# Meteorological Variables Nowcasting using Machine Learning and Deep Learning Techniques

**A Thesis**

submitted to
Indian Institute of Science Education and Research Pune
in partial fulfillment of the requirements for the
BS-MS Dual Degree Programme

by

**Hrishikesh Kailas Haral**


Indian Institute of Science Education and Research Pune
Dr. Homi Bhabha Road,
Pashan, Pune 411008, INDIA.

May, 2023

Supervisor: Dr Bipin Kmar (IITM Pune)
TAC Expert: Dr Amit Apte (IISER Pune)
Co-Supervisor: Dr MCR Kalpureddy (IITM Pune)

# Certificate

This is to certify that this dissertation entitled **Meteorological Variables Nowcasting using Machine Learning and Deep Learning Techniques** towards the partial fulfillment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune, represents the work carried out by Hrishikesh Kailas Haral at the Indian Institute of Tropical Meteorology, Pune, under the supervision of Dr Bipin Kumar, during the academic year 2022-2023.

**Dr Bipin Kumar**
Scientist E, IITM Pune

Committee
Supervisor : Dr Bipin Kumar
TAC Expert : Dr Amit Apte
Co-supervisor : Dr MCR Kalpureddy

***I dedicate this thesis to my family***

# Declaration

I hereby declare that the matter embodied in the report entitled **Meteorological Variables Nowcasting using Machine Learning and Deep Learning Techniques** are the results of the work carried out by me at the Indian Institute of Tropical Meteorology, Pune, under the supervision of Dr Bipin Kumar, and the same has not been submitted elsewhere for any other degree

Hrishikesh Kailas Haral

# Acknowledgements

I am grateful to my supervisor Dr Bipin Kumar for allowing me to work under his supervision at IITM Pune. I thank my TAC expert Dr. Amit Apte for his valuable suggestions and comments during my mid-year evaluation. I thank Co-supervisor Dr MCR Kalpureddy for his valuable input to the study. Finally, I thank my friends and family for their encouragement and support throughout my studies.

# Abstract

In this work, we address two nowcasting problem statements. The first is the $PM_{2.5}$ concentration temporal nowcasting over Delhi NCR. It has been observed that the high levels of $PM_{2.5}$ concentrations in Delhi, particularly for the winter season, are affecting many people's health and causing various respiratory diseases. In this work, we try different Machine Learning and Deep Learning techniques for $PM_{2.5}$ nowcasting with a lead time of up to six hours. The second is Precipitation nowcasting using Bhopal Radar Data. It is essential because it provides critical information to protect people, property, and infrastructure from the impacts of extreme weather events. In this work, we develop a deep learning-based ConvLSTM model to perform spatiotemporal nowcasting.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Weather Forecasting is imperative because it helps reduce loss of life and property. It allows us to take better decisions with the available information. Forecasting Weather is a complex process because of the various factors affecting the phenomenon. Also, the Earth's atmosphere dynamically changes across temporal and spatial scales. In recent times, air pollution has been on the rise. The air quality index (AQI) has plummeted in Northern India. The National Capital Region is a part where New Delhi is among the regions of highest ambient particulate matter exposure in the country. $PM_{2.5}$ causes very much Air Pollution. These particles, with a size of $\leq$2.5 microns, penetrate deep into the lung and corrode the alveolar wall. These causes impair lung function and other respiratory diseases. Hence, it is vital to Forecast the magnitude of $PM_{2.5}$ concentrations in advance to provide timely information to vulnerable communities and individuals to help them minimise their exposure to $PM_{2.5}$ air pollution. Similarly, Precipitation Now-casting is also very important to make well-informed decisions for any calamities or extreme events that can happen.

Currently, we use the Numerical Weather Prediction (NWP) Model for Meteorological Forecasting. NWP computer models process current weather measurements to predict future weather. The model's structure and output are based on the most recent weather observations. Other models use Statistical Forecasts routinely to enhance the results of the dynamical forecasts at operational weather stations worldwide and employ the least squared regression method for forecasting (Wilks, 2011, Chapter 6). One such model is ARIMA(Box et al., 2015). This model is also used as a benchmark to test other models. But it gives poorer performance for long-term forecasts and is

computationally expensive(One, 2021).

Nowadays, Machine Learning is applied for Weather Forecasting, which is robust to perturbations and can generate more accurate weather forecasts for larger periods(Holmstrom et al., 2016). Machine Learning algorithm like Random Forest Classifier provides a low-cost and portable solution(Singh et al., 2019). Machine Learning algorithms efficiently analyse and model spatially and temporally variable environmental data but cannot capture spatiotemporal dependencies(Amato et al., 2020). This drawback can be resolved using Deep Learning which can capture spatiotemporal relations using Convolution Neural Network (CNN)(Wang et al., 2020). Certain deep learning algorithms, such as Recurrent Neural Networks, LSTM, and GRU, accept sequential data as input and provide predictions based on the data's individual time steps. LSTM Networks are used for temporal weather prediction tasks as seen in (Salman et al., 2018). For spatiotemporal Precipitation nowcasting (Shi et al., 2015) proposed Conv-LSTM, which has outperformed FC-LSTM and ROVER algorithm as validated by the authors. Figure 1.1 shows the ML/DL pipeline. In this thesis, we have worked on statistical,


Figure 1.1: AI Based Forecasting(M, 2021)

machine learning and deep learning-based algorithms to forecast variables like $PM_{2.5}$ and Precipitation

The thesis is organised as follows:

In the next section, we define the problem statement. Chapter 2 consists of the theoretical background related to our work. In Chapter 3, we discuss the

data and methodology used. Results and discussion is covered in chapter 4. Finally, in chapter 5, we have a conclusion and future work.

## 1.1 Problem Statement

Weather forecasting can be thought of sequence forecasting problem. For any model to produce a forecast, it learns the pattern in the data using the recent time steps in the past; e.g. it takes $m$ lag steps to create a lead of $n$ steps. One forecasting model, such as ARIMA, works on the same principle. It produces output auto regressively over the predicted timestep(Box et al., 2015) Similarly, the Machine Learning models require the data in split sequences to learn the temporal relationship in data to output the Forecast (Brownlee, 2020).

Let $X_1$, $X_2$, $X_3$ , ..., $X_t$ represent a sequence of observations. The temporal sequence forecasting problem predicts the most likely K-length sequence in the future, given the previous J observations, including the current one(Shi et al., 2015). The following can be framed as given in equation 1.1.

$$\widetilde{X}_{t+1}, \ldots, \widetilde{X}_{t+K} = arg\max p(X_{t+1}, \ldots, X_{t+K} | \widehat{X}_{t-J+1}, \widehat{X}_{t-J+2}, \ldots, \widehat{X}_t) \quad (1.1)$$

This implementation can be used for both temporal and spatiotemporal weather forecasting. For spatiotemporal, let's suppose we have a $M \times N$ grid. Inside each grid point, $P$ variables are measured for any time, and the observation can be represented as $X \in R^{P \times M \times N}$. For e.g. in a 3-dimensional array with dimensions of $time \times x \times y$, where $x, y$ is the number of rows and columns of an array, we need to predict $K$ arrays given the previous sequence of $J$ arrays. Similarly, we would have data in a 2-dimensional format for temporal sequence, but formulation will be the same, i.e. predict K timesteps in future given the previous information of J timesteps.

# Chapter 2

# Theoretical Background

## 2.1 Machine Learning

A computer science and artificial intelligence subfield called "machine learning" focuses on using data and algorithms to mimic how people learn. It has two subcategories, supervised and unsupervised learning. In the former, a labelled dataset is used to train algorithms to classify or predict outcomes, while the latter contains unstructured data which can be analyzed or clustered using machine learning algorithms. Common Supervised Learning tasks include Regression and Classification, while Clustering, Dimensionality Reduction, and Anomaly Detection can be made using unsupervised machine learning.
The Machine Learning model inputs the data and generates the predictions, which can be tested using metrics like RMSE in Regression, accuracy in classification, etc. If the predictions do not match, then the model is repeatedly trained to minimise the defined cost function to improve itself.

### 2.1.1 Linear Regression

Linear Regression comes under the supervised machine learning technique. The model finds the best fit linear line or, to say, it captures the linear relationship between the dependent and independent variables. Let $X_1, X_2, \ldots, X_n$ be the independent variables in our dataset and corresponding, each row has labelled dependent feature $Y$, so our task is to find the best fit linear line which maps from $X$ to $Y$. The linear equation 2.1 has weights $b_1, b_2, \ldots, b_n$

and bias $b_0$ to get the least mse.

$$Y = b_0 + b_1 X_1 + b_2 X_2 + \ldots + b_n X_n \tag{2.1}$$

It uses mean squared error as the cost function to adjust the weights and the biases. Here $y_i$ is the ground truth value, and $\hat{y}_i$ is the predicted one.

$$\text{MSE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2.2}$$

The above cost function 2.2 has to be minimized using some optimization algorithm. Since this is a convex cost function with only one minimum, we can use Gradient Descent Algorithm to reduce the loss. The model then can be evaluated using R2 Score or other metrics like MSE, RMSE etc., to check the performance.

$$\text{R}^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \tag{2.3}$$

### 2.1.2 Gradient Descent

It is used for finding a local minimum of a differentiable function. Consider the above model 2.1 for only one independent variable (n = 1). So, we have $Y_{pred} = b_0 + b_1 X$, here $Y_{pred}$ is predicted output, $b_0$ is intercept, $b_1$ is slope and $X$ is input sample. The task is to reduce cost function 2.2 using some optimization. This can be achieved using the gradient descent algorithm on $J(w, b)$ where J is cost function. It randomly initializes some weight($b_1$) and bias($b_0$) which are updated using the steps below where $\alpha$ is the learning rate. The $\alpha$ generally lies between 0 and 1. The smaller the learning rate, the slower the step will be, and the bigger the learning rate, the larger the steps will be. Equations 2.4 and 2.5 update the weight and bias by calculating the partial derivatives.

