A TIME SERIES APPROACH TO MODELING GROUNDWATER DYNAMICS

A Project Report Submitted
in Partial Fulfilment of the Requirements
for the Degree of

MASTER OF SCIENCE

in

Mathematics and Computing

by

Ritika Verma

(Roll No. 212123045)

to the

DEPARTMENT OF MATHEMATICS
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, INDIA

April 2023

CERTIFICATE

This is to certify that the work contained in this report entitled **"A TIME SERIES APPROACH TO MODELING GROUNDWATER DYNAMICS"** submitted by **Rtika Verma** (**Roll No: 212123045**) to Department of Mathematics, Indian Institute of Technology Guwahati towards the requirement of the course **MA699 Project** has been carried out by her under my supervision.

Guwahati - 781 039
April 2023

(Dr. Arabin Kumar Dey)
Project Supervisor

ABSTRACT

Groundwater is a crucial resource for both human and environmental sustainability, and the impacts of climate change are expected to have a significant influence on its demand and availability in the future. Effective management of water resources requires addressing various challenges, such as forecasting groundwater levels, predicting water quality, estimating function, and ensuring data quality. In this context, a recent study has focused on predicting the long-term impact of climate change on groundwater levels, taking into account human interventions as well.

To achieve this, the study employed a specific type of recurrent neural and transfomer based network capable of capturing spatio-temporal features, which was trained using different architectures to predict groundwater levels. The performance of each model was evaluated based on the mean absolute error (MAE), with a combined regression technique which provides the best results.

The study also provides a prediction for the groundwater levels over the next 20 years, along with a test set prediction, which demonstrates that the models were able to capture the signals reasonably well. Overall, the research contributes to a better understanding of how climate change and human activities can impact groundwater levels and offers insights for improving water resource management strategies.

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

As per the fifth Assessment Report (AR5) by IPCC, there is a predicted decline in the average precipitation in many subtropical and mid-latitude areas, along with a high level of confidence in the temperature and solar radiation variation. This could lead to water scarcity or droughts (IPCC 2014). The interlink between climate and groundwater is a challenging issue. The climate has an impact on groundwater recharge and utilization (Green 2011). The ground aquifers affect atmospheric processes at the boundary level significantly (Maxwell et al. 2011). Precipitation is a crucial factor in water balance, while evapotranspiration is considered as the prominent output parameter. The contribution of evapotranspiration in groundwater is notable (Cohen et al. 2006). Additionally, solar energy absorption by humidity, temperature, and soil moisture plays a crucial role in water energy balance (Green 2011; Alkhaier et al. 2012a, b; Whan et al. 2015; Berg et al. 2014). In light of this, it is important to develop long-term strategies to deal with climate change and support decision-makers in managing water resources. The variability in groundwater is the outcome of natural and human-induced factors. The prediction of the groundwater level can help achieve sustainable development in these resources. Groundwater modelling can help tackle these challenging issues (Mackay et al. 2014).

The use of numerical modelling, which often involves large datasets, has garnered significant attention. However, this approach can pose challenges for small scale research projects (Mohanty et al. 2010; Nayak et al. 2006). Machine learning offers an effective solution for predictive analysis with datasets of varying sizes. Crucial determinants for time series prediction include nonlinearity, cross correlation, and heteroscedasticity (Rakhshandehroo et al. 2018). This report focuses on predicting groundwater levels based on observed factors in the Varuna river basin and its subregions over a specific timeframe. Recurrent neural networks are utilized to model the behavior of groundwater level fluctuations and inform necessary management practices.

Coulibaly et al. (1999) wrote a review article on the use of Artificial Neural Networks (ANN) to predict ground water fluctuations. Style (2009) conducted a second review on this topic, and several works by Uddameri (2006), Shaoyuan et al. (2007), Mohanty et al. (2010), and Chitsazan et al. (2015) attempted to address different problems related to climate data and ground water level prediction using ANN. Various approaches such as Fuzzy Inference System (FIS) (Nourani & Mousavi, 2016),

Adaptive Neuro Fuzzy Inference System (ANFIS) (Djurovic et al., 2015), Support Vector Regression (SVR) (Suryanarayana et al., 2014), and Nonlinear Autoregressive Networks with Exogenous Input (NARX) (Wunsch et al., 2018) have been used to investigate the effectiveness of spatio-temporal ANN models. Although most studies on groundwater level prediction using ANN are limited to short-term predictions, Rakhshandehroo et al. (2018) conducted long-term groundwater level predictions using a WNN model trained by a novel improved algorithm to forecast one year ahead groundwater level conditions.

Zhang et al. (2018) utilized an LSTM based on recurrent neural network to predict ground water levels in agricultural areas of Inner Mongolia, China, but the proposed LSTM model could only capture temporal features. Our study proposes the use of certain variations of LSTM models capable of capturing spatio-temporal features, to predict groundwater and determine the best results. While these architectures are commonly used in technological developments by companies such as Google, Amazon, and Microsoft, our methodology presents a way to predict long-term groundwater levels using an RNN model based on a climate parametric database. Our selection of climate factors is different from Zhang et al.'s (2018) as we chose factors based on Northern Indian climatic conditions, including rainfall, evapotranspiration, temperature, and solar radiation as input data to predict groundwater level output.

1.1 Research region

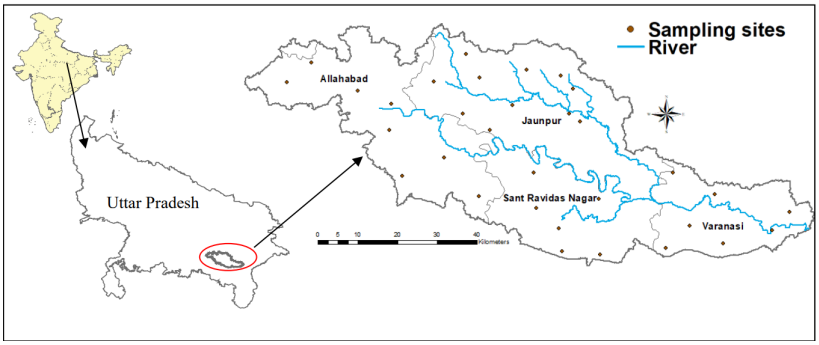


Figure 1.1: Location map of Varuna river basin, Uttar Pradesh, India

The Varuna river basin (Figure 1) is located in the central Ganga alluvial plain, covering an area of 3675.45 sq. km, including parts of the districts of Allahabad, Jaunpur, Sant Ravidas Nagar, and Varanasi. The basin's latitude ranges from 25°39'28.71"N to 25°19'44.61"N, and its longitude ranges from 81°45'57.46"E to 82°03'06.59"E. The area experiences a semi-arid to sub-humid tropical climate, with the rainy season brought by the Southwest monsoon and an annual rainfall ranging from 572 to 897mm. The temperature varies widely between summer and winter, with maximum temperatures ranging from approximately 22°C to 48°C and minimum temperatures ranging from approximately 15°C to 1°C. The general geology of the area comprises alluvium deposits of Quaternary age that began forming during the Pleistocene period after the final cataclysm of the Himalayas. Groundwater

occurs in unconsolidated sediments and primary porosity of the alluvium deposits, with shallower aquifers present down to a depth of 50 mbgl. The deep tubewells are confined down to depths of 250 to 400m below ground level. The rich, fertile alluvium soil in the area is ideal for agriculture and is one of the most productive regions in India. Although there is sufficient water during the monsoon season, the pre-monsoon season usually results in insufficient withdrawal of groundwater (Figure 2a), with the area receiving less rainfall than in previous years (Figure 2b). India's food grain production is largely dependent on Indian Summer Monsoon Rainfall, which can affect the potentiality of food production, particularly kharif production in this region (Kumar et al., 2004; Mall et al., 2006).

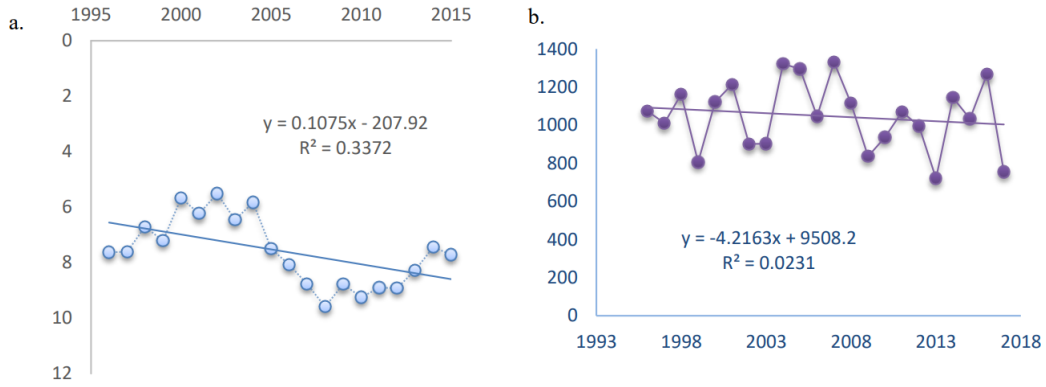


Figure 1.2: (a)Water level variation and (b) Rainfall variation in the study area for the time span of 1996 to 2017

Chapter 2

DESCRIPTION OF DATASET

To simulate the groundwater level, an empirical dataset was created using secondary data sources. Specifically, groundwater level data from bore wells in the study area between 1996 and 2017 was obtained from India-WRIS, a web-based GIS interface for water resources information developed by the National Remote Sensing Center (NRSC). Bore well draft data from the Central Ground Water Board was also used as input data, which was obtained from a linear equation. Rainfall and temperature data from the India Meteorological Department was collected for the period of 1996 to 2015, while relative humidity data was gathered from the NASA POWER Data Access Viewer website. Potential evapotranspiration (PET) was calculated based on the Hargreaves equation under FAO 1998.
PET = $0.0023(\text{Tmean} + 17.8)*(\text{Tmax} - \text{Tmin})_{0.5}*R_a$

Where:
PET : Potential Evapotranspiration
Tmax: Mean maximum temperature
Tmin: Mean minimum temperature
Tmean: Mean temperature calculated as (Tmax + Tmin)/2
R_a: Extraterrestrial radiation obtained from standard graphs given in FAO.

PET was further multiplied by 0.8 to get actual evapotranspiration (AET)

The multivariate dataset comprises interdependent variables, specifically the ground water level data of four regions in Uttar Pradesh: Allahabad, Bhadohi, Jaunpur, and Varanasi. For each region, climate datasets from 1996 to 2015 were selected, and six factors, including the ground water level, were reported for all regions during the mentioned time frame. The entire dataset was analyzed in various forms.

Cross correlation indices at 95% confidence intervals showed that the variables have high correlations among each other. Rainfall has showing negative relationship with all the parameters whereas other parameters are strongly positive relationship between each other.

The analysis of the rainfall data reveals a clear pattern of serial dependence, indicating that the values of the variable are correlated with their past values. Additionally, there is a possibility that the variance of the data may vary over time.

