Improved Neural Network based Modeling for Wind Speed Prediction in the Indian Ocean Region

A project report submitted in partial fulfillment

of the requirements for the degree of

Bachelor of Technology

in

Electronics & Computer Engineering

by

Deeptimaan Banerjee 19BLC1162

Prakhar Narain Srivastava 19BLC1019

Tarunveer Singh Sidhu 19BLC1186

School of Electronics Engineering,

Vellore Institute of Technology, Chennai,

Vandalur-Kelambakkam Road,

Chennai - 600127, India.

April 2023

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Declaration

I hereby declare that the report titled ***Improved Neural Network based Modeling for Wind Speed Prediction in the Indian Ocean Region*** submitted by me to the School of Electronics Engineering, Vellore Institute of Technology, Chennai in partial fulfillment of the requirements for the award of **Bachelor of Technology** in **Electronics and Computer Engineering** is a bona-fide record of the work carried out by me under the supervision of ***Dr. Saurav Gupta***.
I further declare that the work reported in this report, has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or University.

Sign: _____

Name & Reg. No.: _____

Date: _____

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Electronics Engineering

Certificate

This is to certify that the project report titled ***Improved Neural Network based Modeling for Wind Speed Prediction in the Indian Ocean Region*** submitted by ***Deeptimaan Banerjee* (19BLC1162)** ***Prakhar Narain Srivastava* (19BLC1019)** ***Tarunveer Singh Sidhu,* (19BLC1186)** to Vellore Institute of Technology Chennai, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in **Electronics and Computer Engineering** is a bona-fide work carried out under my supervision. The project report fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

Supervisor

Signature: .....

Name: .....

Date:

Head of the Department

Signature: .....

Name: .....

Date:

Examiner

Signature: .....

Name: .....

Date:

(Seal of the School)

Abstract

One area that has not received adequate attention in contemporary times is Wind Speed Prediction. Accurate Wind Speed Prediction is important for various industries such as those concerning shipping, offshore oil and gas exploration, and cyclone predictions. Wind Speed can be predicted accurately over a short period of time and is majorly focused on a particular longitude, and latitude. This research paper aims to predict wind speed over a large region, i.e. particularly in the Indian Ocean region using next frame prediction via neural network models constituting multiple longitudes and latitudes. The study compares to predict wind speed and represent it as a 2D vector, researchers have employed an Artificial Neural Network model as well as a modified architecture known as Convolutional Long-Short Term Memory (ConvLSTM). The technique of Next-Frame Prediction, which utilizes the dense architecture of either Convoluted Neural Network (CNN) layers or Artificial Neural Network (ANN) layers, has been used to implement Deep Learning Models. The findings demonstrate the neural network model's ability to predict wind speed in the Indian Ocean region as 2D images for up to 6, 12, and 24 hours into the future. The study suggests that neural network models can be a valuable tool for predicting wind speed in the Indian Ocean region.

Acknowledgements

We wish to express our sincere thanks and deep sense of gratitude to our project guide, Dr.Saurav Gupta, Professor, School of Electronics Engineering, for her consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to Dr. Susan Elias, Dean Dr. Reena Monica, Associate Dean (Academics) & Dr. John Sahaya Rani Alex, Associate Dean (Research) of the School of Electronics Engineering, VIT Chennai, for extending the facilities of the School towards our project and for his unstinting support.

We express our thanks to our Head of the Department Dr. Annis Fathima A for her support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

Contents

List of Figures

Chapter 1

Introduction

The region encompassing the Indian Ocean is known to be highly susceptible to tropical cyclones and extreme weather phenomena, making accurate Wind Speed Prediction crucial for disaster management and maritime operations.

Traditional methods for Wind Speed Prediction rely on numerical weather prediction models, which are computationally expensive and require high-resolution input data. The development of remote sensing and satellite data has led to the emergence of machine learning algorithms as an effective tool for Wind Speed Prediction.

Wind Speed Prediction (WSP) is a crucial task due to its impact on various activities such as renewable energy generation, weather forecasting, and air traffic control on land and fishing, transportation, and offshore operations in oceanic regions.

WSP has been achieved over the years using a range of techniques, from simpler ones such as Autoregressive integrated moving average (ARIMA)[1] and Support vector machines (SVM) [2] to more advanced machine learning techniques [3][4].

Optimization algorithms such as Whale Optimization (WOA) [5], Grey Wolf (GWA) [6], Fireworks (FWA) [7] and AdaBoost [8] have been utilized to improve WSP models.

Novel Neural Network models such as Shared Hidden Layer- Deep Neural Network (SHL-DNN) [9], nonlinear autoregressive network with exogenous inputs (NARX), Radial basis function (RBF) and Back Propagation (BPN) [10] have also been developed for WSP.

A different research effort [11] suggested a two-dimensional framework for forecasting the wind speed of a specific location in the near future, similar to the one proposed in

this study.

Machine Learning and Neural Networks have been considered as well as favoured for the prediction of other weather parameters like Pressure, Temperature [12][13], Dew Point, Humidity [14], visibility and Precipitation [15].

This paper focuses on WSP which is an important factor for predicting tropical cyclones. Studies have been conducted where machine learning models have been used to efficiently predict other natural disasters like floods [16], typhoons [17], hurricanes [18] and tornadoes [19]. Intensity Prediction and tracking of Tropical cyclones has been made a possibility due to models like [20],[21],[22] and [23].

The novelty of this research paper is the creation of a two-dimensional model that employs two machine learning approaches to forecast wind speed across the extensive Indian Ocean region, enabling accurate WSPs for almost 40,000 locations simultaneously. This proposed method can be implemented in real-time scenarios, such as disaster management and maritime operations. The effectiveness of the proposed neural network model is assessed using metrics such as mean squared error (MSE) and mean absolute error (MAE).

Chapter 2

Literature Survey

2.1 Wind Speed Prediction Based on Seasonal ARIMA model

This research paper uses Seasonal Autoregressive Integrated Moving Average (SARIMA) model for predicting wind speed for the next 1 hour for a wind farm in Morocco with a collection interval of 10 minutes. Dataset is 1-dimesional and of the year 2018 and predicted wind speed for the months of March, July and October. This study provided a good understanding for using simpler algorithms like ARIMA for wind speed prediction.

2.2 Short-Time Wind Speed Forecast Using Artificial Learning-Based Algorithms

This research paper provided a basic understanding of the use of different Machine Learning and artificial intelligence algorithms in the field of wind speed prediction. The paper compares CNN, ANN, LSTM, SVR and ConvLSTM models for wind speed prediction. The dataset used is of Lake Alan Henry, Texas, USA. It is based on a single point with an interval of 5 minutes and is averaged to 30 minutes and 1 hour for the prediction. The study proved that ConvLSTM gives the least error out of all 5 chosen algorithms for prediction of next 5, 30 and 60 minutes.

2.3 Wind speed prediction using a hybrid model of the multi-layer perceptron and whale optimization algorithm

This research paper compares the performance of Multilayer Perceptron (MLP), MLP optimized by Whale optimization Algorithm (WOA), and MLP optimized by Genetic Algorithm (GA) to predict wind speed for a scattered 10-point dataset based in Iran. It concludes that MLP with WOA has superior performance as compared to the other two algorithms for predicting daily wind speeds. The paper also points out importance of wind speed prediction for calculating wind energy potential.

2.4 An artificial neural network optimized by grey wolf optimizer for prediction of hourly wind speed in Tamil Nadu, India

This research paper compares the performance of multiple state of the art metaheuristic algorithms like: Feed-Forward Multilayer Perceptron (FF-MLP) with Artificial Bee Colony (ABC), Ant Colony Optimization (ACO), Biogeography Based Optimization (BBO), Evolutionary Strategy (ES), Genetic Algorithm (GA), Grey-Wolf-Optimizer (GWO), Population-Based Incremental Learning (PBIL), Particle Swarm Optimization (PSO), Tree-Seed Algorithm (TSA) to predict wind speed for 5 cities in Tamil Nadu. Dataset is 1-dimesional and of the years 1980-2018. The GWO successfully manages to predict wind speed for the next 8 years as 29 years of data is used for training.

2.5 Wind speed prediction using measurements from neighboring locations and combining the extreme learning machine and the AdaBoost algorithm

This research paper utilized Extreme Learning Method (ELM) paired with AdaBoost algorithm to predict wind speed for a scattered 17-point dataset from a geographical region in China, with a collection interval of 10 minutes. Comparison between single point and multi point ELM paired with and without AdaBoost is done in this study, and Multiple-point ELM with Adaboost outperforms its single point counterparts in predicting wind speed for the next 10, 20 and 30 minutes.

2.6 Transfer learning for short-term wind speed prediction with deep neural networks

This research paper provides an insight in the use of Deep Neural Networks in the field of wind speed prediction. 4 models namely: Deep Neural Network (DNN), Support Vector Regression (SVR), Extreme Learning Method (ELM) and Shared Hidden Layer (SHL)- DNN. Out of which SHL-DNN performs best for predicting wind speed for next 10 minutes, 20 minutes, 1 hour and 2 hours. The data set is based on 4 scattered wind farms in China with a collection interval of 10 minutes.

2.7 Improved Prediction of Wind Speed using Machine Learning

This research paper uses different types of Artificial Neural Network (ANN) models like Nonlinear Autoregressive Network with Exogenous inputs (NARX), Back-Propagation Network (BPN) and Radial Basis Function (RBF) Network with the combination of Mutual Information Feature Selection (MIFS) to predict wind speed for data gathered from the University of Waterloo weather station between 2013 and 2015 at intervals of 15 minutes. Results show that NARX paired with MIFS outperforms the other models and networks used in predicting wind speed for the next 256 minutes (4 hours and 16 minutes).

2.8 2-D regional short-term wind speed forecast based on CNN-LSTM deep learning model

This research paper was helpful in understanding the use of Machine Learning to predict wind speed in a 2-Dimensional dataset. An amalgamation of Convoluted Neural Network (CNN) and Long Short-Term Memory (LSTM) was used to predict wind speed for an area of 2 km x 2 km made into a 10 x 10 array. The data was collected from satellite over Indiana, USA. The proposed model was compared with ANN, LSTM and Persistence models out which it emerged to be the best for predicting wide-area wind speed for the next 2 hours.