Repeat until convergence:

$$w \leftarrow w - \alpha \frac{\partial J(w, b)}{\partial w} \tag{2.4}$$

$$b \leftarrow b - \alpha \frac{\partial J(w, b)}{\partial b} \tag{2.5}$$

Each gradient descent step traverses the curve figure in the steepest instantaneous direction. The model is said to be trained after the weights and

biases are found for the minimum of the cost function. Figure 2.1 shows the working of the algorithm.

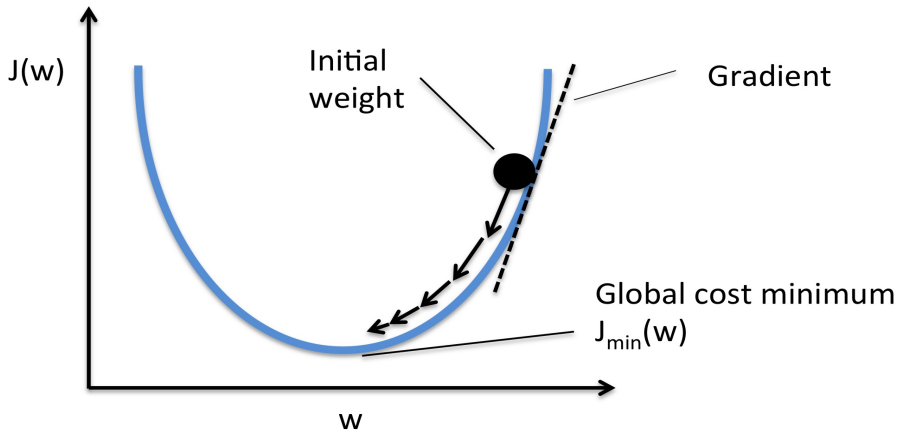



Figure 2.1: Gradient Descent Algorithm(Raschka, 2023)

### 2.1.3 XGBoost

A tree-based machine learning model called Gradient Boosting uses gradient descent for optimization. An extension of this is Extreme Gradient Boosting, also known as XGBoost, which can avoid over-fitting using L1 and L2 regularization techniques. It can handle sparse data. It uses multiple cores of CPU for faster computing. It has high performance and accuracy.

## 2.2 Deep Learning

Deep Learning is a subfield of Machine Learning. It consists of a neural network made up of artificial neurons. The neural network attempts to simulate the human brain's behaviour; it can learn from extensive data. Nowadays, deep learning technology is used in speech recognition, self-driving cars, natural language processing etc. In some cases, like image detection, it outperforms humans(He et al., 2015). It can work on unstructured data like images and text without some data preprocessing generally required in machine learning.

### 2.2.1 Artificial Neural Network

The most fundamental unit of any deep learning model is Artificial Neuron. (figure 2.2). It receives one or more inputs and sums them up after giving each input some weight and bias randomly. Further, it applies non-linear transformation using some activation function and produces the output. The errors are compared using some cost function like MSE(see 2.2), and weights are updated using the Gradient Descent algorithm(see 2.4,2.5) during the backpropagation.

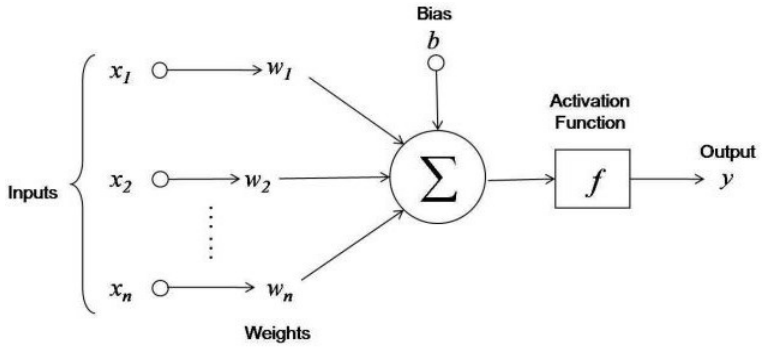



Figure 2.2: Artificial Neuron(Quiza and Davim, 2011)

More such neurons are added layer-wise to extract features from the inputs, and finally, all these layers are connected to the output layer. In this way, an Artificial Neural Network is created. Because of many parameters, these networks usually require a lot of time to train. If the Neural Network contains more than three layers, it is called Deep Neural Network. In figure 2.3, inputs are connected to more than one neuron, and a hidden layer is connected. The neurons use activation functions to learn non-linearity in data. The sigmoid activation function is used to scale the values between range [0,1]; it is represented by $\sigma(x)$ as given in Equation 2.6.

$$\sigma(x) = \frac{e^x}{1 + e^x} \tag{2.6}$$

Another commonly used activation function is the 'Rectified Linear' (ReLU) unit is used to scale the values in the range $[0,\infty)$ given in the equation 2.7. For values less than zero, it replaces them with zero, and the same for values

greater than zero.

$$g(x) = \max(0, x) \tag{2.7}$$

Hyperbolic tangent is used to scale the values in the range (-1,1). It is similar to sigmoid but centred at zero. It is given in equation 2.8.

$$g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.8}$$

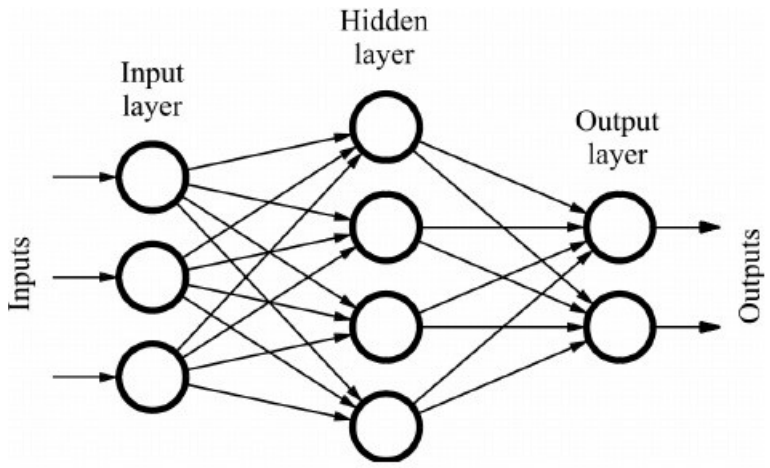



Figure 2.3: Artificial Neural Network (Quiza and Davim, 2011)

The neural network uses the cost function to calculate the difference between the predictions of the network and the actual truth. The model learns by minimizing the cost function. The commonly used cost functions include 'mean squared error', 'cross entropy', 'mean absolute error' etc. During the forward propagation, the neural network generates some output, and the errors get back propagated such that weights and biases are adjusted using an optimizer. Generally, 'Adam', 'RMSProp', and 'Gradient Descent' algorithms are used for optimization. The training data can be provided in batches; this is a hyperparameter a larger batch size means a lower number of training batches and vice versa. Finally, the model can be evaluated using metrics like 'rmse' for regression or 'accuracy' for classification problems.

### 2.2.2 Convolutional Neural Network

The ANN can handle image data, but its computational complexity increases due to increased parameters with the image size; hence, convolutional neural

networks were introduced, which can extract relevant features from the images and thereby reduce the number of parameters to learn in the data. The convolution operation passes a filter over the image to extract the features. Figure 2.4 shows convolution operation on a 7×7 image with a 3×3 filter to generate a new image of size 5×5. We see that element-wise multiplication occurs, which is added to create the final output. A bias is added to this output; an activation function is applied to the final output. If we provide

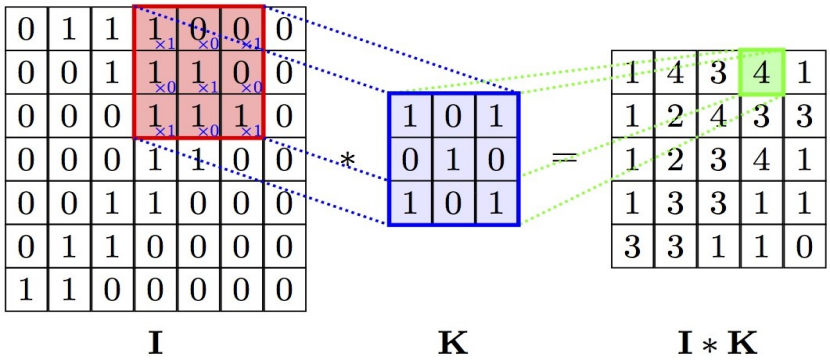



Figure 2.4: Convolution Operation on an Image

more filters, then that number of image arrays will get output. The filter size and the dimensionality can be adjusted. The greyscale images use `Conv2D` filter while the RGB channel images can use both `Conv2D & Conv3D`. Two common types of padding are applied in CNN, namely 'Same' and 'Valid'. For 'Same', the size of the input array is maintained after performing the convolution operation. While in the case of 'valid', the input array size is reduced. Other operations like 'max pooling' and 'average pooling' are used to reduce the spatial size of the image. Figure 2.5 shows 'max' and 'average' pooling applied over a 4×4 size image to reduce spatial dimension to 2×2. In max pooling operation, it takes the maximum value of the elements in the pooling window. While the average pooling averages all the elements in the pooling window. This is particularly helpful in the case of large-size images where not a large amount of information will be lost.

