Indian Institute of Information Technology Vadodara

(Gandhinagar Campus)

Data-Driven Weather Forecast Using Deep Convolution Neural Network

Submitted by

Priyanshu Anand-(202151121)
Navneet Tewatia-(202151097)
Prashant Singh-(202151116)

Under the supervision of

Dr. Pratik Shah

***Abstract*—A Convolutional Neural Network (CNN) regression model for weather forecasting is proposed in this study. The model is trained on historical weather data and is developed with Keras with a TensorFlow backend. The process of preparing data entails importing the dataset, dividing it into training and testing sets, removing non-numeric features, and converting date columns. Scaling the characteristics and target variable involves using standardization. A 1D convolutional layer, a max pooling layer, and fully linked layers make up the CNN architecture. The mean squared error loss function is used to train the model once it has been compiled using the Adam optimizer. The performance of the model is evaluated using measures like Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).**

Keywords: Weather Forecasting, Convolutional Neural Network, Regression, Keras, TensorFlow, Data Preprocessing, Model Training, Evaluation Metrics.

I. Introduction

A number of industries, including agriculture, transportation, energy, and disaster relief, rely heavily on weather forecasting. Making educated decisions and reducing potential dangers requires accurate forecasts of the weather for the future. Conventional weather forecasting techniques rely on meteorological observations and physical models, however they might not be as good at accurately describing complicated weather patterns. The use of data-driven methods, especially deep learning techniques, to anticipate weather has gained popularity in recent years. From large-scale meteorological data, deep convolutional neural networks (CNNs) have demonstrated potential in learning complex spatiotemporal patterns. In this paper, we offer a deep CNN architecture-based data-driven weather forecasting method. In order to train the CNN model to forecast future weather, we preprocess previous weather data. This paper presents the methodology, experimental results, and implications of our CNN-based weather forecasting approach.

II. Background

For numerous sectors and facets of daily life, weather forecasting is essential. In addition to assuring effective transportation by anticipating weather-related disruptions, it helps farmers plan ahead for natural calamities like hurricanes and floods.

Weather predictions have historically been based on intricate physical models that were derived from atmospheric science and fluid dynamics theories. These models have disadvantages, such as being computationally costly and sensitive to initial conditions, even if they have improved over time.

The application of machine learning, particularly deep learning, to improve weather forecasting has gained popularity recently. Deep learning does not require intricate physical models in order to learn complicated patterns straight from data. Deep learning methods can identify minute patterns and connections by examining big meteorological datasets.

Deep learning has the benefit of being able to manage complicated, high-dimensional data well. While recurrent neural networks (RNNs) are helpful for time-series data like weather forecasts, convolutional neural networks (CNNs) excel at processing spatial data like weather maps.

Nonetheless, there are still issues with deep learning for weather forecasting, such as the requirement for huge datasets, comprehending the model's predictions, and adjusting to novel weather patterns. Making deep learning models effective enough for real-time forecasting is also crucial.

In order to overcome these obstacles and develop more dependable and effective weather prediction systems, this proposal looks into how deep learning might increase the accuracy of weather forecasts.

III. Proposed Neural Network Architectures

We used CNN structure,

A. Input Layer

The input layer accepts the input data with the shape (*number of samples*, *number of features*, 1), where:

- *number of samples* represents the number of instances in the dataset.
- *number of features* represents the number of input features.
- 1 indicates that the data is univariate.

B. Convolutional Layer (Conv1D)

The convolutional layer consists of 32 filters with a kernel size of 3 and ReLU activation function. The output of this layer is obtained by convolving the input data with the filters to extract features.

C. Pooling Layer (MaxPooling1D)

Max pooling with a pool size of 2 is applied to reduce the dimensionality of the feature maps. This layer helps in capturing the most important features while reducing computational complexity.

D. Flattening Layer (Flatten)

The output from the convolutional and pooling layers is flattened into a one-dimensional array. Flattening prepares the data to be fed into the fully connected layers.

E. Fully Connected Layers (Dense)

Two dense layers are added after flattening to learn complex patterns in the data. The first dense layer consists of 128 neurons with ReLU activation function. The second dense layer consists of a single neuron with linear activation function, which serves as the output layer for regression.

F. Model Compilation

The model is compiled using the Adam optimizer and mean squared error loss function, suitable for regression tasks.

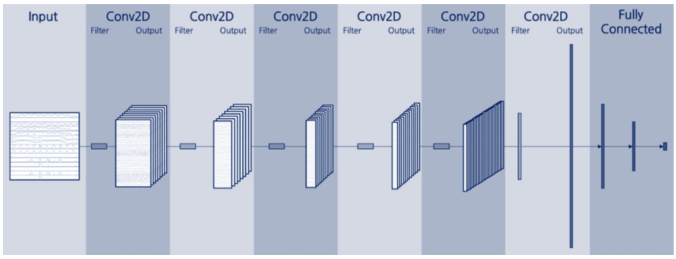

Fig. 1. Figure 1. Model structure of convolution neural network (CNN) for temperature forecasting.

G. Model Training

The model is trained on the training data with 10 epochs and a batch size of 32. During training, a validation split of 20% is used to monitor the model's performance on unseen data.

H. Model Evaluation

After training, the model is evaluated on the testing data. Predictions are made on the test data, and evaluation metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are calculated to assess the model's performance.

IV. CNN Regression Model for Weather Forecasting

In this section, we outline the process of implementing a Convolutional Neural Network (CNN) for regression using Keras with TensorFlow backend for weather forecasting.

A. Data Preprocessing

The data is loaded from a CSV file using Pandas. The 'Date.Full' column is converted to datetime format. Non-numeric columns are dropped to keep only numerical features. Features (X) and the target variable (y) are separated.

