Air Quality Analysis and Prediction in Tamil Nadu: AN ANN and CNN approach

Department of Electronics and Communication Engineering

Velammal College of Engineering and Technology (Autonomous), Madurai

Abstract— Air quality is a major environmental and health concern, especially in developing countries like India. Tamil Nadu, a state in southern India, faces severe air pollution problems due to various sources such as vehicular emissions, industrial activities, biomass burning, and dust storms. To monitor and forecast air quality, conventional methods such as ground-based stations and satellite observations have limitations in terms of spatial and temporal coverage, cost, and maintenance. Therefore, there is a need for alternative methods that can leverage the advances in machine learning and big data analytics to provide high-resolution and real-time air quality information. In this paper, we propose a novel approach for air quality analysis and prediction in Tamil Nadu using artificial neural networks (ANNs) and convolutional neural networks (CNNs). These are powerful machine learning techniques that can learn complex nonlinear relationships from large and diverse datasets, and can handle missing and noisy data. We use historical data of air pollutant concentrations, meteorological variables, and land use features from various sources to train and test our models. We compare the performance of our models with existing methods and evaluate their robustness and generalizability. We also provide insights into the spatiotemporal patterns and trends of air quality in Tamil Nadu, and identify the key factors and drivers of air pollution. Our approach can be useful for policy makers, researchers, and citizens to understand and improve the air quality situation in Tamil Nadu and other regions with similar characteristics.

Keywords—Machine Learning, Time Series, Air Quality Index.

I. INTRODUCTION

Air quality is a crucial factor for human health and well- being, as well as for the environment and the economy. In this paper, we propose a novel approach for air quality analysis and prediction in Tamil Nadu using artificial neural networks (ANN) and convolutional neural networks (CNN). ANN and CNN are powerful machine learning techniques that can learn complex nonlinear relationships from large and diverse datasets, and can handle missing and noisy data. In the face of escalating global concerns about air quality and its profound implications for public health, our research aims to contribute significantly by integrating advanced technologies, specifically Convolutional Neural Networks (CNNs) and Artificial Neural Networks (ANNs), into air quality analysis and prediction. The body of literature encompassing our study reflects a rich tapestry of innovative methodologies employed to unravel the intricacies of air pollution dynamics. Studies such as [1] advocate for the deployment of wireless sensor networks, addressing limitations associated with globally trained models. Simultaneously, the integration of CNN- LSTM models, demonstrated in [2], has proven effective in predicting PM10 concentrations. Regional air quality analysis, as exemplified in [3], integrates decision trees and clustering techniques to identify pollution hotspots, while [4] underscores the enhanced predictive capabilities of LSTM over traditional methods. Regression techniques contribute to the precise prediction of air quality in smart cities [5], and models like A3T-GCN showcase excellence in predicting NO2 concentrations [6]. Unconventional approaches, such as leveraging CNNs for PM2.5 estimation from natural images [7], provide noteworthy accuracy. Moreover, localized linear regression studies offer valuable insights into predicting AQI classifications [8]. The exploration extends to studies such as ASIA-AQ's plans for Asian air quality study using satellites [9] and urban air quality data mining and visualization using SARIMAX and Prophet [10]. The severity of air pollution, as discussed in [11], emphasizes the health impacts of fine particulate matter (PM2.5) and proposes a

predictive model using multiple neural networks [12]. In unconventional domains, studies address tool wear prediction for CNC machine tools [13], spacecraft debris removal using Faster R-CNN [14], and comparison of MPPT technologies [15]. The significance of batteries in electric vehicles is addressed, proposing an ANN- based passive cell balancing [16]. SNNs emerge for handling asynchronous data [17], and a novel deep learning model, ST- OR-ResNet, is proposed for air pollution prediction [18]. The framework for predicting air pollution levels unfolds through the integration of CNNs, GRUs, and KELM [19]. Lastly, a hybrid CNN-LSTM model for PM2.5 prediction in Beijing is introduced [20]. Our research synthesizes insights from these diverse studies, proposing a novel integrated framework leveraging CNNs and ANNs for air quality analysis and prediction. Through this endeavor, we aim to contribute meaningfully to the field of air quality analysis, fostering advancements in environmental monitoring and public health.

This paper contains

Section-II METHODOLOGY AND DESIGN Section-III RESULT AND DISCUSSION Section-IV CONCLUSION

II. METHODOLOGY

Analysis and prediction of air quality is a topic that involves the use of various methods and techniques to measure, monitor, and forecast the levels of pollutants in the atmosphere. Air quality is important for human health, environmental protection, and economic development. Some of the common pollutants that affect air quality are particulate matter (PM), ozone (O3), nitrogen dioxide (NO2), sulfur dioxide (SO2), carbon monoxide (CO), and volatile organic compounds (VOCs). There are many sources of data and information on air quality, such as government agencies, research institutions, private companies, and citizen scientists. Some of the data are collected from sensors installed on vehicles, buildings, or satellites. Some of the data are derived from satellite images or ground-based observations. Some of the data are also obtained from historical records or models.

One of the challenges in analyzing and predicting air quality is to deal with the complexity and uncertainty of the atmospheric processes that affect pollutant formation, transport, transformation, and deposition. There are many factors that influence air quality, such as weather conditions, meteorological variables, topography, land use, emission sources, chemical reactions, biological processes, and human activities. These factors can vary over time and space, creating variability and unpredictability in air quality.

To address this challenge, various methods and techniques have been developed to analyze and predict air quality using different types of data and information. Some of these methods are based on machine learning techniques and Deep learning techniques that learn from data using algorithms that can recognize patterns or make predictions. In this paper ANN and CNN algorithms are used.

Data description

The dataset provided contains information related to air quality in the state of Tamil Nadu, India. The key components of the dataset

Sampling Date: The date on which the air quality measurements were taken. It is crucial for understanding temporal variations and trends in air quality.

State: Indicates the state within India where the air quality measurements were recorded. It helps in identifying the geographical location of the air quality monitoring.

City/Town/Village/Area: Specifies the specific location (city, town, village, or area) within the state where the air quality monitoring station is situated. The granularity of location is important for localized analysis and identification of pollution sources.