### 2.2.3 Recurrent Neural Network

The ANN can take only fixed-size input but cannot process sequential information such as time series, speech, and natural language, so 'Recurrent

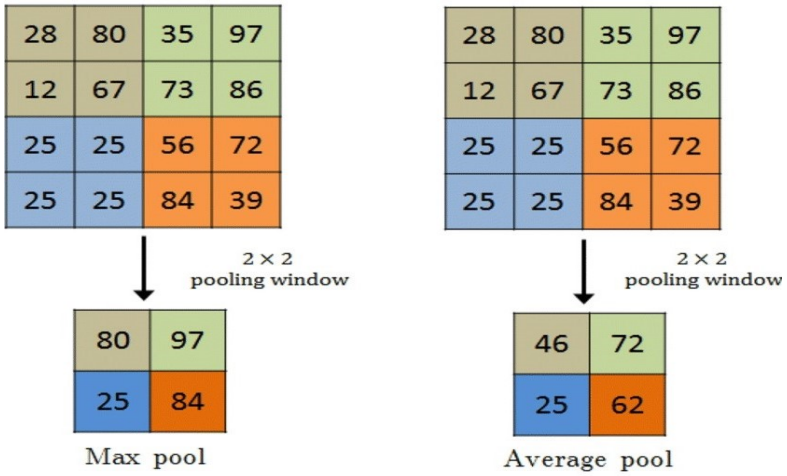



Figure 2.5: Pooling operation on an Image

Neural Networks' were introduced, which can take sequences of inputs of arbitrary length and process them sequentially. The output of the previous cell, called the hidden state is passed on with the following input so that the network learns the sequential information. Figure 2.6 shows the unrolled 'RNN' that takes input $X_t$ inputs, producing a hidden state $h_t$ at each timestep that is passed on with the following input in the sequence. In this way, the network learns to retain the sequential information of previous timesteps.

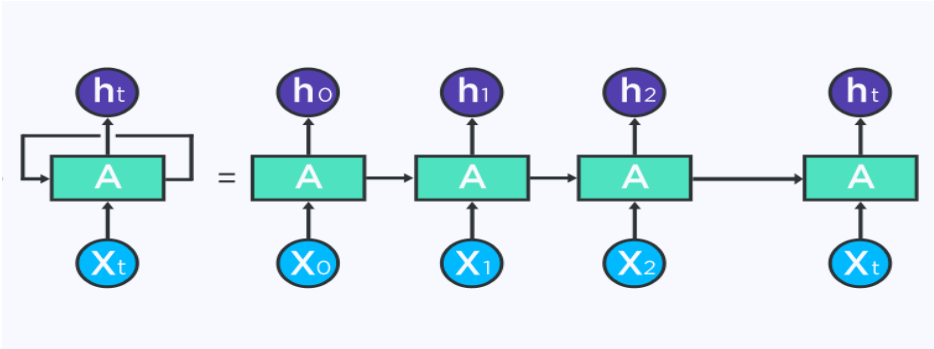



Figure 2.6: Unrolled Recurrent Neural Network(RNN)(Donges, 2023)

### 2.2.4 LSTM

'RNN' fail to capture correlation in sequences which are far apart due to the problem of vanishing/exploding gradient. The Long Short-Term Memory(LSTM)(Hochreiter and Schmidhuber, 1997) was introduced to counter this. It has memory cell $C_t$, input gate, forget gate and output gate and hidden state like in 'RNN'. In Figure 2.7 the cell takes input $X_t$ along with

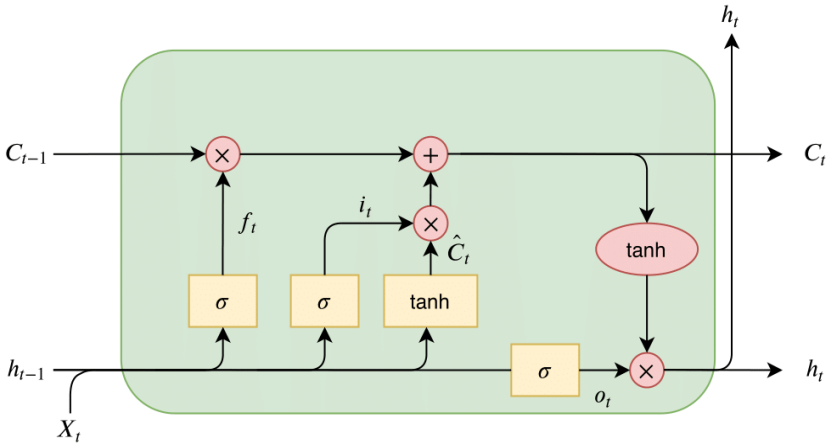



Figure 2.7: LSTM Cell

$C_{t-1}$ and $h_{t-1}$ from previous cell. The '$\times$' in the figure performs element-wise multiplication, $i_t$ is the input gate, $f_t$ is the forget gate, and $o_t$ is the output gate. These control the flow of information in and out of the memory cell.

### 2.2.5 ConvLSTM

For spatiotemporal precipitation nowcasting (Shi et al., 2015) introduced the ConvLSTM model, which is similar to LSTM architecture but uses convolution operation instead of matrix multiplication in LSTM, which extracts spatial patterns in data. The equations are similar to the LSTM. A ConvLSTM cell contains an input, forget, and output gate with cell state $C_t$. In the figure 2.8 the '*' denotes the convolution operation, '$\times$' denotes Hadamard product and '+' denotes addition.

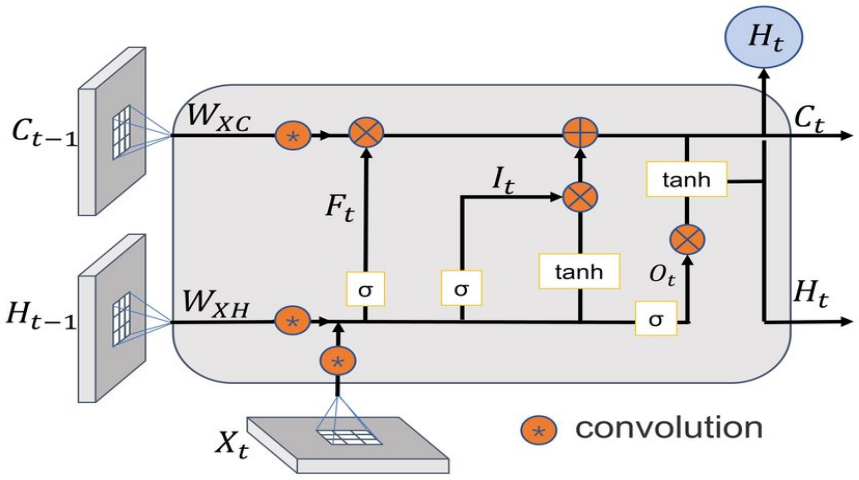



Figure 2.8: ConvLSTM Architecture(Shi et al., 2022)

The equations in the ConvLSTM cell are given below. $W$ is the filter applied over $X_t$ to extract spatial features. $\circ$ gives the hadamard product between two arrays.$\sigma$ and $tanh$ are the activation functions. $H_t$ and $C_t$ is the hidden and cell state respectively.

$$
\begin{aligned}
i_t &= \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \circ C_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \circ C_{t-1} + b_f) \\
\tilde{C}_t &= tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + b_c) \\
C_t &= f_t \circ C_{t-1} + i_t \circ \tilde{C}_t \\
o_t &= \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \circ C_t + b_o) \\
H_t &= o_t \circ tanh(C_t)
\end{aligned}
$$

# Chapter 3

# Data and Methodology

We have addressed two problem statements for the meteorological variables nowcasting and used different data for each. This is because the first problem statement is temporal nowcasting of $PM_{2.5}$,with up to six hours, and the second is spatiotemporal precipitation nowcasting using radar data. Data used for each will be explained in the upcoming sections.

## 3.1 Temporal Nowcasting of $PM_{2.5}$

The data was taken from the Center for Pollution Control Board(CPCB)[1] website, which consists of $PM_{2.5}$ and $PM_{10}$ Air Pollution Data. The data is generated using the stations monitoring the air pollutant concentrations. For our model building, we have taken thirty nine station's $PM_{2.5}$ and $PM_{10}$ data in the Delhi NCR from the year 2017 to the year 2022 with Hourly temporal resolution.