Chapter 3

Methodology

3.1 Models Used

3.1.1 Convolutional LSTM (ConvLSTM)

ConvLSTM is a type of LSTM that incorporates convolutional operators to process input data and state-to-state transitions rather than matrix multiplication [16]. It is designed to handle 3-dimensional data as an input and is trained on spatial information from the dataset. Additionally, each LSTM cell's gate undergoes a convolution operation to exchange matrix multiplication. It can then integrate the identification of underlying spatial features within multidimensional data which becomes crucial for many applications in fields such as remote sensing, climate modelling, and image analysis.[3]. The equations that define the ConvLSTM architecture are given below, In which the convolution operator is denoted by $*$ and the Hadamard product is represented by $\odot$.:

$$i_t = \sigma\left(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \odot C_{t-1} + b_i\right) (1)$$

$$f_t = \sigma\left(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \odot C_{t-1} + b_f\right) (2)$$

$$C_t = f_t \odot C_t - 1 + i_t \odot \tanh\left(W_{xc} * X_t + W_{hc} * \mathcal{H}_{\sqcup -1} + b_c\right) (3)$$

$$o_t = \sigma\left(W_{xo} * X_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \odot C_t + b_o\right) (4)$$

$$\mathcal{H} = o_t \odot \tanh\left(C_t\right) (5)$$

Equations (1) to (5) define the input, forget, and output gates of each timestamp in the ConvLSTM architecture. The variables used in these equations are i, f, and o for the

gates, $\mathcal{H}$ for the hidden state, C for the cell output, and X for the inputs. W stands for weighted connections between the states, and represents the activation.[18]

3.1.2 Artificial Neural Network (ANN)

Artificial neural networks (ANNs) are a machine learning model that mimics the functioning of biological neurons in the human brain [15]. ANNs are composed of interconnected nodes or neurons organized in layers that can learn to identify patterns in data through a process known as training.
The basic mathematical formula for a single neuron in an ANN is:

$$y_i = f\left(\sum_{i=0}^{n}(w_i x_i) + b\right) (6)$$

In which yi represents the neuron's output, f is the activation function, wi denotes the input weight, xi corresponds to the input value, and b indicates the bias. There are various types of ANNs, including feedforward (FFNN), convolutional (CNN)[1], and recurrent neural networks (RNNs). RNNs, in particular, are capable of processing sequential data by incorporating feedback loops that allow information to persist over time. The Long Short-Term Memory (LSTM) network is a popular type of Recurrent Neural Network (RNN). LSTM was designed to overcome the limitations of RNN in processing long-term memory [7]. The mathematical equations for an LSTM network are more complex than those for a standard RNN, but the basic structure involves input, forget, and output gates which are responsible for controlling the flow of information in and out of the memory cells in an LSTM network. Artificial neural networks (ANNs) have been proven effective in multiple fields, including predictive modeling, natural language processing, and image and speech recognition.

3.1.3 Kernel Initializer – he_uniform

The "he_uniform" kernel initializer is a type of weight initialization method used in deep learning neural networks. It was proposed by [26], in their 2015 paper. The he_uniform initializer initializes weights in a way that takes into account the number of input units to a layer, which is important because the scale of the weights can affect the performance of the network. The he_uniform initializer, in particular, selects values from a uniform distribution that spans a range dependent on the quantity of input units present in the layer. The initialization of weights in a suitable manner for each layer can aid in the optimization process during training, which is important for achieving good performance of the neural network. Overall, the he_uniform initializer has been shown to be effective

in improving the training and performance of deep neural networks, particularly in tasks related to computer vision.

3.1.4 Why Relu?

The Rectified Linear Unit (ReLU) is a preferred activation function in deep learning. It is a simple non-linear function that replaces any negative input with zero, while passing through any positive input. ReLU activation function can be mathematically defined as shown in Formula (7):

$$f(x) = max(0, x) \quad (7)$$

he ReLU activation function has several advantages over other activation functions, including simplicity, computational efficiency, and the ability to alleviate the Rectified Linear Unit (ReLU) activation function is effective in addressing the issue of vanishing gradients during backpropagation and is also observed to perform well in practical applications, leading to improved accuracy and faster training times in many deep learning applications. Graphical representation of ReLu is shown in Figure 3.1.

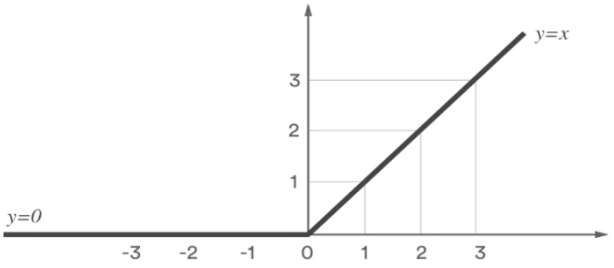


FIGURE 3.1: Graphical Representation of ReLu activation function

3.2 Data Set

Data has been collected from Earth Nullschool [24] website obtains its weather data from the Global Forecast System (GFS) of the National Oceanic and Atmospheric Administration (NOAA) and Ocean Currents data from the Ocean Surface Current Analyses Real-time (OSCAR) project of Earth and Space Research (ESR), which is supported by NASA. The website offers a variety of data, including but not limited to wind speed and its direction, current speed and its direction, humidity, temperature and more for historical weather data. A data mining algorithm had been implemented to collect wind speed data from the website from 1st January 2015 to 31st October 2015 after every 6

hours of interval. For Eg. 0000Z, 0600Z, 1200Z, 1800Z of 1st January 2015 and so on for all the other dates. Z is called Zulu time or more commonly referred to as UTC, Coordinated Universal Time. Therefore, each day has 4 time slots and there should be a total of 1216 time slots of data available. As shown in figure 3.2, data had been scraped over the zone of 30°N, 60°E to 10°S, 100°E with jumps of 0.20 degree. 201x202 are the number of wind speed values for the longitudes and latitudes and these are stacked for 8 consecutive time slots. For Eg. 0000Z, 0600Z, 1200Z, 1800Z of 1st January 2015 and 0000Z, 0600Z, 1200Z, 1800Z of 2nd January 2015 are stacked consecutively and a prediction is made for 0000Z of 3rd January in the 6 hours forward prediction model whereas a prediction 0600 of 3rd January is made in the 12 hours forward prediction model and 0000 of 4th February's prediction is made in 24 hours forward model.

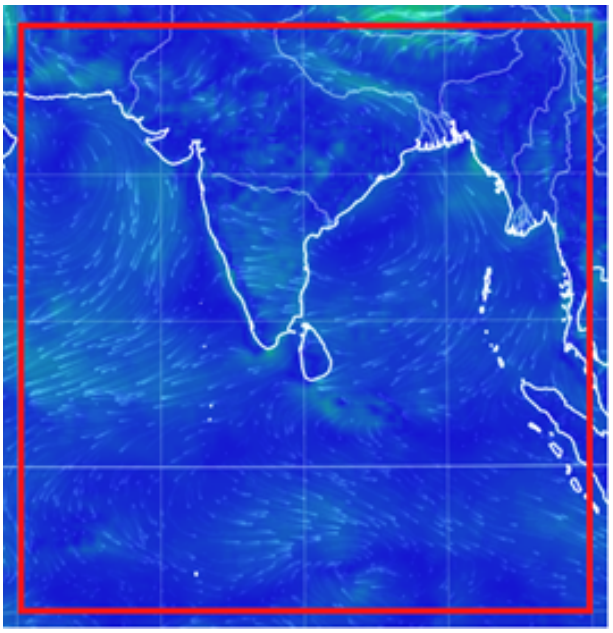


FIGURE 3.2: Region of Interest (Boxed in Red)

3.3 Data Exploration

There were two types of missing data in the dataset. One, was missing time slots summing up to 3. There were only 1213 time slots out of 1216, which is negligible missing data, and the other was missing wind speed values in the time slots, which on inspection were also negligible compared to the entire 201x202 matrix. Missing data was replaced by the values of their nearest neighbours. The collected wind speed data unit is in km/hr which was converted into m/s and then normalized by dividing it by 100 making all the values below 1 as ML models work better in the range of 0 to 1.

The input data structure used for the ConvLSTM-based model had dimensions of 201 x 202 x 8 x 1 per 3D. tensor as depicted in figure 3.3. 201 x 202 are the number of wind speed values for the longitudes and latitudes and these are stacked for 8 consecutive time slots. For Eg. 0000Z, 0600Z, 1200Z, 1800Z of 1st January 2015 and 0000Z, 0600Z, 1200Z, 1800Z of 2nd January 2015 are stacked consecutively and a prediction is made for 0000Z of 3rd January in the 6 hours forward prediction model whereas a prediction 0600 of 3rd January is made in the 12 hours forward prediction model and 0000 of 4th February's prediction is made in 24 hours forward model.

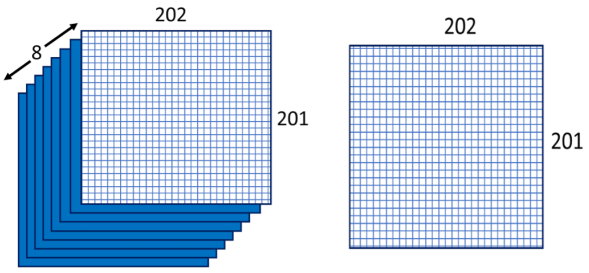


FIGURE 3.3: Input (left) and Output (right) shape of Conv-LSTM based model

The data structure fed into the ANN model has 8 columns with the data of 8 consecutive time slots t1, t2, t3, t4, t5, t6, t7, t8. As shown in figure 3.4. For eg. 0000Z, 0600Z, 1200Z, 1800Z of 1st January 2015 and 0000Z, 0600Z, 1200Z, 1800Z of 2nd January 2015 are stacked consecutively in t1, t2, t3, t4, t5, t6, t7, t8 and a prediction is made for the t9 time slot which is 0000Z of 3rd January in the 6 hour forward prediction model; t1, t2, t3, t4, t5, t6, t7, t8 will also have wind speed values for 0600Z, 1200Z, 1800Z of 1st January 2015, 0000Z, 0600Z, 1200Z, 1800Z of 2nd January 2015 and 0000Z of 3rd January respectively and a prediction is made for 0600Z of 3rd January and so on. The 2D structure is converted into 1D therefore 201 X 202 shape is converted into 40,602 size 1D shape.