Location of Monitoring Station: Describes the precise location or address of the monitoring station. This information is crucial for pinpointing the monitoring station and understanding its proximity to potential pollution sources.

Agency: Indicates the organization or agency responsible for conducting the air quality monitoring. Different agencies may follow distinct monitoring protocols, and knowing the responsible agency is essential for assessing data reliability.

Type of Location: Classifies the type of environment where the monitoring station is located (e.g., industrial, residential, traffic). This classification helps in understanding the specific setting and potential sources of pollution.

SO2 (Sulfur Dioxide): Represents the concentration of sulfur dioxide in the air. SO2 is a common air pollutant originating from industrial processes and combustion of fossil fuels. The relation between SO2 and Sampling date is shown in the figure 2.1.

Figure 2.1: Scatter Plot between SO2 and Sampling Date

NO2 (Nitrogen Dioxide): Indicates the concentration of nitrogen dioxide in the air.NO2 is primarily produced by combustion processes, especially in vehicles and industrial

facilities. The relation between SO2 and Sampling date is shown in the figure 2.2.

Figure 2.2: Scatter Plot between NO2 and Sampling Date

RSPM/PM10 (Respirable Suspended Particulate Matter / Particulate Matter less than 10 micrometers): Represents the concentration of particulate matter in the air with a diameter of 10 micrometers or less.PM10 includes particles that can be inhaled into the respiratory system and have various sources, including vehicle emissions and industrial activities. The relation between RSPM/PM10 and Sampling date is shown in the figure 2.3.

Figure 2.3: Scatter Plot between RSPM/PM10 and Sampling Date

PM 2.5 (Particulate Matter less than 2.5 micrometers): Indicates the concentration of fine particulate matter with a diameter of 2.5 micrometers or less.PM2.5 is associated with more severe health impacts as these particles can penetrate deeper into the lungs. The relation between PM 2.5 and Sampling date is shown in the figure 2.4.

Figure 2.4: Scatter Plot between PM 2.5 and Sampling Date

AQI (Air Quality Index):

The Air Quality Index (AQI) is a numerical scale used to communicate the overall quality of air in a specific location. It is calculated based on the concentrations of various air pollutants, including SO2, NO2,

RSPM/PM10, and PM 2.5. The AQI provides a standardized way to convey the potential health risks associated with the observed pollutant levels.

AQI = SO2 + NO2 + RSPM_PM10 + PM 2.5

where,

SO2 - (Sulfur Dioxide) NO2 - (Nitrogen Dioxide)

RSPM/PM10 - (Respirable Suspended Particulate Matter /Particulate Matter less than 10 micrometers) PM 2.5 - (Particulate Matter less than 2.5 micrometers)

Figure 2.5: AQI vs Sampling Date

The AQI is typically categorized into different air quality levels, ranging from "Good" to "Hazardous," with corresponding health advisories. It serves as a valuable tool for policymakers, researchers, and the general public to assess and communicate the impact of air pollution on human health. The table 2.1 explains the calculation or classification of AQI.

AQI calculation:

1, Calculate Sub-Indices: For each pollutant, calculate its sub-index using the concentration and breakpoints specific to that pollutant.

2, Determine AQI: The overall AQI is then determined by the maximum sub-index among all pollutants.

3, Mapping to Health Categories: The AQI is typically mapped to health categories (e.g., Good, Moderate, Unhealthy) to make it more interpretable for the public.

Figure 2.6 describes the dataset. The dataset is utilized to analyze trends, assess the impact of different pollutants, and contribute to a deeper understanding of air quality in Tamil Nadu. Additionally, by incorporating AQI values, you can provide a comprehensive evaluation of overall air quality conditions and their potential health implication

Table 2.1: AQI calculation

Figure 2.6: Data Set

A. Algorithm used

The algorithm used in this paper is CNN and ANN, let us first get a detailed information about the algorithm,

CNN (Convolution Neural Network):

CNN is a type of neural network designed for processing structured grid data, such as images. It uses convolutional layers to learn spatial hierarchies of features automatically and adaptively.

Training and Prediction:

Train the CNN using historical air quality data and deploy it for predicting future air quality. Weight Update in Backpropagation:

Figure 2.8: CNN Architecture

ANN (Artificial neural Network):

- **Definition:** ANN is a computational model inspired by the structure and function of the human brain. It consists of interconnected nodes (neurons) organized into layers, including an input layer, one or more hidden layers, and an output layer.

- **Fully Connected Layers:** Connect the output of convolutional layers to fully connected layers for prediction.

- Process :

Figure 2.9: Structure of ANN

- Data Preparation: Collect and preprocess air quality data, including features like pollutant concentrations, meteorological conditions, and other relevant factors.

$$Normalized\ Value = Original\ Value - Min\ Value$$

$$Max\ Value - Min\ Value$$

Normalize the features to a common scale, which helps in training the neural network.

- Network Architecture: Design the ANN architecture, specifying the number of layers, neurons per layer, and activation functions.

No of Parameters in a Layer =

(No of Neurons in Previous Layer + 1) × No of Neurons in Current Layer

The formula calculates the number of parameters (weights and biases) in a fully connected layer of the neural network.

- Training: Use historical air quality data to train the ANN. The network learns to map input features to the corresponding air quality output.

New Weight = Old Weight − Learning Rate × ∂Loss

∂Old Weight

Adjust the weights during training using gradient descent to minimize the loss function.

- Validation and Testing: Evaluate the trained model on validation data to ensure generalization. Test the model on unseen data to assess its predictive performance.

Mean Squared Error for Regression:

N

MSE = $N1$ ∑(Actual$i$ − Predicted$i$)2

i=1

Accuracy for Classification

Accuracy= (Number of Correct Predictions)/(Total Number of Predictions)

Use appropriate evaluation metrics like Mean Squared Error for regression tasks and accuracy for classification tasks to assess model performance.

- Prediction: Deploy the trained ANN to predict air quality based on new input data.