Other variables are taken from ERA5. The data is available to download using Climate Data Store(CDS)[2].

We took data on wind speed, relative humidity, temperature, surface Pressure, boundary layer height, precipitation, and wind direction from the CDS for Delhi Coordinates (28.7°N, 77.1°S) to generate temporal time series from 2017 to 2022 to match the duration of $PM_{2.5}$ data.

[1] https://cpcb.nic.in/

[2] https://cds.climate.copernicus.eu/

### 3.1.1 Data Information

#### $PM_{2.5}$

This is the target variable, the concentration we want to predict. It is measured in $\mu g/m^3$ and has a mean of $105.8 \mu g/m^3$. It was found that the concentration of $PM_{2.5}$ varies seasonally.

#### $PM_{10}$

It has a good correlation with the target variable, so it is also considered for the study. It is measured in $\mu g/m^3$ and has a mean of $216 \mu g/m^3$. Similar seasonality was observed in $PM_{10}$ concentration as that of $PM_{2.5}$.

#### Precipitation

Precipitation has a notable impact on $PM_{2.5}$ concentration. It has a wet scavenging effect meaning an increase in precipitation decreases $PM_{2.5}$ concentration level and vice-versa(Liu et al., 2020). Therefore, precipitation is taken into account.

#### Boundary Layer Height

It is a major factor in the dispersion of $PM_{2.5}$ pollutants in the atmosphere. Hence it is taken into consideration.

#### Relative Humidity

The abrupt and accumulative rise in $PM_{2.5}$ concentrations could be due to RH fluctuations(Zhang et al., 2017). So, it is an important variable to include in our study.

#### Surface Pressure

It is found that $PM_{2.5}$ has a positive correlation with surface pressure (Li et al., 2017). Hence it is taken into consideration.

#### Wind Velocity ($u$,$v$)

Wind speed's u-component, v-component and magnitude were taken because changes in the wind patterns affect the $PM_{2.5}$ concentrations, as studied in (Liu et al., 2020) paper.

All the variables used in the model building are summarized in the table 3.1

| Sr.No. | Variable | Units | Source |
|---|---|---|---|
| 1 | $PM_{2.5}$ | $\mu g/m^3$ | CPCB |
| 2 | $PM_{10}$ | $\mu g/m^3$ | CPCB |
| 3 | Precipitation | mm/hr | ERA5 |
| 4 | Relative Humidity | none | ERA5 |
| 5 | Wind Speed ($u$,$v$) | m/s | ERA5 |
| 6 | Boundary Layer Height | m | ERA5 |
| 7 | Temperature | K | ERA5 |
| 8 | Surface Pressure | Pa | ERA5 |

Table 3.1: Variables in the Data

### 3.1.2 Data Preprocessing

#### UTC to IST conversion

ERA5 reanalysis data in Coordinated Universal Time(UTC) format is converted to Indian Standard Time(IST) to match the $PM_{2.5}$ station data.

#### Imputation of Missing Values

For the CPCB website data, it was observed that many stations in Delhi NCR do not have continuous data, i.e. values for some duration were missing. To resolve this, we took the average of all forty station data in NCR to generate a single time series for 2017 to 2022. Missing values in the ERA5 reanalysis data were linearly interpolated.

#### Data Slicing

The air pollution in Delhi NCR increases in the winter season, so data is sliced for the winter period of each year from September to February of the following year while maintaining the hourly temporal resolution. This is done to make the model learn the seasonality and trend in this particular winter duration rather than providing a whole year's data. Temporal slicing is carried out for all the variables in the dataset. After slicing, the data was reduced to Time(18144) $\times$ Variables(8) from Time(48912) $\times$ Variables(8) dimensions. Doing the steps mentioned above, we recovered the final data for model building.

### 3.1.3 Model Building

#### Normalization

The data variables have different units (see table 3.1), but Machine Learning or Deep Learning models are not scale-independent. Hence, data is scaled using Normalization. It re-scales the values so that they range between 0 and 1.

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.1}$$

here, $X_{max}$ is maximum value and $X_{min}$ is minimum value in data.

#### Supervised Splits

Supervised Machine Learning problems require data mapping from X to Y, i.e. each input of row must have a defined output value, but time-series data does not follow this, so we need to convert it using supervised splits. Let $X_1, X_2, .., X_n$ represent the values in some time series. Suppose the splits are formed using twenty-four lag time steps and producing a forecast of six-time steps(see table 3.2). This generates the required mapping for supervised machine learning.

#### Train-Validation-Test Split

Data was divided into a train-validation-test split on winter data separated using temporal slicing.(section 3.1.2). The Data in Table 3.3 refers to the duration of winter from September start to February end of the following year.

| $\boldsymbol{X}$ (Input) | $\boldsymbol{Y}$ (Output/Forecast) |
|---|---|
| $X_{t1}$, $X_{t2}$, $X_{t3}$,...,$X_{t24}$ | $X_{t25}$, $X_{t26}$, $X_{t27}$,...,$X_{t30}$ |
| $X_{t2}$, $X_{t3}$, $X_{t4}$,...,$X_{t25}$ | $X_{t26}$, $X_{t27}$, $X_{t28}$,...,$X_{t31}$ |
| $X_{t3}$, $X_{t4}$, $X_{t5}$,...,$X_{t26}$ | $X_{t27}$, $X_{t28}$, $X_{t29}$,...,$X_{t32}$ |
| ⋮ | ⋮ |
| $X_{tn-30}$, $X_{tn-29}$, $X_{tn-28}$,...,$X_{tn-6}$ | $X_{tn-5}$, $X_{tn-4}$, $X_{tn-3}$,...,$X_{tn}$ |

Table 3.2: Supervised Split

| **Training Set** | **Validation Set** | **Test Set** |
|---|---|---|
| 2017 to 2019 | 2020 | 2021 |

Table 3.3: Train Test Split

### Models

Different models were tried for nowcasting, XGBoost, LSTM, Random Forest etc. For the XGBoost model, n_estimators is 1000, and the cost function is 'squared error', and other hyperparameters are unchanged. For LSTM Model, the loss function is mse, and batch_size is 15. The following table 3.4 shows the LSTM model architecture(Hochreiter and Schmidhuber, 1997).

| **Layer no.** | **Layer** | **No. of Units** | **Activation** |
|---|---|---|---|
| 1 | `LSTM` | 32 | `relu` |
| 2 | `Dense` | 64 | `relu` |
| 3 | `Dense` | 128 | `relu` |
| 4 | `Dense` | 64 | `relu` |
| 5 | `Dense` | 1 | `linear` |

Table 3.4: LSTM Model Architecture

## 3.2 Spatiotemporal Precipitation Nowcasting

### 3.2.1 Data Information

For spatiotemporal nowcasting, we have used Bhopal Radar Data[3]. This precipitation data is taken for the JJAS months of 2021. Data were provided in the polar coordinates, and the temporal resolution of the data was 20 minutes.

### 3.2.2 Data Preprocessing

#### Data Conversion

Since the data was in polar coordinates and our model accepts the data in the gridded format, the data was converted to grid coordinates at the specific height of 2 km with respect to the ground. This height was decided because clouds at this have a high percentage of precipitation. After conversion, we obtained a grid of 250 $\times$ 250 dimensions( Latitude $\times$ Longitude).

#### Data Capping

We tried to apply inter-quartile based filtering, but the lower range was converted to negative values, and negative values are not defined for precipitation. So, to remove the outliers in the converted dataset, we applied percentile-based filtering. All the values above $99^{th}$ percentile were capped to $99^{th}$ percentile, and the values below one percentile were limited to one percentile. The data range was reduced from [0,1359]mm to [0,8)mm. This seems to be valid because the value of 1359 mm in the data is an outlier. The data capping is shown in figure 3.1.

[3]Dr MCR Kalpureddy, IITM Pune provided the data

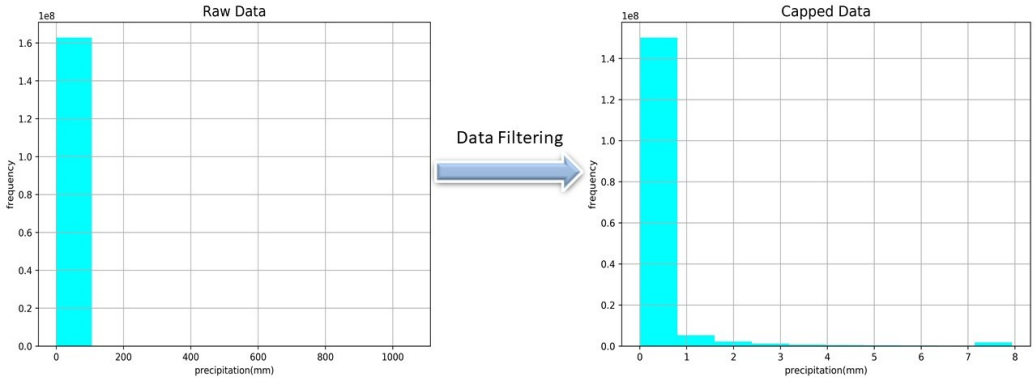



Figure 3.1: Percentile Based Data Capping

### Data Normalization

The capped data were scaled using Normalization (equation 3.1) to bring the data in the [0,1] range. This was necessary because we used exponential transformation for the next step, and we did not want the range to exceed 8 mm.