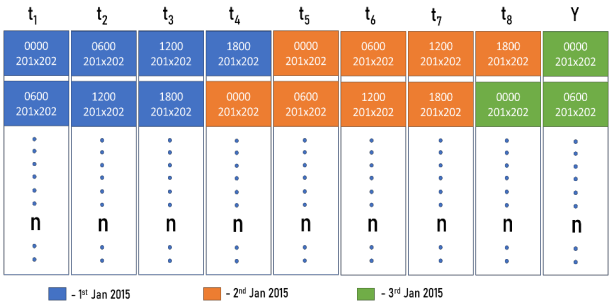


FIGURE 3.4: Input (t1, t2, t3, t4, t5, t6, t7, t8) and Output (Y or t9) shape of ANN model

3.4 Proposed Models

3.4.1 ConvLSTM based Model

The proposed ConvLSTM - based model has an Input Layer that takes in a tensor shape of 201 x 202 x 8 x 1 as shown in Figure 3.5 201 x 202 are the wind speed values for the respective latitudes and longitudes for one time slot. 8 means that 8 subsequent time slots are stacked together. A Permute Layer is then used to reshape the tensor before adding some gaussian noise using a Gaussian Noise Layer. Then the output from the Gaussian Layer is used; once to slice out the last time slot's wind speed values using a Lambda Layer which will later be used directly in the final Add Layer. This technique is used in next frame prediction techniques where the current image is used as a palette for the next frame's prediction [25].

The same output from the Gaussian Layer is again reshaped via a Permute layer to make it suitable to be used as an input for two parallel ConvLSTM layers with 6 filters each and their kernel size is 3x3 and 9x9 respectively. The output from these layers is fed into their own separate Conv2D layers as an input and both these layers have only 1 filter and their kernel size is 3x3. The output from these layers is of the shape (201 x 202 x 1) which is equivalent to a grayscale image.

The outputs from these two Conv2D layers is added to the output from the Lambda Layer that had the latest time-slot's wind speed using the Add Layer. This is the final output. The activation function for the ConvLSTM Layers and the Conv2D layers is 'ReLu' and all these layers were initialized with the 'he uniform' kernel initializer. During model training, the loss function employed is the mean square error and the optimizer used is "Adam.". The final output is of the shape 201 x 202 x 1. This output is then processed using the np.squeeze function that removes the axes of length 1 from a given numpy array. Therefore, the final array shape becomes 201 x 202.

As seen in Table 1 (Figure 3.6), The ConvLSTM based model has the same number of total parameters as trainable parameters., that are weights which is 15278, the ConvLSTM2D layer that has a kernel shape of 3X3 and has 1536 trainable parameters, the ConvLSTM2D layer that has a kernel shape of 9X9 has 13632 trainable parameters. Their respective Conv2D layers with 3X3 kernel shape have 55 trainable parameters respectively.

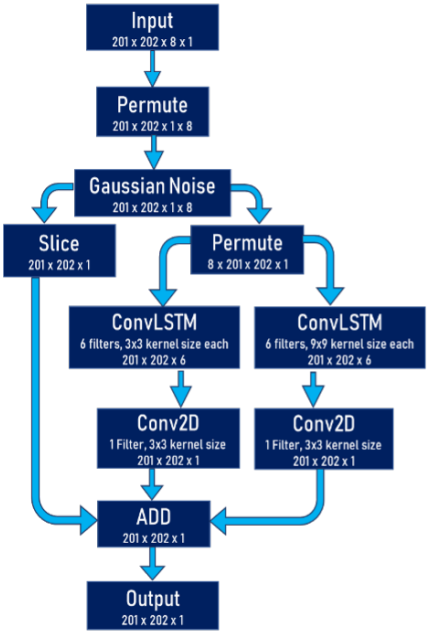


FIGURE 3.5: Conv-LSTM based model

Layer (Type)	**Output shape**	**Parameters**	**Connected To**
input_1 (InputLayer)	[(None, 201, 202, 8, 1)]	0	[]
permute (Permute)	(None, 201, 202, 1, 8)	0	['input_1[0][0]']
gaussian_noise (GaussianNoise)	(None, 201, 202, 1, 8)	0	['permute[0][0]']
permute_1 (Permute)	(None, 8, 201, 202, 1)	0	['gaussian_noise[0][0]']
conv_lstm2d (ConvLSTM2D)	(None, 201, 202, 6)	1536	['permute_1[0][0]']
conv_lstm2d_1 (ConvLSTM2D)	(None, 201, 202, 6)	13632	['permute_1[0][0]']
lambda (Lambda)	(None, 201, 202, 1)	0	['gaussian_noise[0][0]']
conv2d (Conv2D)	(None, 201, 202, 1)	55	['conv_lstm2d[0][0]']
conv2d_1 (Conv2D)	(None, 201, 202, 1)	55	['conv_lstm2d_1[0][0]']
add (Add)	(None, 201, 202, 1)	0	['lambda[0][0]', 'conv2d[0][0]', 'conv2d_1[0][0]']
Total params: 15,278			
Trainable params: 15,278			
Non-trainable params: 0			

FIGURE 3.6: Table 1.Conv-LSTM based model parameters

3.4.2 ANN based Model

The other model used was an ANN model with only two layers. The first Dense Layer contains 15 filters, and its output is transmitted to the second Dense Layer, which has a single filter. The input of the first Dense Layer takes in 8 columns that are 8 consecutive time slots as shown in Figure 3.7. This model can be almost considered as a logistic regression model with multi variable inputs (8 time slots values to provide the 9th slot's output). While logistic regression is typically employed for binary classification tasks, in this case, the model's output is utilized to predict wind speed values for the 9th time slot. The activation function 'relu' is employed in the dense layers, while the loss function used for model training is 'mean square error', and the optimizer is 'adam'. The dataset had 201 x 202 latitudes and longitudes for one time slot. This model does not take into consideration the neighboring wind speed values but is used to predict the 9th time slot's output for the given latitude and longitude using its previous 8 time slot values. Therefore, the 2D (201 x 202) matrix was reshaped into 1D, which translates to the size of 40,602 for one time slot value. The next 8 subsequent time slots data were used in the neighboring 8 columns.

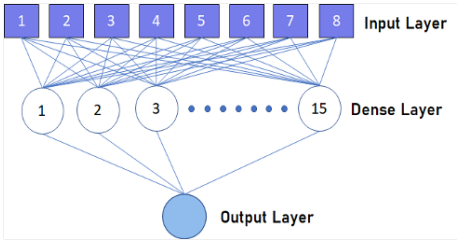


FIGURE 3.7: ANN model

Layer (Type)	Output Shape	Param #
dense (Dense)	(None, 15)	135
dense_1 (Dense)	(None, 1)	16
Total params: 151		
Trainable params: 151		
Non-trainable params: 0		

FIGURE 3.8: Table 2. ANN model parameters

The ANN model has very few trainable parameters compared to that of ConvLSTM based model and has a total of 151 parameters which is also equal to the number of trainable parameters as shown in Table 2 (Figure 3.8). The first dense layer with 15 neurons has 135 parameters and the final output dense layer with only one neuron has 16 trainable parameters.

3.5 Model Evaluation

Mean Square Error (MSE) and Mean Absolute Error (MAE) are frequently used loss functions in machine learning for regression tasks that compute the difference between actual and predicted wind speed values, it is mathematically defined as shown in formula (1) and (9).

$$MSE = \frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y_i})^2$$

$$MAE = \frac{1}{n}\sum_{i=0}^{n}|y_i - \hat{y_i}|$$

One can calculate the deviation between predicted and actual wind speed values at a given time step i using the formulas (6) and (7), where yi represents the actual wind speed value, y'i represents the predicted wind speed value, and number of samples is given as n [4] [13]. In the context of these models, accuracy, precision, recall, and F1 score are not relevant as they are evaluation parameters commonly used for classification models, whereas the models in question are numeric value prediction models. In these models, which involve predicting numeric values, accuracy, precision, recall, and F1 score are not used as model evaluation parameters. In classification models, the output layer has that many number of dense filters/neurons as there are classes or categories in the dataset and each neuron in the output layer predicts a value between 0 to 1 that determines if the given value belongs to that class or not. In prediction models, generally there is one neuron in the output layer that predicts a numeric value based on the input parameters. The model evaluation parameters are Mean Square Error and Mean Absolute Error, since the proposed model predicts a numeric wind speed value.

3.6 Model Training

Table 3 (Figure 3.9) denotes the graphs of training losses and validation losses of the ANN and ConvLSTM based architectures for 6 hours, 12 hours and 24 hours forward predictions respectively. As the dataset was large and would crash the RAM if loaded at once. The training was done in 4 batches with 33% of the data in each batch taken as Validation data and the data has not been shuffled as this is time series data and the validation data is taken from the end for both the ANN and ConvLSTM based model.

The abrupt variations observed in the graph of the ANN models can be attributed to the use of a fresh batch of data for model training, as well as the conversion of the 2D data to 1D. Therefore, the possibility of same data increases due to the presence of

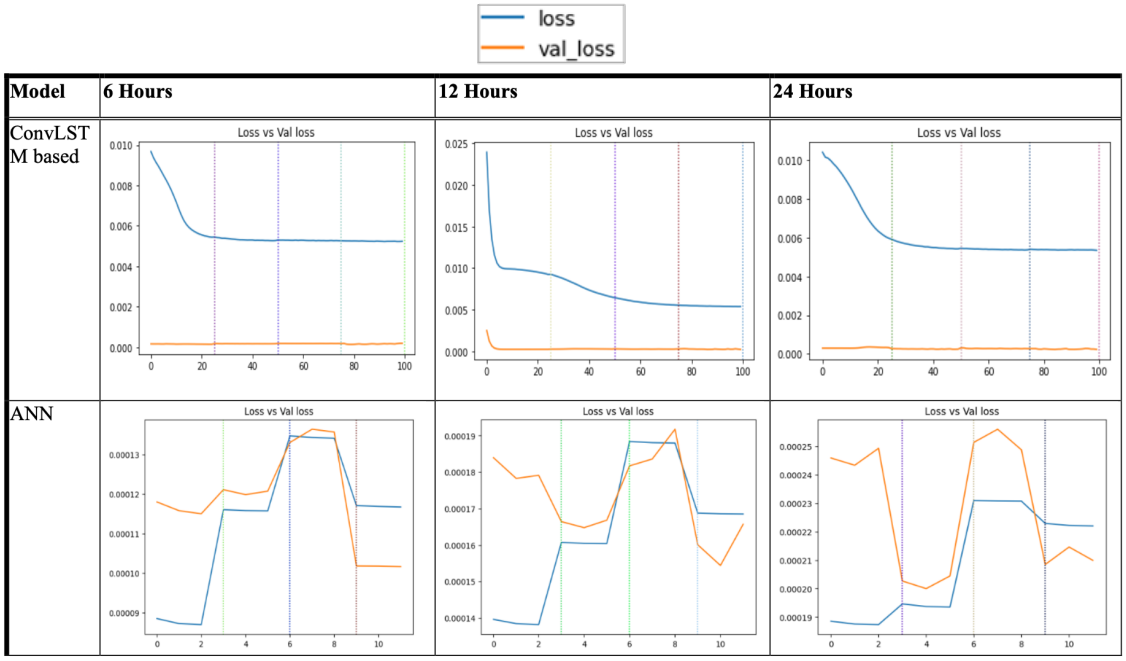


FIGURE 3.9: Table 3. Training Loss of ConvLSTM and ANN models

same values for long period of times in the neighbouring cells. For each batch of data, the ANN model was trained for 3 epochs. Thus, the model was trained for a total of 12 epochs, covering all the four batches. In the final batch, validation loss drops below the training loss. Though the loss range of both the validation and training data is low that suggests the ANN model is robust. Meanwhile, the ConvLSTM based model was trained for 25 epochs for each batch of data. It was trained for a total of 100 epochs over all the 4 batches. The table shows that the training loss steadily decreases, and the validation loss remains low. It is because this model is dependent on the neighbouring values and can differentiate between copy values with minor changes in the times slots across different coordinates unlike the ANN. The ANN model cannot differentiate between longitudes and latitudes and the wind speed coordinate is irrelevant for the ANN.

