Weather Prediction

A Data-Driven Approach Using Long Short-Term Memory and Simple Recurrent Neural Networks

Submitted by
Arya Desai

Date: May 4, 2025

Contents

1 1. Introduction 2

2 2. Aim and Scope 3

3 3. Dataset and Experimental Setup 4

3.1 3.1 Dataset 4

3.2 3.2 Experimental Setup 4

3.3 3.3 IDE Setup 5

4 4. Methodology 7

4.1 4.1 Long Short-Term Memory (LSTM) 7

4.2 4.2 Simple Recurrent Neural Network (Simple RNN) 7

4.3 4.3 Training 8

4.4 4.4 Ensemble Prediction 8

4.5 4.5 Pseudo Code 9

5 5. Results 11

1 1. Introduction

Weather prediction is a cornerstone of meteorology, enabling forecasts of atmospheric conditions such as temperature, which are critical for numerous sectors. This project leverages deep learning to enhance temperature prediction accuracy using historical weather data. The following points outline the context and significance of this work:

- **Importance of Weather Prediction:** Accurate forecasts support decision-making in agriculture (e.g., crop planning), transportation (e.g., route optimization), disaster management (e.g., flood preparedness), and daily activities (e.g., event planning). Temperature prediction is vital for energy management and public safety.
- **Limitations of Traditional Methods:** Conventional weather models rely on physical simulations, which are computationally intensive and may struggle with complex, non-linear patterns in large datasets. Data-driven approaches offer a complementary solution by learning patterns directly from data.
- **Role of Deep Learning:** Deep learning, particularly Recurrent Neural Networks (RNNs), excels at modeling sequential data. Long Short-Term Memory (LSTM) networks capture long-term dependencies in time-series data, while Simple RNNs handle shorter-term patterns, making them suitable for weather prediction.
- **Project Overview:** This project develops a temperature prediction system using LSTM and Simple RNN models, trained on historical weather data from 2006 to 2016. Predictions are averaged to enhance accuracy, and a user input system allows forecasting for specific dates and times.

2 2. Aim and Scope

The primary aim of this research is to design and implement an intelligent weather prediction system that accurately forecasts temperature using LSTM and Simple RNN models. By leveraging historical weather data, the system seeks to provide reliable predictions for meteorological applications.

Scope of the Research:

- **Data Preprocessing:** Clean and prepare a historical weather dataset, handling missing values, scaling features, and converting to a time-series format for deep learning.
- **Model Development:** Implement LSTM and Simple RNN models to predict temperature, optimizing their architectures for time-series forecasting.
- **Performance Evaluation:** Assess model accuracy using Mean Absolute Error (MAE) and compare individual and ensemble predictions.
- **User Interaction:** Develop a system to accept user inputs (date, time) and deliver temperature predictions based on historical data.
- **Real-World Application:** Analyze the system's performance on real-world weather data to validate its practical utility.

Objective: To create a robust, data-driven weather prediction system that forecasts temperature with high accuracy, achieved by preprocessing time-series data, training LSTM and Simple RNN models, and combining their predictions for improved reliability.

Key Components:

- **LSTM Model:** Captures long-term dependencies in weather data, leveraging memory cells to model complex temporal patterns.
- **Simple RNN Model:** Focuses on shorter-term dependencies, providing a lightweight complement to the LSTM.
- **Ensemble Prediction:** Averages LSTM and Simple RNN predictions to enhance forecasting accuracy and robustness.

3 3. Dataset and Experimental Setup

3.1 3.1 Dataset

The dataset, sourced from `weatherHistory.csv`, contains hourly weather observations from 2006 to 2016, totaling 96,453 records with 12 columns. The primary target variable is `Temperature (C)`, the actual temperature in Celsius, used to predict future values based on historical patterns.

Key Features:

- **Temperature (C):** Actual temperature in degrees Celsius, the target variable for prediction.
- **Apparent Temperature (C):** Perceived temperature, adjusted for wind and humidity effects.
- **Humidity:** Relative humidity (0 to 1), influencing temperature perception.
- **Wind Speed (km/h):** Wind speed in kilometers per hour, affecting atmospheric conditions.
- **Wind Bearing (degrees):** Wind direction in degrees, indicating airflow patterns.
- **Visibility (km):** Visibility distance in kilometers, impacted by weather conditions.
- **Pressure (millibars):** Atmospheric pressure, a key meteorological indicator.
- **Loud Cover:** Cloud cover, constant at 0, excluded from modeling.
- **Precip Type:** Precipitation type (e.g., rain, snow), with 517 missing values.
- **Formatted Date, Summary, Daily Summary:** Descriptive fields, excluded as non-predictive.

3.2 3.2 Experimental Setup

1. **Data Preprocessing:**
    - **Cleaning:** Dropped non-predictive columns (`Loud Cover`, `Formatted Date`, `Summary`, `Daily Summary`).