### Imputation of missing values

The precipitation values range from [0,$\infty$), and our converted gridded data contains NaN values, so we cannot simply impute them with zero as this is a valid precipitation value. Hence we use the exponential transformation $\exp^{kx}$ where $k = 2$ for all the scaled values and have replaced the NaN values with zero. Here the value of $k$ is chosen to be two to scale the data back to the capped range of [0,8)mm. This transformation will bring the data to the [0,$\infty$) range. Where the valid precipitation values(non-NaN grid points) range between [1,$\infty$) and NaN values equal zero as seen in figure 3.2. A similar technique has been applied in (Kumar et al., 2022).

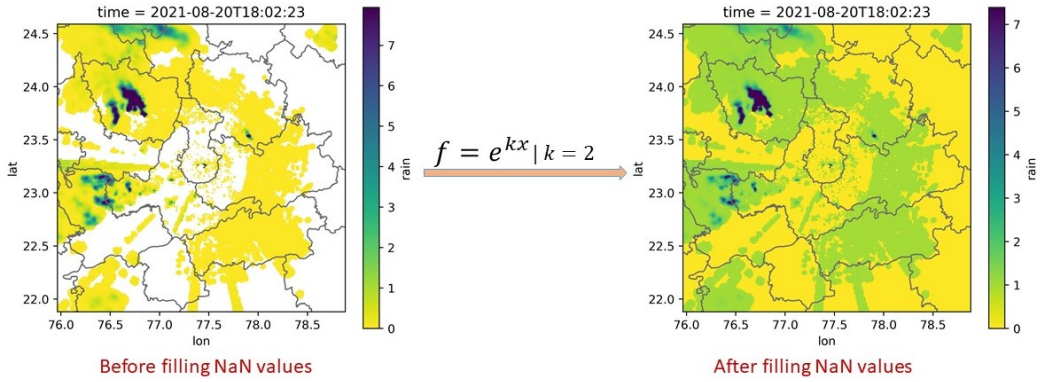



Figure 3.2: Transformation to Exponential Space

### 3.2.3 Model Building

#### Supervised Splits

Each timestamp has a 2-dimensional Grid of Latitude× Longitude dimensions. Our problem statement is to predict K likely sequences given some previous J sequences. E.g., we take both J and K to be five. This will create a 3D array of dimensions J×Latitude×Longitude as Input and Output(see figure 3.3). We can hyper-tune J and K as per our needs. So a sliding window can be applied over the data to create supervised splits(see table 3.2).

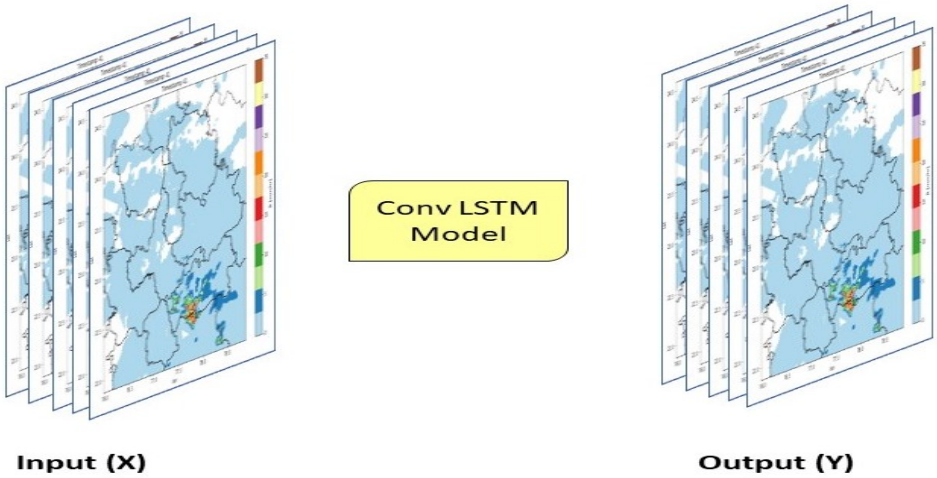



Figure 3.3: Spatiotemporal Supervised Split

### Train-Validation-Test Split

Since we had Precipitation data for four months of JJAS with 20 minutes resolution for 2021, we divided the data of 8459 timestamps as follows, for training first 70%, which is up to 5921 timestamps were separated, for validation, the next 10% from 5922 to 6767 timestamps data was separated, and finally remaining 20% from 6768 to 8459 data was used for testing.

### Model

We have used the ConvLSTM model for spatiotemporal nowcasting. The model is built with Keras with TensorFlow as the backend. We have taken 'mse' as the loss function. "Convolutional LSTM" layers in architecture are used to learn long-term spatiotemporal dependencies in the data. The padding is kept the same for each layer so that the grid size does not change after convolution operations. The convolutional layers in the model enable the model to learn spatial relations in the data. We have used the 'relu' activation function to learn the non-linear relationships in the data. Kernel size of (3,3) is applied for 2D layers and (3,3,3) for 3D layers like `Conv3D` with the stride of one. We have used TensorBoard for 'loss vs epochs' visualization. Since we used only the precipitation variable to predict itself, the number of channels in the input dimensions was set to one. We have used two approaches for nowcasting and created two separate models, Iterative and Direct Nowcasting. We found that increasing `ConvLSTM2D` layers in iterative nowcasting improved the model's performance.

For the iterative model, we have obtained supervised splits with a lag time of 5 timesteps and a lead of 1 timestep. So our model outputs only one lead time, but we have used numpy operations to use lead time one and generated other lead time in total up to five timesteps. For the direct model, we have used `return_sequences = True` hyperparameter of the ConvLSTM layer to output simultaneously generate five lead timesteps. The supervised splits were done accordingly to the model requirements. We used five lag timesteps for direct nowcasting to produce an output of 5 lead times. Both models were trained for 100 epochs. Table 1.5 shows iterative nowcasting model architecture, and Table 1.6 shows direct nowcasting model architecture.

| Layer no. | Layer | Activation | Kernel size | # Filter |
|---|---|---|---|---|
| 1 | `ConvLSTM2D` | `relu` | (3,3) | 4 |
| 2 | `ConvLSTM2D` | `relu` | (3,3) | 8 |
| 3 | `ConvLSTM2D` | `relu` | (3,3) | 8 |
| 4 | `ConvLSTM2D` | `relu` | (3,3) | 16 |
| 5 | `Conv2D` | `relu` | (3,3) | 15 |
| 6 | `Conv2D` | `relu` | (3,3) | 1 |

Table 3.5: Iterative Model Architecture

| Layer no. | Layer | Activation | Kernel size | # Filter |
|---|---|---|---|---|
| 1 | `ConvLSTM2D` | `relu` | (3,3) | 8 |
| 2 | `ConvLSTM2D` | `relu` | (3,3) | 8 |
| 3 | `ConvLSTM2D` | `relu` | (3,3) | 16 |
| 4 | `Conv3D` | `relu` | (3,3,3) | 15 |
| 5 | `Conv3D` | `relu` | (3,3,3) | 1 |

Table 3.6: Direct Model Architecture

| Hyperparameter | Value |
|---|---|
| Input Timesteps | 5 |
| Input size | $5 \times 250 \times 250 \times 1$ |
| Batch Size | 32 |
| Optimizer | Adam( learning rate = $10^{-4}$) |

Table 3.7: Hyperparameters provided during training for Iterative Model

# Chapter 4

# Results and Discussion

## 4.1 Temporal Nowcasting

In this section, we discuss the results for nowcasting of air pollutant $PM_{2.5}$ concentration.

### 4.1.1 Evaluation Metric

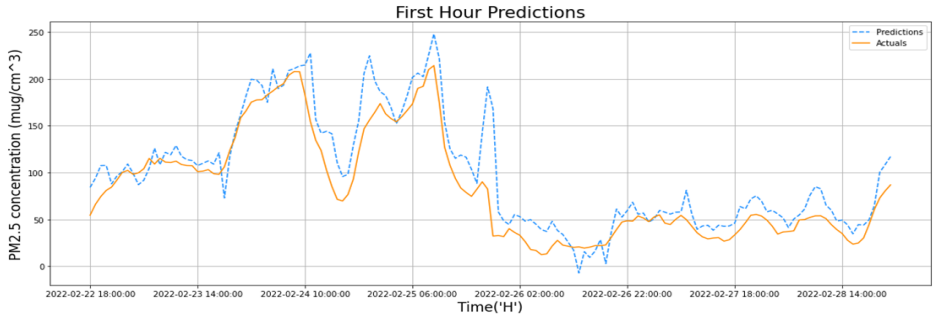
We have used three evaluation metrics, '$R^2$', 'MBE' and 'MAPE'; here, R-squared is a commonly used metric to check the performance of regression models. Its range lies in [0,1], where closer to one signifies a good fit and closer to zero is a bad fit. Mean Bias error(MBE) is used to calculate how much deviation from the true values have from predicted on an average. Mean absolute percentage error gives the total percentage of the prediction error. These metrics give a clear idea of the model's performance.