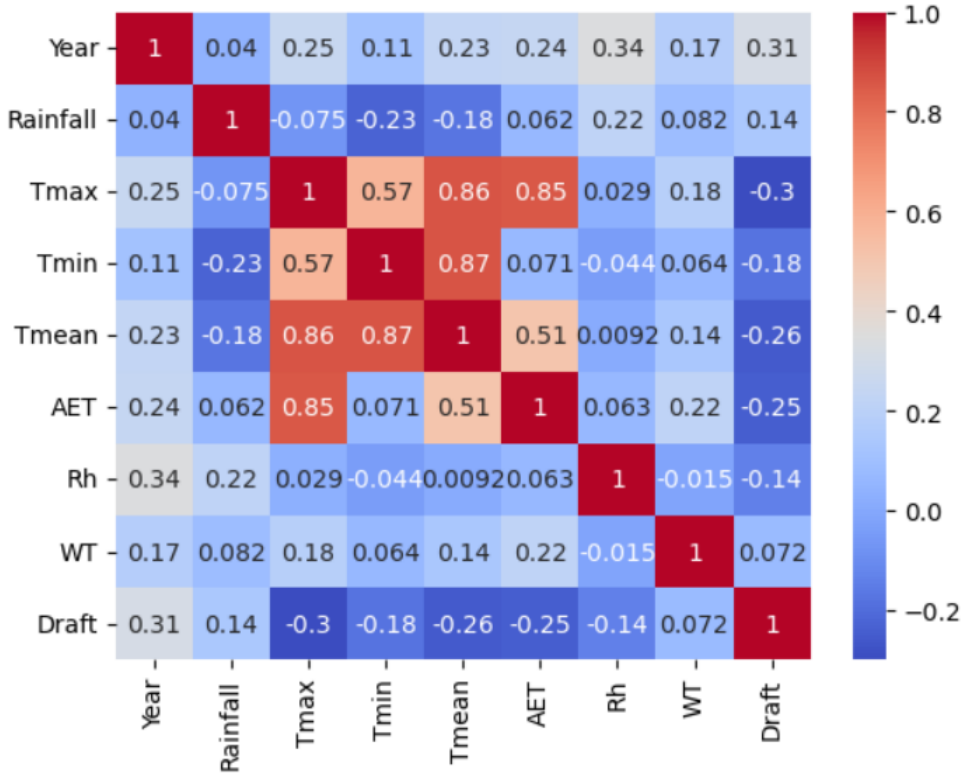


Figure 2.1

However, it is important to note that the report under consideration does not focus on investigating the underlying dependency structure of the data.

It should be highlighted that the presence of even a single variable exhibiting temporal dependence can render the entire set of variables dependent. This implies that the dependence of the system is influenced by the presence of any form of time-dependence in one of its variables, which can impact the reliability of the results obtained from the analysis. The figure above displays the heat map for the given set of features in the dataset.

Chapter 3

DATA SET PREPROCESSING

3.1 Benchmark

We have compared different machine learning models in terms of Signal Accuracy. The criteria taken for comparision of different models is **mean absolute error** . The results are shown in the result section below.

3.2 Dataset separation

LSTM network was trained and tested using 1 year of ground water level data.
80% of the data was used for training the network.
10% of the remaining data was used for validation the network in each case.
The remaining 10% of the data was used for testing the network in each case.
The goal is to evaluate the performance of different network on predicting ground water levels.
The approach involved splitting the data into training, validation and testing sets to assess the mean absolute error(MAE) and validation loss of the model.

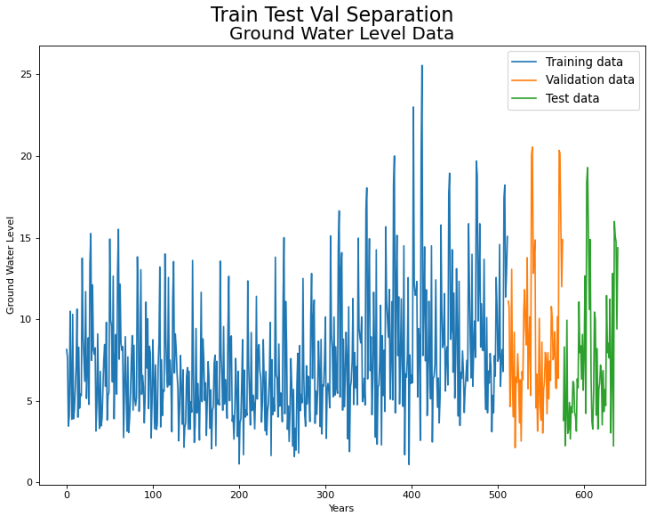


Figure 3.1: Dataset Separation

3.3 Parameter Tuning

3.3.1 Optimization Algorithm

Optimization is an iterative process in which the model is trained iteratively and results in a maximum and minimum function evaluation. There are three different optimization algorithms are explored while learning. The algorithms are as follows:

1. Stocastic Gradient Descent
2. Adaptive Gradient Descent
3. Adam

Stocastic Gradient Descent: In Stocastic Gradient Descent, a few n samples are choosen randomly instead of the whole dataset for the iteration. It is an iterative algorithm used to minimize the cost function.

Adaptive Gradient Descent : In Adaptive Gradient Descent, the learning rate is adaptively scaled for each parameter on the basis of previous gradient information. These type of optimization algorithm avoids oscillations.

Adam: In Adam, the learning rate is adaptively scaled for each parameter on the basis of moving average of the gradient and the squared gradient. Like Adaptive Gradient Descent, Adam also avoids the oscillations.

Adaptive Gradient Descent and Adam provides dynamic weights, whereas Stocastic Gradient provides static weights.

Among all the three algorithms, we get the best fitting in the case of mean absolute error in the presence of ADAM optimizer. All the following results have been generated by using the ADAM while learning the model.

3.3.2 Lookback Number

Lookback number is the number of the dataset point that our model is going to consider at a time to predict the future. We have observed that MAE decreases with the increase in lookback number till some certain point and after it MAE starts increasing with the increase in the lookback number.
The optimal lookback number at which MAE is the least as compared to others is around 20. So, here we have observed the global minimum in MAE.

3.3.3 Dropout

Dropout is a regularization method in the neural network. It prevents the over fitting of the model. The parameter dropout is the fraction of total units which will be randomly selected and turned off.

3.4 Window Feature Scaling

In machine learning, feature scaling is a critical component of training models, and data normalization is a commonly used method for achieving this. Normalization involves scaling the data feature-wise to transform it into values between 0 and 1, with the min-max scaling approach being a popular choice. While there are different normalization techniques available, this report employs a specific normalization relationship given as:

$$norm = \frac{X - X_{min}}{X_{max} - X_{min}}$$

The objective is to evaluate the impact of feature scaling on the performance of machine learning models. Although normalization is a well-established technique, it may not be effective for highly volatile time series data. When training a model with a constant lookback, such as an LSTM, it is more beneficial to normalize the data in each window of the lookback length. This process involves normalizing each training sequence between 0 and 1, and then de-normalizing the predictions made by the model. Window feature scaling improves training by allowing the model to capture the relative variance in the data within a small neighborhood. This method enhances learning speed by enabling the weights to capture the relative ordering in the input rather than a universal absolute ordering. Consequently, the model can capture increasingly complex or highly volatile time series data more effectively.

Chapter 4

DESCRIPTION OF WEAK LEARNERS

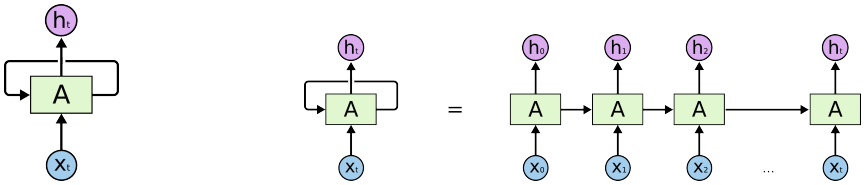
Recurrent Neural Networks (RNNs) are neural networks that are designed to process sequential data. They can take variable-length inputs and maintain an internal memory of previous inputs to capture temporal dependencies in the data. RNNs use a feedback loop that allows the output from one time step to be fed back into the network as input for the next time step. The hidden state of the network serves as its memory and allows it to capture context and dependencies of previous inputs.

RNNs are well-suited for tasks such as time-series analysis.Training RNNs can be challenging due to the vanishing gradient problem and exploding gradient problem, which occurs when gradients become very small and very large respectively and cause the network to stop learning effectively .Various techniques have been developed to address the vanishing gradient problem and improve the training of RNNs. So, to address this issue, we will study the following models.

4.1 LSTM

4.1.1 Introduction to LSTM

LSTM (Long Short-Term Memory) is a type of Recurrent Neural Network that has the ability to learn and remember long-term dependencies. The design of LSTM is aimed at addressing the long-term dependency problem often encountered in traditional RNNs. With LSTM networks, information can be retained and remembered over extended periods of time.



(a) Single unit (b) Unrolled unit

LSTM networks consist of a series of repeating modules, but unlike traditional

RNNs that have a single neural network layer in each module, LSTMs have four interacting layers. It consists of three gates namely:

1. Forget gate
2. Input gate
3. Output gate

It works on two states:

1. Hidden state
2. Cell state

The diagram depicted below illustrates how each line represents an entire vector that is carried from the output of one node to the input of another node. The learned neural network layers are represented by the boxes, and the circle represents the pointwise vector operations, such as vector addition. When lines merge, it indicates concatenation, whereas a line forking means that its content is copied and the copies are sent to different locations.

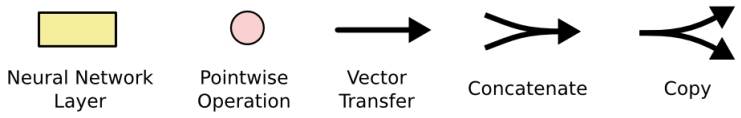


Figure 4.2: Some useful notations

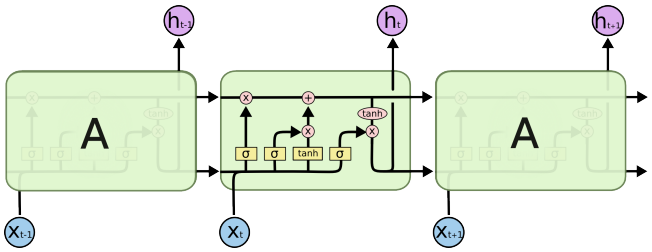


Figure 4.3: LSTM network

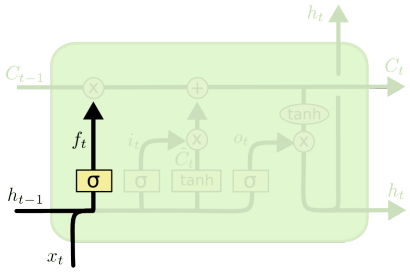
Core idea about LSTM:

The key aspect of LSTMs is the cell state, which can be modified by the gating mechanism. The gates allow the information to pass through them selectively and can be opened or closed as needed. Each gate comprises a sigmoid neural network layer and a pointwise multiplication operation.

The sigmoid layer produces an output of either 0 or 1. A value of 0 indicates that no information is allowed through the gate, whereas a value of 1 means that all the information is allowed to pass through the gate. As mentioned earlier, LSTMs have three gates that regulate the flow of information.

FIRST STEP:

Information which is not required is removed from the cell state which is decided by a sigmoid layer known as 'forget gate layer'. Here consider:
h_{t-1}: Information carried from the last layer from time t-1.
x_t: Information passed at the time t.



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

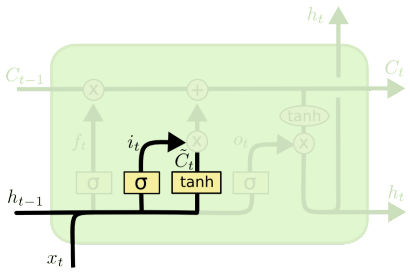
Figure 4.4

The forget gate layer processes the vectors h_{t-1} and x_t, resulting in a value between 0 and 1 for each value in the cell state. A value of 1 indicates that the corresponding information is retained entirely, while a value of 0 indicates that the corresponding information will be discarded from the cell state. This concept is governed by the following equation:
$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$

So, we have multiplied the old state with f_t, which results in the removal of information which is not required.