M

Prediction Output = Activation(∑(Weight i × Input i) + Bias)

i=1

Apply the trained model to new input data to obtain predictions

LOAD DATASET:

data=pd.read_csv('D:\Kaggle_India_Air_Quality_Data_ For_TamilNadu.csv')

pd.read_csv: This is a function provided by the Pandas library to read data from a CSV file. It creates a Data Frame, a two- dimensional tabular data structure, from the content of the CSV file.

data = : The result of the `pd.read_csv` operation is assigned to the variable `data`. This variable now holds a Pandas Data Frame containing the data from the CSV file.

CALCULATE AQI:

The function `calculate_aqi` is used to create a new column 'AQI' in the Pandas Data Frame named 'data'. This is a Python function named `calculate_aqi` that takes four parameters: `so2`, `no2`,

`rspm_pm10`, and `pm25`.

- It calculates the Air Quality Index (AQI) by summing up the values of these parameters.

-The calculated AQI is then returned by the function.

Applying the Function to Create 'AQI' Column:

-`data['AQI']`: This creates a new column named 'AQI' in the 'data' Data Frame to store the calculated AQI values.

-`data.apply(...)`: This is used to apply a function along the axis of the DataFrame. In this case, the function being applied is a lambda function.

-`lambda row: calculate_aqi(row['SO2'], row['NO2'], row['RSPM/PM10'], row['PM 2.5'])`: This lambda function takes a row of the Data Frame (`row`) and extracts the values of 'SO2', 'NO2', 'RSPM/PM10', and 'PM 2.5' from that row. It then calls the

`calculate_aqi` function with these values.

- `axis=1`: Specifies that the lambda function should be applied along the rows (i.e., for each row of the Data Frame).

HANDLE MISSING VALUES:

Subsetting Columns:

- The next line creates a new Data Frame called `numeric_data` by selecting specific columns from the original Data Frame (`data`).
- The selected columns are 'AQI', 'SO2', 'NO2', 'RSPM/PM10', and 'PM 2.5'.

Mean = Σxi

PREPARE FEATURE AND TARGET:

Feature Matrix (`X`):

- Here, you are selecting a subset of columns ('SO2', 'NO2', 'RSPM/PM10', 'PM 2.5') from the DataFrame `numeric_data` to create the feature matrix `X`. This matrix contains the input features that will be used to predict the target variable.

Target Variable Vector (`y`):

- The `apply` function is used to apply the `calculate_aqi` function to each row of the `numeric_data` DataFrame along the specified axis (axis=1 indicates that the function should be applied to rows).

- The `calculate_aqi` function takes the values from the columns

`'SO2'`, `'NO2'`, `'RSPM/PM10'`, and `'PM 2.5'` in each row and calculates the Air Quality Index (AQI) using the formula `so2 + no2 + rspm_pm10 + pm25`.

- The resulting values form the target variable `y`, which represents the AQI for each corresponding row in the dataset.

In summary, the feature matrix `X` is defined with columns related to air quality parameters and calculated the target variable

`y` representing the AQI based on these parameters. This data will be used for training and evaluating a predictive model.

DATA PREPROCESSING:

Convert 'Sampling Date' to DateTime Format:

- This line converts the 'Sampling Date' column in the DataFrame to a datetime format. The `pd.to_datetime` function is used for this purpose. The `format='%d-%m-%Y'` parameter specifies the expected date format in the column, where `%d` represents the day, `%m` represents the month, and `%Y`

Where,

n

Xi = Values in each column

N = Number of the values in each column

represents the year.

Drop Rows with Missing Values in 'Sampling Date':

- The `dropna` method is used to remove rows where the

Filling NaN Values with Mean: numeric_data.fillna(numeric_data.mean(), inplace=True)

- This line fills missing (NaN) values in the `numeric_data` DataFrame with the mean value of each respective column.

- The `fillna` method is used for this purpose, and it takes the mean value of each column as an argument (`numeric_data.mean()`).

- The `inplace=True` parameter ensures that the changes are applied directly to the original DataFrame (`numeric_data`), and it returns `None` to indicate that no new DataFrame is created.

In summary, the code is creating a subset of the original DataFrame containing only numeric columns of interest (`'AQI', 'SO2', 'NO2', 'RSPM/PM10', 'PM 2.5'`) and then filling any missing values in this subset with the mean value of each respective column. This is a common data preprocessing step to handle missing data before further analysis or modeling.

Figure 2.10: Histogram between Health rate and Count

'Sampling Date' column has missing values. This ensures that only rows with valid date information are retained.

Set 'Sampling Date' as the Index:

- The 'Sampling Date' column is set as the index of the DataFrame using `set_index`. This is a common practice when dealing with time series data, as it allows for convenient time- based indexing.

Convert Numeric Columns to Numeric Type:

- The `apply` method, along with `pd.to_numeric`, is used to convert all columns in the DataFrame to numeric type. The

`errors='coerce'` parameter ensures that if any value cannot be converted to numeric, it is replaced with NaN.

Drop Rows with Missing Values in Numeric Columns:

- Rows with missing values in the numeric columns (previously selected in `numeric_data`) are dropped using `dropna`. This step is performed after converting columns to numeric to ensure that only rows with complete numeric information are retained.

Resample Data to Monthly Frequency, Taking Mean of AQI for Each Month:

- The `resample` method is used to resample the data to monthly frequency ('M'). The `mean` function is applied to calculate the mean of each month. This step is beneficial when dealing with time series data, as it helps in summarizing the data on a monthly basis, smoothing out short-term fluctuations.

These steps collectively prepare the data for further analysis, ensuring that the date columns are in the correct format, missing

values are handled appropriately, and the data is aggregated to a monthly frequency for better temporal analysis.

BUILDING A MODEL:

Model Training:

The model is trained using the `fit` method, which trains the model for a fixed number of epochs (iterations on a dataset). It takes as input the training data and labels, the number of epochs, the batch size,

and the validation split. The validation split fraction of the training data will be used as validation data. The model will not be trained on this data. This method returns a `History` object, which is a record of training loss values and metrics values at successive epochs, as well as validation loss values and validation metrics values.