### 4.1.2 Methods

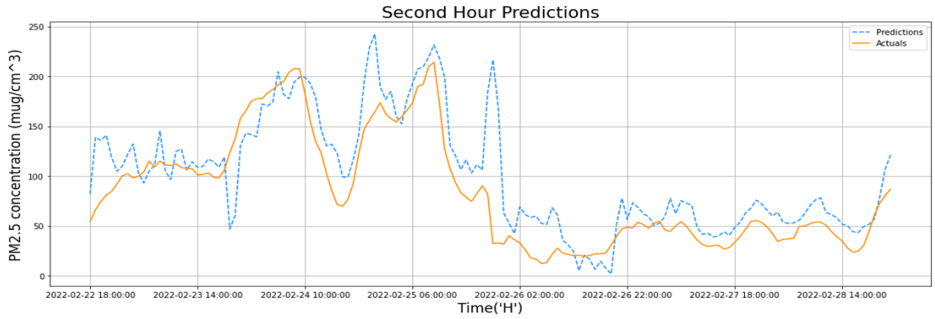
As discussed in the earlier chapter, we have used XGBoost, LSTM, and other models for $PM_{2.5}$ nowcasting. Iterative nowcasting is applied for all the models where lead time one's predictions are used to generate the following lead times predictions.

### 4.1.3 Results

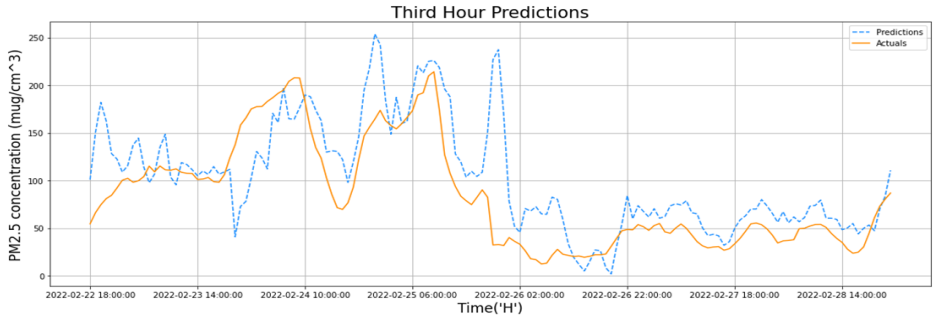
After Data Preprocessing XGBoost, LSTM and other models were tested on the data. The results in the figure below show the ground truth vs predictions for the XGBoost model.




(a) Lead Hour 1, $R^2$: 0.93, MAPE: 17%




(b) Lead Hour 2, $R^2$: 0.89, MAPE: 21%




(c) Lead Hour 3 , $R^2$: 0.83, MAPE: 25%

Figure 4.1: Comparison of forecast of $PM_{2.5}$ concentration with ground truth for different lead hours

From Figures 4.1a, 4.1b and 4.1c, we see that $R^2$ is decreasing; this is due to the fact that the model uses previous hour predictions to generate next-lead time steps. We know that the predictions have errors associated with them, and this error gets again added up for successive predictions. Hence, the same can be seen through MAPE, increasing from 1 to 3. The $R^2$ of 0.93 for the First Hour shows that the model has well-fitted the data; correspondingly, the MAPE gives a lower MAPE of 17%. All of the scores are summarized below in Table 4.1. It shows $R^2$ score of 0.66 for $6^{th}$ Hour.

| **Lead Hour** | **$R^2$ Score** | **MAPE** | **Mean Bias Error** |
|---|---|---|---|
| 1 | 0.93 | 17% | -7.69 |
| 2 | 0.89 | 21% | -8.27 |
| 3 | 0.83 | 25% | -8.68 |
| 4 | 0.77 | 30% | -9.40 |
| 5 | 0.71 | 33.7% | -9.88 |
| 6 | 0.66 | 36.7% | -10.24 |

Table 4.1: XGBoost Forecast Scores for the forecast of $PM_{2.5}$ concentration

The XGBoost model, however, has a negative bias associated with every lead hour prediction (Table 4.1). It can be due to the steps taken in preprocessing the data. The $PM_{2.5}$ data was averaged over forty stations in Delhi NCR, and we know that the average is affected by the outliers in the data, and the XGBoost model is over-predicting them, resulting in overall negative bias. This negative bias can be reduced by handling outliers and checking the mean and standard deviation of the resulting data. Such that cleaner data is given to the model for better performance.

| **Sr.No.** | **Model Name** | **$R^2$ LD1** | **$R^2$ LD6** | **Mean Bias LD1** |
|---|---|---|---|---|
| 1 | XGBoost | 0.93 | 0.66 | -7.69 |
| 2 | LSTM | 0.98 | 0.57 | 3.81 |
| 3 | Random Forest | 0.85 | 0.52 | -14.6 |

Table 4.2: Model Comparison for forecast of $PM_{2.5}$ concentration

From the results above, we see that XGBoost gave better $R^2$ for successive lead hours as compared to other models. All the models take 24 Hours of Lag Time to generate 6 Hours of Lead Time. The LSTM model provides a good

$R^2$ Score for the lead hour one but lower than XGBoost for the lead hour six. The Random forest $R^2$ Scores are lower than the XGBoost and LSTM model; hence we can conclude XGBoost has performed better compared to other models in the study.

## 4.2 Spatiotemporal Precipitation Nowcasting

In this section, we discuss the results of precipitation nowcasting using radar data.

### 4.2.1 Evaluation Metric

We have used 'root mean squared error' and 'correlation' to evaluate the model. We can measure the strength of the linear relationship between two variables using the Pearson coefficient. It is generally used in regression analysis to check for collinearity. It does not depend on the scale of the variables. The range of 'r' lies in [-1,1]. Higher values of r towards 1 or -1 denote highly correlated variables, and the sign indicates the direction of the relationship. E.g., -1 represents a negative correlation, whereas +1 shows a positive correlation. If r = 0, then there does not exist any relationship. Also, from the value of 'r,' we cannot infer the causal relationship between two variables. We have chosen this metric to capture the linear relationship between the two arrays, ground truth and predicted. If the correlation between these two arrays is high, i.e. close towards 1, our model generates correct predictions since the correlation between two identical arrays is 1.

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}} \tag{4.1}$$

In the equation above, $\bar{x}$ and $\bar{y}$ are the mean of arrays x and y, respectively. $x_i$ is value of $i^{th}$ x variable and $y_i$ is value of $i^{th}$ y variable.

### 4.2.2 Iterative vs Direct Prediction

We have compared two models, namely Iterative Model and Direct Model, to decide which is a better approach for precipitation nowcasting. As described earlier in the iterative nowcasting model architecture, we set the hyperparameter in the ConvLSTM layer, `return_sequences = False` to generate

only a single lead time prediction and Lead 1 predictions are used to generate the following lead time predictions. While for Direct nowcasting, we set the hyperparameter in the ConvLSTM layer as `return_sequences = True` to generate five lead time predictions. Direct nowcasting can be applied only if the Input Time-steps (Lag) are the same as Output Time-steps (Lead); this condition has to match the dimensionality of the ground truth array to the prediction array to calculate loss during training. Suppose we have Day 1, Day 2 and so on, up to Day N data of some variable; then the supervised split is created by supplying five lead time steps corresponding to five lag time steps(see table 4.4). At the same time, in the former, we provided only one lead time corresponding to 5 lag time steps.(see table 4.3) Also, note that iterative nowcasting cannot be applied for multivariate spatiotemporal time series as we won't have the data of other variables except the target variable, which our model predicts to generate following lead time predictions.

| **Input (X)** | **Output (Y)** |
|---|---|
| Day 1, . . . , Day 5 | Day 6 |
| Day 2, . . . , Day 6 | Day 7 |
| ⋮ | ⋮ |
| Day N-5, . . . , Day N-1 | Day N |

Table 4.3: Iterative Nowcasting

| **Input (X)** | **Output (Y)** |
|---|---|
| Day 1, . . . , Day 5 | Day 6,. . . , Day 10 |
| Day 2, . . . , Day 6 | Day 7,. . . , Day 11 |
| ⋮ | ⋮ |
| Day N-9, . . . , Day N-5 | Day N-4,. . . , Day N |

Table 4.4: Direct Nowcasting

### 4.2.3 Results

For both models, Iterative and Direct, we have used only one variable, i.e. precipitation, as an input to generate an output of itself using the supervised

splits mentioned in the previous section. Lead times one and so on up to five are generated for both model types. Figures 4.2 and 4.3 show the correlation density plot generated with the corresponding lead time. The temporal correlation coefficient describes the value of correlation corresponding to each latitude×longitude grid value (250×250 in our study).

