Weather Forecast

1st Nishant Dev
Computer Science Engineering
IIIT Vadodara
202151100@iiitvadodara.ac.in

2nd Sankalp Sagar
Information Technology
IIIT Vadodara
202152337@iiitvadodara.ac.in

3rd Siddhant Kumar Chaudhary
Computer Science Engineering
IIIT Vadodara
202151155@iiitvadodara.ac.in

4th Sourabh Totod
Computer Science Engineering
IIIT Vadodara
202151173@iiitvadodara.ac.in

5rd Tanmay Meshram
Information Technology
IIIT Vadodara
202152341@iiitvadodara.ac.in

***Abstract*—This project presents an advanced weather forecasting system utilizing TensorFlow/Keras and LSTM-based deep learning models, designed for accurate predictions of temperature, humidity, and pressure. It segments data into time slices (morning, afternoon, evening, night) and leverages 5G network slicing principles to meet diverse QoS requirements, such as high accuracy, low latency, and balanced performance. Feature-specific models ensure precise predictions, while multiprocessing and GPU/CPU resource allocation optimize computational efficiency. Preprocessing includes scaling input data for compatibility and inverse transformations for interpretability. The system dynamically selects models based on timestamps and prioritizes scalability, flexibility, and real-world robustness. With applications in agriculture, smart cities, and disaster management, this solution integrates AI and 5G slicing to deliver scalable, actionable weather insights.**

I. Introduction

This project focuses on developing a sophisticated weather forecasting system using LSTM-based deep learning models implemented in TensorFlow/Keras. The system predicts key weather parameters—temperature, humidity, and pressure—by processing sequential time-series data divided into distinct time slices (morning, afternoon, evening, and night). It incorporates 5G network slicing principles to address diverse Quality of Service (QoS) requirements, such as high accuracy, low latency, and balanced performance, ensuring efficient resource utilization with dynamic GPU/CPU allocation. Feature-specific models enhance prediction precision, while scalable preprocessing methods normalize and inverse-transform input data for clarity. By dynamically loading the appropriate models and leveraging multiprocessing, the system delivers robust, real-time weather insights.

II. Literature Review

A. What is Network Slicing

Network slicing is a concept in 5G networks that enables the creation of multiple virtual networks, or ”slices,” tailored to specific use cases and Quality of Service (QoS) requirements. Each slice can be optimized for distinct needs such as high throughput, low latency, or massive device connectivity. In the context of weather forecasting, network slicing ensures that different types of predictions, such as high-accuracy forecasts or low-latency predictions, are handled efficiently by allocating the appropriate computational resources. The system dynamically selects and loads models based on the time slice (morning, afternoon, evening, or night) and the required QoS level (high accuracy, medium priority, low latency). This approach enhances the scalability and responsiveness of the weather forecasting system, ensuring real-time predictions and efficient resource utilization.

B. How it is used in our project

In our project, network slicing is utilized to tailor weather forecasting models based on the specific Quality of Service (QoS) requirements for different use cases. The system defines three types of network slices: high accuracy, medium priority, and low latency, corresponding to different weather prediction tasks. For example, high-accuracy predictions are allocated to more computationally intensive resources, while low-latency predictions leverage faster but less resource-demanding models. Each time slice (morning, afternoon, evening, and night) dynamically selects the appropriate model, ensuring that weather data is processed according to the required QoS level. This slicing approach allows the system to handle various forecasting needs efficiently, balancing computational resources and ensuring real-time, scalable predictions for applications like agriculture, disaster management, and smart cities.

III. Software Components

A. TensorFlow/Keras

TensorFlow and Keras are deep learning libraries that provide the tools for building and training sophisticated machine learning models. In this project, they are primarily used for implementing LSTM (Long Short-Term Memory) networks to predict weather parameters such as temperature, humidity, and pressure. Keras, as a high-level API, simplifies model creation, training, and saving, ensuring that the models are both flexible and scalable for future upgrades.