SECOND STEP:
New Information which is going to be stored in the cell state. It consists of two parts.

1. Sigmoid layer: INPUT GATE LAYER, which is responsible for the updation in the values
2. Tanh layer, which is responsible for the generation of new candidate values $\tilde{C}_t$ that can be added to the cell state at time t.

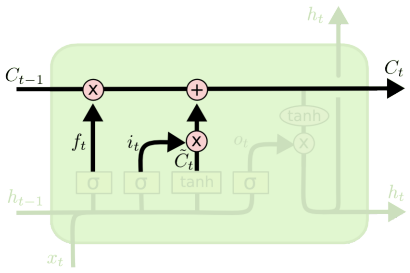


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figure 4.5

Hence, as h_{t-1} and x_t vectors passes through sigmoid layer and Tanh layer, so i_t and $\tilde{C}_t$ are produces respectively, where

$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i)$
$\tilde{C}_t = \tanh(W_c.[h_{t-1}, x_t] + b_c)$



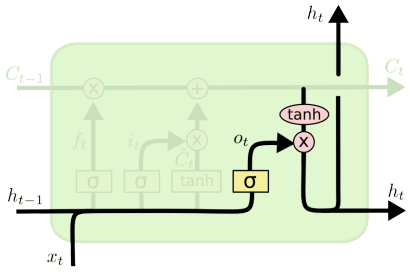
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figure 4.6

Here we will do the pointwise multiplication of the two vectors viz. i_t and $\tilde{C}_t$. As the information which is not required has been already removed from the cell state, so, we will add $i_t * \tilde{C}_t$ to cell state C_{t-1}, which gets updated to the new cell state C_t .

Hence, mathematically we can represent it as :
$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$

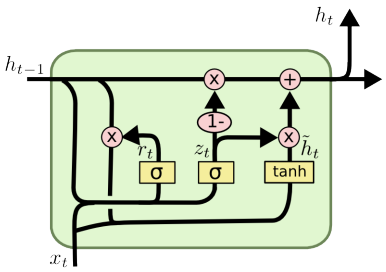
THIRD STEP:
At this stage, the LSTM produces an output based on the information stored in the cell state. The cell state is first passed through a sigmoid layer, which acts as a filter and generates a vector o_t. Then, the cell state is passed through a tanh function to generate values ranging from -1 to 1. The output vector h_t for time t is produced as the pointwise product of o_t and $\tanh(\tilde{C}_t)$, is given as:
$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$
$h_t = 0_t * \tanh(C_t)$



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

Figure 4.7

Hence, we get the final LSTM as follows:



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 4.8

4.2 BIDIRECTIONAL LSTM

4.2.1 Introduction to BIDIRECTIONAL LSTM:

Bidirectional long-short term memory (Bi-LSTM) is a technique that enables neural networks to incorporate sequence information in both forward (past to future) and backward (future to past) directions. Unlike regular LSTM networks that allow input flow in only one direction, Bi-LSTM networks enable input flow in both directions to retain both past and future information. Thus, Bi-LSTM is different from regular LSTM, where input flow can be either in the forward or backward direction.

Core idea about LSTM:

FIRST STEP:
Forward Pass: In the forward pass of a Bidirectional LSTM network, the input sequence is processed in the same way as in a regular LSTM network, where the input data flows from the first time-step to the last time-step. This forward pass generates a set of hidden states that encode the input sequence information in a forward direction.

SECOND STEP:
Backward Pass: In the backward pass of a Bidirectional LSTM network, the input sequence is processed in a reverse order, starting from the last time-step to the first time-step. This backward pass generates a separate set of hidden states that encode the input sequence information in a backward direction.

THIRD STEP:
Concatenation: In the final stage, the outputs of the forward and backward passes are concatenated element-wise, resulting in a final sequence of hidden states that contains both forward and backward information. This concatenated output sequence is then used as the input to the next layer in the network or as the output of the Bidirectional LSTM network. By combining the forward and backward information, Bidirectional LSTMs can capture both past and future context of the input sequence and thus, provide better performance than regular LSTMs.

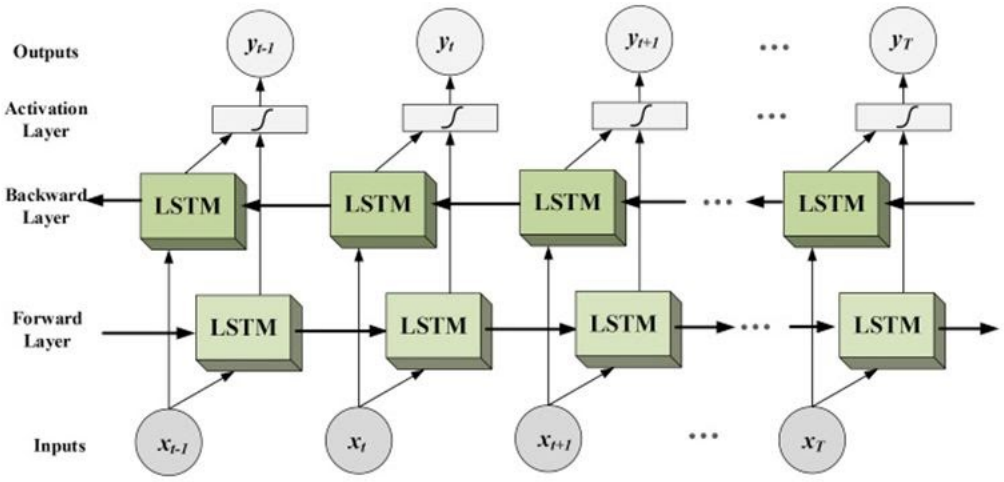


Figure 4.9

4.2.2 Architechture of BIDIRECTIONAL LSTM:

The Architecture which is used in Bidirectional LSTM is as follows:

The models.Sequential() function initializes a sequential model.The Bidirectional() function wraps an LSTM layer to create a bidirectional LSTM layer. The first Bidirectional layer has 500 LSTM units with ReLU activation and returns sequences. It takes an input tensor of shape (X_{train}.shape[1], X_{train}.shape[2]), where X_{train} is the training set.

The second Bidirectional layer has 100 LSTM units and aggregates the output from the first Bidirectional layer to create a final output. A Dense layer is added with 7 output units and sigmoid activation function.

A ModelCheckpoint callback is created to save the best model based on validation loss.The model is compiled using Adamax optimizer, mean squared error (mse) loss function and two metrics - mean absolute error (mae) and mean absolute percentage error (mape).

The model is trained on the training set X_{train} and y_{train} for 50 epochs with a batch size of 32. The training process is monitored using the ModelCheckpoint callback, and the best model is saved. The validation set (X_{val}, y_{val}) is used to evaluate the performance of the model during training.

Once the training is complete, the history object contains the training and validation losses, as well as the metrics for each epoch.

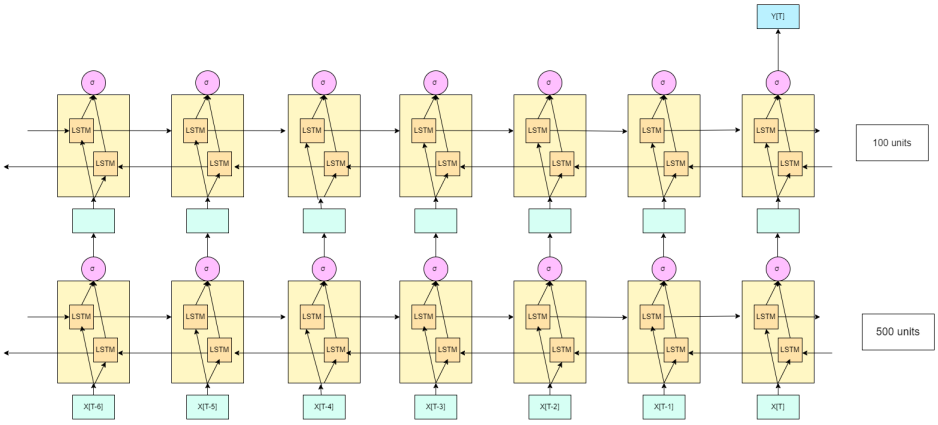


Figure 4.10: **Architecture of Bidirectional LSTM**

4.2.3 Performance of BIDIRECTIONAL LSTM

Here are the results of Bidirectional LSTM on the Ground Water level Data set:
Validation loss: 0.0031
MAE(Mean Absolute Error): 0.0538

Predictions on training data:

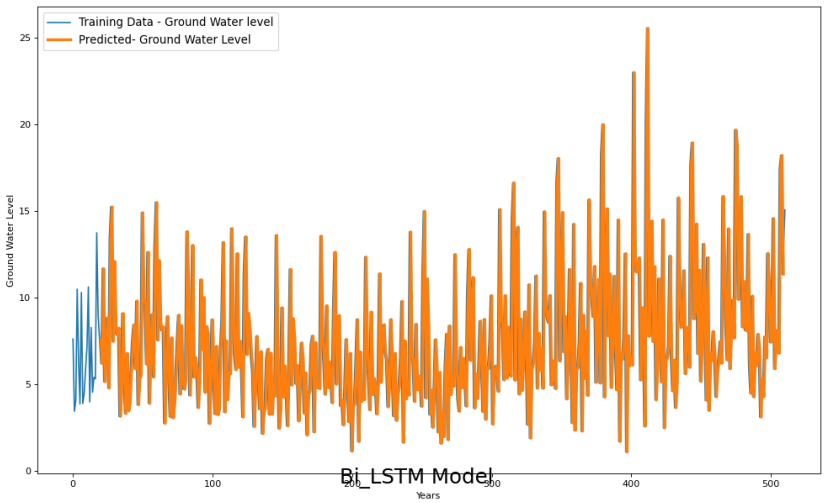


Figure 4.11: Bidirectional LSTM prediction on training data

Predictions on validation data:

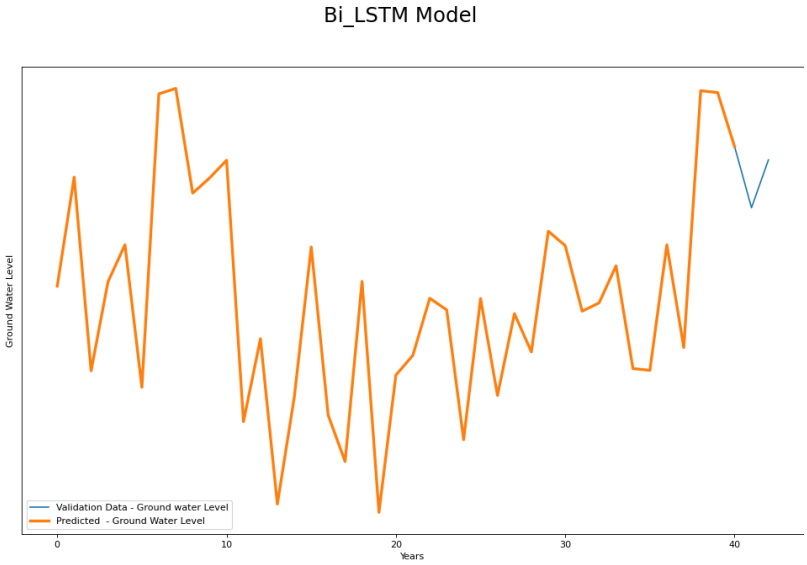


Figure 4.12: Bidirectional LSTM prediction on validation data

Predictions on testing data:

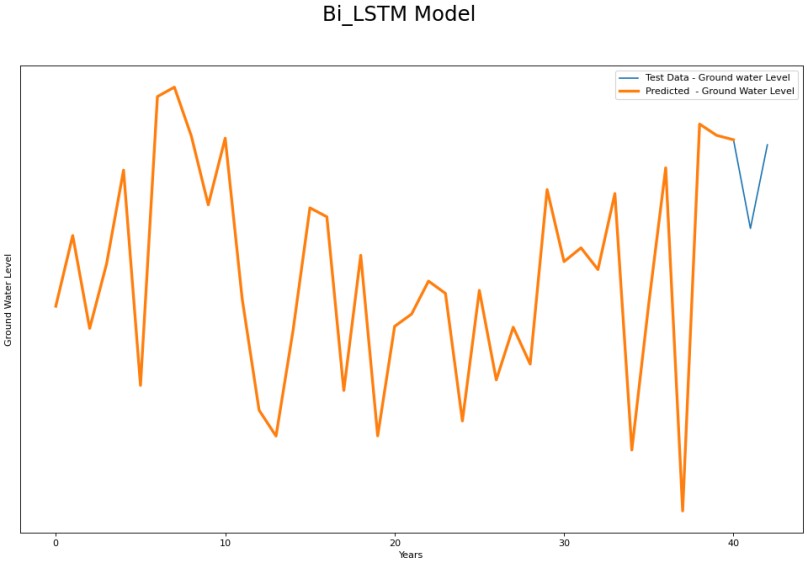


Figure 4.13: Bidirectional LSTM prediction on testing data

4.3 GRU

4.3.1 Introduction to GRU:

GRU, or Gated Recurrent Units, is a type of Recurrent Neural Network that addresses the issue of vanishing or exploding gradients during training. Unlike LSTM networks, GRUs operate solely on the hidden state and do not have a separate cell state. GRUs also employ gates to regulate the amount of information retained before updating the hidden state, similar to LSTM networks. However, GRUs have fewer gates compared to LSTM networks. Despite their relative simplicity, GRUs have demonstrated strong performance on various tasks and are widely used in the field of deep learning. GRU consists of two gates namely:

1. Reset gate
2. Update gate

Core idea about GRU: Similar to LSTM here also we have the following terminologies:
h_{t-1} = hidden state at previous timestep t-1
x_t = input vector at current timestep t
h_t = hidden state at current timestep t

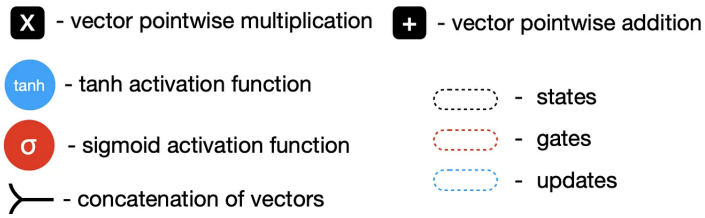


Figure 4.14

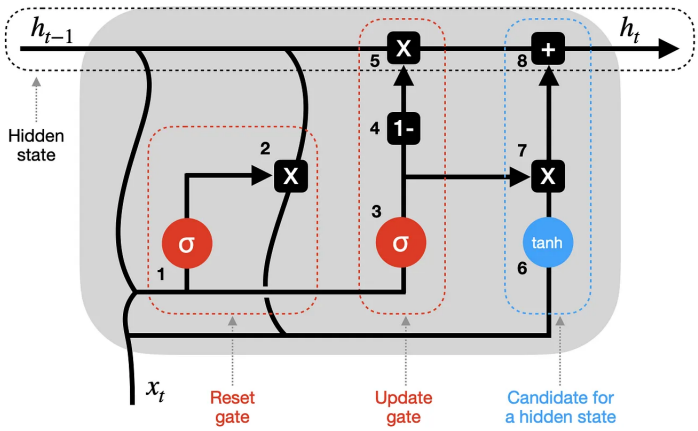


Figure 4.15

FIRST STEP:
To calculate the current hidden state in a GRU, the previous hidden state h_{t-1} and

the current input state x_t are multiplied by their respective weights and biases, and then added together. This sum is passed through a reset gate, similar to the forget gate in LSTM networks. The reset gate produces an output between 0 and 1, where 0 indicates information that should be discarded, 1 indicates information that should be retained completely, and values between 0 and 1 indicate partial retention of information. The previous hidden state is then reset by multiplying it with the output vector obtained from the reset gate.

SECOND STEP:
To update the hidden state in a GRU, the previous hidden state h_{t-1} and the current input state x_t are multiplied by their respective weights and biases, and then added together. This sum is passed through an update gate, which uses different weights and biases to scale these vectors. The output vector obtained from the update gate is subtracted from a vector containing all 1s, and this result is multiplied by the previous hidden state to obtain the final output vector. This final output vector updates the hidden state with the new information.

THIRD STEP:
As the previous hidden state is being reset earlier, so the outputs are combined with x_t, i.e the new input at timestep t. Before passing through a tanh To generate a potential hidden state candidate in a GRU, the previous hidden state h_{t-1} and the current input state x_t are multiplied by their respective weights and added biases. This result is passed through an activation function to obtain the candidate hidden state. The candidate hidden state is then multiplied by the output of an update gate and added to the previously modified hidden state h_{t-1} to generate the new hidden state h_t.

Hence, the same process repeats for next iteration for timestep $t + 1$, til it processes the entire sequence.

4.3.2 Architechture of GRU combined with Bidirectional LSTM:

The architecture which is used in GRU combined with Bidirectional LSTM is as follows:

The models.Sequential() function initializes a sequential model.The Bidirectional() function wraps an LSTM layer to create a bidirectional LSTM layer. The first Bidirectional layer has 500 LSTM units with ReLU activation and returns sequences. It takes an input tensor of shape (X_{train}.shape[1], X_{train}.shape[2]), where X_{train} is the training set. A GRU layer is added with 300 units and ReLU activation. A Dropout layer is added with a rate of 0.2 to prevent overfitting. A Dense layer is added with 7 output units and sigmoid activation function.

A ModelCheckpoint callback is created to save the best model based on validation loss. The model is compiled using Adamax optimizer, mean squared error (mse) loss function, and two metrics - mean absolute error (mae) and mean absolute

percentage error (mape).The model is trained on the training set X_{train} and y_{train} for 50 epochs with a batch size of 32. The training process is monitored using the ModelCheckpoint callback, and the best model is saved. The validation set (X_{val}, y_{val}) is used to evaluate the performance of the model during training.

Once the training is complete, the history object contains the training and validation losses, as well as the metrics for each epoch.

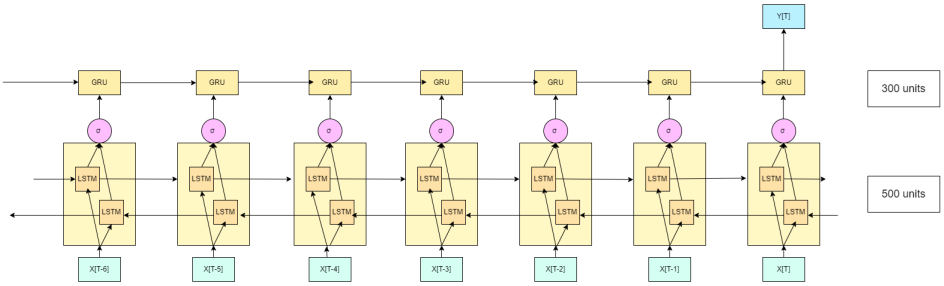


Figure 4.16: **Architecture of GRU combined with BiLstm**.

,

4.3.3 Performance of GRU combined with Bidirectional LSTM:

Here are the results of GRU combined with Bidirectional LSTM on the Ground Water level Data set:
Validation loss: 0.0030
MAE: 0.0532

Predictions on training data:

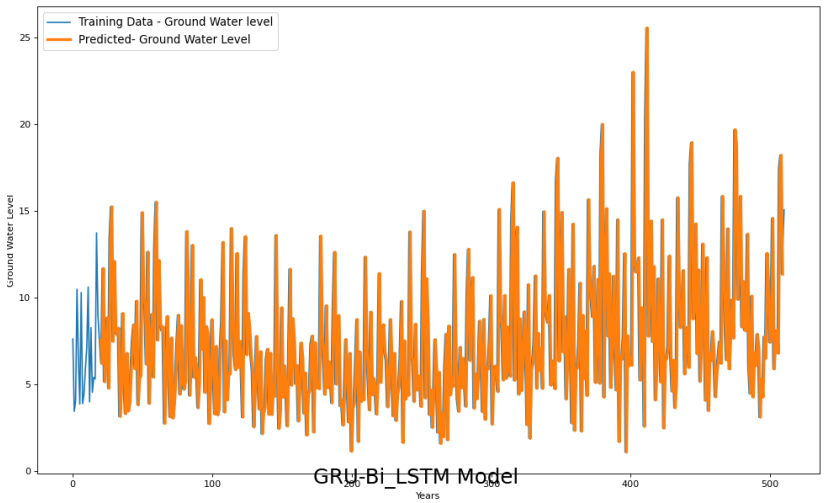


Figure 4.17: GRU-Bidirectional LSTM prediction on training data

Predictions on validation data:

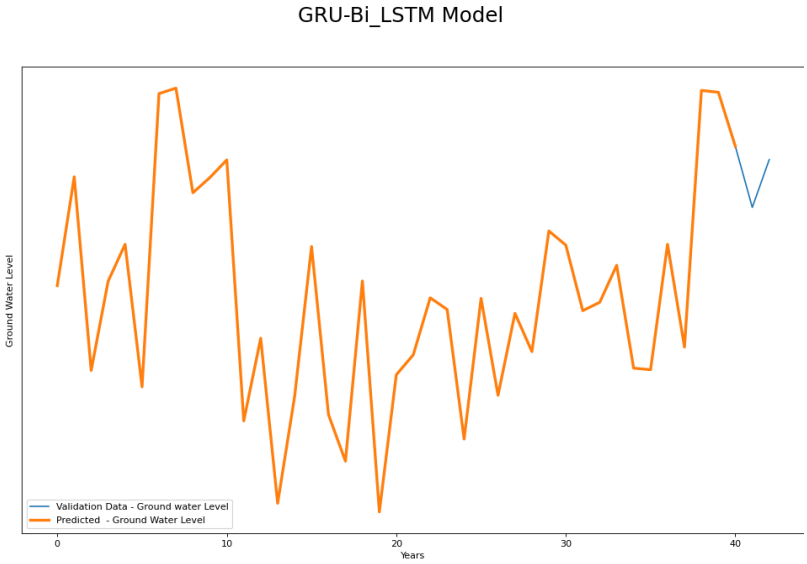


Figure 4.18: GRU-Bidirectional LSTM prediction on validation data

Predictions on testing data:

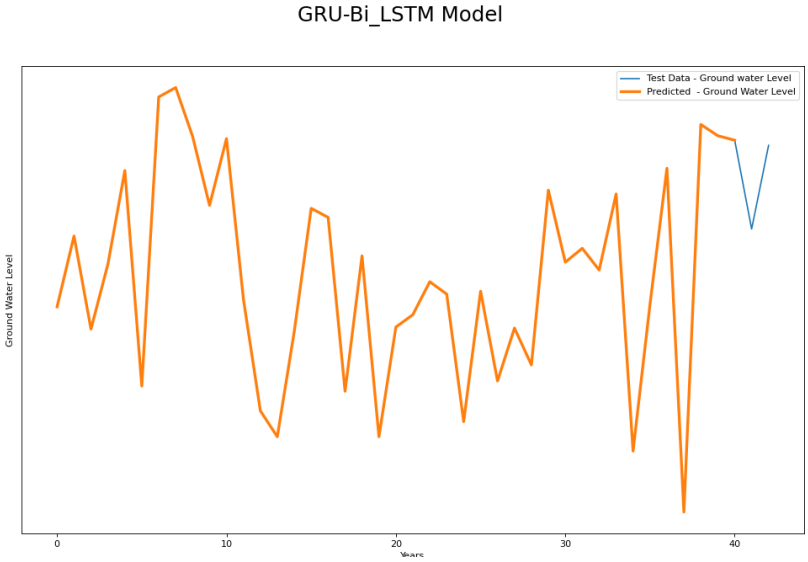


Figure 4.19: GRU-Bidirectional LSTM prediction on testing data

4.4 HYBRID

4.4.1 Introduction to HYBRID

Hybrid models are a type of machine learning model that combine different types of neural networks to improve their performance on a specific task. In this case, a hybrid model is being proposed that combines Bidirectional Long Short-Term Memory (BiLSTM), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) networks.