From Table 4 (Figure 3.10), it is implied that the total data is split into 80:20 train - test split for all the models. The table gives us details about the batch size, batch number, total training data and test data for both the ANN and ConvLSTM based models.

Table 5 (Figure 3.11). shows the Minimum and Maximum Validation and Training Loss over all the batches for the models during training. The loss (MSE) is normalized and to get it in m/s it is multiplied by 100. The model with the best loss parameter is 6 hours forward ANN model. And the ANN model in general seems to have better training and validation parameters for its equivalent ConvLSTM model.

Model	Time	Batch Shape	Number of Batches	Total Training Data	Testing Data
ANN Model	6 hours forward	(9744480, 8) 240 slots	4	3,89,77,920 (960 slots)	97,85,082 (241 slots)
	12 hours forward	(9703878, 8) 239 slots	4	3,88,15,512 (956 slots)	98,66,286 (243 slots)
	24 hours forward	(9622674, 8) 237 slots	4	3,84,90,696 (948 slots)	1,00,28,694 (247 slots)
ConvLSTM based Model	6 hours forward	(240, 201, 202, 8, 1) 240 slots	4	960 slots	240 slots
	12 hours forward	(239, 201, 202, 8, 1) 239 slots	4	956 slots	252 slots
	24 hours forward	(237, 201, 202, 8, 1) 237 slots	4	948 slots	258 slots

FIGURE 3.10: Table 4. Train-Test details

Model	Time	Min Validation Loss	Min Training Loss	Max Validation Loss	Max Training Loss
ANN Model	6 hours forward	0.0001	8.70E-05	0.00014	0.00013
	12 hours forward	0.00015	0.000138	0.00019	0.00019
	24 hours forward	0.0002	0.000187	0.00026	0.00023
ConvLSTM based Model	6 hours forward	0.00014	0.005224	0.00019	0.00968
	12 hours forward	0.00024	0.005387	0.00251	0.02393
	24 hours forward	0.00023	0.005349	0.00036	0.01041

FIGURE 3.11: Table 5. Validation and Training Loss details

Chapter 4

Results and Discussions

4.1 Results and Comparison

From Tables. 6 (Figure 4.1), 7 (Figure 4.2) and 8 (Figure 4.3), It is seen that the ConvLSTM model for all three 6,12 and 24 hours predict the exact same Max and Min values as that of the actual/real values whereas the ANN models do not. The ANN model with the closest Max Predicted Value to the Real Value is the 6 hours forward ANN prediction model. It is also seen that the farther the ANN models try to predict, the lower their Max Predicted values becomes. The Max Difference column shows the max difference between predicted and actual values. The Max Actual, Max Predicted, Min Actual, Min Predicted, Max Difference, MSE and MAE are calculated across all the wind speed values in all latitudes and longitudes in all time slots across all the days in the test dataset for both the ANN and ConvLSTM based models. The Mean Square Error (MSE) and Mean Absolute Error (MAE) values were measured in m/s. The 6 hours forward ANN model had the lowest MSE and MAE values, indicating its superior performance compared to the ConvLSTM based models. To visualize the 1D output from the ANN it was again reshaped into 2D. Wind Speed predictions done by other

Model (6 Hours)	Max Actual	Max Predicted	Min Actual	Min Predicted	Max Difference	MSE	MAE
ConvLSTM based	54.7222	54.7222	0.2777	0.2777	33.6111	1.9642	0.7639
ANN	54.7222	41.8055	0.2777	0	22.6389	1.0736	0.0371

FIGURE 4.1: Table 6 : 6 hours forward ConvLSTM based Model and ANN model

various studies shown in table 9 (Figure 4.4). are usually based on 1- Dimensional data that usually consists of a single point or a few scattered points and is collected over a selected few coordinates and altitude. 2-D models which have been developed until

Model (12 Hours)	Max Actual	Max Predicted	Min Actual	Min Predicted	Max Difference	MSE	MAE
ConvLSTM based	54.7222	54.7222	0.2777	0.2777	39.7222	2.6321	0.8245
ANN	54.7222	39.9328	0.2777	0	26.8773	1.7319	0.2911

FIGURE 4.2: Table 7 : 12 hours forward ConvLSTM based Model and ANN model

Model (24 hours)	Max Actual	Max Predicted	Min Actual	Min Predicted	Max Difference	MSE	MAE
ConvLSTM based	54.7222	54.7222	0.2777	0.2777	46.3888	2.5659	0.4500
ANN	54.7222	20.4119	0.2777	0.3110	45.2405	2.2466	0.1833

FIGURE 4.3: Table 8 : 24 hours forward ConvLSTM based Model and ANN model

Name	Data Type	Prediction Time	Model Used	Accuracy Metrics (m/s)
Tyass et al.[1]	1D	1 hour	SARIMA	MAE: 0.75
Abrahim et al.[3]	1D	5min, 30min, 1 hour	ConvLSTM	MAE:0.177 (mph)
Samadianfard et al.[5]	1D	24 hours	MLP-WOA	RMSE: 0.703
Cinar et al.[6]	1D	8 years	ANN optimized by GWO	MSE: 0.0046
Wang et al.[8]	1D	10min, 20min, 30mins	multiple point ELM with AdaBoost	MAE: 0.2474, 0.3060, 0.3317
Q. Hu et al.[9]	1D	10-min,30min,1hr,2hr	SHL-DNN	MAE: 04679, 0.8057, 0.1364, 1.6406
Kumar P [10]	1D	256 mins (4hr 16min)	(BPN, RBF, NARX)-with MIFS	MAE: 0.6437, 0.5732, 0.4381
Chen et al.[11]	2D	2 hours	CNN, LSTM	MAE: 0.164

FIGURE 4.4: Table 9 : 24 hours forward ConvLSTM based Model and ANN model

now either have a small coverage area or a small prediction time. The proposed models manage to predict wind for an area of more than 10 million sq. kms for 6,12, and 24 hours

In Figures 4.5 to 4.10 it can be seen that the predicted output from both the models gives decent wind speed outputs that are close to the real values. Both models exhibit low MSE and MAE, indicating their robustness and suitability for estimating wind speed in a given area.

4.1.1 Case Study: 2015 CYCLONE CHAPALA

Many studies and developments in the past have shown that wind speed can be used as a primary parameter for predicting cyclones [20]. The proposed models were also

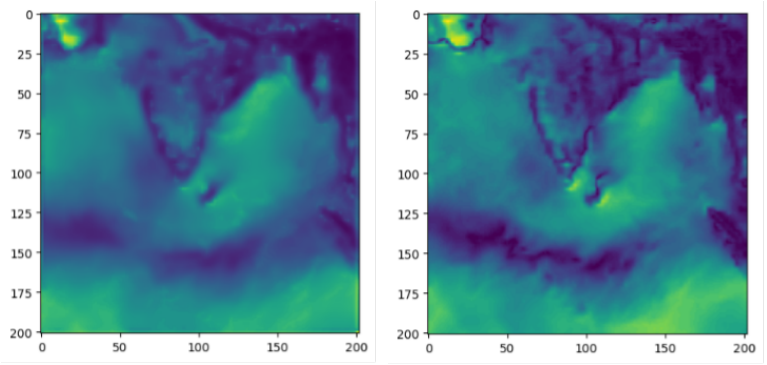


FIGURE 4.5: 6 hours forward ConvLSTM based Model Predicted (left) and Actual (right) Wind Speeds Visualisation

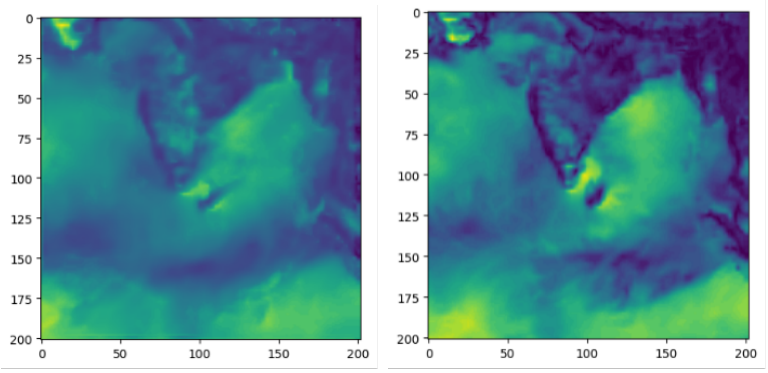


FIGURE 4.6: 12 hours forward ConvLSTM based Model Predicted (left) and Actual (right) Wind Speeds Visualisation

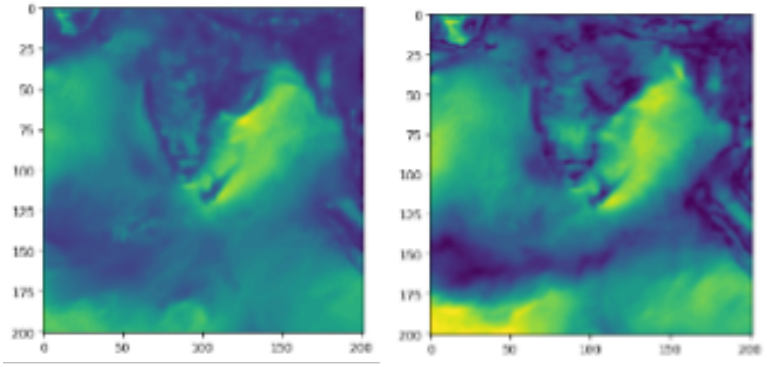


FIGURE 4.7: 24 hours forward ConvLSTM based Model Predicted (left) and Actual (right) Wind Speeds Visualisation

tested to predict the extreme cyclonic storm Cyclone Chapala, which was a powerful tropical cyclone and caused landfall on the Arabian Peninsula, leading to significant damage and displacement of people in late October 2015. It was one of the strongest tropical cyclones ever recorded in the Arabian Sea, and its effects were felt in Yemen, Somalia and Oman. [27]. Tables 10 to 12 (Figures 4.11 to 4.13) shows the prediction of all 6 models during the event. The proposed ConvLSTM-based and ANN models helped in predicting the formation of Cyclone Chapala. The model has the potential to