Building the CNN Model:

The CNN model is built using the Keras functional API. The Input function is used to instantiate a Keras tensor. A Keras tensor is a symbolic tensor-like object, which we augment with certain attributes that allow us to build a Keras model just by knowing the inputs and outputs of the model. For instance, if `a` and `b` are Keras tensors, it becomes possible to do: `model = Model(input=[a, b], output=c)`. The Conv2D function creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. The `Flatten` function flattens the input without affecting the batch size. The `Dense` function implements the operation: `output = activation(dot(input, kernel)+ bias)` where `activation` is the element-wise activation function passed as the `activation` argument, `kernel` is a weights matrix created by the layer, and `bias` is a bias vector created by the layer. The input layer for the Convolutional Neural Network (CNN) is defined. It specifies that the input is a 2x2 grid with one channel (assumed based on reshaping X_2d). Then the addition a 2D convolutional layer with 32 filters and a 2x2 kernel to the CNN. The ReLU activation function is applied to the output. The Flatten layer is used to flatten the output of the convolutional layer into a one-dimensional array. This step is necessary before connecting to the dense layers. Then a fully connected (dense) layer with 64 neurons is incorporated and applies the ReLU activation function is to the CNN. The final dense layer with one neuron and a linear activation function is added. This layer produces the output for the CNN.

Building the ANN Model:

The ANN model is built similarly to the CNN model but without convolution and flattening layers. The input layer for the Artificial Neural Network (ANN) is defined. The shape is determined by the number of features in X_train_scaled. Then a fully connected layer with 64 neurons and ReLU activation to the ANN was added. Adds another fully connected layer with 32 neurons and ReLU activation to the ANN. The final dense layer with one neuron and a linear activation function is added. This layer produces the output for the ANN.

Combining the CNN and ANN Models:

The outputs of the CNN and ANN models are concatenated together. This combined model is then passed through two more dense layers. The outputs of the CNN and ANN models are concatenated into a single vector. A fully connected layer is concatenated with 64 neurons and ReLU activation to the combined model, Where the ReLU activation function helps introduce non-linearity in neural networks, enabling them to learn complex patterns. It also prevents the vanishing gradient problem and 64 neurons in a layer strike a balance between learning capacity and computational efficiency, allowing the model to capture detailed patterns without excessive complexity. The final dense layer with one neuron and a linear activation function is added. This layer produces the final output of the combined model.

Compiling the Model:

The model is compiled with the Adam optimizer which efficiently, combines the benefits of momentum and adaptive learning rates, and the mean squared error loss function which measures the squared difference between predicted and actual values, penalizing larger errors more heavily. The `compile` method configures the model for training.

Training the Combined Model:

The combined model is trained in the same way as the initial model. The Keras functional API is used to build and train a combined Convolutional Neural Network (CNN) and Artificial Neural Network (ANN) model. The CNN extracts spatial features, while the ANN learns non-spatial features. Their outputs are combined and passed through dense layers for the final prediction. The model is trained with the Adam optimizer and mean squared error loss for regression tasks. Performance is evaluated on a validation set to monitor unseen data during training and prevent overfitting.

Hyperparameters like epochs and batch size impact performance and require tuning based on the task and resources. The Keras functional API allows flexibility in constructing models with multiple inputs and outputs. Functions like `Input`, `Conv2D`, `Flatten`, and `Dense` create layers. The `concatenate` function combines CNN and ANN outputs. The `Model` function instantiates the Keras model. `compile` configures it, and `fit` trains it, providing a `History` object for performance tracking.

Keras Model:

The `Model` function is used to instantiate a Keras model given its inputs and outputs. This creates a model that includes all layers required in the computation of `outputs` given `inputs`.

Compiling the Model:

The `compile` method configures the model for training by setting the optimizer, loss function, and metrics. The optimizer determines how the network will be updated based on the loss function. It implements a specific variant of stochastic gradient descent (SGD). The loss function, or objective function, is the function that will be minimized by the optimizer. For this model, we use 'adam' as the optimizer and 'mean_squared_error' as the loss function.

Training the Combined Model:

The combined model is trained in the same way as the initial model. The `fit` function trains the model for a fixed number of epochs (iterations on a dataset). It takes as input the training data and labels, the number of epochs, the batch size, and the validation split. The validation split fraction of the training data will be used as validation data. The model will not be trained on this data. This method returns a `History` object, which is a record of training loss values and metrics values at successive epochs, as well as validation loss values and validation metrics values. This code showcases the creation of a sophisticated model by combining Convolutional Neural Network (CNN) and Artificial Neural Network (ANN) components using the Keras functional API. The CNN is adept at capturing spatial features, while the ANN excels in understanding non-spatial features. The outputs from both models undergo further processing through dense layers to generate the final predictions. This strategy enables the model to effectively learn both spatial and non-spatial characteristics from the data, enhancing performance on intricate tasks.

Figure 2.11: Model Training of CNN

For training, the model employs the Adam optimizer and the mean squared error loss function, particularly suitable for regression tasks. Evaluation is conducted on a validation set, a subset of the training

data unseen during model training. This monitoring process aids in preventing overfitting and ensures the model's adaptability to new, unseen data.

The hyperparameters, including the number of epochs and batch size, are pivotal for optimizing model performance. The number of epochs signifies how many times the model undergoes training on the complete dataset, while the batch size determines the number of samples the model processes at each iteration. Balancing these parameters is crucial, as a higher number of epochs may lead to better performance but raise the risk of overfitting and require more computational resources. Similarly, a larger batch size accelerates training but may compromise gradient accuracy, leading to slower convergence. The Keras functional API is used for building intricate models with multiple inputs and outputs. Key functions such as `Input`, `Conv2D`, `Flatten`, `Dense`, `concatenate`, `Model`, `compile`, and `fit` are utilized. The `History` object is employed to record training loss and metric values across epochs, along with validation loss and metrics if applicable. This comprehensive approach highlights the flexibility of the Keras functional API in constructing complex neural network architectures.

Validation Split:

The validation split is the fraction of the training data that is used as validation data. A larger validation split provides a more accurate estimate of the model's performance on unseen data but reduces the amount of data the model is trained on. The value of the validation split is set as 0.2.