B. LSTM Models

LSTM models are used in this system due to their ability to handle sequential weather data. They are categorized into three types: the High Accuracy model for precise predictions, the Medium Priority model for balanced performance, and the Low Latency model for fast response times. Additionally, feature-specific models are developed to predict individual weather parameters like temperature, humidity, and pressure, enhancing the system's adaptability to various forecasting needs.

C. Pandas

Pandas is a data manipulation and analysis library that aids in reading, cleaning, and structuring weather data. It enables operations like converting date-time information and selecting specific time slices (morning, afternoon, evening, night) based on the datetime column. Pandas also helps in handling and transforming large datasets, ensuring that they are prepared correctly for the input of the model.

D. NumPy

NumPy is a fundamental package for scientific computing in Python, providing support for large, multi-dimensional arrays and matrices. In this project, it is used to reshape the weather data into the required format for LSTM models and to perform numerical operations like element-wise arithmetic needed during data preprocessing and post-processing phases. It ensures smooth data manipulation and conversion between formats.

E. Scikit-learn (MinMaxScaler)

Scikit-learn's MinMaxScaler is crucial for normalizing features in the weather data, which helps in stabilizing the training process of machine learning models. It scales features like temperature, humidity, and pressure to a specified range (usually 0-1), ensuring that the models are trained effectively without being biased by variations in the scale of input features. This normalization step is essential for the LSTM models to perform optimally.

F. Pickle

Pickle is used to serialize and deserialize Python objects, enabling the saving and loading of complex objects like the MinMaxScaler. This ensures that the preprocessing steps applied during model training are exactly replicated during prediction. By saving the scaler object to a .pkl file, it can be easily reused for any future predictions, ensuring consistency and avoiding the need for re-scaling each time.

G. Time Slicing Logic

Time slicing divides the weather data into specific segments based on the time of day—morning, afternoon, evening, and night. This segmentation ensures that the model can be trained and make predictions specific to the weather patterns associated with different times of day, making the forecasting more contextually accurate. The approach allows for a more granular and tailored analysis of weather data, which is particularly useful for dynamic environments.

H. Model Management Module

The model management component loads and organizes the trained models, including those for different time slices (morning, afternoon, evening, night) and QoS levels (high_accuracy, medium_priority, low_latency). This module ensures that the correct models are used for predictions based on the time of day, optimizing the prediction process according to specific priorities.

I. Prediction Module

This module is responsible for generating predictions based on the input data and the loaded models. It applies different QoS models on GPU or CPU according to their priority and also predicts specific weather features, such as temperature, humidity, and pressure, for each time slice. The output from this component is the core prediction that will be used in decision-making processes.

IV. Working

A. Initialization and WiFi Connection

The system initializes essential hardware components, such as sensors and the NodeMCU, and establishes a robust WiFi connection to ensure seamless data transmission.

B. Data Acquisition from Sensors

Once the WiFi connection is established, the system gathers crucial environmental data from sensors, including temperature, humidity, barometric pressure, and air quality measurements.

C. Data Handling by Node.js Server

The collected sensor data is transmitted to a Node.js server, which serves as the central hub for data processing. Here, the data is meticulously processed and stored in a CSV file, ensuring structured storage for subsequent analysis.

D. Machine Learning Prediction

Leveraging historical sensor data, an LSTM (Long Short-Term Memory) model engages in predictive analytics to forecast future environmental conditions. By continuously learning from past patterns, the model generates accurate predictions for the upcoming hours.

E. Data Output and Visualization

The predicted and real-time data are seamlessly integrated into a MongoDB database, allowing for efficient storage and retrieval. Additionally, an API is created to facilitate easy access to the stored data, enabling users to visualize and compare environmental parameters through intuitive web-based interfaces.