The proposed hybrid model would leverage the strengths of each of these three types of networks. The BiLSTM component would provide a wider context of the input data, the LSTM component would help retain long-term dependencies, and the GRU component would add computational efficiency and prevent overfitting.

By combining these three types of networks, the hybrid model would be expected to achieve better performance.

4.4.2 Architechture of HYBRID:

The architecture which is used in Hybrid is as follows:

The model is defined as a Sequential model, and the layers are added one after the other using the model.add() method.

The first layer is a Bidirectional LSTM layer with 500 units, a ReLU activation function, and the return sequences parameter set to True. The input shape of the layer is set to the shape of the training data.

The second layer is another Bidirectional LSTM layer with 100 units, and the return sequences parameter is also set to True.

The third layer is a GRU layer with 300 units, a ReLU activation function, and the return sequences parameter set to True. A Dropout layer with a rate of 0.2 is added after the GRU layer to prevent overfitting.

The fourth layer is an LSTM layer with 100 units, and a Dropout layer with a rate of 0.5 is added after it.

The final layer is a Dense layer with 7 units and a sigmoid activation function, which is used for multi-label classification.

The model is compiled using the Adamax optimizer, mean squared error loss function, and three evaluation metrics - mean absolute error (MAE), mean absolute percentage error (MAPE), and loss.

During training, a ModelCheckpoint callback is used to save the best performing model based on the validation loss. The model is trained for 50 epochs with a batch

size of 32, and the training and validation data are passed to the fit() method.

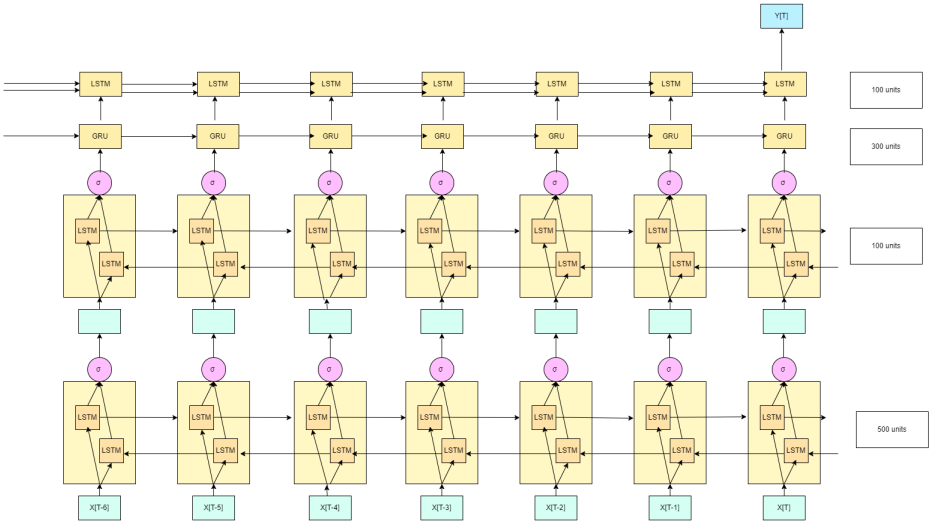


Figure 4.20: **Architecture of Hybrid**

4.4.3 Performance of HYBRID

Here are the results of Hybrid on the Ground Water level Data set:
Validation loss: 0.0031
MAE : 0.0557

Predictions on training data:

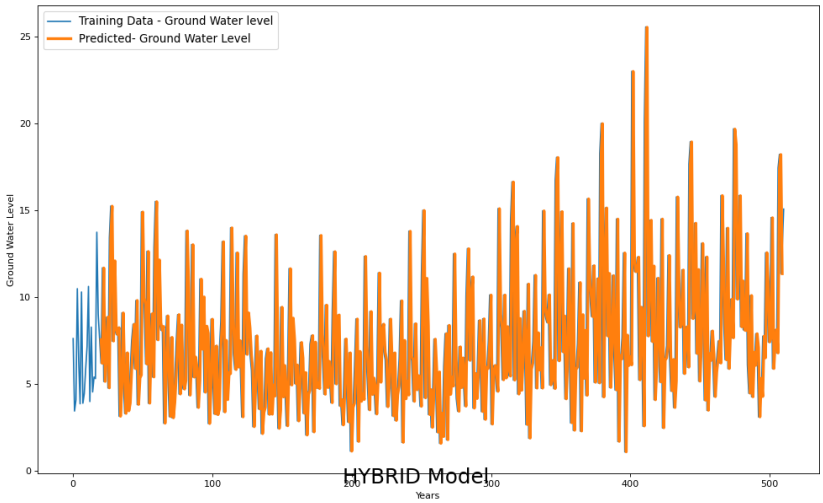


Figure 4.21: Hybrid prediction on training data

Predictions on validation data:

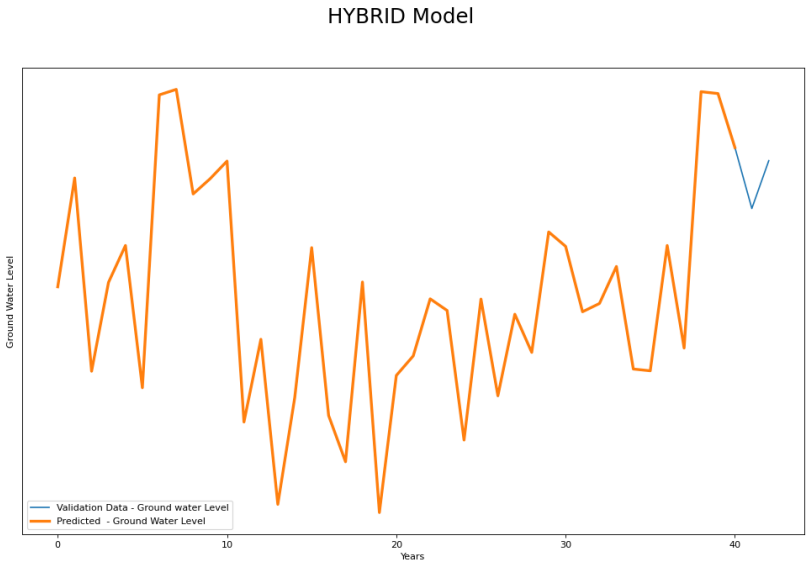


Figure 4.22: Hybrid prediction on validation data

Predictions on testing data:

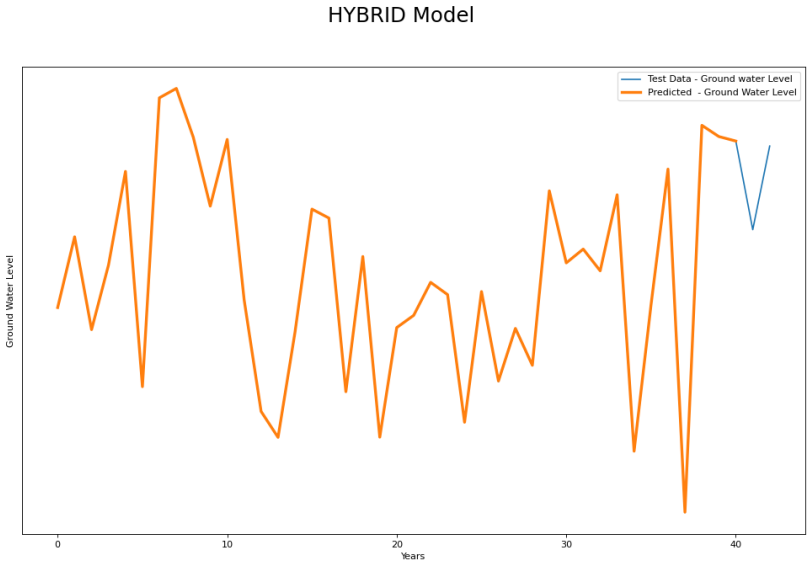


Figure 4.23: Hybrid prediction on testing data

4.5 HIGHWAY BIDIRECTIONAL LSTM

4.5.1 Introduction to HIGHWAY BIDIRECTIONAL LSTM

It is well-known that deeper neural networks can enhance the accuracy of a model. However, training deeper networks is not an easy task. It can be considerably difficult to optimize such models. To tackle this issue, a learning gating mechanism can be used to regulate the flow of information, which is similar to the approach used in LSTM. This gating mechanism allows the neural network to have different paths for information flow, which are referred to as "Information highways". Neural networks that incorporate these highways are called "Highway Networks".

Core idea of Highway Bidirectional LSTM

Let us consider a plain feedforward neural network which consists of L layers where the $l-th$ layer applies a non-linear transform H on the input, say x and generates the output, say y. It is governed by the mathematical equation if we omit biases for the sake of clarity:
$y = H(x, W_H)$
Hence, for the $i-th$ unit:
$y_i = H_i(x)$
After computing y_i it is then passes to the next layer.

In Highway network, there are two non-linear tranforms namely T and C where T is the **Transform Gate** and C is the **Carry Gate**,such that it alters the output as follows:
$y = H(x, W_H).T(x, W_T) + x.C(x, W_C)$
It reflects that how much of the output is produced by transforming and carrying the input, respectively.

As we have already seen that Bidirectional LSTM updates the hidden layer at each timestep t from the hidden layer h_{t-1}, where the hidden layer h_{t-1} is not saved anywhere. In order to improve Bidirectional LSTM, Highway Network adds a forget gate in feedward network to save the previous hidden h_{t-1}.

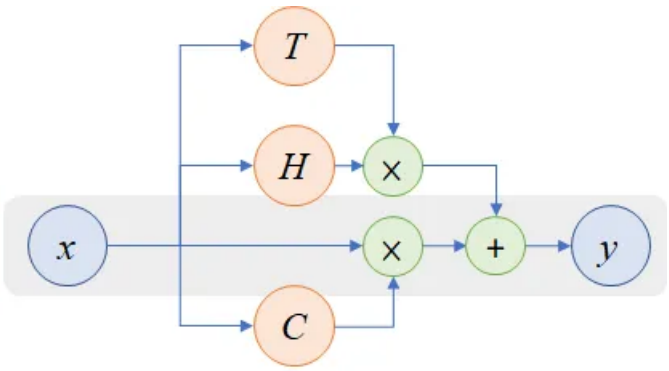


Figure 4.24

It is defined as:
$g_T = \sigma(W_T x + b_T)$
$g_C = \sigma(W_C x + b_C)$

$y = x \odot g_C + tanh(Wx + b) \odot g_T$

Here tanh($Wx + b$) is a feedward network, x is the input.

4.5.2 Architechture of HIGHWAY BIDIRECTIONAL LSTM:

The architecture which is used in Highway Bidirectional LSTM is as follows:

The models.Sequential() function initializes a sequential model. A Bidirectional LSTM layer is added with 500 LSTM units, ReLU activation, and return sequences. It takes an input tensor of shape (X_{train}.shape[1], X_{train}.shape[2]), where X_{train} is the training set.

Another Bidirectional LSTM layer is added with 300 LSTM units, ReLU activation, and return sequences. A third Bidirectional LSTM layer is added with 200 LSTM units, ReLU activation, and no return sequences. A Dropout layer is added with a rate of 0.5 to prevent overfitting.

A HighwayBlock layer is added with 400 units, a bias of -2.0, ReLU activation for the highway and sigmoid activation for the transform gate. Another Dropout layer is added with a rate of 0.2.

A Dense layer is added with 7 output units and sigmoid activation function. A ModelCheckpoint callback is created to save the best model based on validation loss.

The model is compiled using Adamax optimizer, mean squared error (mse) loss function, and two metrics - mean absolute error (mae) and mean absolute percentage error (mape).

The model is trained on the training set X_{train} and y_{train} for 50 epochs with a batch size of 32. The training process is monitored using the ModelCheckpoint callback, and the best model is saved. The validation set (X_{val}, y_{val}) is used to evaluate the performance of the model during training.

Once the training is complete, the history object contains the training and validation losses, as well as the metrics for each epoch.

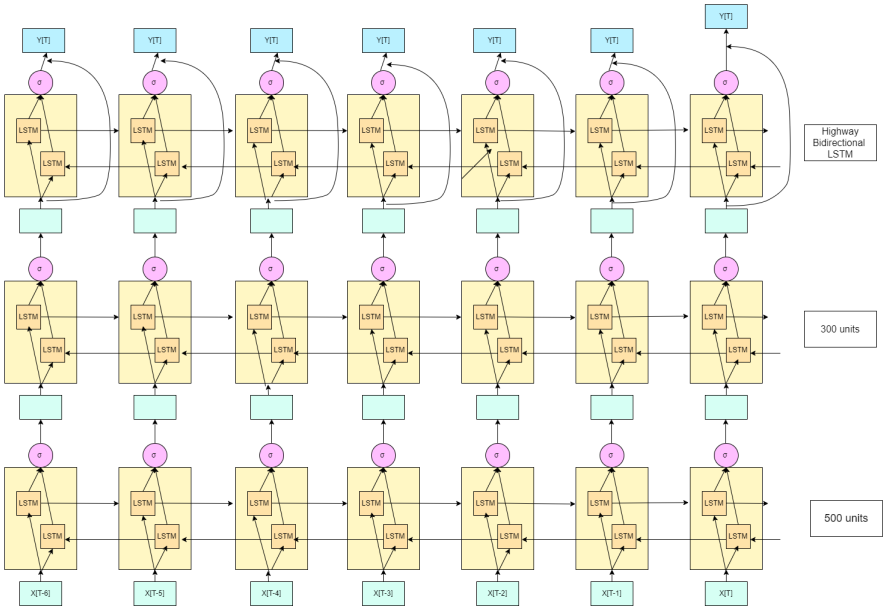


Figure 4.25: **Architecture of HIGHWAY BIDIRECTIONAL LSTM**

4.5.3 Performance of HIGHWAY BIDIRECTIONAL LSTM

Here are the results of Highway Bidirectional LSTM on the Ground Water level Data set:
Validation loss: 0.0030
MAE: 0.0533

Predictions on training data:

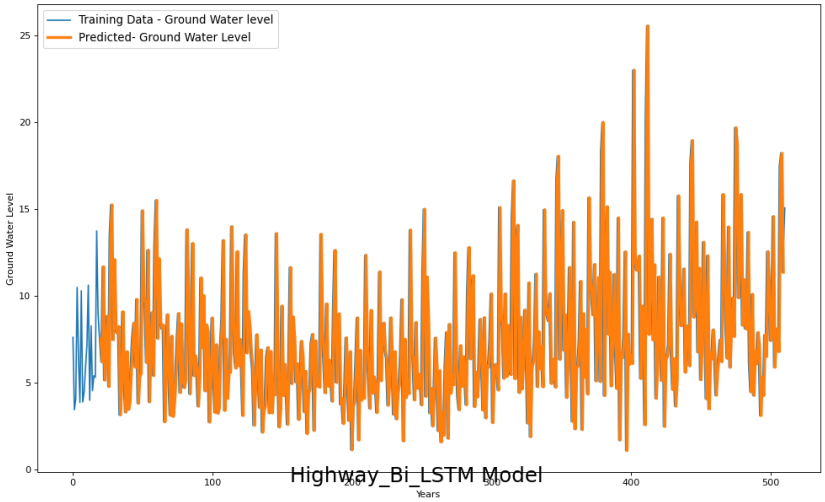


Figure 4.26: Highway Bidirectional LSTM prediction on training data

Predictions on validation data:

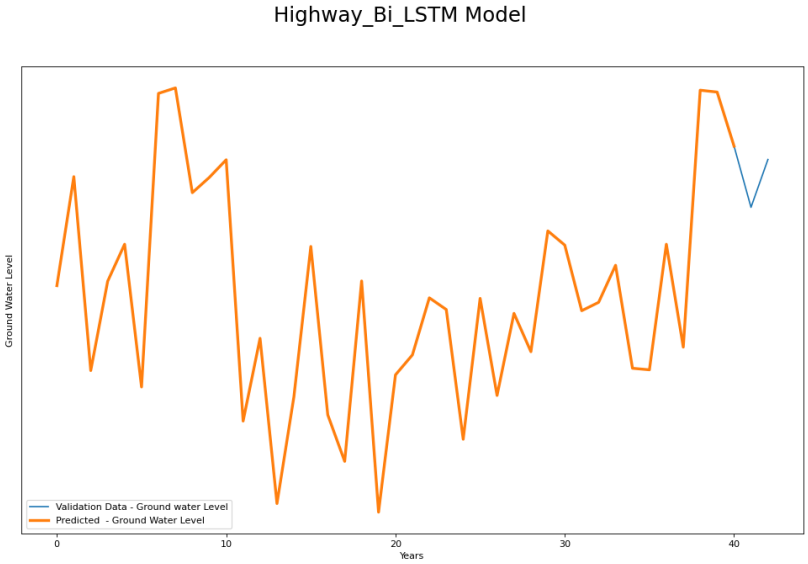


Figure 4.27: Highway Bidirectional LSTM prediction on validation data

Predictions on testing data:

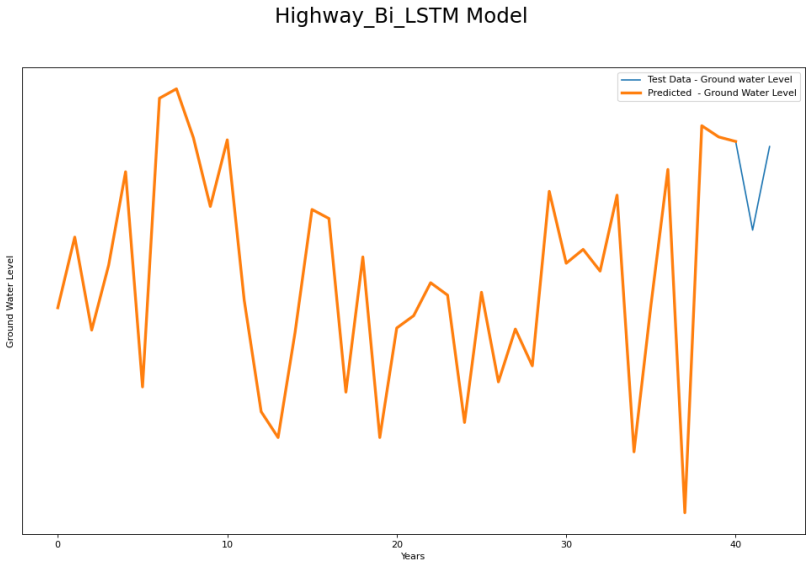


Figure 4.28: Highway Bidirectional LSTM prediction on testing data

4.6 TRANSFORMER

4.6.1 Introduction to TRANSFORMER

Transformers are a type of neural network used for processing sequences. Unlike recurrent neural networks, they do not have recurrent connections, which means they lack explicit memory of previous states. However, this allows the model to process the entire sequence at once rather than element-by-element, and it overcomes this lack of memory by perceiving the entire sequence simultaneously.
Transformers require more memory during training.

Core Idea about Transformer:
1) **Attention Mechanism in Transformer Architecture**:

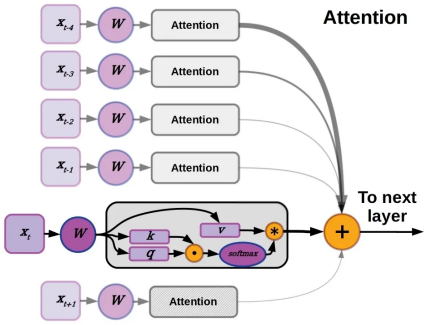


Figure 4.29

Attention is a technique used to selectively weight different elements in the input data. The weights are used to create an impact on the hidden layers in the downstream layers. Attention is implemented by parsing the input into key, query, and value vectors. The attention weights are obtained from the dot product of the key and query for a given position. The attention weights are then passed through the softmax function so that they sum to 1. The value vectors corresponding to each element are then summed according to their attention weights and passed to the next layers.

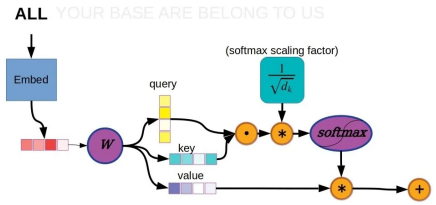


Figure 4.30

As shown the diagram, It can be easily seen that the attention layer W produces three output vectors based on the input, which are termed as above i.e. key, query

and value.

Since the entire sequence is fed into the model at once, the attention mechanism is applied in parallel to each element of the vector sequence. Therefore, every element is assigned an attention score simultaneously. These scores are multiplied by the square root of the number of units in the key vector and then passed through the softmax function to ensure that the attention weights are scaled between 0 and 1. This ensures that the sum of all attention weights is equal to 1, and the corresponding value vectors are multiplied by their weights to indicate the significance of each vector in the original sequence. The resulting vectors are then summed up, resulting in a resulting vector that is passed through the feed-forward fully connected layer.

Encoder layer of transformer:
The aforementioned layer is commonly referred to as the self-attention layer. When it is paired with a dense feed-forward layer, it forms a single encoder layer. The feed-forward layer consists of two linear layers separated by a ReLU activation function. Initially, the input is passed through the first linear layer to undergo a transformation, followed by setting the resulting values to either be 0 or always greater than 0. Finally, the resulting values are passed through the second linear layer to produce the final output for the feed-forward layer. Multiple transformers may use several encoder layers, each comprising a self-attention layer and other feed-forward layers.

Encoder layer of transformer:
Decoder layer consists of three layers, viz. Self-attention layer, Encoder-Decoder attention layer and feed-forward layer.

4.6.2 Architechture of TRANSFORMER:

The architecture which is used in Transformer is as follows:

The model consists of three TransformerEncoder layers, each with a specified number of heads, feedforward dimension, and key/value dimension. A Time2Vector layer is also used to encode the time component of the input sequence.

The input sequence is defined as a tensor with shape (seq-len, 7), where seq-len is the length of the input sequence and 7 is the number of features. The input sequence is fed into the time-embedding layer, and the resulting output is concatenated with the input sequence along the feature axis using the Concatenate layer.

The concatenated sequence is then passed through the three TransformerEncoder layers to learn the temporal dependencies between the input features. The output of the final TransformerEncoder layer is passed through a GlobalAveragePooling1D layer to reduce the temporal dimension to a single value. This value is then passed through two Dropout and Dense layers to produce the final output tensor with shape (7,) representing the predicted values for each of the 7 features in the output sequence.