Batch Size:

The batch size is the number of samples the model is trained on at a time. A larger batch size can speed up training, but can also lead to less accurate estimates of the gradient and slower. Here batch size is considered as 32.

Number of Epochs:

The number of epochs is the number of times the model is trained on the entire training dataset. A larger number of epochs can lead to better performance, but also increases the risk of overfitting and requires more computational resources. The epoch's value is 400.

Using the Keras functional API, the construction and training of a combined Convolutional Neural Network (CNN) and Artificial Neural Network(ANN) model. The CNN captures spatial features, while the ANN focuses on non-spatial features. Their outputs are merged through additional dense layers for the final prediction, enhancing the model's performance on complex tasks. The training utilizes the Adam optimizer and mean squared error loss function, suitable for regression tasks. Performance evaluation occurs on a validation set, preventing overfitting by monitoring the model's performance on unseen data. Tunable hyperparameters include the number of epochs and batch size, impacting model performance and computational requirements. A larger validation split provides a more accurate estimate of the model's performance on unseen data but reduces training data.

The Keras functional API allows for flexibility, facilitating the creation of complex models with multiple inputs and outputs. The `History` object records training and validation metrics across epochs. This comprehensive approach demonstrates the API's adaptability in constructing advanced neural network architectures.

In conclusion, this approach enables the model to learn both spatial and non-spatial features, enhancing performance on complex tasks. Training involves the Adam optimizer and mean squared error loss function, with performance evaluated on a validation set to prevent overfitting. Hyperparameters such as the number of epochs and batch size are tunable, impacting performance and computational resources. A larger validation split provides a more accurate estimate of the model's performance on unseen data but reduces training data. The Keras functional API is showcased for creating complex models with multiple inputs and outputs. This demonstration highlights the flexibility of the Keras functional API in constructing advanced neural network architectures, combining both CNN and ANN for improved learning capabilities.

MODEL EVALUATION AND PREDICTION:

The prediction of the output of test data using the trained model. The prediction method in Keras is used to generate output predictions for the input samples. The input to this function is a list containing two elements: X_test and X_test_scaled. These are the test data that the model has not seen during training.

The calculation of the Mean Squared Error (MSE) between the true output values for the test data and the predicted output values. MSE is a common loss function for regression problems and it measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value.

The calculation of the Mean Absolute Error (MAE) between the true output values for the test data (`y_test`) and the predicted output values (`y_pred`). MAE is another common loss function for regression problems and it measures the average of the absolute differences between the estimated values and the actual value. It's less sensitive to outliers compared to MSE.

The calculation of the R-squared (Coefficient of Determination) regression score function. This score function computes the coefficient of determination of predictions for true data (`y_test`) and predicted data (`y_pred`). It provides a measure of how well future samples are likely to be predicted by the model. The best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of `y`, disregarding the input features, would get an R^2 score of 0.0.

These metrics (MSE, MAE, and R^2) are commonly used to evaluate the performance of regression models. They provide different perspectives on the performance of the model, and looking at all of them together can give a more complete picture of the model's performance. For example, a model with a low MSE and a high R^2 score is generally considered a good model. However, it's also important to consider the specific context and requirements of the task when evaluating a model's performance. For instance, in some cases, it might be more important to minimize the number of large errors (which would be reflected in a lower MSE), while in other cases, it might be more important to get the overall trend right (which would be reflected in a higher R^2 score).

DATA ANALYSIS AND VISUALIZATION:

The scatter plot is created for visualization where the x-axis represents the actual AQI values (y_test), and the y-axis represents the predicted AQI values (y_pred). The alpha = 0.5 parameter controls the

transparency of the points, making it easier to observe overlapping points. Figure 2.8 gives the model's performance by plotting the actual AQI values against the predicted AQI values.

Figure 2.12: Scatter Plot between Actual AQI and Predicted AQI

Figure 2.9 shows how well a model can predict the AQI (Air Quality Index) of different locations based on some features. The x-axis is the actual AQI measured by sensors, and the y-axis is the predicted AQI calculated by the model. The closer the data points are to the line of best fit, the more accurate the predictions are. The graph indicates that the model is fairly good at predicting low to moderate AQI values, but it tends to underestimate high AQI values.

VISUALIZATION:

A violin plot is used for visualization, it is used for AQI vs Type of location because it can show how the air quality varies across different locations and how they are related to each other and a violin plot can show if there are any clusters or groups of locations with similar AQI values, or if there are any outliers or extreme values in the data. A violin plot can also show if there is any correlation between AQI and other factors, such as weather, geography, or population density.

A violin plot is a type of statistical graph that shows the frequency or count of each value using a violin-shaped curve. The length of the curve represents the frequency or count of each value, and the position of the curve relative to the mean represents the relative frequency or proportion.

Figure 2.13: Violin Plot between City/ Area and AQI

This graph shows how the air quality index (AQI) varies across different locations in Tamil Nadu, India. AQI is a number that indicates how polluted the air is and what health effects it may have on people. The graph uses a violin plot to show the distribution of AQI values for each location.

Figure 2.14: Violin Plot between Type of Location and AQI

The figure shows that industrial areas have a higher and more concentrated level of pollution compared to residential, rural, and other areas. The plot for the industrial area is taller and narrower, indicating a higher concentration of pollution levels. The plot for residential, rural, and other areas is shorter and wider, indicating lower but more varied pollution levels.

Figure 2.13: Violin Plot between Agency and AQI

III. RESULT AND DESCRIPTION

CORRELATION HEATMAP AND RESIDUALS:

A correlation heatmap is a graphical tool that displays the correlation between multiple variables as a color-coded matrix1. It’s like a color chart that shows us how closely related different variables are. In a

correlation heatmap, each variable is represented by a row and a column, and the cells show the correlation between them. The color of each cell represents the strength and direction of the correlation, with darker colors indicating stronger correlations1.

Understand correlation heatmap:

Look at the color of each cell to see the strength and direction of the correlation1. Darker colors indicate stronger correlations, while lighter colors indicate weaker correlations1.