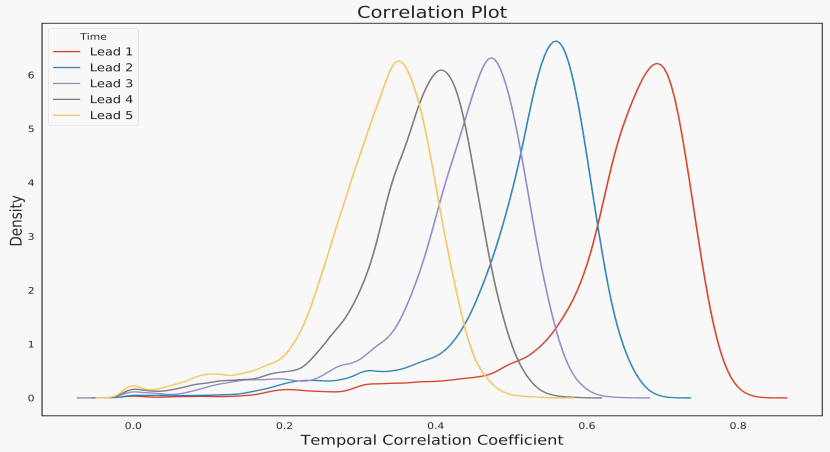



Figure 4.2: Iterative Nowcasting

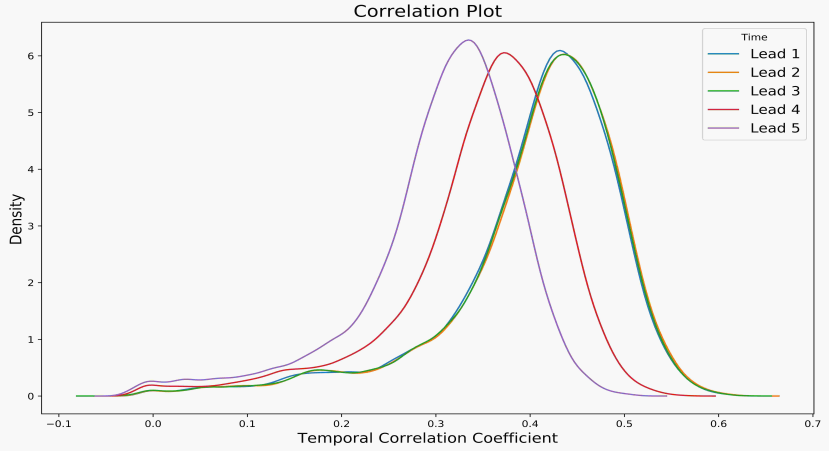



Figure 4.3: Direct Nowcasting

From the figure 4.2 and 4.3, we see that the iterative model has higher correlation values than the direct model. For the lead time one, it shows a

correlation of greater than 0.7 for most of the values. Also, the less spread for the lead one density plot in the iterative model shows a lesser variance than the lead one density plot of the direct model's lead one correlation. For iterative nowcasting, we observe that the correlation coefficient is dropped for consecutive lead times because predictions generated contain errors, which add up to forecasts of the following lead times. For Direct nowcasting, the correlation value for lead one is close to 0.5, shows similar values up to lead three, and falls following lead times.

We conclude from the above two comparisons that the iterative model performs better than the direct model for nowcasting. Although this is true in our case, these results might change from problem to problem as both models have drawbacks; the iterative model can be used for only one variable nowcasting, which is not the case with direct nowcasting, which can be used for multiple variables. Also, the exact lead times are produced in direct nowcasting as the input lag times. At the same time, this is not the case with iterative nowcasting, where lead times can be generated, concatenating the lead one's prediction to the input array.

### Comparison with Baseline

In the previous section, we conclude that iterative nowcasting performs better than direct. We use Persistence as our baseline model for the comparison to validate the model performance. Here in the method of Persistence, the last lag time is taken to be the forecast. Our model uses five lag time steps to produce the next lead time step, so the fifth lag time step is the forecast that will be compared with the lead time one.

Figure 4.4 shows the comparison of Persistence with our Model. Here the temporal correlation values of the model have less standard deviation than the persistence, and the mean of the distribution is also high. Similar statistics are seen for the spatial correlation. Here the spatial correlation is measured for a lat-by-lon grid for a particular time. The temporal correlation for each grid point can be plotted spatially and is compared in figure 4.5. This figure gives a clear idea of which part of the location has high or low values of correlation captured.

We use skill score and rmse to compare the model with persistence further. The defined skill score in literature is `1 - (mse of forecast) / (mse of baseline)`. If the skill score exceeds zero, we can conclude that the model performs better than the baseline. The skill score plotted in Figure 4.6a

shows that the skill score is above zero for most of the values; hence, we can conclude that model performs better than persistence. From the RMSE density 4.6b plot, we see that model's predictions have a low lead1 RMSE than the persistence, plus the standard deviation in the RMSE values is lesser compared to persistence.

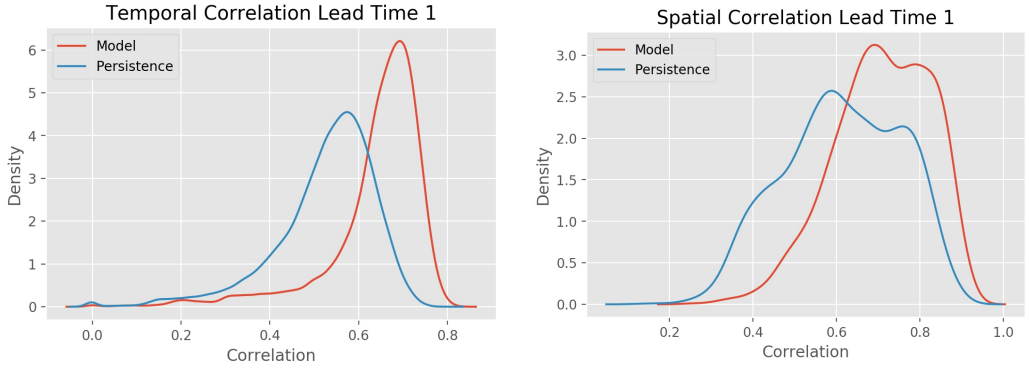



Figure 4.4: Comparison with Baseline Model

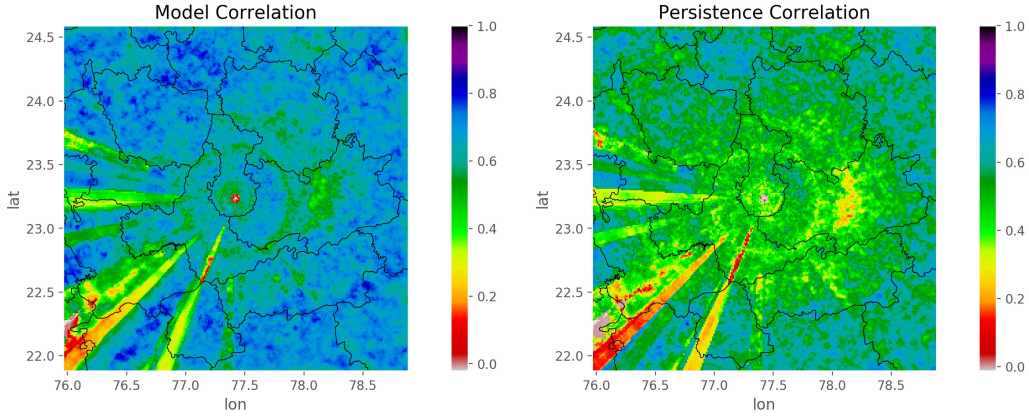
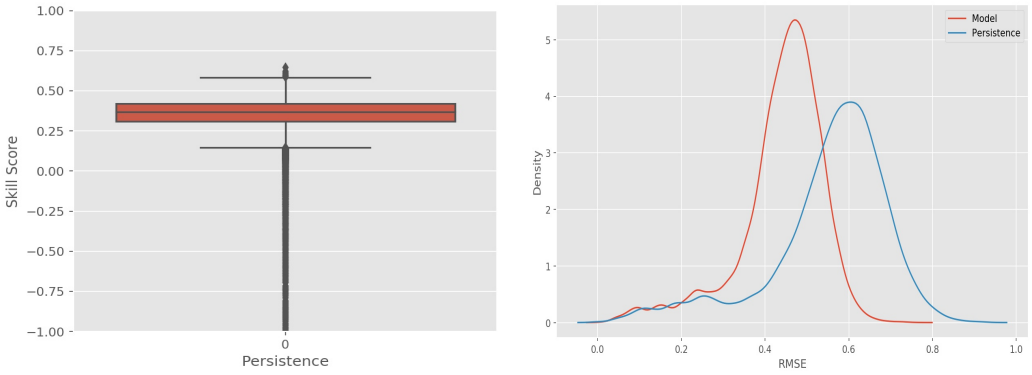



Figure 4.5: Comparison of temporal correlation of precipitation for LD1




(a) Skill Score (b) RMSE Comparison

Figure 4.6: Model vs Persistence

The model's temporal and spatial correlation is higher than persistence; the other metrics, skill score and RMSE, also confirm that the model performs better than persistence. Hence, from this discussion, our model out-competes Persistence.