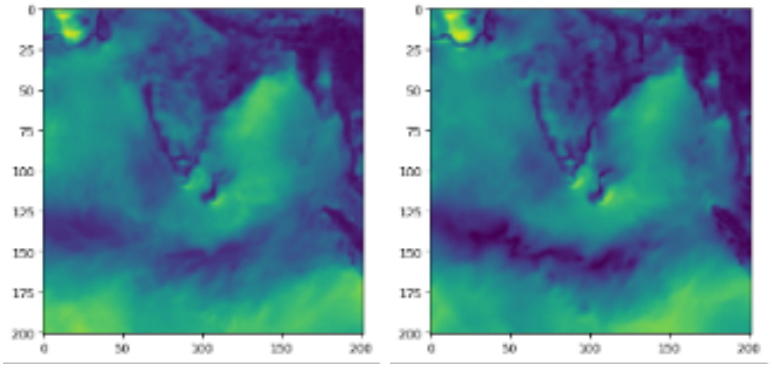


FIGURE 4.8: 6 hours forward ANN Model Predicted (left) and Actual (right) Wind Speeds Visualisation

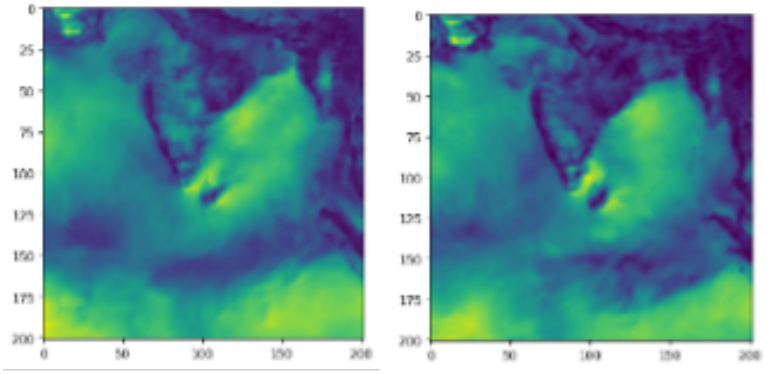


FIGURE 4.9: 12 hours forward ANN Model Predicted (left) and Actual (right) Wind Speeds Visualisation

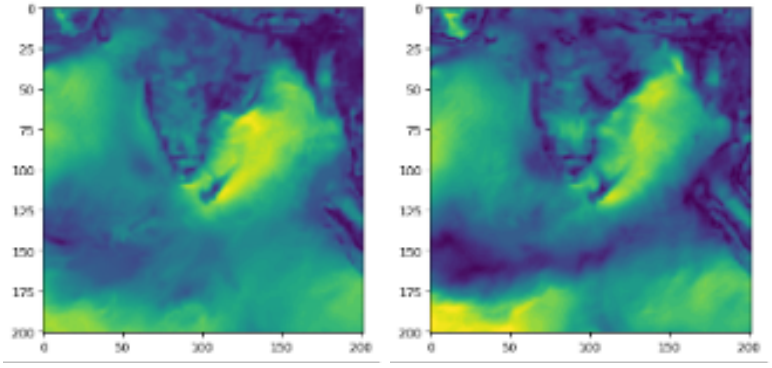


FIGURE 4.10: 24 hours forward ANN Model Predicted (left) and Actual (right) Wind Speeds Visualisation

be enhanced to forecast cyclones in the Indian Ocean, Arabian Sea, and Bay of Bengal. This improvement could lead to better disaster response efforts, and ultimately save lives. By accurately predicting cyclones, officials can take preventive steps to evacuate at-risk populations and secure infrastructure, reducing the impact of such storms in the future. Both the models help in properly estimating the cyclone paths and wind speed for the region, even though there is some noise in the visualization. The best forecast

was produced by the 6 hours forward ANN model followed by their ConvLSTM based counterpart.

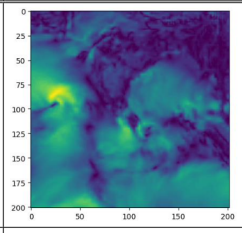
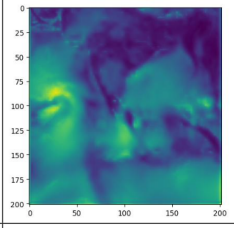
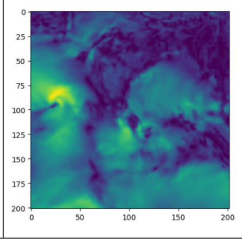
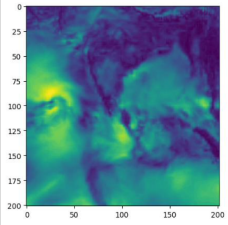
Model	Actual 6 hours	Predicted 6 hours
ConvLSTM based		
ANN		

FIGURE 4.11: Table 9 : 6 hours forward actual and predicted wind speed visualization for Cyclone Chapala

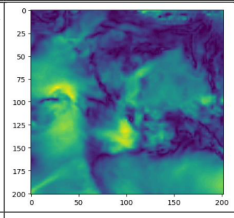
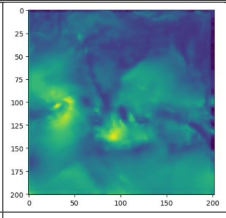
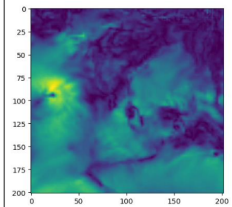
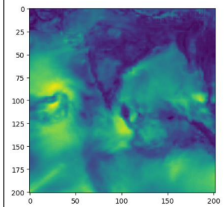
Model	Actual 12 hours	Predicted 12 hours
ConvLSTM based		
ANN		

FIGURE 4.12: Table 10 : 12 hours forward actual and predicted wind speed visualization for Cyclone Chapala

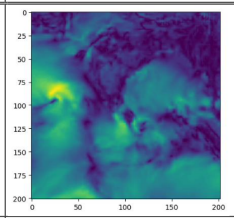
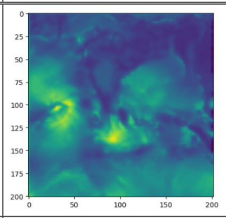
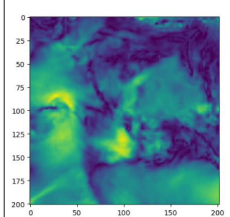
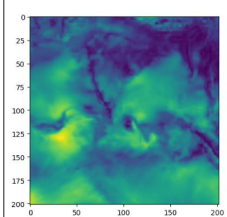
Model	Actual 24 hours	Predicted 24 hours
ConvLSTM based		
ANN		

FIGURE 4.13: Table 11 : 24 hours forward actual and predicted wind speed visualization for Cyclone Chapala

Chapter 5

Conclusion and Future Scope

The study employed two distinct models, namely the ConvLSTM-based model and an ANN model, to forecast wind speed in the Indian Ocean region at 6, 12, and 24-hour intervals. Both the models take 8 consecutive time slots as an input for 6, 12 and 24 hours forward WSP. ConvLSTM models are used in next frame prediction and this technique also could help in predicting wind speeds. The ANN model was found to have an edge over ConvLSTM based model as the visualizations were clearer in the ANN based forecasting model than ConvLSTM model. ConvLSTM based models might work better if the time slot dimensions were increased but are limited to the current hardware advancements. MSE and MAE values calculated in the testing phase were in m/s and best prediction was given by 6 hours forward ANN model. The models were put to practical use by predicting wind speed values of a real-life cyclone.