F. Continuous Operation and Display

To ensure real-time monitoring, the system operates continuously, updating the web page with new predictions and real-time sensor readings at regular intervals. This dynamic display provides users with a comprehensive overview of environmental conditions, promoting proactive decision-making and analysis.

V. Machine Learning Model

The machine learning component of this project employs an LSTM (Long Short-Term Memory) network, a type of recurrent neural network (RNN) well-suited for time series prediction. This model is pivotal in predicting environmental conditions such as temperature, humidity, and air pressure by analyzing patterns found in historical data.

A. LSTM Network Architecture

LSTM networks are uniquely capable of learning long-term dependencies because unlike traditional feed-forward neural networks, LSTMs have a feedback connection that helps them retain information over longer time periods. This makes LSTMs ideal for time series data like that generated from environmental sensors.

B. Training Process

- **Data Preparation**: Data collected from sensors is structured into a CSV file format by the Node.js server. Each entry contains timestamped records of temperature, humidity, air pressure, and other relevant measurements. This data is then preprocessed to normalize the values, making them suitable for training the LSTM model.
- **Model Training**: The LSTM model, developed using Python and TensorFlow, is trained on a combination of freshly collected and externally sourced data. Specifically, the training dataset includes sensor readings from the past 24 hours, supplemented with additional datasets obtained from the internet. This comprehensive approach enriches the training data, ensuring the model is well-versed in a variety of environmental scenarios. The model employs a rolling window technique, continuously updating its predictions for the upcoming six hours based on the most recent data inputs. This dynamic training method enhances the model's accuracy and responsiveness to changes in environmental conditions.

C. Model Observation and Evaluation

1) Data Distribution:

- Actual Temperature is represented by blue circles connected with a line, showing the real-time measured values from the sensors.
- Predicted Temperature is represented by orange crosses connected with a line, indicating the values predicted by the LSTM model.

2) Trend Analysis:

- Both actual and predicted data lines follow similar trends throughout the dataset, indicating that the LSTM model has effectively learned the general pattern of temperature changes over time.
- There are peaks and troughs in both datasets, which suggests a recurring pattern or cycle in the temperature data, possibly influenced by daily natural variations such as day and night cycles.

3) Accuracy and Deviations:

- There are points where the predicted values closely match the actual values, showing high accuracy of the model at these points.
- However, there are also notable deviations at some time points where the predicted temperatures either underestimate or overestimate the actual temperatures significantly. This could be due to anomalies in the environment that were not present in the training data, or due to the model's limitations in capturing certain subtleties of the data.

4) Model Summary:

a) Model Training:

- The model appears well-trained on the general pattern of the dataset but might benefit from further training or refinement to handle anomalies or unusual data points better.

b) Potential Improvements:

- Incorporating more diverse training data covering a broader range of environmental conditions could help improve model predictions.
- Adjusting model parameters or experimenting with a different architecture might also reduce prediction errors.

5) Usability:

- Despite some errors, the model likely provides sufficiently accurate predictions for practical use, depending on the specific requirements of the application.

VI. Conclusion

This project successfully demonstrates a comprehensive pipeline for weather forecasting that integrates advanced modeling techniques with emerging 5G network slicing concepts. By segmenting the data into time slices—morning, afternoon, evening, and night—and training different LSTM-based models tailored to distinct Quality of Service (QoS) requirements (e.g., high accuracy, low latency), we were able to align the model's performance with the needs of various 5G network slices. Through the careful preparation of data, including feature scaling and sequence creation, and the selection of appropriate neural network architectures, the models can forecast key meteorological parameters (temperature, humidity, and pressure) several steps into the future.

The project addressed and resolved several technical challenges, including ensuring compatibility with updated Keras versions, managing proper input and output dimensions for LSTM layers, and maintaining consistency between training and inference stages. The final implementation includes model

saving in a format compatible with newer Keras APIs, reliable loading and preprocessing steps, and a prediction script that accurately returns scaled results back to their original units.