### Model Performance with Ground Truth Spatial

We compared the model's predictions with some examples in the test set, and the following figures show the same.

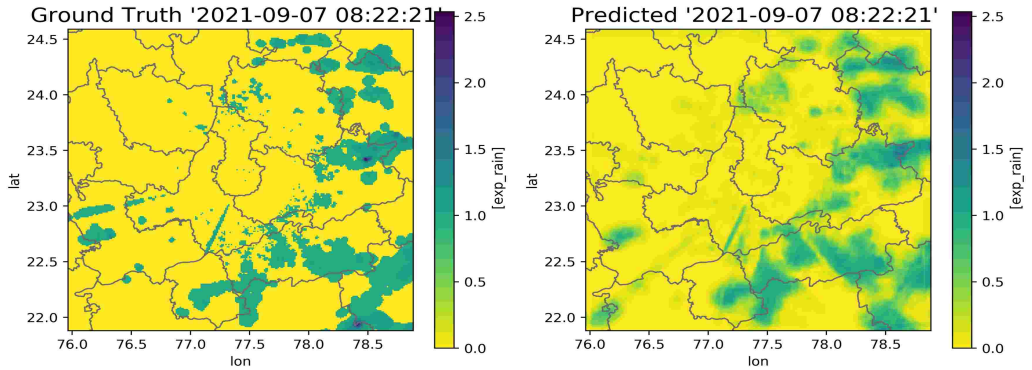



Figure 4.7: Spatial Comparison LD1

From Figure 4.7, we see that the model has correctly predicted most of the spatial patterns in the data; the spatial correlation comes out as 0.7,

meaning the model has learnt to discard zero values in the data and predicted only the values which are non-NaN values. The intensity seems to be under-predicted, but this can be improved using hyperparameter tuning. Hence we can conclude that overall, our model performs well and reasonably accurately predicts the ground truth.

### Model Performance with Ground Truth Temporal

For latitude = 22.0°N and longitude = 76.44°E, we have temporal series in the test set as the ground truth, compared with the model's predictions. Figure

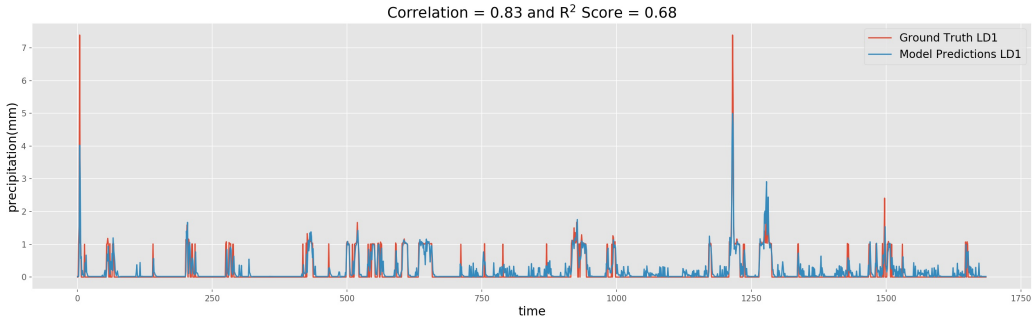



Figure 4.8: Ground Truth vs Predicted for lat = 22.0°N and lon = 76.44°E

4.8 we note that the model can correctly capture the temporal relations in data with up to a reasonable accuracy for Lead time 1. We get a Pearson correlation of 0.83 and the model $R^2$ Score of 0.68, which shows that the model has provided correct predictions for the ground truth in the testing set.
Hence with this discussion above, we conclude that the proposed model gives good performance for spatiotemporal precipitation nowcasting.

# Chapter 5

# Conclusion

We applied XGBoost, LSTM and other models for Temporal Nowcasting of $PM_{2.5}$ concentration. XGBoost performed best among all the models with a decent $R^2$ of 0.66 for the Lead time of the sixth hour. We also developed a deep learning-based ConvLSTM model for precipitation nowcasting up to a lead time of around two hours. The developed model is found to perform better than the persistence. We also compared two methods and found iterative nowcasting to perform better in our case. In iterative nowcasting Temporal Correlation of 0.75 was observed for most of the values on the spatial grid, and for successive lead times, it fell.

## 5.1 Future Work

- For temporal nowcasting US embassy's $PM_{2.5}$ data can be used in place of CPCB data. As we did an average of $PM_{2.5}$ over all the stations in NCR, this step can be avoided, and we get a US embassy's single station continuous data for the model.
- Other meteorological variables can be added and Hyper-parameter tuning can be applied to improve the model's performance.
- Other models like Support Vector Machine and K Nearest Neighbour can be applied to see if these improve the nowcasting.
- For spatiotemporal precipitation nowcasting, the current model takes only one variable as input, i.e. precipitation; its skill can be improved

by using multi-variable input. The data was available only for only the JJAS duration year 2021. More data can enhance the model skill. The model's performance can also be improved by hyperparameter tuning

- . Other models like ConvGRU can be used to compare the accuracy with the ConvLSTM model.

# Bibliography


Adarsh M. The possibilities and limitations of weather forecasting using ai, 2021. URL https://blog.accubits.com/the-possibilities-and-limitations-of-weather-forecasting-using-ai/.

Sebastian Raschka. What are gradient descent and stochastic gradient descent?, 2023. URL https://sebastianraschka.com/faq/docs/gradient-optimization.html.

Ramon Quiza and J Paulo Davim. Computational methods and optimization. *Machining of hard materials*, pages 177–208, 2011.

Niklas Donges. A guide to recurrent neural networks: Understanding rnn and lstm networks, 2023. URL https://builtin.com/data-science/recurrent-neural-networks-and-lstm.

Changjiang Shi, Zhijie Zhang, Wanchang Zhang, Chuanrong Zhang, and Qiang Xu. Learning multiscale temporal–spatial–spectral features via a multipath convolutional lstm neural network for change detection with hyperspectral images. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–16, 2022.

Daniel S Wilks. *Statistical methods in the atmospheric sciences*, volume 100. Academic press, 2011.

George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

Capital One. Understanding arima models for machine learning, 2021. URL https://www.capitalone.com/tech/machine-learning/understanding-arima-models/.

Mark Holmstrom, Dylan Liu, and Christopher Vo. Machine learning applied to weather forecasting. *Meteorol. Appl*, 10:1–5, 2016.

Nitin Singh, Saurabh Chaturvedi, and Shamim Akhter. Weather forecasting using machine learning algorithm. In *2019 International Conference on Signal Processing and Communication (ICSC)*, pages 171–174. IEEE, 2019.

Federico Amato, Fabian Guignard, Sylvain Robert, and Mikhail Kanevski. A novel framework for spatio-temporal prediction of environmental data using deep learning. *Scientific reports*, 10(1):22243, 2020.

Senzhang Wang, Jiannong Cao, and Philip Yu. Deep learning for spatio-temporal data mining: A survey. *IEEE transactions on knowledge and data engineering*, 2020.

Afan Galih Salman, Yaya Heryadi, Edi Abdurahman, and Wayan Suparta. Single layer & multi-layer long short-term memory (lstm) model with intermediate variables for weather forecasting. *Procedia Computer Science*, 135:89–98, 2018.

Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 2015.

Jason Brownlee. How to develop lstm models for time series forecasting, 2020. URL https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Zhen Liu, Luming Shen, Chengyu Yan, Jianshuang Du, Yang Li, and Hui Zhao. Analysis of the influence of precipitation and wind on pm2. 5 and pm10 in the atmosphere. *Advances in Meteorology*, 2020:1–13, 2020.

Liyuan Zhang, Yan Cheng, Yue Zhang, Yuanping He, Zhaolin Gu, Chuck Yu, et al. Impact of air humidity fluctuation on the rise of pm mass concentration based on the high-resolution monitoring data. *Aerosol and Air Quality Research*, 17(2):543–552, 2017.

X Li, YJ Feng, and HY Liang. The impact of meteorological factors on pm2. 5 variations in hong kong. In *IOP Conference Series: Earth and Environmental Science*, volume 78, page 012003. IOP Publishing, 2017.

Bipin Kumar, Namit Abhishek, Rajib Chattopadhyay, Sandeep George, Bhupendra Bahadur Singh, Arya Samanta, BSV Patnaik, Sukhpal Singh Gill, Ravi S Nanjundiah, and Manmeet Singh. Deep learning based short-range forecasting of indian summer monsoon rainfall using earth observation and ground station datasets. *Geocarto International*, pages 1–28, 2022.

Ritesh Kalbande, Bipin Kumar, Sujit Maji, Ravi Yadav, Kaustubh Atey, Devendra Singh Rathore, and Gufran Beig. Machine learning based quantification of voc contribution in surface ozone prediction. *Chemosphere*, page 138474, 2023.

Manmeet Singh, Nachiketa Acharya, Pratiman Patel, Sajad Jamshidi, Zong-Liang Yang, Bipin Kumar, Suryachandra Rao, Sukhpal Singh Gill, Rajib Chattopadhyay, Ravi S Nanjundiah, et al. A modified deep learning weather prediction using cubed sphere for global precipitation. *Frontiers in Climate*, 2023.