Positive correlations (when one variable increases, the other variable tends to increase) are usually represented by warm colors, such as red or orange1.

Negative correlations (when one variable increases, the other variable tends to decrease) are usually represented by cool colors, such as blue or green1.

Understanding correlation heatmaps can help us identify patterns and relationships between multiple variables1. So next time you analyze data with many variables, think like an artist and use a correlation heatmap to see the colors of the relationships. (width=10, height=8) - Specifies the size of the figure for the heatmap.

A heatmap is created using Seaborn (`sns`). The heatmap represents the correlation matrix of the numeric columns in the `numeric_data` DataFrame. The

`annot=True` parameter adds numerical annotations to the heatmap cells. The color map (`cmap`) is set to 'coolwarm', and the format (`fmt`) for the annotations is '.2f'. The `linewidths` parameter controls the width of the lines between cells. Figure 2.8 gives the correlation matrix between dates.

Figure 3.1: Correlation Heatmap

The correlation heatmap shows how different air quality indices are related to each other. A correlation coefficient is a number between -1 and 1 that measures how strongly two variables are associated. A positive correlation means that the variables tend to increase or decrease together, while a negative correlation means that they tend to move in opposite directions. A correlation close to 0 means that there is little or no relationship between the variables. From Figure 2.10 the air quality index (AQI) is most strongly correlated with PM 2.5, which is a measure of fine particulate matter in the air. This means that as PM 2.5 levels increase, the AQI also increases, indicating worse air quality. The AQI is also moderately correlated with RSPM/PM10, which is another measure of particulate matter, but less correlated with SO2 and NO2, which are gases that contribute to air pollution. The image also shows that SO2 and NO2 have very low correlations with each other and with the other variables, meaning that they vary independently of each other and of the particulate matter levels.

A residual is the difference between an observed value and a predicted value in a model. It is calculated as Residual = Observed value - Predicted value.

Residuals are used to understand the discrepancy between the model prediction and the actual data. Analyzing residuals can help diagnose whether the assumptions of a statistical model have been met and guide improvements to the model. A residual plot is a graphical tool that is used to assess the residuals (errors) of a regression model. Residuals are the differences between the observed and predicted values.

In a residual plot:

The x-axis typically represents the predicted values or the independent variables.

The y-axis represents the residuals.

Random scatter: If the points are randomly scattered around the horizontal axis, this suggests that the model's errors are random, and the model is probably a good fit for the data.

Figure 3.2: Residual Plot

A good model should have residuals that are close to zero and randomly distributed. The image shows that the model has mostly small residuals, but there is one outlier that has a large negative residual. A scatter plot is created where the x-axis represents the actual AQI values (`y_test`), and the y-axis represents the residuals (the differences between actual and predicted AQI values). The transparency is set as alpha = 0.5, Added a horizontal line at y=0 (the x-axis) in red ('r') with a dashed line style ('--'). This type of plot helps you assess how well your model's predictions match the actual values and identify patterns or trends in the residuals. The distribution of residuals refers to the spread of the residuals (errors) in a regression model. Residuals are the differences between the observed and predicted values.One of the key assumptions of linear regression is that the residuals are normally distributed1. This assumption can be checked by creating a Q-Q plot, which is a type of plot that can be used to determine whether or not the residuals of a model follow a normal distribution. If the residuals are normally distributed, they would form a bell-shaped curve when plotted1. This is because the normal distribution is symmetric and bell-shaped. However, suppose the residuals are not normally distributed. In that case, they might show patterns such as skewness (if the residuals are skewed to the left or right), kurtosis (if the residuals have heavy tails or light tails compared to a normal distribution), or heteroscedasticity (if the variance of the residuals is not constant across all levels of the independent variables). Analyzing the distribution of residuals can help diagnose whether the assumptions of a statistical model have been met and guide improvements to the model.

Funnel shape (Heteroscedasticity): If the residuals form a funnel shape with the spread of residuals increasing with the predicted value, this suggests heteroscedasticity, i.e., the variability of the error changes across levels of the independent variable. The residuals should be normally distributed with a mean of zero and a small standard deviation, indicating that the model fits the data well and has low error. The figure 2.12 shows that most of the residuals are around -2.5, which means that the model tends to overestimate the actual values by a small amount. The graph also shows that there are very few residuals that are far from the mean, which means that there are no outliers or extreme values in the data.

AQI CATEGORIES:

Categorizing Actual AQI Values:

`y_test` contains the actual AQI values. The `apply` method is used to apply the `classify_aqi` function to each element in `y_test`. The `classify_aqi` function categorizes AQI values into predefined categories ('Good', 'Moderate', 'Satisfactory', 'Poor', 'Very Poor', and 'Severe').

Categorizing Predicted AQI Values:

`y_pred` contains the predicted AQI values. `y_pred.flatten()` is used to flatten the array, and then the `apply` method applies the `classify_aqi` function to each element. This step categorizes the predicted AQI values into the same categories as the actual values.

Creating a DataFrame for Comparison:

By comparing 'Actual' and 'Predicted', a new data set is created. The 'Actual' column contains the categories of the actual AQI values (`y_test_categories`), and the 'Predicted' column contains the categories of the predicted AQI values (`y_pred_categories`).

Printing the Comparison DataFrame:

printing the data frame that shows the actual and predicted AQI categories side by side for comparison. The purpose of these steps is to evaluate how well the model is performing in terms of categorizing AQI values. The `classify_aqi` function defines categories based on the AQI values, and the comparison DataFrame allows you to visually inspect where the model predictions align or differ from the actual categories. This is useful for understanding the model's performance in classifying air quality.

Output is finalized by:

Mean Squared Error (MSE): This is a measure of how close the predicted values of a regression model are to the actual values of the target variable. It is calculated by taking the average of the squared differences between the predicted and actual values. The formula for MSE is:

$$MSE = \frac{1}{n}\sum(y - \hat{y})^2$$

Figure 3.3: Distribution of Residuals

$$\sum_{i=1}^{n}$$

where n is the number of observations, yi is the actual value, and is $\hat{y}i$ the predicted value. The lower the MSE, the better the model fits the data. However, MSE can be sensitive to outliers and large errors, so it may not be a good indicator of model performance if there are many extreme values in the data.