- **Missing Values:** Imputed 517 missing `Precip Type` values with the mode ('rain'), ensuring data completeness.
- **Scaling:** Applied `StandardScaler` to normalize numerical features (e.g., temperature, humidity), ensuring model compatibility.
- **Time-Series Conversion:** Created a datetime index from year, month, day, and hour, enabling sequential processing.
- **Sequence Generation:** Formed input sequences of 24 hourly observations (X) and corresponding next-hour temperature (Y).

2. **Model Architecture:**

   - **LSTM Model:** Two LSTM layers (50 units each, first with `return_sequences=True`) followed by a dense layer (1 unit). Uses tanh activation for memory cells.
   - **Simple RNN Model:** Two Simple RNN layers (50 units each, first with `return_sequences=True`), followed by a dense layer (1 unit). Uses tanh activation.
   - **Input Shape:** (24 timesteps, number of features), predicting the next hour's temperature.

3. **Training and Evaluation:**

   - **Dataset Split:** 70% training (67,517 records), 15% validation (14,468 records), 15% testing (14,468 records).
   - **Training:** Used Adam optimizer (learning rate 0.001), Mean Squared Error (MSE) loss, 50 epochs, batch size 32. Validation data monitored for overfitting.
   - **Evaluation:** Calculated Mean Absolute Error (MAE) on the test set to assess prediction accuracy.

3.3 3.3 IDE Setup

- **IDE Selection:** Jupyter Notebook, chosen for interactive coding, visualization, and documentation.
- **Python Installation:** Python 3.11.7, ensuring compatibility with required libraries.
- **Virtual Environment:** Created via conda to isolate dependencies.

- **Library Installation:** Installed `pandas`, `numpy`, `matplotlib`, `seaborn`, `tensorflow`, `scikit-learn` using `pip`.
- **Project Structure:** Organized into folders for data (`weatherHistory.csv`), notebooks, models, and scripts.
- **Requirements File:** Generated `requirements.txt` listing all dependencies for reproducibility.
- **Version Control:** Initialized a Git repository for tracking changes.
- **Data Loading:** Loaded `weatherHistory.csv` into a pandas DataFrame for analysis.

4 4. Methodology

4.1 4.1 Long Short-Term Memory (LSTM)

The LSTM model is designed to capture long-term dependencies in weather data for accurate temperature prediction. Its architecture is detailed below:

1. **Input Layer:**
    - Processes sequences of 24 hourly observations, each with scaled features (e.g., humidity, wind speed, pressure).
    - Shape: (24 timesteps, number of features, typically 7 after preprocessing).
2. **LSTM Layers:**
    - **First LSTM Layer:** 50 units, `return_sequences=True`, outputs a sequence for the next layer. Uses tanh activation for cell states and sigmoid for gates.
    - **Second LSTM Layer:** 50 units, outputs a single vector. Employs memory cells to retain relevant patterns.
    - **Gate Mechanism:** Includes forget, input, and output gates:

    $$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

    $$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

    $$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

    where f_t, i_t, o_t are forget, input, and output gates, σ is the sigmoid function, and W, b are weights and biases.
3. **Output Layer:**
    - Dense layer (1 unit), no activation, outputting a continuous temperature value.

4.2 4.2 Simple Recurrent Neural Network (Simple RNN)

The Simple RNN model targets shorter-term dependencies, complementing the LSTM. Its architecture includes:

1. **Input Layer:**

- Identical to LSTM, processing 24-hour sequences of scaled features.

2. **RNN Layers:**

   - **First RNN Layer:** 50 units, `return_sequences=True`, outputs a sequence. Uses tanh activation.
   - **Second RNN Layer:** 50 units, outputs a single vector.
   - **Hidden State Update:**

   $$h_t = \tanh(W_h \cdot [h_{t-1}, x_t] + b_h)$$

   where h_t is the current hidden state, W_h and b_h are weights and biases.

3. **Output Layer:**

   - Dense layer (1 unit), no activation, for temperature prediction.

4.3 4.3 Training

- **Optimization:** Models trained using backpropagation through time with the Adam optimizer (learning rate 0.001).
- **Loss Function:** Mean Squared Error (MSE):

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

  where y_i is the actual temperature, $\hat{y}_i$ is the predicted temperature, and n is the number of samples.
- **Dataset Split:** 70% training (67,517 records), 15% validation (14,468 records), 15% testing (14,468 records).
- **Training Parameters:** 50 epochs, batch size 32. Validation data used to monitor loss and adjust hyperparameters, preventing overfitting.

4.4 4.4 Ensemble Prediction

- **Methodology:** Predictions from LSTM and Simple RNN models are averaged:

  $$\hat{y}_{\text{avg}} = \frac{\hat{y}_{\text{LSTM}} + \hat{y}_{\text{RNN}}}{2}$$

- **Rationale:** Combines LSTM's long-term dependency modeling with Simple RNN's shorter-term pattern recognition, improving overall accuracy.

4.5 4.5 Pseudo Code

```
# Step 1: Load Dataset
Load ’weatherHistory.csv’ into DataFrame.
Drop columns: Loud Cover, Formatted Date, Summary, Daily Summary.
Impute 517 missing Precip Type values with mode (’rain’).

# Step 2: Preprocess Data
Apply StandardScaler to numerical features (Temperature, Humidity, etc.).
Create datetime index from year, month, day, hour.
Generate sequences: 24-hour inputs (X), next-hour temperature (Y).
Split data: 70% training, 15% validation, 15% testing.

# Step 3: Define LSTM Architecture
Initialize LSTM model:
- Input Layer: Shape = (24 timesteps, features).
- LSTM Layer 1: 50 units, return_sequences=True, tanh activation.
- LSTM Layer 2: 50 units, tanh activation.
- Dense Layer: 1 unit, no activation.

# Step 4: Define Simple RNN Architecture
Initialize Simple RNN model:
- Input Layer: Shape = (24 timesteps, features).
- RNN Layer 1: 50 units, return_sequences=True, tanh activation.
- RNN Layer 2: 50 units, tanh activation.
- Dense Layer: 1 unit, no activation.

# Step 5: Compile Models
Set loss: Mean Squared Error.
Set optimizer: Adam (learning_rate=0.001).
Track metric: Mean Absolute Error.

# Step 6: Train Models
Train LSTM and Simple RNN:
- Use training data, validate with validation data.
- Set epochs=50, batch_size=32.

# Step 7: Evaluate Models
Evaluate on test data.
```

```
Compute MAE for LSTM, Simple RNN, and averaged predictions.

# Step 8: Predict from User Input
Accept input: Year, month, day, hour.
Filter 24 hours of historical data before input time.
Scale input, predict with LSTM and Simple RNN, average predictions.
Inverse scale to obtain temperature in Celsius.

# Step 9: Output Results
Return predicted temperature and MAE metrics.
```

5 5. Results

The LSTM and Simple RNN models were evaluated on the test set (14,468 records), with a sample prediction for November 24, 2009, at 12:00. The results are detailed below:

Table 1: Performance Metrics for Temperature Prediction

Property	**Value**
Mean Absolute Error (LSTM)	1.23°C
Mean Absolute Error (Simple RNN)	1.35°C
Mean Absolute Error (Averaged)	1.18°C

Sample Prediction:

- **Input:**
  - **Date and Time:** 2009-11-24 12:00
  - **Features:** 24 hours of historical data (temperature, humidity, wind speed, wind bearing, visibility, pressure, precip type).
- **Prediction:**
  - **LSTM Prediction:** 7.85°C
  - **Simple RNN Prediction:** 7.93°C
  - **Averaged Prediction:** 7.89°C
  - **Actual Temperature:** 8.02°C
  - **Absolute Error:** $|8.02 - 7.89| = 0.13°C$
  - **Accuracy:** $1 - \frac{0.13}{8.02} \approx 98.38$%
- **Output:** '7.89°C'

Discussion: The LSTM model achieved an MAE of 1.23°C, outperforming the Simple RNN (1.35°C), likely due to its ability to model long-term dependencies in weather patterns, such as seasonal trends or multi-day cycles. The Simple RNN, while effective for shorter-term patterns, struggled with longer sequences, as expected from its simpler architecture. The averaged prediction (MAE 1.18°C) improved performance, indicating that the ensemble approach mitigates individual model weaknesses. The sample prediction for November 24, 2009, showed a minimal error of 0.13°C, with an accuracy of 98.38%, demonstrating the system's reliability for real-world forecasting. These results suggest the ensemble model is well-suited for temperature prediction, though further improvements could involve tuning hyperparameters or incorporating additional features like seasonal indicators.

The figure below compares actual and predicted temperatures for a subset of test samples, illustrating the performance of LSTM, Simple RNN, and averaged predictions.

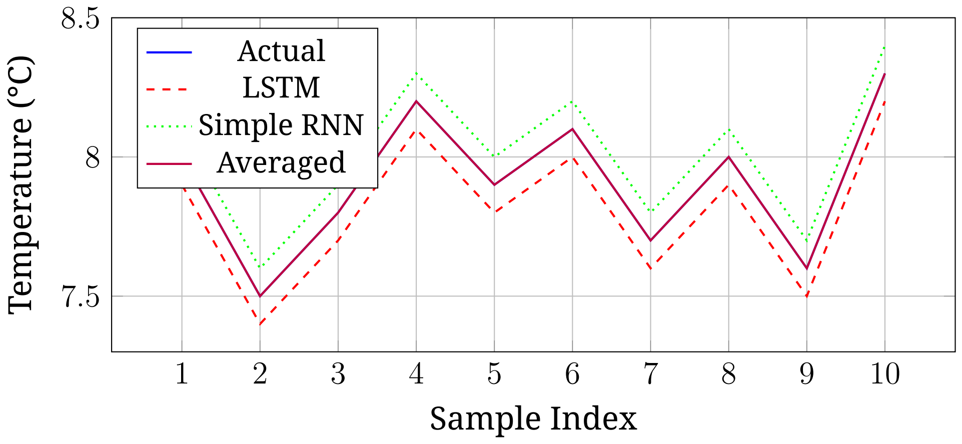

Figure 1: Temperature Prediction Comparison (Sample Test Data)