exploding during training.

## 4.4 Learning Process

The learning process of LSTM involves the following steps:

**Data Preprocessing:** this learning process of LSTM for load forecasting is data preprocessing. This step involves preparing the input data to be used in the LSTM model. The data is normalized to scale it to a common range, making it easier for the LSTM to learn from the input features. The normalized data is then split into training, validation, and test sets.

**Model Architecture:** In this step, the architecture of the LSTM model is defined. This involves deciding on the quantity of time steps to be used, the number of LSTM layers, and the number of neurons in each layer. The model learns to recognize patterns and relationships in the incoming data by feeding it into the LSTM layers**.**

**Model Training:** The LSTM model is trained using the training set. During training, the LSTM learns to predict future energy demand based on historical data.

The training process involves minimizing the difference between the predicted and actual output using backpropagation.

**Hyperparameter Tuning:** Hyperparameters, like learning rate, batch size, and number of epochs, are model parameters that are not learned during training. Finding the best values for these parameters is known as hyperparameter tuning, and it is used to enhance the LSTM model's performance.

**Model Evaluation:** After the model is trained, it is evaluated on the validation and test sets. The evaluation metrics typically used for load forecasting comprise the coefficient of determination (R-squared), mean absolute error (MAE), and root mean square error (RMSE). The model is iteratively evaluated and refined until satisfactory performance is achieved.

**Model Deployment:** Once the LSTM model has been trained and evaluated, it is deployed to make predictions on new data. In the case of load forecasting, the LSTM takes in the historical load data as input and predicts the future load demand.

Overall, the learning process of LSTM for load forecasting involves data preprocessing, model architecture, model training, hyperparameter tuning, model evaluation, and model deployment. This process is iterative, with each step informing the next to improve the accuracy and performance of the LSTM model.

## 4.5 Advantages and Disadvantages of Using LSTM in Short Term load forecasting

**Advantages:**

- *Ability to handle long-term dependencies*: One of the key advantages of LSTM is its ability to handle long-term dependencies in time series data. This makes it well-suited for load forecasting, where the demand patterns may vary over longer periods of time.

- *Non-linear modeling*: Since LSTM is a non-linear model, it can represent intricate correlations between the goal and input variables. This makes it suitable for modeling the highly nonlinear and dynamic nature of energy demand.
- *Data-driven approach*: LSTM is a data-driven approach to modeling, which means that it can learn patterns and relationships in the data automatically. This makes it well-suited for load forecasting, where there may be several factors influencing energy demand.

**Disadvantages:**

- *High computational complexity*: LSTM models have a high computational complexity, which can make them slower to train and evaluate compared to other models.
- *Data requirements:* For LSTM models to train efficiently, a lot of historical data is needed. There may occasionally be insufficient previous data available to provide precise load forecasts.
- *Overfitting:* Because LSTM models are prone to overfitting, they may end up performing poorly on fresh data because they have learned to fit the training set too closely. Regularizing the model or use a bigger validation set can help to lessen this.
- *Black-box nature:* LSTM models are often seen as black boxes, making it difficult to interpret the internal workings of the model and understand how it arrived at a particular prediction.

Overall, STLF using LSTM is a promising approach in energy management that offers significant advantages in accurately predicting energy demand in the short term. However, it requires careful selection of input variables, model architecture, and training data to achieve high accuracy and avoid overfitting.

# Chapter 5

# Bidirectional Long Short Term Memory Neural Network

## 5.1 Introduction

Bidirectional Long Short-Term Memory (BiLSTM) represents an advance in recurrent neural network (RNN) architecture aimed at solving the problem of capturing past and future context in sequential data. While traditional RNNs and their LSTM variant excel at maintaining long-term dependencies in sequential data, they have a limited ability to incorporate information from past and future contexts simultaneously.

BiLSTM addresses this constraint by introducing bidirectionality, enabling the model to analyze input sequences in both forward and reverse directions. This bidirectional approach enables the network to grasp dependencies not only from past observations but also from future ones, enhancing its comprehension of sequential data.

Like LSTM, BiLSTM employs memory cells and gating mechanisms to regulate information flow across time steps. However, by examining the input sequence bidirectionally, BiLSTM enhances its capacity to grasp contextual nuances and discern intricate relationships within the data.

## 5.2 Bidirectional Long Short Term Memory Neural Network Architecture

The architecture of a Bidirectional Long Short-Term Memory (BiLSTM) network builds upon the foundation of LSTM, enhancing it with bidirectionality to capture both past and future dependencies in sequential data. The structure of a BiLSTM, depicted in Figure 4.1, shares similarities with LSTM but incorporates additional components to process input sequences in both forward and backward directions simultaneously.

Like LSTM, BiLSTM contains memory cells responsible for transferring information along a sequence. These memory cells are regulated by gating mechanisms that control the flow of information. However, in BiLSTM, the input sequence is processed in two directions: forward

and backward. This bidirectional processing allows the network to capture contextual information from both previous and subsequent elements in the sequence, facilitating a more comprehensive understanding of the data.

In a BiLSTM module, the fundamental elements mirror those of an LSTM, featuring three essential gates: the forget gate, the input gate, and the output gate. However, in BiLSTM, each gate functions independently in both forward and backward directions, enabling the network to assimilate insights from both past and future contexts.

The forget gate, for instance, comprises a sigmoid layer, which takes inputs of the previous hidden state (ht−1) and the current input (xt), generating an output ranging between 0 and 1. This output, denoted as the forget gate output (ft), plays a crucial role in determining which information to retain or discard. A value of zero indicates the information should be forgotten, while a value of one signifies it should be retained.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

where,

bf =connection bias at t, Wf = weight matrix between forget gate and input gate , t=timestamp

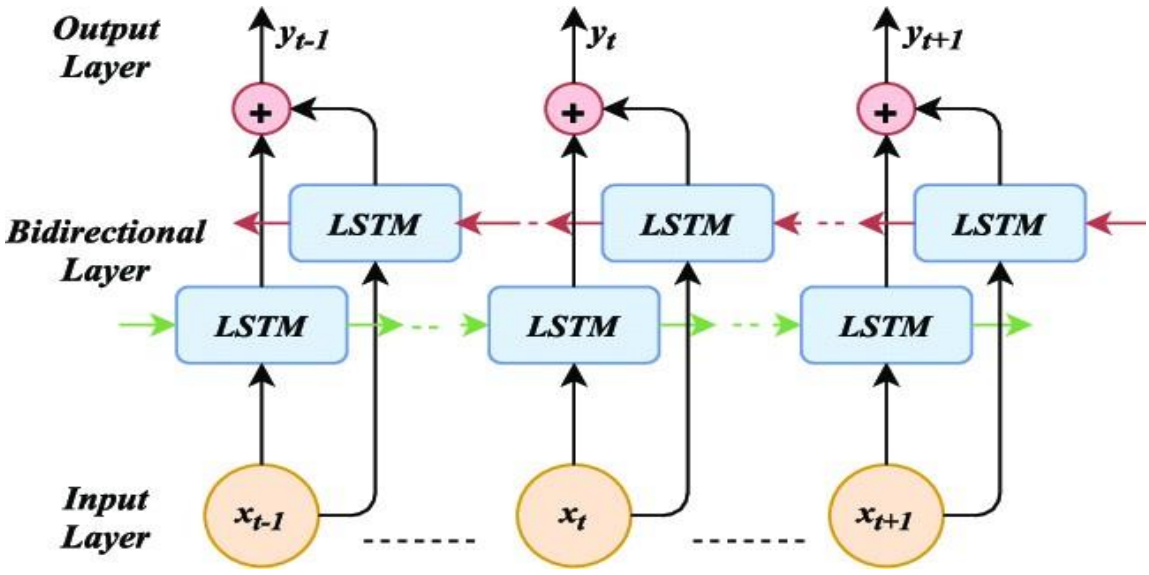



Figure 4.1: Bidirectional Long short-term memory (LSTM) architecture

Subsequently, the forget gate employs a combination of a sigmoid and a hyperbolic tangent (tanh) function to determine the information to be incorporated into the cell state. Both functions take the previous hidden state (ht−1) and the current input (xt) as input parameters. The sigmoid function output assesses the significance of the current information, while the tanh function adjusts the network by compressing the value within the range of -1 to +1. Ultimately, the outputs of both functions are multiplied together.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\widetilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Where,