Mean Squared Error (MSE): 0.7758493326851073

- MSE is a measure of the average squared difference between the actual and predicted values.
- In this case, the MSE is 0.776, which is relatively low. Lower MSE values indicate that, on average, the model's predictions are close to the actual values.

Mean Absolute Error (MAE):

This is another measure of how close the predicted values of a regression model are to the actual values of the target variable. It is calculated by taking the average of the absolute differences between the predicted and actual values. The formula for MAE is:

our data.

R-squared (R2): 0.9998385590491591

- R-squared is a measure of how well the model's predictions match the actual data.
- R2 value of 0.9998 is very close to 1, suggesting that the model explains a very high proportion of the variance in the target variable. This indicates an excellent fit.

Root Mean Squared Error (RMSE):

The Root Mean Squared Error is a measure that provides the standard deviation of the residuals, which are the differences between the actual and predicted values. It is an extension of the Mean Squared Error (MSE) and is calculated by taking the square root of the MSE. The formula for RMSE is:

n

1

$RMSE = \sqrt{\quad} \sum(yi - \hat{y}i)2$

n

i=1

1 n where n is the number of observations, yi is the actual value, and $\hat{y}i$

$MAE = n \sum | yi - \hat{y}i |$

i=1

where n is the number of observations, yi is the actual value, and y^i is the predicted value.

The lower the MAE, the better the model fits the data. Unlike MSE, MAE does not take into account whether errors are positive or negative, so it gives equal weight to all errors regardless of their direction. This makes MAE more robust to outliers and less biased than MSE.

Mean Absolute Error (MAE): 0.03715586600411847

- MAE is a measure of the average absolute difference between the actual and predicted values.
- The MAE is 0.037, indicating that, on average, the absolute difference between the actual and predicted values is very small.

R-squared (R2): This is a measure of how well a regression model explains or predicts variation in a target variable. It ranges from 0 to 1, where 0 means that none of the variation in y can be explained by x, and 1 means that all of the variation in y can be explained by x. The formula for R2 depends on whether you use least squares or maximum likelihood estimation to fit your model. For least squares estimation, R2 can be calculated as:

$\sum n \quad (y - \hat{y})2$

is the predicted value.

The RMSE is particularly useful because it provides an interpretable scale that is in the same units as the target variable.

Root Mean Squared Error (MSE): 0.4522174134442851

It indicates the standard deviation of the residuals. A lower RMSE suggests that, on average, the model's predictions deviate from the actual values by around 0.452 units. In the context of your regression model, this is a relatively good performance, indicating that the model's predictions are close to the actual values. It's essential to compare the RMSE to the scale of your target variable to assess its practical significance.

Actual vs. Predicted Categories DataFrame:

- The DataFrame shows a comparison between the actual and predicted AQI categories for the first 100 instances.
- Each row represents a data point, and the "Actual" and "Predicted" columns show the corresponding AQI categories.
- It seems like there might be some missing values in the "Actual" column, indicated by "NaN" values.

	Actual	Predicted
0	NaN	Poor
1	NaN	Satisfactory
2	NaN	Satisfactory
3	NaN	Satisfactory

$R2 = 1 - \quad i=1 \quad i \quad i$

$\sum n \quad (y - \bar{y})2$

$i=1$

where $\bar{y}$ is the mean value of y. For maximum likelihood estimation, R2 can be calculated as:

$R2 = r2$

where r is an estimate of correlation coefficient between x and y, given by:

4	NaN	Poor
...	...	...
12017	Poor	NaN
12067	Very Poor	NaN
12247	Satisfactory	NaN
12291	Satisfactory	NaN
12302	Poor	NaN

$$\sum n$$

$$xi\ yi\ - n(\bar{x}y1\ - \bar{x})(\bar{y}1\ - \bar{y}\ )$$

$$r = \quad i=1$$

$$\sqrt{\sum n}\quad x2 - n(\bar{x}\ )2\sqrt{\sum n}\quad y2 - n(\bar{y})2$$
$$i=1\quad i\ i=1\quad i$$

where $\bar{x}$ and $\bar{y}$ are means of x and y, respectively.

The higher R-squared value indicates that your model fits your data well and captures most of its variation. However, R-squared does not tell you anything about whether your model has any causal relationship between its variables or whether it overfits or underfits

7	ANN	0.045	-	0.63
8	Linear Regression	11.9	8.89	-
9	LSTM-3, RCNN	27.33	-	0.7267
10	LSTM-6, RCNN	27.94	-	0.7206
11	LSTM-12, RCNN	26.50	-	0.735
12	LSTM-24, RCNN	27.10	-	0.729
13	LSTM-48, RCNN	26.79	-	0.7321
14	ARIMA, RCNN	31.29	-	0.6871
15	SARIMA, RCNN	28.88	-	0.7112

16	ST-ResNet, RCNN	20.20	-	0.798
17	ST-OR-ResNet_SO, RCNN	19.20	-	0.808
18	ST-OR-ResNet_SRIP, RCNN	17.60	-	0.824

IV CONCLUSION

In conclusion, addressing the formidable challenge of air quality in Tamil Nadu, especially in the context of developing nations like India, demands innovative solutions. This paper proposes a groundbreaking approach utilizing Artificial Neural Networks (ANNs) and Convolutional Neural Networks (CNNs), harnessing the power of machine learning and big data analytics. By surpassing the limitations of traditional monitoring methods, such as ground-based stations and satellite observations, our models offer high-resolution, real-time air quality insights. The robustness and generalizability of our models, validated against existing methods, underscore their potential significance. Moreover, our analysis sheds light on spatiotemporal air quality patterns, revealing key factors and drivers of pollution. This comprehensive approach not only equips policymakers with valuable tools but also empowers researchers and citizens to comprehend and actively contribute to improving air quality in Tamil Nadu and analogous regions facing similar environmental challenges.

V REFERENCES

[1] M. Banach, Z. Dlugosz, R. Dlugosz, and T. Talaska, “The Use of Artificial Neural Networks in Predicting Air Pollution in Cities-Hardware Implementation Issues,” in Proceedings of the International Conference on Microelectronics, ICM, Institute of Electrical and Electronics Engineers Inc., Sep. 2021, pp. 271–274. doi: 10.1109/MIEL52794.2021.9569046.

[2] L. Jovova and K. Trivodaliev, “Air Pollution Forecasting Using CNN-LSTM Deep Learning Model,” in 2021 44th International Convention on Information, Communication and Electronic Technology, MIPRO 2021 - Proceedings, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 1091–1096. doi: 10.23919/MIPRO52101.2021.9596860.

[3] R. S. Kumar, A. Arulanandham, and S. Arumugam, “Air quality index analysis of Bengaluru city air pollutants using Expectation Maximization clustering,” in 2021 International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation, ICAECA 2021, Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/ICAECA52838.2021.9675669.

[4] F. Naz et al., “Comparative Analysis of Deep Learning and Statistical Models for Air Pollutants Prediction in Urban Areas,” IEEE Access, vol. 11, pp. 64016–64025, 2023, doi: 10.1109/ACCESS.2023.3289153.

[5] S. Al-Eidi, F. Amsaad, O. Darwish, Y. Tashtoush,

A. Alqahtani, and N. Niveshitha, “Comparative Analysis Study for Air Quality Prediction in Smart Cities Using Regression Techniques,” IEEE Access, vol. 11, pp. 115140–115149, 2023, doi: 10.1109/ACCESS.2023.3323447.

[6] D. Iskandaryan, F. Ramos, and S. Trilles, “Graph Neural Network for Air Quality Prediction: A Case Study in Madrid,” IEEE Access, vol. 11, pp. 2729– 2742, 2023, doi: 10.1109/ACCESS.2023.3234214.

[7] “IMAGE- BASEDAIRQUALITYANALYSISUSINGDEEP CONVOLUTIONALNEURALNETWORK.” [Online]. Available: http://www.stateair.net/web/historical/1/1.html

[8] S. B. Sonu and A. Suyampulingam, “Linear Regression Based Air Quality Data Analysis and Prediction using Python,” in Proceedings of the IEEE Madras Section International Conference 2021, MASCON 2021, Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/MASCON51689.2021.9563432.

[9] J. H. Crawford et al., “The Airborne and Satellite Investigation of Asian Air Quality (Asia-Aq): An Opportunity for International Collaboration,” in International Geoscience and Remote Sensing Symposium (IGARSS), Institute of Electrical and Electronics Engineers Inc., 2022, pp. 6506–6509. doi: 10.1109/IGARSS46834.2022.9883819.

[10] V. Gupta, S. Kapadia, and C. Bhadane, “Time Series Analysis and Forecasting of Air Quality in India,” in 2023 5th International Conference on Electrical, Computer and Communication Technologies, ICECCT 2023, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/ICECCT56650.2023.10179673.

[11] L. Lyu, J. Kong, and Y. Peng, “Urban Ambient Air Quality Data Mining and Visualisation,” in Proceedings - 2022 International Conference on Artificial Intelligence of Things and Crowdsensing, AIoTCs 2022, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 616–620. doi: 10.1109/AIoTCs58181.2022.00101.

[12] P. W. Soh, J. W. Chang, and J. W. Huang, “Adaptive Deep Learning-Based Air Quality Prediction Model Using the Most Relevant Spatial- Temporal Relations,” IEEE Access, vol. 6, pp. 38186–38199, Jun. 2018, doi: 10.1109/ACCESS.2018.2849820.

[13] F. C. Zegarra, J. Vargas-Machuca, and A. M. Coronado, “Comparison of CNN and CNN-LSTM Architectures for Tool Wear Estimation,” in Proceedings of the 2021 IEEE Engineering International Research Conference, EIRCON 2021, Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/EIRCON52903.2021.9613659.

[14] Z. Wang, Y. Cao, and J. Li, “A Detection Algorithm Based on Improved Faster R-CNN for Spacecraft Components,” in 2023 IEEE International Conference on Image Processing and Computer Applications, ICIPCA 2023, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 1804–1808. doi: 10.1109/ICIPCA59209.2023.10257992.

[15] T. Fathi, J. Houda, and A. Mami, “Comparative between ANN algorithms in optimizing MPPT control autonomous photovoltaic,” in Proceedings of the International Conference on Advanced Systems and Emergent Technologies, IC_ASET 2020, Institute of Electrical and Electronics Engineers Inc., Dec. 2020, pp. 355–361. doi: 10.1109/IC_ASET49463.2020.9318275.

[16] S. V. P. Singh and P. Agnihotri, “ANN Based Modelling of Optimal Passive Cell Balancing,” in 2022 22nd National Power Systems Conference, NPSC 2022, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 326–331. doi: 10.1109/NPSC57038.2022.10069440.

[17] A. K. Kosta, M. P. E. Apolinario, and K. Roy, “Live Demonstration: ANN vs SNN vs Hybrid Architectures for Event-based Real-time Gesture Recognition and Optical Flow Estimation,” in IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, IEEE Computer Society, 2023, pp. 4148–4149. doi: 10.1109/CVPRW59228.2023.00436.

[18] L. Zhang, D. Li, and Q. Guo, “Deep Learning from Spatio-Temporal Data Using Orthogonal Regularizaion Residual CNN for Air Prediction,” IEEE Access, vol. 8, pp. 66037–66047, 2020, doi: 10.1109/ACCESS.2020.2985657.

[19] J. Tan, W. Xiong, and Z. Tu, “The Air Quality Prediction on Deep Spatiotemporal Feature Extraction with a Transductive Kernel Extreme Learning Machine,” in IEEE International Symposium on Industrial Electronics, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 104–109. doi: 10.1109/ISIE51582.2022.9831755.

[20] T. Li, M. Hua, and X. Wu, “A Hybrid CNN-LSTM Model for Forecasting Particulate Matter (PM2.5),” IEEE Access, vol. 8, pp. 26933–26940, 2020, doi: 10.1109/ACCESS.2020.2971348.