The model is compiled using the mean squared error loss function, the Adam

optimizer, and metrics for mean absolute error and mean absolute percentage error.

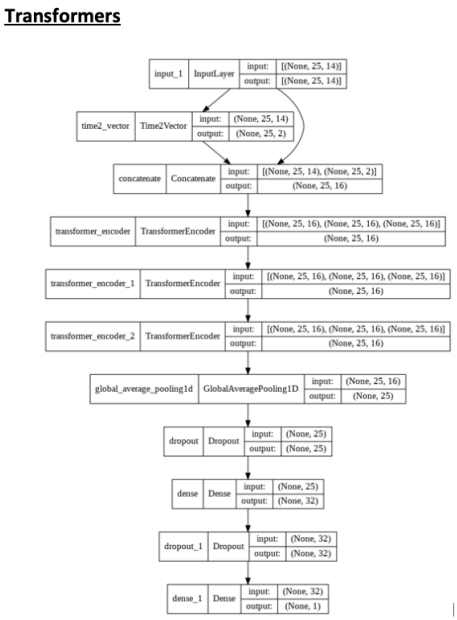


Figure 4.31

4.6.3 Performance of TRANSFORMER

Here are the results of Transformer on the Water level Data set:
Validation loss: 0.00049440
MAE: 0.0474

Predictions on training data:

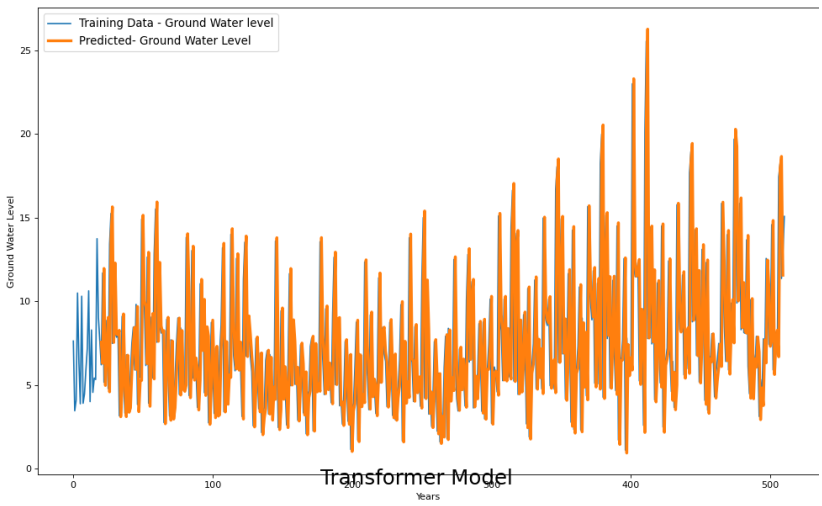


Figure 4.32: Transformer prediction on training data

Predictions on validation data:

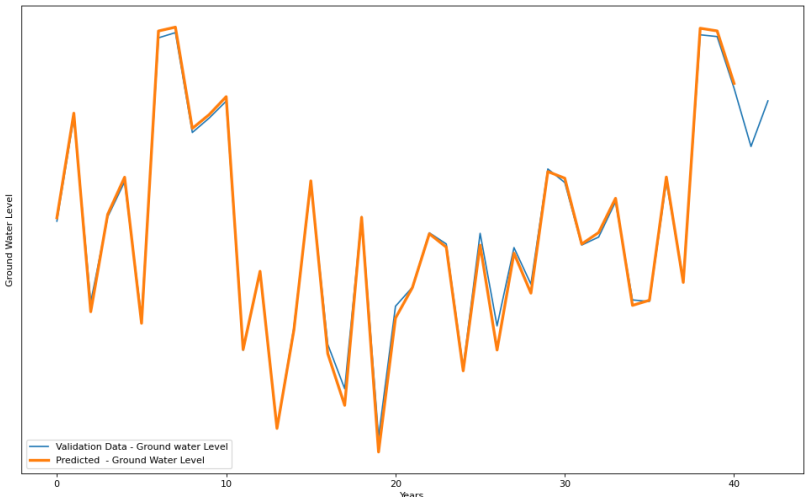


Figure 4.33: Transformer prediction on validation data

Predictions on testing data:

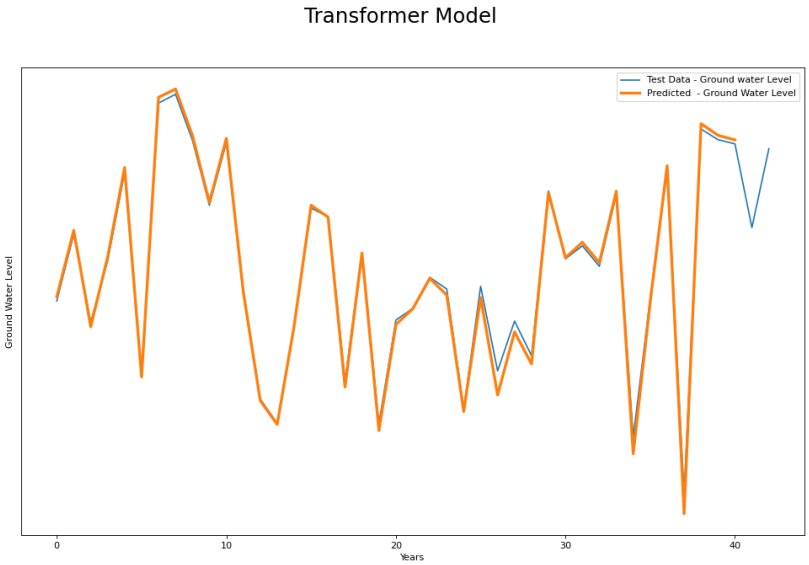


Figure 4.34: Transformer prediction on training data

Chapter 5

DESCRIPTION OF COBRA TECHNIQUE

We will be starting with an original data set D_n. It will be divided into two data sequences, D_k (starting k points) and D_l (remaining $n-k$ points).

We have a set of M models which takes in a certain input, and gives an estimate r*. These basic estimators – basic machines – are assumed to be generated using only the first subsample (D_k). Hence, each model predicts provides an estimation of r*(x) on the basis ofD_k alone. Cobra provides a method to combine all the results of the model, and hence obtain a better estimate using the given models.

The goal of COBRA is to predict response for a new input point x. Using the threshold level as ϵ, we would create a ϵ-ball around the point (x_i,y_i), such that $|r_m(x_i) - r_m(y_i)| \leq \epsilon$. For the corresponding y_i's, we would be taking the average (or some other function) for the prediction of x.

Let us consider the weighted average of the corresponding values of y_i. We define the collective estimator T_n to be:

$$T_n(r_k(x)) = \sum_{i=1}^{l} W_{n,i}(x) Y_i$$

where the random weights $W_{n,i}(x)$ take the form

$$W_{n,i} = \frac{1_{\cap_{m=1}^{M} |r_{k,m}(X_i) - r_{k,m}(X_i)| \leq \epsilon_l}}{\sum_{j=1}^{l} 1_{\cap_{m=1}^{M} |r_{k,m}(X_i) - r_{k,m}(X_i)| \leq \epsilon_l}}$$

where ϵ_1 has been chosen appropriately using a certain algorithm. The restriction of the above ϵ_1-ball can be relaxed by imposing a fixed fraction $\alpha \in \frac{1}{M}, \frac{2}{M}, ..., 1$ such that:

$$W_{n,i} = \frac{1_{\cap_{m=1}^{M} |r_{k,m}(X_i) - r_{k,m}(X_i)| \leq \epsilon_l}}{\sum_{j=1}^{l} 1_{\cap_{m=1}^{M} |r_{k,m}(X_i) - r_{k,m}(X_i)| \leq \epsilon_l}}$$

Asymptomatically speaking ($\alpha \to 1$), the behaviour remains the same.

The Weak learners defined above will be used in the COBRA.

Architecture of COBRA:

The approach being taken involves the use of a pre-trained ensemble of deep learning models to make predictions on a given dataset, and then analyzing the differences between the consecutive predictions of each individual model using the Extracount method.

In order to optimize the performance of the ensemble, a grid search is being performed over a range of values for the hyperparameters epsilon and alpha.
The ensemble is defined and trained using the MyCobra class, and during each iteration of the grid search, the ensemble is fitted to the training data and used to make predictions on a separate validation set.

The mean absolute error (MAE) between the predicted and true labels for the validation set is then calculated, and the values of the best MAE, epsilon, and alpha are updated if the current MAE is better than the previous best. Overall, the goal is to find the best hyperparameters for the ensemble using a combination of pre-trained deep learning models, grid search, and validation data, in order to optimize its performance on the given dataset.

Performance of COBRA:
Here are the results of Transformer on the Water level Data set:
Validation loss: 0.00024220
MAE: 0.0337

Predictions on training data:

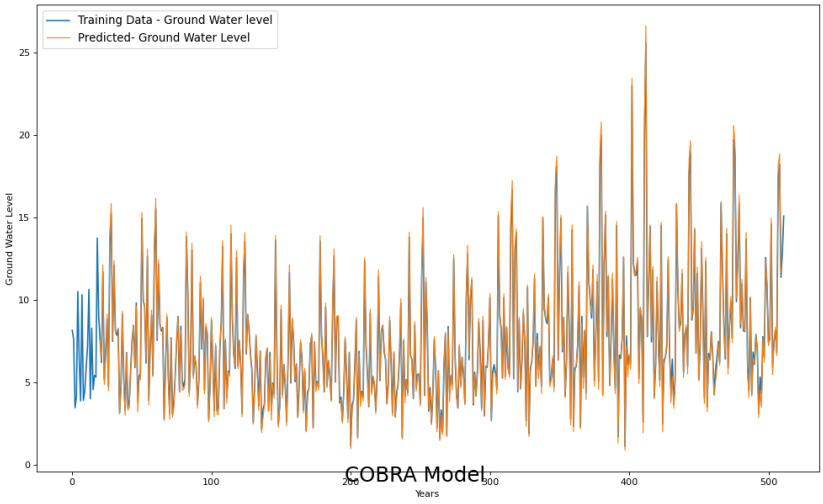


Figure 5.1: COBRA prediction on training data

Predictions on validation data:

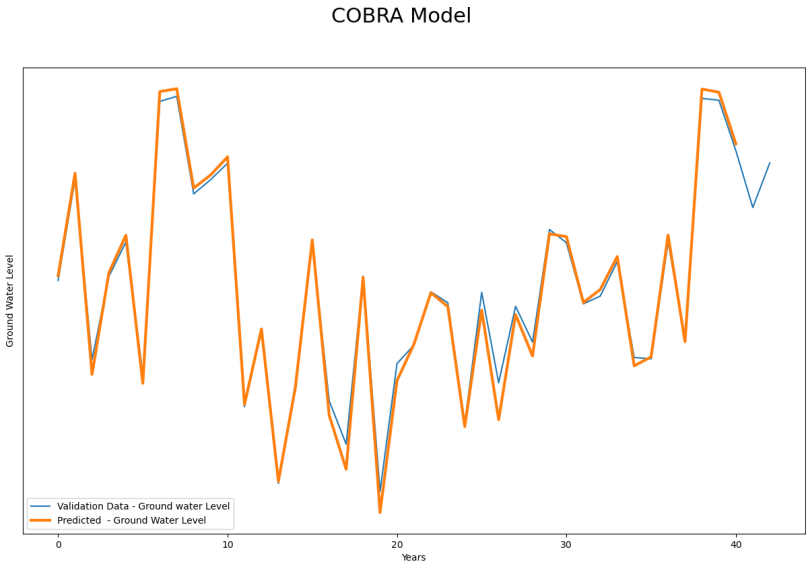


Figure 5.2: COBRA prediction on validation data

Predictions on testing data:

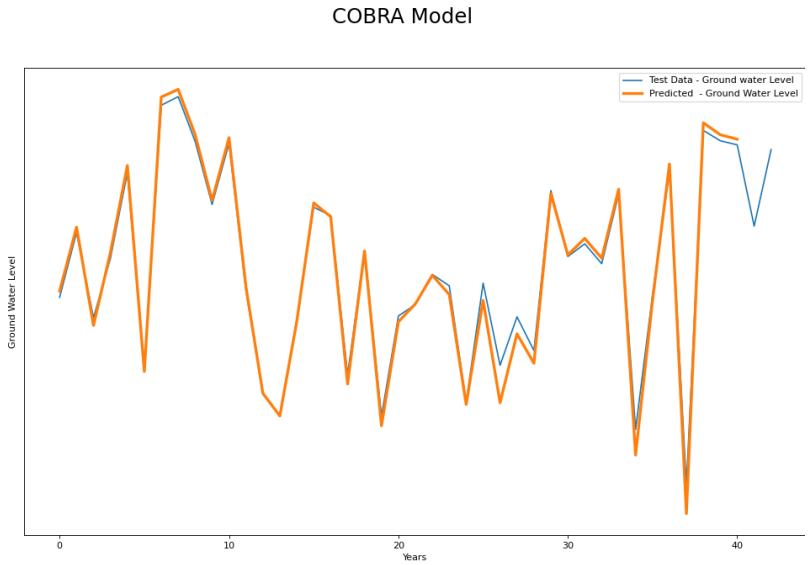


Figure 5.3: COBRA prediction on testing data

Chapter 6

RESULTS

The results of our implementation of the above described models, namely Bidirectional LSTM, GRU with Bidirectional LSTM, Transformer, Hybrid of Bidirectional LSTM, LSTM and GRU, Highway Bidirectional LSTM, and COBRA, are presented below. For each model, the plots of predictions on training, validation, and testing data were obtained, which are shown in the previous chapters. The metrics used for evaluation are:

Mean Absolute Error (mae)

Loss.

Models	Test		Val		Train	
	mae	loss	mae	loss	mae	loss
Bidirectional LSTM	0.0549	0.0032	0.0536	0.0031	0.0536	0.0031
GRU with Bidirectional LSTM	0.0545	0.0032	0.0532	0.0030	0.0532	0.0030
Hybrid	0.0546	0.0032	0.0534	0.0031	0.0533	0.0030
Highway Bidirectional LSTM	0.0545	0.0032	0.0532	0.0030	0.0532	0.0030
Transformer	0.0192	0.0006	0.0178	0.0005	0.0186	0.0005
COBRA	0.0066	0.0002	0.0071	0.0002	0.0066	0.0002

Table 6.1: Evaluation of Weak learners and COBRA

Chapter 7

CONCLUSION

The validation loss results of the different models applied to the Ground water level dataset provide insights into their effectiveness for this particular task.

The transformer model achieved the lowest validation loss of 0.00018, which is an impressive result and indicates that this model is well-suited for this type of data. The transformer model has shown great success in various natural language processing tasks and has recently been applied to time-series data as well, with promising results.

The hybrid model with stacked bidirectional LSTMs and GRU achieved the second lowest validation loss of 0.00301, indicating that it is also an effective model for this task. This model combines the strengths of both LSTM and GRU networks and uses bidirectional layers to improve the learning capabilities.

The hybrid model with 3 stacked bidirectional LSTMs with a Highway block also achieved a validation loss of 0.00301, indicating that this model is also a promising approach for this task. The Highway block is a modification of the standard LSTM cell that helps to improve the flow of information through the network.

The stacked bidirectional LSTM model achieved a validation loss of 0.00308, which is slightly higher than the hybrid models with GRU and Highway block. This model uses bidirectional LSTM layers to capture the temporal dependencies in the data and has shown success in various time-series tasks.

The hybrid model with 2 stacked bidirectional LSTMs, a GRU layer, and an LSTM layer achieved the highest validation loss of 0.00311 among the models listed. This model combines multiple layers and types of recurrent networks to capture complex patterns in the data.

In conclusion, the transformer model outperformed the other models, while the hybrid models with stacked bidirectional LSTMs and GRU or a Highway block showed promising results as well. It is recommended to further investigate and compare the performance of these models on other datasets to determine their effectiveness in various applications. Overall, the results demonstrate the importance of choosing an appropriate model architecture and hyperparameters for the given data.

However, "cobra", which involves an ensemble of multiple weak learners working in conjunction with each other, has demonstrated superior performance and produced the most favorable outcomes among all the models evaluated.

Chapter 8

FUTURE PLANS

Our proposed plan is to deploy the BERT model onto the groundwater level dataset, effectively capturing the spatiotemporal features of the data. However, the constraints of time have impeded our ability to execute this implementation.

Additionally, we intend to implement a combined regression technique, which encompasses an ensemble model of bidirectional LSTM, GRU combined with bidirectional LSTM, highway bidirectional LSTM, and transformer with dynamically modulated epsilon values and varying alpha, with the aim of further refining the accuracy of our predictions.

Bibliography

1. IPCC, Climate Change (2014) Synthesis Report, Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change [Core Writing Team, R.K. Pachauri and L.A. Meyer (eds.)]. IPCC, Geneva, Switzerland, 2014,151

2. Alkhaier F, Flerchinger GN, Su Z, (2012 a), Shallow groundwater effect on land surface temperature and surface energy balance under bare soil conditions: modeling and description, Hydrol. Earth Syst. Sci., 16, 1817–1831, doi:10.5194/hess-16-1817-2012

3. Alkhaier F, Su Z, Flerchinger GN (2012 b) Reconnoitering the effect of shallow groundwater on land surface temperature and surface energy balance using MODIS and SEBS, Hydrol. Earth Syst. Sci., 16, 1833–1844, doi:10.5194/hess-16-1833-2012

4. Berg A, Lintner BR, Findell KL, Malyshev S, Loikith PC, Gentine P (2014) Impact of soil moisture– atmosphere interactions on surface temperature distribution. J Clim 27(21):7976–7993. doi:10.1175/ JCLID-13-00591.1

5. Chitsazan M, Rahmani G and Neyamadpour A (2015) Forecasting Groundwater Level by Artificial Neural Networks as an Alternative Approach to Groundwater Modeling, Journal Geological Society of India, Vol. 85, 98 – 106.

6. Cohen D, Person M, Daannen R, Locke S, Dahlstrom D, Zabielski V, Winter TC, Rosenberry DO, Wright H, Ito E, (2006) Groundwater- Supported evapotranspiration within glaciated watersheds under conditions of climate change, Journal of Hydrology, 320 (3-4) 484-500. 15

7. Coulibaly P, Anctil F, Bobee B (1999) Hydrological forecasting using artificial neural networks: the state of art. Can J Civ Eng 26(3):293–304.

8. Cui Z, Ke R,Wang Y(2017) Deep Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction, https://www.researchgate.net/publication/322328703 Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction.

9. Djurovic N, Domazet M, Stricevic R, Pocuca V, Spalevic V, Pivic R, Gregoric E, Domazet U (2015) Comparison of Groundwater Level Models Based on Artificial Neural Networks and ANFIS, The ScientificWorld Journal, 742138, 1-13, doi:http://dx.doi.org/10.1155/2015/742138.

10. Green TR, Taniguchi M, Kooi H, Gurdak JJ, Allen DM, Hiscock KM, Treidel H, Aureli A, (2011) Beneath the surface of global change: Impacts of climate change on groundwater, Journal of Hydrology 405 532– 560.

11. Hochreiter S and SchmidhuberJ(1997) Long Short-Term Memory, Neural Comput. 9: 1735–1780.

12. Kumar KK, Kumar KR, Ashrit RG, Deshpande NR and Hansen JW (2004) Climate Impacts on Indian Agriculture. Int. J. Climatol. 24: 1375 – 1393.

13. Kumar S, Halder S, Singhal DC (2011) Groundwater Resources Management through Flow Modeling in Lower Part of Bhagirathi - Jalangi Interfluve, Nadia, West Bengal, Journal Geological Society of India, 78, 587-598

14. Kurata G, Ramabhadran B, Saon G and Sethy A (2017) Language Modeling with High- way LSTM, arXiv:1709.06436.

15. Mackay JD, Jackson CR, Wang L, A lumped conceptual model to simulate groundwater level time-series (2014), Environmental Modelling & Software, 61, 229 – 245.

16. Mall RK and Anandha KJ (2010) Impact of Climate Change on estimation of groundwater resources of India, proceedings of workshop: GroundwaterResources estimation, February 23 – 24, 2010, Central Ground Water Board & IIT-Delhi, New Delhi. pp 166-174.

17. Maxwell RM (2011) Development of a coupled groundwater-atmosphere model. Mon. Weather Rev., 139(1): 96-116. doi: 10.1175/2010MWR3392.117.

18. Mohanty S, Jha MK, Kumar A and Sudheer KP(2010) Articial Neural Network Modeling for Ground- water Level Forecasting in a River Island of Eastern India Water Resource Manage 24, 1845 – 1865.

19. Natural Resources Management and Environment Department (NR) under Food and Agriculture Organization (FAO) of the United Nations (1998) Crop Evapotranspiration – Guidelines for Computing Crop Water Requirements. FAO Irrigation and Drainage Papers – 56.

20. Nayak PC, Satyajirao YR, Sudheer KP (2006), Groundwater Level Forecasting in a Shallow Aquifer Using Artificial Neural Network Approach, Water Resources Management 20: 77–90 DOI: 10.1007/s11269-006- 4007-z.

21. Nourani V and Mousavi S (2016) Spatiotemporal groundwater level modeling using hybrid artificial intelligence-meshless method, Journal of Hydrology, 536 (C), 10 – 25.

22. Rakhshandehroo G, Akbari H, Igder MA, Ostadzadeh E (2018) Long-Term Groundwater-Level Forecasting in Shallow and Deep Wells Using Wavelet Neural Networks Trained by an Improved Harmony Search Algorithm, Journal of Hydrologic Engineering, 23(2): 04017058, DOI: 10.1061/(ASCE)HE.1943-5584.0001591

23. Style G (2009) Application of artificial neural network in the field of Geohydrology, University of the free State, South Africa.

24. Shaoyuan F, Shaozhong K, Zailin H, Shaquin C and Xiaomin M (2007). Neural network to simulate regional ground water levels affected by human activities. Groundwater, 46 (1), 80 – 90.

25. Suryanarayana Ch, Sudheer Ch., Mahammood V, Panigrahi BK (2014) An integrated wavelet-support vector machine for groundwater level prediction in Visakhapatnam, India. Neurocomputing 145: 324335.

26. Uddameri V (2006). Using statistical and artificial neural network models to forecast potentiometric levels at a deep well in South Texas. Environ. Geol., 51, 885 – 895.

27. Whan K, Zscheischler J, Orth R, Shongwe M, Rahimi M, Asare EO, Seneviratne SI (2015) Impact of soil moisture on extreme maximum temperatures in Europe. Weather Clim Extremes 9:57–67. doi:10. 1016/j.wace.2015.05.001

28. Wunsch A and Liesch Tand Broda S (2018). Forecasting Groundwater Levels using Nonlinear Autoregressive Networks with Exogenous Input (NARX). Journal of Hydrology. 10.1016/j.jhydrol. 2018.01.045.

29. Zhang J, Zhu Y, Zhang X, Ye M, Yang J (2018), Developing a Long Short-Term Memory (LSTM) based model for predicting water table depth in agricultural areas, Journal of Hydrology 561, 918–929