Chapter 6

Appendix

Code for ANN Model

```
"!wget "https://www.thedaytimestories.ml/capstone.zip"
!nvidia-smi
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import os
import csv

import zipfile
import os
zip_ref = zipfile.ZipFile('/content/capstone.zip', 'r')
#Opens the zip file in read mode
zip_ref.extractall('/tmp')
#Extracts the files into the /tmp folder
zip_ref.close()

file="/tmp/"
import pathlib
data_dir=pathlib.Path(file)
data_dir
dataset_dict = {
    'wind_speed' : list(data_dir.glob('wind_speed/*')),
}
```

```
print(len(dataset_dict['wind_speed']))

dataset_dict['wind_speed'].sort()
dataset_dict['wind_speed']

import pandas as pd

def batchwise_6(start,end):
  X=[]

  # Create List of list
  for i in range(8):
    X.append([])

  y=[]
  for i in range(start,end-8):
    print(dataset_dict['wind_speed'][i:i+8])
    print(dataset_dict['wind_speed'][i+8])
    tempo=dataset_dict['wind_speed'][i:i+8]
    img=[]
    counter=0
    for j in tempo:
      speed=[]
      with open(str(j), 'r') as read_obj:
        csv_reader = csv.reader(read_obj)
        # convert string to list
        speed = list(csv_reader)
        speed=np.array(speed)
        speed=speed.astype(np.float)

        for k in range(0,201):
          for l in range(0,202):
            if(speed[k,l]<1):
              speed[k,l]=speed[k-1,l]

        speed = speed.flatten()
        X[counter].append(speed)
        counter+=1
```

```
  speed=[]
  with open(str(dataset_dict['wind_speed'][i+8]), 'r') as read_obj:
      csv_reader = csv.reader(read_obj)
      # convert string to list
      speed = list(csv_reader)
  speed=np.array(speed)
  speed=speed.astype(np.float)
  for k in range(0,201):
    for l in range(0,202):
      if(speed[k,l]<1):
        speed[k,l]=speed[k-1,l]
  speed = speed.flatten()
  y.append(speed)


X=np.array(X)
y=np.array(y)

X=X.astype(np.float)
y=y.astype(np.float)

#Normalization and converting to m/s
X=X*(5/1800)
y=y*(5/1800)

x=[]


for i in range(8):
  temp=X[i].flatten()
  x.append(temp)

X=x
X=np.array(X)
x=[]
```

```
  y=y.flatten()

  data = {'one': X[0],
          'two': X[1],
          'three': X[2],
          'four': X[3],
          'five': X[4],
          'six': X[5],
          'seven': X[6],
          'eight': X[7],
          }
  df = pd.DataFrame(data)
  X=df


  return(X,y)


def batchwise_12(start,end):
  X=[]

  # Create List of list
  for i in range(8):
    X.append([])

  y=[]
  for i in range(start,end-9):
    print(dataset_dict['wind_speed'][i:i+8])
    print(dataset_dict['wind_speed'][i+9])
    tempo=dataset_dict['wind_speed'][i:i+8]
    img=[]
    counter=0
    for j in tempo:
      speed=[]
      with open(str(j), 'r') as read_obj:
        csv_reader = csv.reader(read_obj)
        # convert string to list
        speed = list(csv_reader)
        speed=np.array(speed)
```

```
        speed=speed.astype(np.float)

        for k in range(0,201):
          for l in range(0,202):
            if(speed[k,l]<1):
              speed[k,l]=speed[k-1,l]

        speed = speed.flatten()
        X[counter].append(speed)
        counter+=1



    speed=[]
    with open(str(dataset_dict['wind_speed'][i+9]), 'r') as read_obj:
        csv_reader = csv.reader(read_obj)
        # convert string to list
        speed = list(csv_reader)
    speed=np.array(speed)
    speed=speed.astype(np.float)
    for k in range(0,201):
      for l in range(0,202):
        if(speed[k,l]<1):
          speed[k,l]=speed[k-1,l]
    speed = speed.flatten()
    y.append(speed)


X=np.array(X)
y=np.array(y)

X=X.astype(np.float)
y=y.astype(np.float)

#Normalization and converting to m/s
X=X*(5/1800)
y=y*(5/1800)

x=[]
```

```
  for i in range(8):
    temp=X[i].flatten()
    x.append(temp)

  X=x
  X=np.array(X)
  x=[]

  y=y.flatten()

  data = {'one': X[0],
          'two': X[1],
          'three': X[2],
          'four': X[3],
          'five': X[4],
          'six': X[5],
          'seven': X[6],
          'eight': X[7],
          }
  df = pd.DataFrame(data)
  X=df


  return(X,y)


def batchwise_24(start,end):
  X=[]

  # Create List of list
  for i in range(8):
    X.append([])

  y=[]
  for i in range(start,end-11):
    print(dataset_dict['wind_speed'][i:i+8])
    print(dataset_dict['wind_speed'][i+11])
```

```
    tempo=dataset_dict['wind_speed'][i:i+8]
    img=[]
    counter=0
    for j in tempo:
      speed=[]
      with open(str(j), 'r') as read_obj:
        csv_reader = csv.reader(read_obj)
        # convert string to list
        speed = list(csv_reader)
        speed=np.array(speed)
        speed=speed.astype(np.float)

        for k in range(0,201):
          for l in range(0,202):
            if(speed[k,l]<1):
              speed[k,l]=speed[k-1,l]

        speed = speed.flatten()
        X[counter].append(speed)
        counter+=1



    speed=[]
    with open(str(dataset_dict['wind_speed'][i+11]), 'r') as read_obj:
        csv_reader = csv.reader(read_obj)
        # convert string to list
        speed = list(csv_reader)
    speed=np.array(speed)
    speed=speed.astype(np.float)
    for k in range(0,201):
      for l in range(0,202):
        if(speed[k,l]<1):
          speed[k,l]=speed[k-1,l]
    speed = speed.flatten()
    y.append(speed)


X=np.array(X)
```

```
  y=np.array(y)

  X=X.astype(np.float)
  y=y.astype(np.float)

  #Normalization and converting to m/s
  X=X*(5/1800)
  y=y*(5/1800)

  x=[]


  for i in range(8):
    temp=X[i].flatten()
    x.append(temp)

  X=x
  X=np.array(X)
  x=[]

  y=y.flatten()

  data = {'one': X[0],
          'two': X[1],
          'three': X[2],
          'four': X[3],
          'five': X[4],
          'six': X[5],
          'seven': X[6],
          'eight': X[7],
          }
  df = pd.DataFrame(data)
  X=df


  return(X,y)


# 6 hours
```

```
start=[]
end=[]

i=0
j=i+248

while(j<len(dataset_dict['wind_speed'])):
  start.append(i)
  end.append(j)
  i=j-9
  j=i+248

start.append(i)
end.append(len(dataset_dict['wind_speed']))

print(start)
print(end)



# 12 hours

# start=[]
# end=[]

# i=0
# j=i+248

# while(j<len(dataset_dict['wind_speed'])):
#   start.append(i)
#   end.append(j)
#   i=j-10
#   j=i+248

# start.append(i)
# end.append(len(dataset_dict['wind_speed']))

# print(start)
# print(end)
```

```
# 24 hours

# start=[]
# end=[]

# i=0
# j=i+248

# while(j<len(dataset_dict['wind_speed'])):
#   start.append(i)
#   end.append(j)
#   i=j-12
#   j=i+248

# start.append(i)
# end.append(len(dataset_dict['wind_speed']))

# print(start)
# print(end)

X,y=batchwise_6(0,248)
X.shape
y.shape

#Model

import tensorflow as tf
from tensorflow import keras

model = keras.Sequential([
    keras.layers.Dense(15,input_shape=(8,), activation="relu"),
    keras.layers.Dense(1, activation="relu"),
])

model.compile(optimizer='adam', loss="mse")

model.summary()
```

```
model_history=[]
from sklearn.model_selection import train_test_split

for i in range(0,4):
  X,y=batchwise_6(start[i],end[i])
  X_train, X_valid, y_train, y_valid = train_test_split(X, y,
  test_size=0.33, shuffle=False)
  history=model.fit(X_train,y_train,validation_data=
  (X_valid,y_valid), epochs =3, verbose=1)
  model_history.append(history)



  #vallosscsloss
  plt.title('Loss vs Val loss')
  plt.plot(history.history['loss'], label='loss')
  plt.plot(history.history['val_loss'], label='val_loss')

  plt.legend()
  plt.show()


print(len(model_history[0].history['loss']))

loss=[]
val_loss=[]
for temp in model_history:
  for i in temp.history['loss']:
    loss.append(i)
  for i in temp.history['val_loss']:
    val_loss.append(i)

print(len(loss))
print(len(val_loss))



#vallosscsloss
plt.title('Loss vs Val loss')
plt.plot(loss, label='loss')
```

```
plt.plot(val_loss, label='val_loss')

plt.legend()
plt.show()


print("Max value element (loss) : ", max(loss))
print("Max value element (val_loss) : ", max(val_loss))

print("Min value element (loss) : ", min(loss))
print("Min value element (val_loss) : ", min(val_loss))


import random as random
#vallosscsloss
plt.title('Loss vs Val loss')
plt.plot(loss, label='loss')
plt.plot(val_loss, label='val_loss')
j=1
for i in range(25,101,25):
  color_1= random.random()
  color_2= random.random()
  color_3= random.random()
  color = (color_1, color_2, color_3)
  plt.axvline(x=i, color=color, linestyle = (0,(0.1,2)),
  dash_capstyle = 'round', label='batch'+str(j))
  j+=1

plt.legend(bbox_to_anchor=(1.05, 1.0), loc='upper left')
plt.show()

model.save('model_6.h5')
model.load_weights('model_6.h5')

#Testing
#Batch 1

X,y=batchwise_6(956,1084)
X.shape
```

```
pred = model.predict(X)

pred = np.squeeze(pred)
pred.shape

y = np.squeeze(y)
y.shape

subs=np.subtract(pred, y)
mae=subs.mean()
mse=np.square(subs).mean()
print("MAE : ",mae)
print("MSE : ",mse)

pred=pred.reshape(120,201,202)
plt.imshow(pred[0])
y=y.reshape(120,201,202)
plt.imshow(y[0])

#kmph

kmy=y*(1800/5)
kmpred=pred*(1800/5)

subs=np.subtract(kmpred, kmy)
mae=subs.mean()
mse=np.square(subs).mean()
print("MAE : ",mae)
print("MSE : ",mse)

#Removing normalisation only mps

ny=y*100
npred=pred*100

subs=np.subtract(npred, ny)
mae=subs.mean()
mse=np.square(subs).mean()
print("MAE : ",mae)
```

```
print("MSE : ",mse)

#Min Max and Difference

print("Max value element (pred) : ", np.max(pred))
print("Max value element (y) : ", np.max(y))

print("Min value element (pred) : ", np.min(pred))
print("Min value element (y) : ", np.min(y))

subs=np.subtract(pred, y)
print("Min value element (Subs) : ", np.min(subs))
print("Max value element (Subs) : ", np.max(subs))

y[0,0,0]
pred[0,0,0]

#Batch 2
X,y=batchwise_6(1084,1213)
X.shape
pred = model.predict(X)

pred = np.squeeze(pred)
pred.shape

y = np.squeeze(y)
y.shape

subs=np.subtract(pred, y)
mae=subs.mean()
mse=np.square(subs).mean()
print("MAE : ",mae)
print("MSE : ",mse)

#kmph

kmy=y*(1800/5)
kmpred=pred*(1800/5)
```

```
subs=np.subtract(kmpred, kmy)
mae=subs.mean()
mse=np.square(subs).mean()
print("MAE : ",mae)
print("MSE : ",mse)

#Removing normalisation only mps

ny=y*100
npred=pred*100

subs=np.subtract(npred, ny)
mae=subs.mean()
mse=np.square(subs).mean()
print("MAE : ",mae)
print("MSE : ",mse)

#Min Max and Difference

print("Max value element (pred) : ", np.max(pred))
print("Max value element (y) : ", np.max(y))

print("Min value element (pred) : ", np.min(pred))
print("Min value element (y) : ", np.min(y))

subs=np.subtract(pred, y)
print("Min value element (Subs) : ", np.min(subs))
print("Max value element (Subs) : ", np.max(subs))

y[0,0,0]
pred[0,0,0]



#Cyclone
pred=pred.reshape(121,201,202)
plt.imshow(pred[104])
plt.imshow(y[104])"
```