B. Data Splitting

The dataset is split into training and testing sets using `train_test_split()` from scikit-learn. 80% of the data is used for training and 20% for testing.

C. Feature Scaling

Standardization is applied separately to features and the target variable using `StandardScaler()` from scikit-learn. This helps in scaling the data to have a mean of 0 and a standard deviation of 1.

D. Reshaping for CNN

The input data is reshaped to have the shape (*samples, time_steps, features*) expected by the `Conv1D` layer in Keras.

E. CNN Model Design

A Sequential model is created. The model consists of a 1D convolutional layer (`Conv1D`) with 32 filters and a kernel size of 3, followed by a ReLU activation function. Max pooling (`MaxPooling1D`) is applied to reduce the dimensionality of the feature maps. The output is flattened (`Flatten`) to be fed into a fully connected layer (`Dense`) with 128 neurons and ReLU activation. Finally, there's a single neuron output layer with a linear activation function.

V. Data

In this section, we introduce the numeric weather data we used and describe the data preprocessing.

A. Data Description

We collected the Weather data From the CORGIS Dataset Project.

Key	Description	Comment	Example Value
Data.Precipitation	Float	The average amount of rain, in inches.	0.0
Date.Full	String	A full string representation of the date this report was made.	"2016-01-03"
Date.Month	Integer	The month of the year that this report was made.	1
Date.Week of	Integer	The first day of the week that this report was made.	3
Date.Year	Integer	The year that this report was made.	2016
Station.City	String	The city that the reporting station sends its data to. Note that the recording station itself might actually be in a different city.	"Birmingham"
Station.Code	String	The unique code representing this station.	"BHM"
Station.Location	String	The exact location of this recording station. Note that the recording station itself might be in a different location than where it sends its data.	"Birmingham, AL"
Station.State	String	The state that the reporting station sends its data to. Note that the recording station itself might actually be in a different state.	"Alabama"
Data.Temperature.Avg Temp	Integer	The average recorded temperature on this week, in degrees Fahrenheit.	39
Data.Temperature.Max Temp	Integer	The highest recorded temperature on this week, in degrees Fahrenheit.	46
Data.Temperature.Min Temp	Integer	The lowest recorded temperature on this week, in degrees Fahrenheit.	32
Data.Wind.Direction	Integer	The average wind direction for that week, in degrees.	33
Data.Wind.Speed	Float	The average windspeed for that week, in Miles per Hour.	4.33

TABLE I
Description of Keys

VI. Experimental Results

The experimental results for the provided code are as follows:

- **Epochs**: 10
- **Batch Size**: 32
- **Loss Function**: Mean Squared Error (MSE)
- **Optimizer**: Adam

The training loss and validation loss values for each epoch are as follows:

Epoch	Training Loss	Validation Loss	Time
1	0.8060	1.0088	1s 2ms/step
2	0.7869	1.0185	0s 1ms/step
3	0.8210	0.9542	0s 1ms/step
4	0.7374	0.9505	0s 1ms/step
5	0.7237	0.9216	0s 1ms/step
6	0.7519	0.9168	0s 1ms/step
7	0.7003	0.9193	0s 1ms/step
8	0.7661	0.8974	0s 1ms/step
9	0.7134	0.8972	0s 1ms/step
10	0.7203	0.8873	0s 1ms/step

TABLE II
Training and Validation Loss for Each Epoch

Note: The training loss measures how well the model fits the training data, while the validation loss assesses how well it generalizes to unseen data. Both are calculated during model training, with the goal of minimizing both to achieve better performance.

Adam: This tells Keras to use the Adam optimizer to minimize the mean squared error loss function during training.

After training, the model is evaluated on the testing data, and the following evaluation metrics are obtained:

- **Mean Absolute Error (MAE)**: 0.5018199074562588
- **(RMSE)**:0.811057280979801

VII. Acknowledgement

For his significant advice and assistance during the course of this project on Data-Driven Weather Forecast Using Deep Convolutional Neural Networks, I would like to sincerely thank Dr. Pratik Shah. His knowledge, advice, and perceptive criticism have greatly influenced this research project. His constant support, tolerance, and hard work have been greatly appreciated; they have been essential to the project's successful conclusion.

VIII. Conclusion

These results indicate how well the CNN model can learn from the data and make predictions about precipitation levels. Lower values of loss and error metrics suggest better performance of the model in forecasting weather conditions.

References

1) **CORGIS Dataset Project:** Weather data can be accessed from the CORGIS Dataset Project at: https://corgis-edu.github.io/corgis/csv/weather/
2) **Google Images:** Images related to "Data-Driven Weather Forecast Using Deep Convolution Neural Network" can be found through a Google search at: https://www.google.com/search?sca_esv=9f7ecf1a9cd93cb6&rlz=1C1CHZN_enIN987IN987&q=Data-Driven+Weather+Forecast+Using+Deep+Convolution+Neural+Network+images&tbm=isch&source=lnms&prmd=ivnsbmtz&sa=X&ved=2ahUKEwjw8Y2t99yFAxW6h68BHck7AAEQ0pQJegQIDRAB&biw=1536&bih=730&dpr=1.25#imgrc=zDoT_iXQiKVzsM
3) **Weather Forecasting Research Papers:**
   - Various research papers on weather forecasting and deep learning techniques can be found on platforms like Google Scholar or IEEE Xplore.
   - Google Scholar: https://scholar.google.com/
   - IEEE Xplore: https://ieeexplore.ieee.org/
4) **Deep Learning Frameworks:**
   - Information about deep learning frameworks like TensorFlow and Keras can be found on their official documentation websites.
   - TensorFlow Documentation: https://www.tensorflow.org/
   - Keras Documentation: https://keras.io/