$i_t$=input gate at t, t= timestamp, $W_i$= weight matrix of sigmoid operator between input gate and output gate, $b_i$= bias vector at t, $C`_t$ =value generated by tanh, $W_c$=Weight matrix of tanh operator between cell state information and network output ,$b_c$= bias vector at t

Using the outputs from both the forget gate and the input gate, the cell state undergoes an update process. This update is achieved through element-wise multiplication between the current cell state and the output of the forget gate. When the output of the forget gate ft=0, the multiplication yields zero, indicating complete removal of the previous value. Conversely, if

ft=1, the previous value is retained. Subsequently, element-wise addition is employed to update the cell state

$$C_t = f_t \times C_{t-1} + i_t \times \widetilde{C}_t.$$

Where, $i_t$=input gate at t, $f_t$=forget gate at t, $C_{t-1}$= Previous cell state

Within this gate, a sigmoid function receives inputs of ht-1 and xt undergoes processing via a hyperbolic tangent (tanh) function. Subsequently, the outputs of the sigmoid and tanh functions are multiplied to ascertain the information that the hidden layer will convey.

$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

where, $W_o$= Weight matrix of output gate $b_o$= bias, $o_t$= output gate, $h_t$= output

Finally, the output gate determines the final output of the BiLSTM network, which also serves as another hidden state (ht). By incorporating information from past and future contexts, the output gate enables the network to generate robust predictions or representations of the input sequence.

## 5.3 Bi-LSTM Network Hyperparameters

**Number of Layers:** This refers to the number of stacked LSTM layers you have in each direction (forward and backward) of your Bi-LSTM network. Here's a breakdown of how this hyperparameter affects your model: It can capture more complex relationships within the sequences you're processing. This becomes especially important for tasks involving long-range dependencies in the data. It also increases training time significantly. With more layers, the model has more parameters to learn, making the training process computationally expensive.

**Number of Units per Layer:** This hyperparameter determines the dimensionality of the hidden state in each LSTM layer. Essentially, it controls how much information each layer can store about the sequence it's processing. Here's what to consider:

Allows the model to capture richer features from the data, potentially leading to better performance. But it also Increases computational cost. More units translate to more complex calculations during training. Again, can contribute to overfitting if not chosen carefully in relation to your dataset size.

**Batch Size:** This hyperparameter defines the number of samples processed together before updating the network weights. Here's how it impacts training: Speeds up training by processing data in chunks but it might lead to poorer convergence, especially if the batch size is too large relative to your dataset. The gradients used for weight updates might become less representative of the entire data.

**Epochs:** This refers to the number of times you pass the entire training dataset through the network.

Here's what to keep in mind:

**Number of Epochs:**Training usually involves a set number of epochs, yet it's vital to monitor validation performance to prevent overfitting. Excessive training epochs may lead the model to memorize the training data rather than learning generalizable patterns.

**Tuning Process:**

Discovering the best combination of these hyperparameters entails an iterative approach. Techniques such as grid search or random search are valuable in exploring various configurations. The primary objective is to achieve a harmonious blend of model complexity and performance while mitigating the risk of overfitting. Continuous monitoring of the model's performance on a distinct validation set is essential. This practice ensures that the model can effectively generalize to new, unseen data.

**Regularization:**

- **Dropout:** Dropout is a potent method for mitigating overfitting. It operates by randomly deactivating a portion of neurons (units) within a layer during the training phase. This mechanism compels the network to avoid excessive reliance on any particular neuron, fostering the acquisition of more resilient features from the data. When implementing dropout, it's essential to consider the following aspects:

    - **Dropout Rate:** This value determines the percentage of neurons to be dropped. Common ranges are from 0.2 to 0.5. Experimenting with different dropout rates can help improve your model's generalizability.

## 5.4 Learning Process:

The learning process of Bi- LSTM involves the following steps:

1. **Input Data:** The learning process starts with input data, typically in the form of sequences. These sequences could be sentences, time series data, or any other sequential data.

2. **Embedding Layer (Optional):** Before feeding the data into the Bi-LSTM network, you may choose to pass it through an embedding layer. This layer converts each token or element in the sequence into a dense vector representation. This step helps in capturing semantic meanings and

relationships between different tokens.

3. **Bi-LSTM Layers:** At the heart of the Bi-LSTM network lie the bidirectional LSTM layers. LSTM (Long Short-Term Memory) units excel at capturing prolonged dependencies within sequential data while addressing the challenge of vanishing gradients. In a bidirectional configuration, the network features two LSTM units: one processes the input sequence in a forward direction, while the other analyzes it in reverse. This setup enables the model to comprehend both past and future contexts for every element within the sequence.

4. **Forward and Backward Passes:** During the training process, the input sequences are fed into the Bi-LSTM network in both forward and backward directions. For each input sequence, the forward pass computes the forward activations, while the backward pass computes the backward activations. These activations capture information about the context of each element in the sequence.

5. **Loss Calculation:** Following both the forward and backward computations, the output produced by the Bi-LSTM layers is juxtaposed with the ground truth labels or targets utilizing a loss function. In sequence-related tasks, prevalent loss functions encompass categorical cross-entropy for classification endeavors and mean squared error for regression undertakings.

6. **Backpropagation Through Time (BPTT):**Upon computing the loss, the gradients of the loss concerning the network parameters are determined through backpropagation through time. This process entails propagating the error gradients in reverse through the entirety of the sequence.

7. **Gradient Descent Optimization:** Once the gradients are computed, the parameters of the Bi-LSTM network undergo updates utilizing an optimization algorithm like stochastic gradient descent (SGD), Adam, or RMSprop. The objective is to minimize the loss function, thereby enhancing the network's capacity to deliver precise predictions on the training dataset.

8. **Repeat:** Steps 4 to 7 are repeated for multiple iterations or epochs until the network converges to a satisfactory solution or until a stopping criterion is met, such as reaching a maximum number of epochs or observing no improvement on a validation set.

9. **Evaluation:** Once the training is complete, the performance of the Bi-LSTM network is evaluated on a separate test set to assess its generalization ability to unseen data.

## 5.5 Advantages and Disadvantages of Using Bi-LSTM in Short Term load forecasting

### Advantages:

**Long-range dependencies:** Regular LSTMs might struggle with capturing long-range dependencies in sequences, but Bi-LSTMs address this by considering both past and future information. This allows them to better capture relationships between distant elements in a sequence.

**Reduced vanishing gradient problem:** LSTMs are designed to alleviate the vanishing gradient problem, but adding the bidirectional aspect further helps by exposing the network to both past and future information, reducing the likelihood of gradients vanishing during training.

**Flexibility:** Bi-LSTMs can be easily integrated into existing neural network architectures and are compatible with various tasks such as sequence labeling, sequence-to-sequence learning, and more.

### Disadvantages:

**Computational complexity:** Bi-LSTMs are computationally more expensive compared to unidirectional LSTMs because they process both forward and backward sequences. This can increase training time and resource requirements, especially for large-scale datasets.

**Memory consumption:** Bidirectional processing effectively doubles the memory footprint required to store activations and gradients during training, which can be a concern for memory-limited environments or when dealing with large models.

**Prediction latency:** In applications where real-time prediction is crucial, the bidirectional nature of Bi-LSTMs introduces a delay because the model needs to process both past and future information before making predictions.

**Overfitting risk:** Bidirectional processing exposes the model to future information during training, which may lead to overfitting, especially when the test data distribution significantly differs from the training data.

# Chapter 6

# Simulations and Result

## 6.1 Introduction

In this chapter we will be discussing the steps included in the modeling of the different deep learning models. Convolution Neural Network and Long Short-term Memory Network are used for solving the problem of short-term load forecasting. There are certain steps that need to be followed for the purpose of training these models so we will be looking into each step-in detail. The model is defined using the Keras and Tensorflow library available in python. In this chapter we will be also looking into the results of these two models and will be comparing them. The steps included in training and testing of the deep learning models are Obtaining Data, Data Preprocessing, Construction of multi-step time series data, Building the forecasting model and Training the proposed model.

## 6.2 Obtaining Data

Kaggle is a platform that hosts a large collection of datasets for machine learning and data analysis projects. One similar dataset is the" Panama Load Forecasting" dataset, available on Kaggle, which contains hourly electricity cargo data from January 2015 to June 2020 for different regions in Panama. This dataset could be used to develop and estimate short- term load forecasting models.

The data set available on Kaggle is in CSV,EXCEL file. There is a CSV file named “continuous.csv”

Which contains the data that was collected from the National Dispatch Centre of the Electric Transmission Company of Panama (ETESA).

The file contains 48,048 rows each representing an hour of the day, and 17 columns shown in Table 6.1.

**Table 6.1 List of Parameters**

| Column name | Description | Unit |
|---|---|---|
| Datetime | Date-time index corresponding to Panama time-zone | UTC-05:00(index) |
| nat_demand | National electricity load (Target or Dependent variable) | MWh |
| T2M_toc | Temperature at 2 meters in Tocumen, Panama City | ºC |
| QV2M_toc | Relative humidity at 2 meters in Tocumen, Panama City | % |
| TQL_toc | Liquid precipitation in Tocumen, Panama City | liters/m2 |
| W2M_toc | Wind Speed at 2 meters in Tocumen, Panama City | m/s |
| T2M_san | Temperature at 2 meters in Santiago city | ºC |
| QV2M_san | Relative humidity at 2 meters in Santiago city | % |
| TQL_san | Liquid precipitation in Santiago city | l/m2 |
| W2M_san | Wind Speed at 2 meters in Santiago city | m/s |
| T2M_dav | Temperature at 2 meters in David city | ºC |
| QV2M_dav | Relative humidity at 2 meters in David city | % |
| TQL_dav | Liquid precipitation in David city | l/m2 |
| W2M_dav | Wind Speed at 2 meters in David city | m/s |
| Holiday_ID | Unique identification number | integer |
| holiday | Holiday binary indicator<br>0 = regular day | 1 = holiday, |
| school | School period binary indicator<br>0 = vacations | 1 = school, |

## 6.3 Data Preprocessing

Data processing is a very important process that has to be done before we give the data to our model for training. The data available on Kaggle is real time data which contains a lot of noise and null values which is caused due to malfunctioning of the devices which are used to record the data. There is a lot of unnecessary information also included in the dataset.Data preprocessing is necessary to clean up the data and prepare it for a machine learning model, which also improves the model's efficiency and accuracy.

To preprocess the Panama Load forecasting Dataset following steps are followed:

**1. Check for Missing values:** Check if the dataset has any missing values. If there are any, replace the null value with the mean value of the adjacent non null values in the corresponding column.

**2. Feature Engineering:** Engineer new features that can help improve the performance of our model. In this case three features have been extracted from the datetime column which are:

- **Hour of the Day**-Is a integer value ranging from 0 to 23 telling which hour of the day it is for example if it is the load forecasting for 1 am then the hour of the day is 1 and if the load forecasting is for the time 11pm then the hour of the day is 23.
- **Day of the week**-This column stores Integer values ranging from 1 to 7 where 1 is for Monday and 7 is for Sunday.
- **Weekend**- This is Boolean column if the weekday is Friday, Saturday or Sunday then the value of weekend is one else it is Zero.

**3. Feature Selection:** There are few of the features in the CSV file which are not required for forecasting. Although in our case there is only one feature in our file which cannot be used i.e. "datetime". As datetime is a string value it can't be used for model training therefore the "datetime" column is removed. So the total coloumn now in the dataset is 19.

**4. Data Normalization:** Normalizing the dataset ensures that all features have the same scale. This can be done using standardization. Normalization makes sure that no feature has more impact on the output just because its numerical value is high.

**5. Split the dataset:** Split the dataset into training and testing sets this is important to evaluate the performance of the model on unseen data so the data of 48048 column is divided in to training dataset and test dataset where training dataset includes first 36,535 rows and testing dataset includes 11,513 rows.

## 6.4 Construction of Multi-step time series data

The input to both the models is the 3 weeks of data and using this data the model makes the prediction of the load for next 2 consecutive hours. But the size of the input tensor to both the deep learning models is different so we need to define different functions for the construction of multi-step time series data for both the deep learning models.

### 6.4.1 Convolution Neural Network

The tensor input to convolution Neural Network is of shape (168,19,3). There are 3 channels of size (168,19) in each channel there is data of one week. As there are 19 features and there are 168 hours in a week. These 3 channels include the data for 3 consecutive weeks. So using python a function is defined where the function converts the 504 input sequence into tensors of size (168,19,3) which is used to predict the load for the next 100 hours.

### 6.4.2 Long Short-term Memory Network

The tensor input to LSTM Network is of shape (504,19). As there are 19 features and there are 168 hours in a week. The data of three weeks is used to form this $2^{nd}$ rank tensor. So using python a function is defined where the function converts the 504 input sequence into tensors of size (504,3) which is used to predict the load for the next 100 hours.

### 6.4.3 Bidirectional Long Short-term Memory Network

The tensor input to Bi-LSTM Network is of shape (200,19). As there are 19 features and there are 168 hours in a week. The data of three weeks is used to form this $2^{nd}$ rank tensor. So using python a function is defined where the function converts the 200 input sequence into tensors of size (200,3) which is used to predict the load for the next 100 hours.

## 6.5 Building the forecasting model

### 6.5.1 Convolution Neural Network

In Chapter 3 we discussed the different hyperparameters (kernel size, stride, dropout rate, learning rate etc) and the different layers (pooling layer, relu layer). Now we will design a CNN architecture which will give the best solution for our model. We divide our training model into a 5:1 ratio into training dataset and validation dataset. We train the model for the training dataset for

a small number of epochs and check the mean absolute error for the validation dataset. According to the results we change the number of convolution layers, dense layers, and the hyperparameters. This process is also called hyperparameter tuning after determining the best hyperparameters our actual model is trained. In our case we got the best results with the model(also shown in figure 5.1) is:

1. There are 11 Convolution layer:
    a. Kernel Size=(5,2) Feature Maps=15 Convolution Stride=(1,1)
    b. Kernel Size=(5,2) Feature Maps=30 Convolution Stride=(1,1)
    c. Kernel Size=(5,2) Feature Maps=60 Convolution Stride=(1,1)
    d. Kernel Size=(5,2) Feature Maps=60 Convolution Stride=(1,1)
    e. Kernel Size=(5,2) Feature Maps=60 Convolution Stride=(1,1)
    f. Kernel Size=(5,2) Feature Maps=60 Convolution Stride=(1,1)
    g. Kernel Size=(5,2) Feature Maps=60 Convolution Stride=(1,1)
    h. Kernel Size=(5,2) Feature Maps=60 Convolution Stride=(1,1)
    i. Kernel Size=(5,2) Feature Maps=60 Convolution Stride=(1,1)
    j. Kernel Size=(5,3) Feature Maps=60 Convolution Stride=(1,1)
    k. Kernel Size=(5,3) Feature Maps=60 Convolution Stride=(1,1)
2. 1 Flatten layer to convert 3$^{rd}$ rank tensor output to 1$^{st}$ rank tensor.
3. 1 Dense layer which contains a Feed forward neural network with 100 output .

### 6.5.2 Long Short-Term Memory Network

Similar to that of convolution neural network in the LSTM case also using validation dataset and training dataset we get the number of layers to be used and the value of the other hyperparameters.The best results for this case are obtained with:

1. There are 10 LSTM Units each with an output size of 64.
2. The output of the LSTM units is flattened.
3. A feed forward network with 2 hidden layers each with 64 neurons.
4. The output of the feed forward network is changed according to requirement.

### 6.5.3 Bidirectional Long Short-Term Memory Network

Similar to that of LSTM in the Bi-LSTM case also using validation dataset and training dataset we get the number of layers to be used and the value of the other hyperparameters.The best results for this case are obtained with:

1. There are 24 LSTM Units each with an output size of 100.

2. The input of Bi-LSTM is encoded in two layers, one processing the input sequence forward, and the other processing it backward.

3. A feed forward network with 2 hidden layers each with 64 neurons.

4. The outputs from the forward and backward LSTM layers are combined at each time step, typically by concatenating or summing them.

## 6.6 Training the Proposed model

After getting the model architecture validation dataset is merged with the training dataset and the model is trained for about 50 epochs.

The model is to be trained until the training loss saturates. The loss function used in the training is Mean Absolute Error (MAE). Total 3 models are trained which are:

**Bi-LSTM** model for prediction of the Net Demand : The variation of training loss and testing loss is plotted in the graph shown in Figure 6.1.

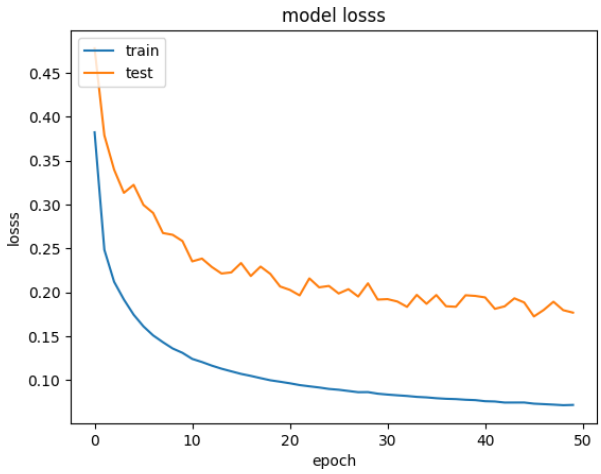



Figure 6.1: variation model loss

**LSTM** model for prediction of the Net Demand : The variation of training loss and testing loss is plotted in the graph shown in Figure 6.2.

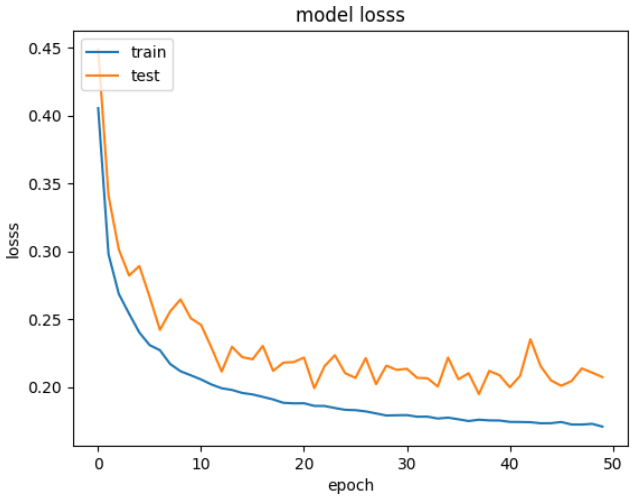



Figure 6.2: variation model loss

**CNN** model for prediction of the Net Demand : The variation of training loss and testing loss is plotted in the graph shown in Figure 6.3.

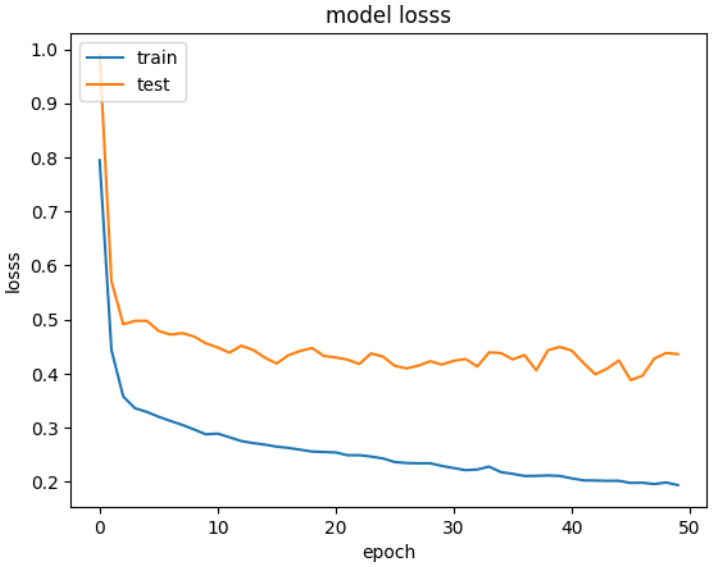



Figure 6.3: variation model loss

## 6.7 Final Observation

**Bi-LSTM for Next 100 hour Prediction**

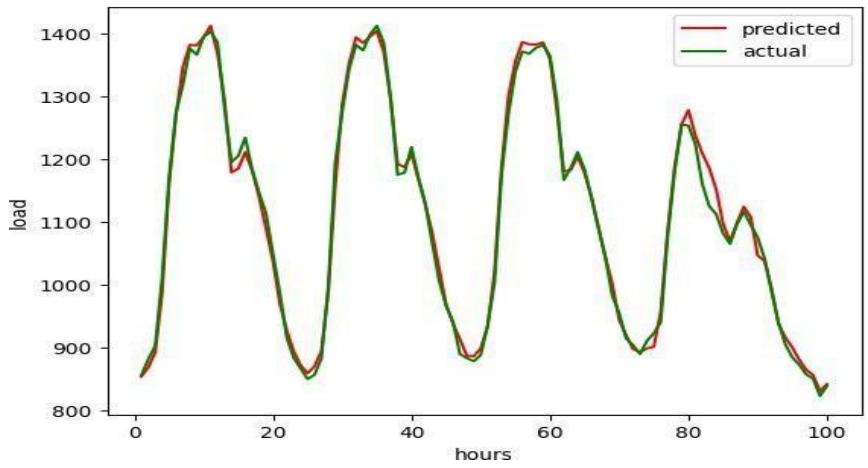



Figure 6.4 : % Error Training Dataset

MAPE= 1.52 %

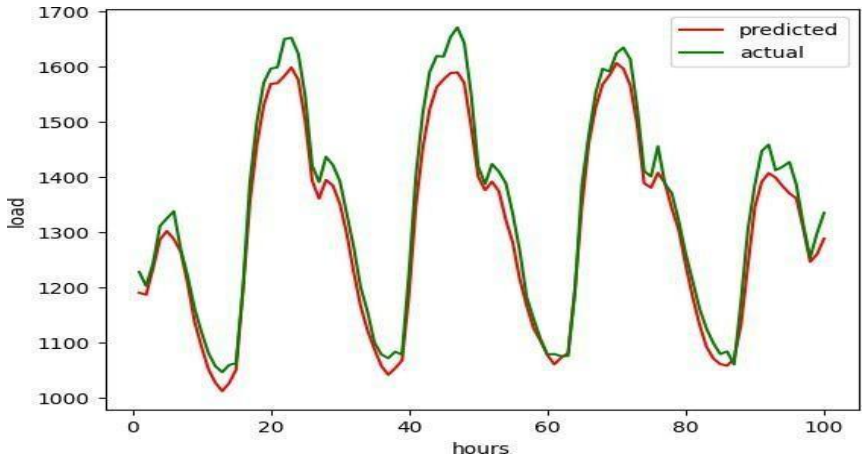



Figure 6.5 : % Error Testing Dataset

MAPE= 1.64 %

**LSTM Network for next 100 hour Prediction**

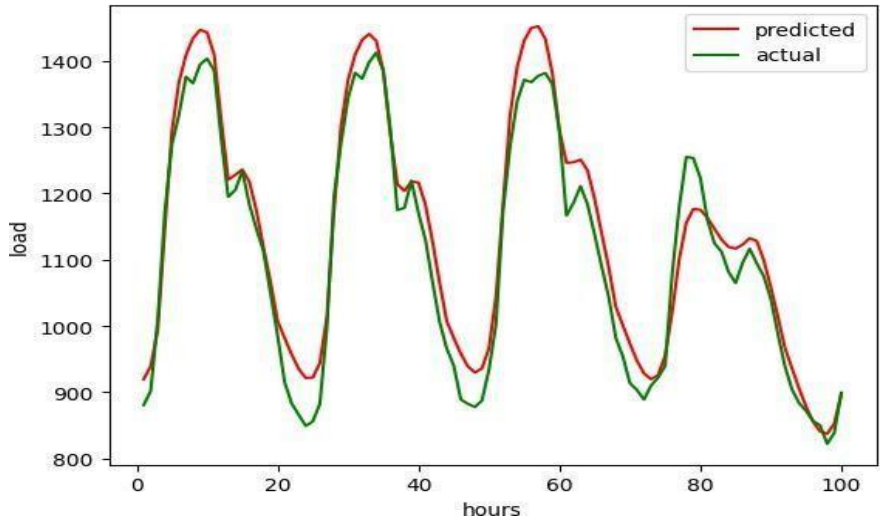



Figure 6.6 : % Error Training Dataset

MAPE= 2.59 %

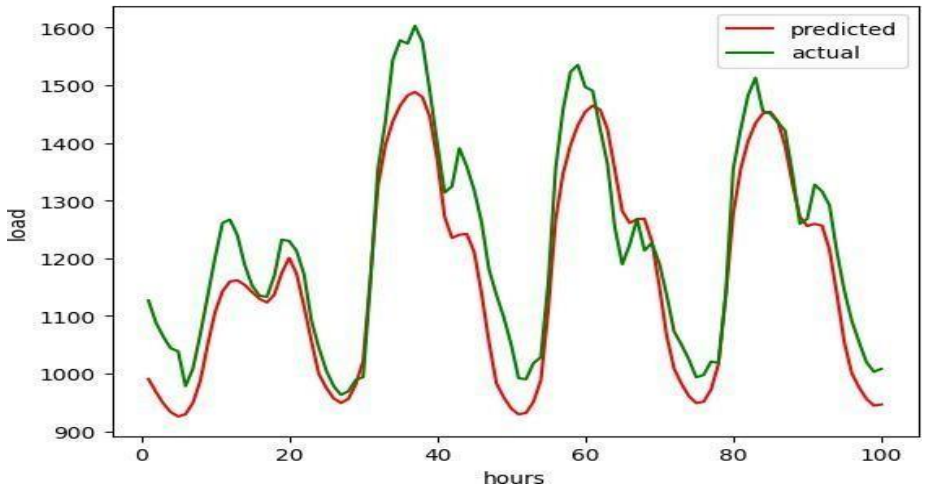



Figure 6.7 : % Error Testing Dataset

MAPE= 3.07 %

**CNN for Next 100 hour prediction**

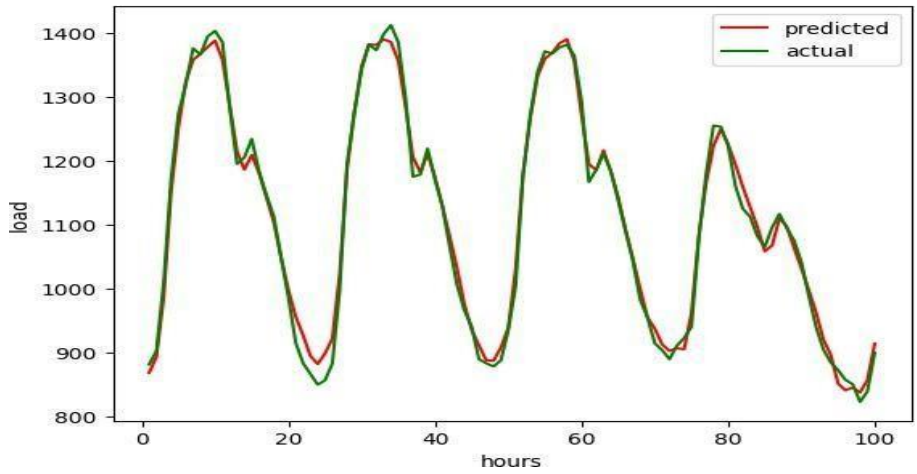



Figure 6.8 : % Error Training Dataset

MAPE= 1.54 %

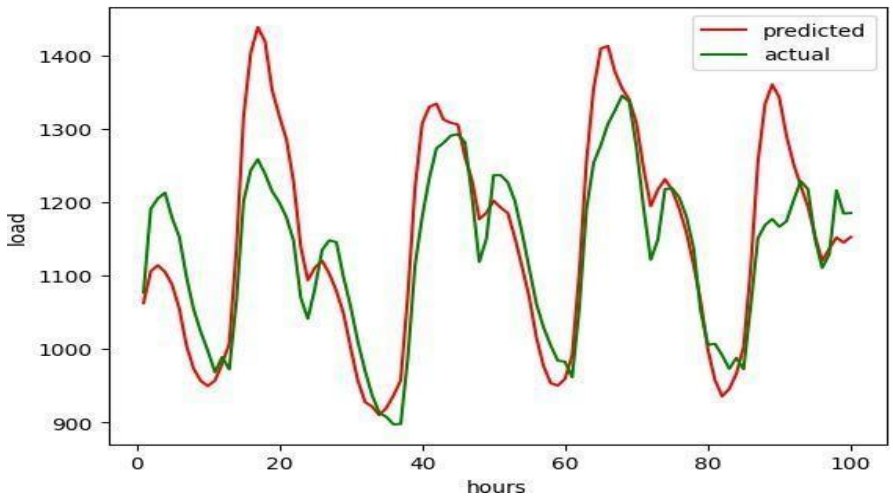



Figure 6.9 : % Error Testing Dataset

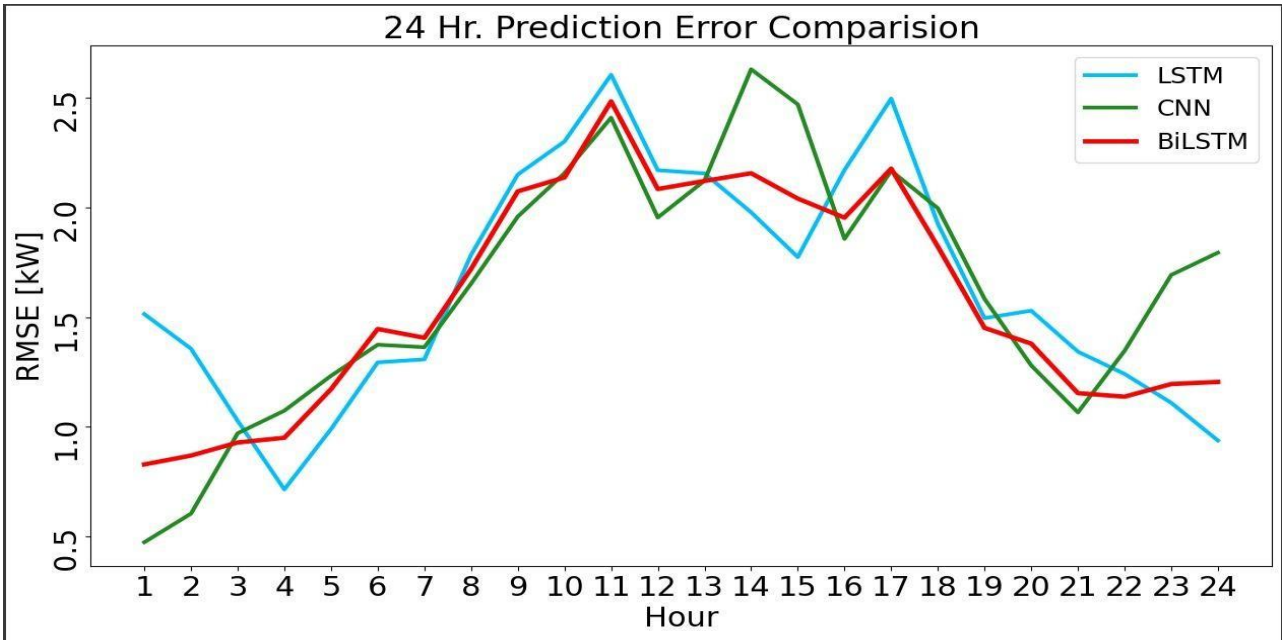
MAPE= 1.78 %

**Table 6.1 Results and comparison between Deep Learning models**

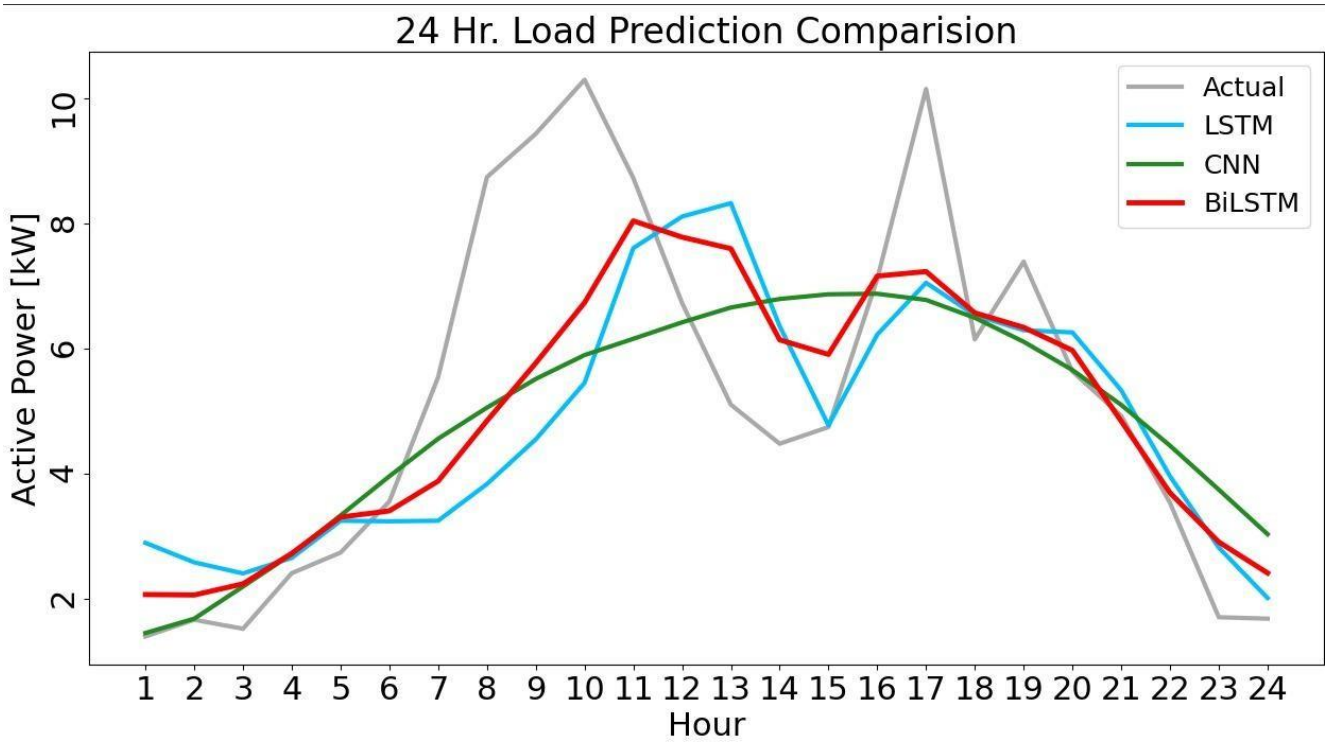
| **Results:** | % Error Training Dataset | % Error Testing Dataset |
|---|---|---|
| Bi-LSTM for Next 100 hour Prediction | 1.52 | 1.64 |
| LSTM Network for next 100 hour Prediction | 2.59 | 3.07 |
| CNN for Next 100 hour Prediction | 1.54 | 1.78 |

## 6.8 Comparison

RMSE Comparison for 24hrs:



Active Power Comparison for 24hrs:




## Summary

In the above chapter we covered all the steps required to design a Short term Load forecasting model using two deep learning algorithms that are Convolution Neural Network and Long Short-term Memory Network and Bidirectional Neural Networking.The effectiveness of a method for load forecasting depends on various factors such as the characteristics of the data, the complexity of the patterns to be captured, and the specific requirements of the forecasting task.

In the end results for each are compared and it has been observed that Bi-LSTM is giving a better result compared to that of LSTM and CNN. By leveraging bidirectional processing and capturing both past and future information, Bi-LSTM offers a powerful framework for modeling the intricate dynamics of load data, ultimately contributing to more efficient energy management systems..

# Chapter 7

# Conclusions and Future Work

- The project report proposes a novel approach to short-term load forecasting (STLF) using deep learning algorithms.
- The importance of STLF in energy management and its associated challenges are highlighted, including weather variability, increasing demand for renewable energy, and the need for real-time predictions.
- The report introduces Long Short-Term Memory (LSTM), Convolutional Neural Networks (CNN) and Bi directional Long Short-Term Memory (Bi-LSTM) as potential solutions to overcome these challenges.
- A real-world dataset of energy consumption in a Panama region is used to implement LSTM,Bi-LSTM and CNN models to forecast energy demand for the next few hours.
- The LSTM model is trained on historical and meteorological data, while the Bi-LSTM and CNN model is trained on a combination of historical and meteorological data with more trainable parameters.
- Experimental results show that the proposed models outperform traditional STLF methods like autoregressive integrated moving average (ARIMA) and support vector regression (SVR).
- The LSTM model achieves an accuracy of 96.93%, the CNN model achieves an accuracy of 98.22% and the Bi-LSTM model achieves an accuracy of 98.36% .
- Incorporating more parameters improves STLF accuracy.
- The study evaluates the sensitivity of the models to the length of the input sequence, with longer input sequences leading to better STLF performance.

- The report concludes by highlighting the potential applications of deep learning algorithms in STLF.
- Future research directions are suggested to improve the accuracy and scalability of these methods.
- The study demonstrates the effectiveness of deep learning algorithms in predicting energy demand accurately and efficiently.
- Accurate STLF can aid in optimal resource allocation and efficient energy distribution in power systems.
- The report presents a valuable contribution to the field of STLF.
- Overall, the project report demonstrates the potential of deep learning algorithms to address the challenges associated with accurate energy demand forecasting.

Finally, from our experimental observations, we concluded that Convolution Neural Network is a sound method to perform load forecasting for a short term period.

**Keeping that in mind, future work involve:**

- **Integration of multiple data sources:** Future work could involve integrating more data sources, such as real-time weather data, social media trends, and economic indicators, to improve the accuracy of STLF using CNN and LSTM.
- **Hybrid models:** Combining different deep learning models, such as CNN and LSTM, could further improve the accuracy of STLF by capturing both short-term and long-term dependencies in the data.
- **Transfer learning:** Applying transfer learning, where a pre-trained model is used as a starting point for a new model, could reduce the amount of training data needed and improve the efficiency of STLF using CNN and LSTM.

- **Uncertainty estimation:** Developing methods to estimate the uncertainty in STLF using CNN and LSTM predictions could provide decision-makers with a better understanding of the accuracy and reliability of the forecast.
- **Integration of energy storage:** Incorporating energy storage data could provide a more accurate representation of energy demand and supply and improve the accuracy of STLF using CNN and LSTM.
- **Scalability:** Ensuring that the STLF using CNN model and LSTM can scale to large datasets and can be deployed in real-time production environments is essential for practical applications.

# References


*[1] Z. Yu, Z. Niu, W. Tang, and Q. Wu, ''Deep learning for daily peak load forecasting–A novel gated recurrent neural network combining dynamic time warping,'' IEEE Access, vol. 7, pp. 17184–17194, 2019.*

*[2] M. Jacob, C. Neves, and D. V. Greetham, ''Short-term load forecasting,'' in Forecasting and Assessing Risk of Individual Electricity Peaks (Mathematics of Planet Earth). Cham, Switzerland: Springer, 2020, pp. 5–37.*

*[3] W. Charytoniuk, M. S. Chen, and P. Van Olinda, ''Nonparametric regression based short-term load forecasting,'' IEEE Trans. Power Syst., vol. 13, no. 3, pp. 725–730, May 1998.*

*[4] C.-M. Lee and C.-N. Ko, ''Short-term load forecasting using lifting scheme and ARIMA models,'' Expert Syst. Appl., vol. 38, no. 5, pp. 5902–5911, May 2011.*

*[5] W. Christiaanse, ''Short-term load forecasting using general exponential smoothing,'' IEEE Trans. Power App. Syst., vol. PAS-90, no. 2, pp. 900–911, Mar. 1971.*

*[6] Y. Chen, P. Xu, Y. Chu, W. Li, Y. Wu, L. Ni, Y. Bao, and K. Wang, ''Short-term electrical load forecasting using the support vector regression (SVR) model to calculate the demand response baseline for office buildings,'' Appl. Energy, vol. 195, pp. 659–670, Jun. 2017*

*[7] H. H. Çevik and M. Çunkaş, ''Short-term load forecasting using fuzzy logic and ANFIS,'' Neural Comput. Appl., vol. 26, no. 6, pp. 1355–1367, Aug. 2015*

*[8] K. B. Sahay and M. M. Triapthi, ''Day ahead hourly load and price forecast in ISO New England market using ANN,'' in Proc. IEEE INDICON, Mumbai, India, Dec. 2013, pp. 1–6.*

*[9] B. S. Bisht and R. M. Holmukhe, ''Electricity load forecasting by artificial neural network model using weather data,'' IJEET Trans. Power Syst., vol. 4, no. 1, pp. 91–99, 2013.*

*[10] E. Banda and K. A. Folly, ''Short term load forecasting using artificial neural network,'' in Proc. IEEE Lausanne Power Tech, Shimla, India, Jul. 2007, pp. 1–5.*

*[11] P.-H. Kuo and C.-J. Huang, ''A high precision artificial neural networks model for short-term energy load forecasting,'' Energies, vol. 11, no. 1, p. 213, Jan. 2018.*

*[12] N. Singh, C. Vyjayanthi and C. Modi, "Multi-step Short-term Electric Load Forecasting using 2D Convolutional Neural Networks," 2020 IEEE-HYDCON, 2020, pp. 1-5, doi: 10.1109/HYDCON48903.2020.9242917.*

*[13] S. H. Rafi and Nahid-Al-Masood, ‘‘Short term electric load forecasting using high precision convolutional neural network,’’ in Proc. Int. Conf. Comput., Electr. Commun. Eng. (ICCECE), Kolkata, India, Jan. 2020, pp. 1–7.*

*[14] S. Muzaffar and A. Afshari, ‘‘Short-term load forecasts using LSTM networks,’’ in Proc. 10th Int. Conf. Appl. Energy, Beijing, China, 2018, pp. 2922–2927.*

*[15] S. Hosein and P. Hosein, "Load forecasting using deep neural networks," 2017 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), 2017, pp. 1-5, doi: 10.1109/ISGT.2017.8085971*

*[16] Ming-Guang Zhang, "Short-term load forecasting based on support vector machines regression," 2005 International Conference on Machine Learning and Cybernetics, 2005, pp. 4310-4314 Vol. 7, doi: 10.1109/ICMLC.2005.1527695.*

*[17] N. Kamel and Z. Baharudin, "Short term load forecast using Burg autoregressive technique," 2007 International Conference on Intelligent and Advanced Systems, 2007, pp. 912-916, doi: 10.1109/ICIAS.2007.4658519.*

*[18] S. H. Rafi, Nahid-Al-Masood, S. R. Deeba and E. Hossain, "A Short-Term Load Forecasting Method Using Integrated CNN and LSTM Network," in IEEE Access, vol. 9, pp. 32436-32448, 2021, doi: 10.1109/ACCESS.2021.3060654.*

*[19] C. Liu, Z. Jin, J. Gu and C. Qiu, "Short-term load forecasting using a long short-term memory network," 2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), Turin, Italy, 2017, pp. 1-6, doi: 10.1109/ISGTEurope.2017.8260110.*

*[20] M. S. Hossain and H. Mahmood, "Short-Term Load Forecasting Using an LSTM Neural Network," 2020 IEEE Power and Energy Conference at Illinois (PECI), Champaign, IL, USA, 2020, pp. 1-6, doi: 10.1109/PECI48348.2020.9064654.*

*[21] M. Voß, C. Bender-Saebelkampf and S. Albayrak, "Residential Short-Term Load Forecasting Using Convolutional Neural Networks," 2018 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), 2018, pp. 1-6, doi: 10.1109/SmartGridComm.2018.8587494.*

*[22] M. Alhussein, K. Aurangzeb and S. I. Haider, "Hybrid CNN-LSTM Model for Short-Term Individual Household Load Forecasting," in IEEE Access, vol. 8, pp. 180544-180557, 2020, doi: 10.1109/ACCESS.2020.3028281.*

# report

ORIGINALITY REPORT

| 14% | 10% | 15% | 8% |
|---|---|---|---|
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | www.atbuftejoste.net<br>Internet Source | 1% |
| 2 | Ali M. Eltamaly, Asmaa H. Rabie. "A Novel Musical Chairs Optimization Algorithm", Arabian Journal for Science and Engineering, 2023<br>Publication | 1% |
| 3 | "Innovations in Machine and Deep Learning", Springer Science and Business Media LLC, 2023<br>Publication | 1% |
| 4 | mdpi-res.com<br>Internet Source | 1% |
| 5 | fastercapital.com<br>Internet Source | 1% |
| 6 | univagora.ro<br>Internet Source | 1% |
| 7 | academic-accelerator.com<br>Internet Source | <1% |

| | | |
|---|---|---|
| 8 | elibrary.tucl.edu.np<br>Internet Source | <1% |
| 9 | "Neural Information Processing", Springer Science and Business Media LLC, 2024<br>Publication | <1% |
| 10 | dokumen.pub<br>Internet Source | <1% |
| 11 | www.iosrjournals.org<br>Internet Source | <1% |
| 12 | "Cognitive Systems and Information Processing", Springer Science and Business Media LLC, 2023<br>Publication | <1% |
| 13 | J. Deny, Siva Kamisetty, Harsha Vardhan Reddy Thalakola, Jagadeesh Vallamreddy, Vijay Kumar Uppari. "Inshort Text Summarization of News Article", 2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS), 2023<br>Publication | <1% |
| 14 | Lakshmi Shrinvisan, Reshma Verma, Kalyani Tekade. "Performance Prediction and Estimation of Battery State of Charge Using DNN for Electric Vehicles", 2023 International Conference on Network, Multimedia and Information Technology (NMITCON), 2023 | <1% |

Publication

| | | |
|---|---|---|
| 15 | Submitted to Liverpool John Moores University<br>Student Paper | <1% |
| 16 | Pranati Rakshit, Avik Sarkar. "A supervised deep learning-based sentiment analysis by the implementation of Word2Vec and GloVe Embedding techniques", Multimedia Tools and Applications, 2024<br>Publication | <1% |
| 17 | Boonsit Yimwadsana. "Determining Empirical Relationship of Rubber Drying Process using Machine Learning", TENCON 2023 - 2023 IEEE Region 10 Conference (TENCON), 2023<br>Publication | <1% |
| 18 | Mohammad Safayet Hossain, Hisham Mahmood. "Short-Term Load Forecasting Using an LSTM Neural Network", 2020 IEEE Power and Energy Conference at Illinois (PECI), 2020<br>Publication | <1% |
| 19 | Sonam Bhardwaj, Mayank Dave. "Integrating a Rule-Based Approach to Malware Detection with an LSTM-Based Feature Selection Technique", SN Computer Science, 2023<br>Publication | <1% |
| 20 | www.mdpi.com<br>Internet Source | <1% |

| | | |
|---|---|---|
| 21 | www.science.gov<br>Internet Source | <1% |
| 22 | Submitted to Coventry University<br>Student Paper | <1% |
| 23 | Submitted to Indian Institute of Technology<br>Student Paper | <1% |
| 24 | www.ai.rug.nl<br>Internet Source | <1% |
| 25 | sciencedocbox.com<br>Internet Source | <1% |
| 26 | www.jocet.org<br>Internet Source | <1% |
| 27 | Submitted to American University of the Middle East<br>Student Paper | <1% |
| 28 | Rusul A. Al Mudhafar, Nidhal K. El Abbadi. "Chapter 50 Comprehensive Approach for Image Noise Analysis: Detection, Classification, Estimation, and Denoising", Springer Science and Business Media LLC, 2024<br>Publication | <1% |
| 29 | ceur-ws.org<br>Internet Source | <1% |

| | | |
|---|---|---|
| 30 | Khanh Nguyen-Trong, Hoai Nam Vu, Ngon Nguyen Trung, Cuong Pham. "Gesture Recognition Using Wearable Sensors With Bi-Long Short-Term Memory Convolutional Neural Networks", IEEE Sensors Journal, 2021<br>Publication | <1% |
| 31 | Submitted to Northern Arizona University<br>Student Paper | <1% |
| 32 | Submitted to Saint Thomas University<br>Student Paper | <1% |
| 33 | Submitted to University of Nottingham<br>Student Paper | <1% |
| 34 | www.ncbi.nlm.nih.gov<br>Internet Source | <1% |
| 35 | Submitted to Tilburg University<br>Student Paper | <1% |
| 36 | Utkarsh Kuchhal, Ved Vyapak, Jeebananda Panda. "Adaptive Arithmetic Coding using Neural Network", 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), 2023<br>Publication | <1% |
| 37 | etd.lib.metu.edu.tr<br>Internet Source | <1% |
| 38 | Submitted to King's College<br>Student Paper | <1% |

39 Submitted to National College of Ireland
Student Paper

<1%

| | | | |
|---|---|---|---|
| Exclude quotes | On | Exclude matches | < 20 words |
| Exclude bibliography | On | | |