Code for ConvLSTM based Model

```
!wget "https://www.thedaytimestories.ml/capstone.zip"
!nvidia-smi

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import os
import csv

import zipfile
import os
zip_ref = zipfile.ZipFile('/content/capstone.zip', 'r')
#Opens the zip file in read mode
zip_ref.extractall('/tmp')
#Extracts the files into the /tmp folder
zip_ref.close()

file="/tmp/"
import pathlib
data_dir=pathlib.Path(file)
data_dir
dataset_dict = {
    'wind_speed' : list(data_dir.glob('wind_speed/*')),
}

print(len(dataset_dict['wind_speed']))

dataset_dict['wind_speed'].sort()
dataset_dict['wind_speed']

def batchwise_6(start,end):
  X=[]
  y=[]
  for i in range(start,end-8):
    print(dataset_dict['wind_speed'][i:i+8])
    print(dataset_dict['wind_speed'][i+8])
```

```
    tempo=dataset_dict['wind_speed'][i:i+8]
    img=[]
    for j in tempo:
      speed=[]
      with open(str(j), 'r') as read_obj:
        csv_reader = csv.reader(read_obj)
        # convert string to list
        speed = list(csv_reader)
        speed=np.array(speed)
        speed=speed.astype(np.float)

        for k in range(0,201):
          for l in range(0,202):
            if(speed[k,l]<1):
              speed[k,l]=speed[k-1,l]

      img.append(speed)
    img=np.array(img)
    img=img.astype(np.float)



    img = np.transpose(img, (1, 2, 0))
    img = np.expand_dims(img, axis=3)
    X.append(img)
    speed=[]
    with open(str(dataset_dict['wind_speed'][i+8]), 'r') as read_obj:
        csv_reader = csv.reader(read_obj)
        # convert string to list
        speed = list(csv_reader)
    speed=np.array(speed)
    speed=speed.astype(np.float)
    for k in range(0,201):
      for l in range(0,202):
        if(speed[k,l]<1):
          speed[k,l]=speed[k-1,l]
    y.append(speed)


X=np.array(X)
```

```
  y=np.array(y)

  X=X.astype(np.float)
  y=y.astype(np.float)

  #Normalization and converting to m/s
  X=X*(5/1800)
  y=y*(5/1800)

  #Expanding y dimension
  y = np.expand_dims(y, axis=3)

  return(X,y)

def batchwise_12(start,end):
  X=[]
  y=[]
  for i in range(start,end-9):
    print(dataset_dict['wind_speed'][i:i+8])
    print(dataset_dict['wind_speed'][i+9])
    tempo=dataset_dict['wind_speed'][i:i+8]
    img=[]
    for j in tempo:
      speed=[]
      with open(str(j), 'r') as read_obj:
        csv_reader = csv.reader(read_obj)
        # convert string to list
        speed = list(csv_reader)
        speed=np.array(speed)
        speed=speed.astype(np.float)

        for k in range(0,201):
          for l in range(0,202):
            if(speed[k,l]<1):
              speed[k,l]=speed[k-1,l]

      img.append(speed)
    img=np.array(img)
    img=img.astype(np.float)
```

```
    img = np.transpose(img, (1, 2, 0))
    img = np.expand_dims(img, axis=3)
    X.append(img)
    speed=[]
    with open(str(dataset_dict['wind_speed'][i+9]), 'r') as read_obj:
        csv_reader = csv.reader(read_obj)
        # convert string to list
        speed = list(csv_reader)
    speed=np.array(speed)
    speed=speed.astype(np.float)
    for k in range(0,201):
      for l in range(0,202):
        if(speed[k,l]<1):
          speed[k,l]=speed[k-1,l]
    y.append(speed)


  X=np.array(X)
  y=np.array(y)

  X=X.astype(np.float)
  y=y.astype(np.float)

  #Normalization and converting to m/s
  X=X*(5/1800)
  y=y*(5/1800)

  #Expanding y dimension
  y = np.expand_dims(y, axis=3)

  return(X,y)

def batchwise_24(start,end):
  X=[]
  y=[]
  for i in range(start,end-11):
    print(dataset_dict['wind_speed'][i:i+8])
```

```
print(dataset_dict['wind_speed'][i+11])
tempo=dataset_dict['wind_speed'][i:i+8]
img=[]
for j in tempo:
  speed=[]
  with open(str(j), 'r') as read_obj:
    csv_reader = csv.reader(read_obj)
    # convert string to list
    speed = list(csv_reader)
    speed=np.array(speed)
    speed=speed.astype(np.float)

    for k in range(0,201):
      for l in range(0,202):
        if(speed[k,l]<1):
          speed[k,l]=speed[k-1,l]

  img.append(speed)
img=np.array(img)
img=img.astype(np.float)



img = np.transpose(img, (1, 2, 0))
img = np.expand_dims(img, axis=3)
X.append(img)
speed=[]
with open(str(dataset_dict['wind_speed'][i+11]), 'r') as read_obj:
    csv_reader = csv.reader(read_obj)
    # convert string to list
    speed = list(csv_reader)
speed=np.array(speed)
speed=speed.astype(np.float)
for k in range(0,201):
  for l in range(0,202):
    if(speed[k,l]<1):
      speed[k,l]=speed[k-1,l]
y.append(speed)
```

```
  X=np.array(X)
  y=np.array(y)

  X=X.astype(np.float)
  y=y.astype(np.float)

  #Normalization and converting to m/s
  X=X*(5/1800)
  y=y*(5/1800)

  #Expanding y dimension
  y = np.expand_dims(y, axis=3)

  return(X,y)


# 6 hours

start=[]
end=[]

i=0
j=i+248

while(j<len(dataset_dict['wind_speed'])):
  start.append(i)
  end.append(j)
  i=j-9
  j=i+248

start.append(i)
end.append(len(dataset_dict['wind_speed']))

print(start)
print(end)


# 12 hours
```

```
# start=[]
# end=[]

# i=0
# j=i+248

# while(j<len(dataset_dict['wind_speed'])):
#   start.append(i)
#   end.append(j)
#   i=j-10
#   j=i+248

# start.append(i)
# end.append(len(dataset_dict['wind_speed']))

# print(start)
# print(end)



# 24 hours

# start=[]
# end=[]

# i=0
# j=i+248

# while(j<len(dataset_dict['wind_speed'])):
#   start.append(i)
#   end.append(j)
#   i=j-12
#   j=i+248

# start.append(i)
# end.append(len(dataset_dict['wind_speed']))

# print(start)
# print(end)
```

```
X,y=batchwise_6(0,248)
X.shape
y.shape

#Model

from keras.models import Sequential
from keras.callbacks import Callback
from keras.layers import Input, Conv2D, ConvLSTM2D, Permute, Conv3D,
    MaxPooling2D, Dropout, Dense, Flatten, UpSampling2D, BatchNormalization
from keras import backend as K
from keras.layers import Lambda, Reshape, Permute, Input, add, Conv3D,
    GaussianNoise, ConvLSTM2D
from keras.models import Model
import tensorflow as tf
from sklearn.model_selection import train_test_split


def slice(x):
    return x[:,:,:,:, -1]

inp = Input((201,202,8,1))
permuted = Permute((1,2,4,3))(inp)
noise = GaussianNoise(0.1)(permuted)
last_layer = Lambda(slice, input_shape=(201,202,8,1),
    output_shape=(201,202,1))(noise)
permuted_2 = Permute((4,1,2,3))(noise)
conv_lstm_output_1 = ConvLSTM2D(6, (3,3), kernel_initializer='he_uniform',
    padding='same', activation='relu')(permuted_2)
conv_lstm_output_2 = ConvLSTM2D(6, (9,9), kernel_initializer='he_uniform',
    padding='same', activation='relu')(permuted_2)
conv_output_1 = Conv2D(1, (3,3), padding="same", kernel_initializer='he_uniform',
    activation='relu')(conv_lstm_output_1)
conv_output_2 = Conv2D(1, (3,3), padding="same", kernel_initializer='he_uniform',
    activation='relu')(conv_lstm_output_2)
combined = add([last_layer, conv_output_1, conv_output_2])
model=Model(inputs=[inp], outputs=[combined])
model.compile(optimizer='adam', loss='mse')
```

```
model.summary()

model_history=[]

for i in range(0,4):
  X,y=batchwise_6(start[i],end[i])
  X_train, X_valid, y_train, y_valid = train_test_split(X, y,
  test_size=0.33, shuffle=False)
  history=model.fit(X_train,y_train,validation_data=
  (X_valid,y_valid), epochs =25, verbose=1)
  model_history.append(history)


  #vallosscsloss
  plt.title('Loss vs Val loss')
  plt.plot(history.history['loss'], label='loss')
  plt.plot(history.history['val_loss'], label='val_loss')

  plt.legend()
  plt.show()


print(len(model_history[0].history['loss']))

loss=[]
val_loss=[]
for temp in model_history:
  for i in temp.history['loss']:
    loss.append(i)
  for i in temp.history['val_loss']:
    val_loss.append(i)

print(len(loss))
print(len(val_loss))


#vallosscsloss
plt.title('Loss vs Val loss')
plt.plot(loss, label='loss')
```

```
plt.plot(val_loss, label='val_loss')

plt.legend()
plt.show()


print("Max value element (loss) : ", max(loss))
print("Max value element (val_loss) : ", max(val_loss))

print("Min value element (loss) : ", min(loss))
print("Min value element (val_loss) : ", min(val_loss))


import random as random
#vallosscsloss
plt.title('Loss vs Val loss')
plt.plot(loss, label='loss')
plt.plot(val_loss, label='val_loss')
j=1
for i in range(25,101,25):
  color_1= random.random()
  color_2= random.random()
  color_3= random.random()
  color = (color_1, color_2, color_3)
  plt.axvline(x=i, color=color, linestyle = (0,(0.1,2)),
  dash_capstyle = 'round', label='batch'+str(j))
  j+=1

plt.legend(bbox_to_anchor=(1.05, 1.0), loc='upper left')
plt.show()

model.save('model_6.h5')
model.load_weights('model_6.h5')

#Testing

X,y=batchwise_6(956,1213)
pred = model.predict(X)
```

```
pred = np.squeeze(pred)
pred.shape

y = np.squeeze(y)
y.shape

subs=np.subtract(pred, y)
mae=subs.mean()
mse=np.square(subs).mean()
print("MAE : ",mae)
print("MSE : ",mse)

plt.imshow(pred[0])
plt.imshow(y[0])

#kmph

kmy=y*(1800/5)
kmpred=pred*(1800/5)

subs=np.subtract(kmpred, kmy)
mae=subs.mean()
mse=np.square(subs).mean()
print("MAE : ",mae)
print("MSE : ",mse)

#Removing normalisation only mps

ny=y*100
npred=pred*100

subs=np.subtract(npred, ny)
mae=subs.mean()
mse=np.square(subs).mean()
print("MAE : ",mae)
print("MSE : ",mse)

#Min Max and Difference
```

```
print("Max value element (pred) : ", np.max(pred))
print("Max value element (y) : ", np.max(y))

print("Min value element (pred) : ", np.min(pred))
print("Min value element (y) : ", np.min(y))

subs=np.subtract(pred, y)
print("Min value element (Subs) : ", np.min(subs))
print("Max value element (Subs) : ", np.max(subs))

y[0,0,0]
pred[0,0,0]


#Cyclone
plt.imshow(pred[232])
plt.imshow(y[232])
```

Bibliography

[1] I. Tyass, A. Bellat, A. Raihani, K. Mansouri, and T. Khalili, "Wind Speed Prediction Based on Seasonal ARIMA model," *E3S Web Conf.*, vol. 336, p. 00034, 2022.

[2] S. Osama, A. Darwish, E. H. Houssein, A. E. Hassanien, A. A. Fahmy, and A. Mahrous, "Long-term wind speed prediction based on optimized support vector regression," in *2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS)*, 2017, pp. 191–196.

[3] M. Ibrahim, A. Alsheikh, Q. Al-Hindawi, S. Al-Dahidi, and H. ElMoaqet, "Short-time wind speed forecast using artificial learning-based algorithms," *Comput. Intell. Neurosci.*, vol. 2020, p. 8439719, 2020.

[4] N. Mandal and T. Sarode, "Prediction of Wind Speed using Machine Learning," *Int. J. Comput. Appl.*, vol. 176, no. 32, pp. 34–37, 2020.

[5] S. Samadianfard, S. Mirjalili, and S. Saremi, "Wind speed prediction using a hybrid model of the multi-layer perceptron and whale optimization algorithm," *Energy Reports*, vol. 6, pp. 1147-1159, 2020.

[6] A. C. Cinar and N. Natarajan, "An artificial neural network optimized by grey wolf optimizer for prediction of hourly wind speed in Tamil Nadu, India," *Intelligent Systems with Applications*, vol. 16, no. 200138, p. 200138, 2022.

[7] B. Shao, D. Song, G. Bian, and Y. Zhao, "Wind speed forecast based on the LSTM neural network optimized by the firework algorithm," *Adv. Mater. Sci. Eng.*, vol. 2021, pp. 1–13, 2021.

[8] L. Wang, Y. Guo, M. Fan, and X. Li, "Wind speed prediction using measurements from neighboring locations and combining the extreme learning machine and the AdaBoost algorithm," *Energy Rep.*, vol. 8, pp. 1508–1518, 2022.

[9] Q. Hu, R. Zhang, and Y. Zhou, "Transfer learning for short-term wind speed prediction with deep neural networks," *Renew. Energy*, vol. 85, pp. 83–95, 2016.

[10] S. Kumar P, "Improved prediction of wind speed using machine learning," *EAI Endorsed Trans. Energy Web*, vol. 6, no. 23, p. 157033, 2019.

[11] Y. Chen, Y. Wang, X. Zhang, S. Yang and Y. Xiong, "2-D regional short-term wind speed forecast based on CNN-LSTM deep learning model," *Energy Convers. Manag.*, vol. 244, p. 114451, 202

[12] Z. Karevan and J. A. K. Suykens, "Transductive LSTM for time-series prediction: An application to weather forecasting," *Neural Netw.*, vol. 125, pp. 1–9, 2020.

[13] M. A. R. Suleman and S. Shridevi, "Short-term weather forecasting using spatial feature attention based LSTM model," *IEEE Access*, vol. 10, pp. 82456–82468, 2022.

[14] A. G. Salman, Y. Heryadi, E. Abdurahman, and W. Suparta, "Single layer multi-layer long short-term memory (LSTM) model with intermediate variables for weather forecasting," *Procedia Comput. Sci.*, vol. 135, pp. 89–98, 2018.

[15] D. N. Fente and D. Kumar Singh, "Weather forecasting using artificial neural network," in *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, 2018, pp. 1757–1761.

[16] M. Moishin, R. C. Deo, R. Prasad, N. Raj, and S. Abdulla, "Designing deep-based learning flood forecast model with ConvLSTM hybrid algorithm," *IEEE Access*, vol. 9, pp. 50982–50993, 2021.

[17] R. Chen, X. Wang, W. Zhang, X. Zhu, A. Li, and C. Yang, "A hybrid CNN-LSTM model for typhoon formation forecasting," *Geoinformatica*, vol. 23, no. 3, pp. 375–396, 2019.

[18] S. Alemany, J. Beltran, A. Perez, and S. Ganzfried, "Predicting hurricane trajectories using a recurrent neural network," *Proc. Conf. AAAI Artif. Intell.*, vol. 33, no. 01, pp. 468–475, 2019.

[19] R. Lagerquist, A. McGovern, C. R. Homeyer, D. J. Gagne II, and T. Smith, "Deep learning on three-dimensional multiscale data for next-hour tornado prediction," *Mon. Weather Rev.*, vol. 148, no. 7, pp. 2837–2861, 2020.

[20] K. Biswas, S. Kumar, and A. K. Pandey, "Tropical cyclone intensity estimations over the Indian ocean using Machine Learning," *arXiv* [physics.ao-ph], 2021.

[21] R. Chen, W. Zhang, and X. Wang, "Machine learning in tropical cyclone forecast modeling: A review," *Atmosphere (Basel)*, vol. 11, no. 7, p. 676, 2020.

[22] Y. Wu, X. Geng, Z. Liu, and Z. Shi, "Tropical cyclone forecast using multitask deep learning framework," *IEEE Geosci. Remote Sens. Lett.*, vol. 19, pp. 1–5, 2022.

[23] C. P. Chand, M. M. Ali, B. Himasri, M. A. Bourassa, and Y. Zheng, "Predicting Indian ocean cyclone parameters using an artificial intelligence technique," *Atmosphere (Basel)*, vol. 13, no. 7, p. 1157, 2022.

[24] C. Beccario, "Earth:: A global map of wind, weather, and ocean conditions," Nullschool.net. [Online]. Available: https://earth.nullschool.net/. [Accessed: 08-Apr-2023].

[25] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," *arXiv [cs.LG]*, 2016.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," *arXiv [cs.CV]*, pp. 1026–1034, 2015.

[27] Wikipedia contributors, "Cyclone Chapala," Wikipedia, The Free Encyclopedia, 06-Mar-2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Cyclone_Chapala&oldid=1143240241. [Accessed: 08-Apr-2023].

Biodata

Name:
Deeptimaan Banerjee
Mobile No.:
8369861428
E-mail:
deeptimaan@hotmail.com
Permanent Address:
46/703, EMP Phase 5, Thakur Village, Kandivali East, Mumbai - 400101 Name:

Tarunveer Singh Sidhu
Mobile No.:

7307075729
E-mail:
tarunsidhu1.0@gmail.com
Permanent Address:
Vill. Gurali, P.O. Fatehgarh Panjtoor Teh. zira, Fategarh Panjtoor, Ferozepur, Punjab-142043

Name:
Prakhar Narain Srivastava
Mobile No.:
9574354320
E-mail:
prakhar.srivastava1511@gmail.com
68, Jaylaxmi Society, Old Padra Road, Vadodara - 390020, Gujarat

Chapter 7

Annexure

ELSEVIER

Pattern Recognition Letters
journal homepage: www.elsevier.com

Improved Neural Network based Modeling for Wind Speed Prediction in The Indian Ocean Region

Deeptimaan Banerjee[a], Prakhar Narain Srivastava[a], Tarunveer Singh Sidhu[a], Dr. Saurav Gupta[b] *

[a] *Vellore Institute of Technology, Chennai 600127, India*
[b] *Vellore Institute of Technology, Chennai 600127, India*

ABSTRACT

One area that has not received adequate attention in contemporary times is Wind Speed Prediction. Accurate Wind Speed Prediction is important for various industries such as those concerning shipping, offshore oil and gas exploration, and cyclone predictions. Wind Speed can be predicted accurately over a short period of time and is majorly focused on a particular longitude, and latitude. This research paper aims to predict wind speed over a large region, i.e. particularly in the Indian Ocean region using next frame prediction via neural network models constituting multiple longitudes and latitudes. The study compares to predict wind speed and represent it as a 2D vector, researchers have employed an Artificial Neural Network model as well as a modified architecture known as Convolutional Long-Short Term Memory (ConvLSTM). The technique of Next-Frame Prediction, which utilizes the dense architecture of either Convoluted Neural Network (CNN) layers or Artificial Neural Network (ANN) layers, has been used to implement Deep Learning Models. The findings demonstrate the neural network model's ability to predict wind speed in the Indian Ocean region as 2D images for up to 6, 12, and 24 hours into the future. The study suggests that neural network models can be a valuable tool for predicting wind speed in the Indian Ocean region.

1. Introduction

The region encompassing the Indian Ocean is known to be highly susceptible to tropical cyclones and extreme weather phenomena, making accurate Wind Speed Prediction crucial for disaster management and maritime operations. Traditional methods for Wind Speed Prediction rely on numerical weather prediction models, which are computationally expensive and require high-resolution input data. The development of remote sensing and satellite data has led to the emergence of machine learning algorithms as an effective tool for Wind Speed Prediction.

Wind Speed Prediction (WSP) is a crucial task due to its impact on various activities such as renewable energy generation, weather forecasting, and air traffic control on land and fishing, transportation, and offshore operations in oceanic regions. WSP has been achieved over the years using a range of techniques, from simpler ones such as Autoregressive integrated moving average (ARIMA)[1] and Support vector machines (SVM) [2] to more advanced machine learning techniques [3][4]. Optimization algorithms such as Whale Optimization (WOA) [5], Grey Wolf (GWA) [6], Fireworks (FWA) [7] and AdaBoost [8] have been utilized to improve WSP models. Novel Neural Network models such as Shared Hidden Layer- Deep Neural Network (SHL-DNN) [9], nonlinear autoregressive network with exogenous inputs (NARX), Radial basis function (RBF) and Back Propagation (BPN) [10] have also been developed for WSP. A different research effort [11] suggested a two-dimensional framework for forecasting the wind speed of a specific location in the near future, similar to the one proposed in this study.

Machine Learning and Neural Networks have been considered as well as favoured for the prediction of other weather parameters like Pressure, Temperature [12][13], Dew Point, Humidity [14], visibility and Precipitation [15]. This paper focuses on WSP which is an important factor for predicting tropical cyclones. Studies have been conducted where machine learning models have been used to efficiently predict other natural disasters like floods [16], typhoons [17], hurricanes [18] and tornadoes [19]. Intensity Prediction and tracking of Tropical cyclones has been made a possibility due to models like [20],[21],[22] and [23].

The novelty of this research paper is the creation of a two-dimensional model that employs two machine learning approaches to forecast wind speed across the extensive Indian Ocean region, enabling accurate WSPs for almost 40,000 locations simultaneously. This proposed method can be implemented in real-time scenarios, such as disaster management and maritime operations. The effectiveness of the proposed neural network model is assessed using metrics such as mean squared error (MSE) and mean absolute error (MAE).

* Corresponding author. e-mail: