Yerevan State University
Institute of physics

FINAL REPORT

Weather Long-term Time Series Forecasting

Student:
Milena Aghabekyan

Yerevan, 2024

Contents

1 Introduction

For various applications such as energy planning, disaster relief, and agriculture, accurate weather forecasting is crucial. This study investigates the use of high-frequency meteorological data to predict air temperature using a Reservoir Neural Network (RNN), a specific type of Reservoir Computing (RC). Over 52,560 data points per variable, captured every 10 minutes throughout 2020, make up the dataset. Twenty meteorological indicators, including temperature, humidity, precipitation, and wind patterns, are included.

Important procedures include building a reservoir of 50 neurons with precisely calibrated parameters, preprocessing the input using Min-Max scaling, and utilizing Ridge Regression to train output weights. Mean Squared Error (MSE) and R2 metrics are used to assess the model.

The accuracy and resilience of the model are validated through visual comparison of actual and forecasted temperatures. This paper demonstrates how reservoir computing may be used for time-series forecasting, providing a low-cost substitute for deep learning models in weather forecasting. Future research might concentrate on expanding the methodology to other meteorological datasets, refining hyperparameters, and adding more variables.

1.1 Background

Data-driven weather forecasting now has more options thanks to the expansion of high-resolution meteorological datasets and developments in machine learning (ML). By modeling intricate, nonlinear interactions in the atmosphere, these methods can produce precise forecasts without only depending on computationally demanding numerical simulations. By using a Reservoir Neural Network (RNN), a kind of reservoir computing model, to anticipate air temperature using comprehensive meteorological data, this study seeks to take advantage of such capabilities.

1.2 Objective

This project intends to create a precise and portable forecasting model that recognizes intricate patterns in meteorological data by utilizing the computational efficiency and temporal modeling capabilities of reservoir computing.

2 Data Description

Description to the Dataset

High-frequency meteorological data collected every 10 minutes during the year 2020 makes up the dataset used in this study. It contains 20 different meteorological indicators and was gathered at a weather station run by the Max Planck Institute. A thorough picture of atmospheric conditions is given by these indicators, which include:

Precipitation, wind speed, humidity, and air temperature are the main weather parameters.

Derived Quantities: Potential temperature, vapor pressure deficit, and other sophisticated meteorological measurements.

With more than 52,560 data points per variable, this high-resolution dataset provides detailed insights into atmospheric dynamics and weather patterns. It is more appropriate for both basic meteorological research and practical forecasting activities when derived quantities are included.

➔Variables

Meteorological Data (Input Features)

The weather indicators used to forecast the target variable (air temperature) are represented by these variables:

- The input vector containing the climatic characteristics at time t is denoted by u(t). There are 20 weather indicators in the dataset, including:
    - **Temperature (T)**: Air temperature, which is the target variable.
    - **Humidity**: Relative humidity of the air.
    - **Wind Speed**: Speed of wind measured at the station.
    - **Precipitation**: Rainfall or snowfall measurements.
    - **Radiation**: Solar radiation or other radiative energy.
    - **Other Derived Variables**: These include vapor pressure deficit, potential temperature, and other meteorological metrics that provide additional context for prediction.

Target Variable

- y(t)y(t): The target variable at time tt, which in this study is the air temperature (denoted as TT). The objective of the model is to predict this temperature value at each time step using the input features over time. This is the value that the model aims to forecast based on various meteorological indicators such as humidity, wind speed, and radiation, among others.

Model Parameters

These variables represent the components that define the structure and behavior of the Reservoir Neural Network:

- n_reservoir: The number of neurons in the reservoir.
- W_in: The input weight matrix. It determines how the input features u(t) are connected to the reservoir neurons. This matrix is initialized randomly and scaled by the input scaling factor.
- W: The recurrent weight matrix of the reservoir. It defines the connections between the neurons in the reservoir. The matrix is initialized randomly, and its spectral radius is adjusted to ensure stability in the reservoir dynamics.
- ρ: The desired spectral radius. It is used to scale the recurrent weight matrix W. The spectral radius controls the stability of the reservoir's dynamics and ensures that it can process time-series data effectively.
- x(t): The reservoir state vector at time t. This represents the internal state of the reservoir at each time step and is updated recursively based on the previous state and the input data.

Training Parameters

These variables are used to train the model:

- **w**: The output weight vector that connects the reservoir states to the target output. These weights are learned through Ridge Regression, which minimizes the error between the predicted and actual target values.
- α: The regularization parameter used in Ridge Regression. It controls the complexity of the output weights and prevents overfitting. A small value of α\alphaα allows the model to fit the training data more closely, while a larger value increases the regularization strength.

Performance Metrics

These variables are used to evaluate the model's prediction accuracy:

- **y^(t)**: The predicted output at time t, which is the forecasted temperature based on the reservoir states and output weights.

- **MSE (Mean Squared Error)**: A metric that quantifies the difference between the predicted values y^(t) and the true values y(t). It is calculated as:

- **R^2 (Coefficient of Determination)**: A metric that indicates how well the model's predictions match the actual data. It is calculated as:

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

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

Data Processing Variables

These variables are used for data preprocessing:

- scaled_feature: The normalized input features, scaled using MinMax scaling to fit within the range [0, 1]. This scaling helps the model converge faster and improves stability.
- scaled_target: The normalized target variable (air temperature), scaled similarly to the input features.
- train_size: The proportion of the dataset used for training. In this case, 80% of the data is used for training, while the remaining 20% is used for testing.

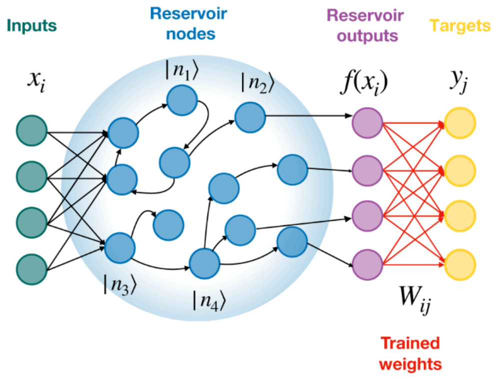
These variables play an essential role in the functioning of the Reservoir Neural Network and its evaluation. By carefully tuning and using them, the model is able to predict air temperature based on historical weather data with high accuracy.

3 Methodology

The methodical process for implementing and assessing the RN for air temperature prediction using meteorological data is described in this section.

3.1 Reservoir computing overview

A computational paradigm for handling temporal data is called reservoir computing. Inputs are mapped into a high-dimensional space via a fixed reservoir, and patterns from the reservoir's states are analyzed using a readout.



Using observed time-series data, reservoir computing is a modern machine learning approach for interpreting information produced by dynamical systems. Crucially, it leverages linear optimization and relatively small training data sets, requiring very little processing power.

This is frequently used for processing temporal data, particularly for applications involving speech recognition, time series prediction, and chaotic system modeling. By merely training a basic output layer and leaving the recurrent portion (the "reservoir") constant, reservoir computing makes training recurrent networks easier. It works especially well with dynamic and nonlinear systems.

Essential Ideas for Reservoir Neural Networks:

- A network of nonlinear nodes with recurrent connections that is fixed and initialized at random.
- Over time, it converts input data into a high-dimensional representation known as a "state space".
- By extracting temporal patterns and nonlinearities from the incoming data, the reservoir functions as a dynamical system.
- Depending on the input and their prior state, reservoir states change.

Layer of Input:

- The input signal is transformed and fed into the reservoir via a layer that is connected to it.
- Frequently, these linkages are fixed and weighted at random.

Layer of Output:

- The reservoir states are mapped to the intended output via a straightforward, trainable linear layer.

Supervised learning, such as ridge regression or linear regression, is used to train only this portion of the network.

A Reservoir Neural Network's Workflow:

Input: The input signal is fed into the reservoir via the input layer.

Reservoir Dynamics: The reservoir updates its states based on the input and internal dynamics

$$x(t + 1) = f\,(W_{res}x(t) + W_{in}u(t))$$

x(t): Reservoir state at time t

u(t): Input signal at time t.

Wres: Reservoir weight matrix.

Win: Input-to-reservoir weight matrix.

f: Nonlinear activation function (e.g., tanh).

Output: The reservoir states are mapped to the output using the trainable output layer:

$$y(t) = W_{out}x(t)$$

- y(t): Output at time t.
- Wout: Trainable output weights.

The reservoir implemented in this study consists of:

- **50 neurons**,
- Randomly initialized input and recurrent weights,
- Input scaling factor of **0.3**,
- Spectral radius of **0.7** for stability.

Reservoir Parameters

1. **Number of Reservoir Neurons (n_reservoir)**:
    - The reservoir consists of **50 neurons**.
    - This value determines the dimensionality of the reservoir state space
2. **Input Weight Matrix (Win)**:
    - The input weight matrix connects the input features to the reservoir neurons.
    - In the code, Win is initialized randomly, with values sampled from a uniform distribution between −0.5 and 0.5.
    - The input weights are scaled by an **input scaling factor** of **0.3** to control the strength of the input signals into the reservoir.

3. **Recurrent Weight Matrix (W)**:
   - The recurrent weight matrix defines the connections between neurons within the reservoir.
   - W is initialized randomly, with values between −0.5 and 0.5.
   - To ensure the reservoir remains stable and exhibits the desired "echo state property," the spectral radius of W is scaled to a predefined value of **0.7**.

4. **Reservoir Update Equation**

$$x(t)=\tanh(Win \cdot u(t)+W \cdot x(t-1))$$

Training the Output Weights

The reservoir itself is not trained. Instead, only the output weights are learned using Ridge Regression. The reservoir states serve as the features for training a linear model to predict the target variable (temperature, T).

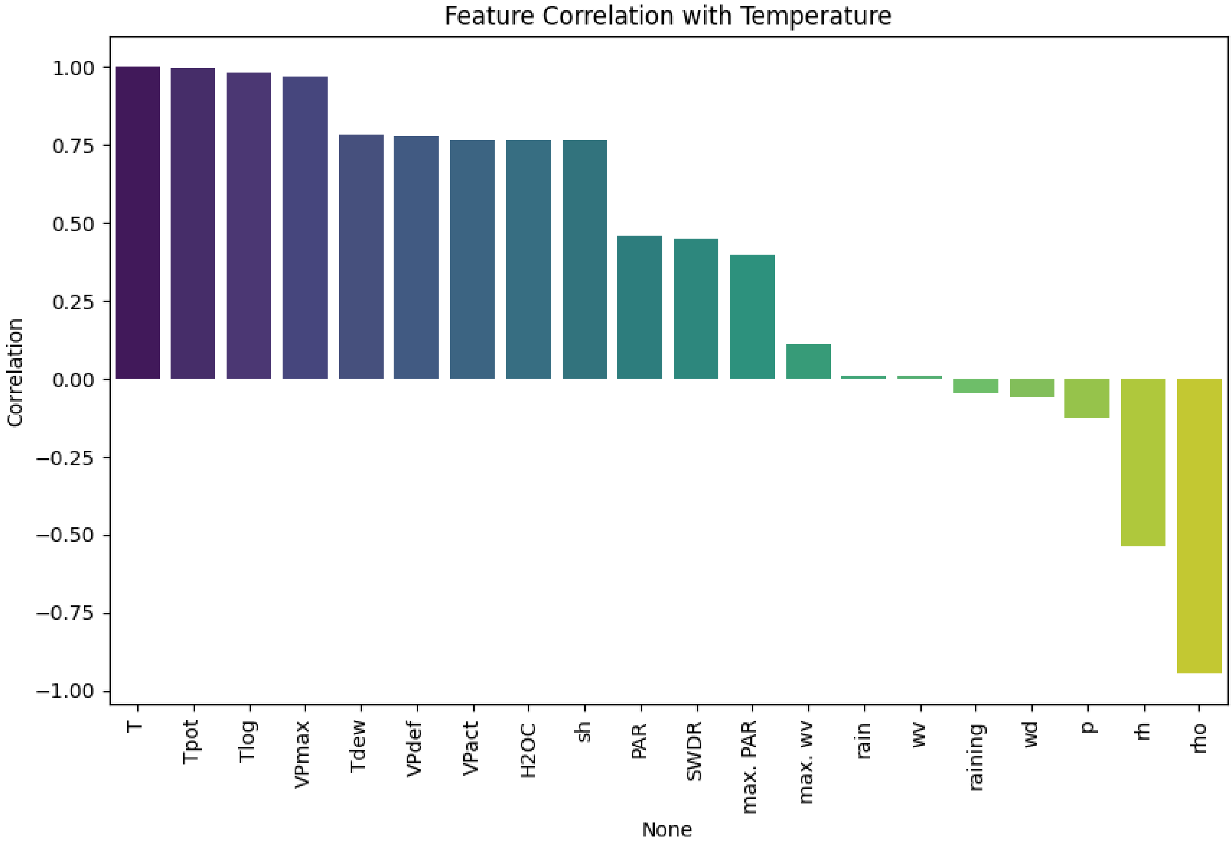
3.2 Data Preprocessing

Preparing the meteorological dataset for the Reservoir Neural Network is the initial stage in the approach. The study's data set includes 20 weather indicators, including temperature, humidity, wind speed, and precipitation, and high-frequency weather measurements taken every 10 minutes during 2020.

Feature Selection: Although the dataset has a number of characteristics, the air temperature (T) is chosen as the goal variable for prediction purposes, with the other variables acting as input features. Since the "date" field has no direct bearing on the prediction, it is removed.

Scaling: MinMax scaling is applied to both the input features and the target variable to guarantee that all features have a comparable scale. By converting the data to a range between 0 and 1, the normalization stage makes sure that every feature makes an equal contribution to the model training process.

Train-Test Split: Training and testing sets are created from the dataset. This division guarantees that the model's generalization ability is tested on unobserved data.



3.3 Prediction and Evaluation

Once the model is trained, it is used to predict air temperature on the test set, and the performance is evaluated using several metrics.

- **Prediction**: The predicted temperature at each time step is computed as the linear combination of the reservoir state and the trained output weights:

$$y\hat{}(t)=x(t)\cdot w$$

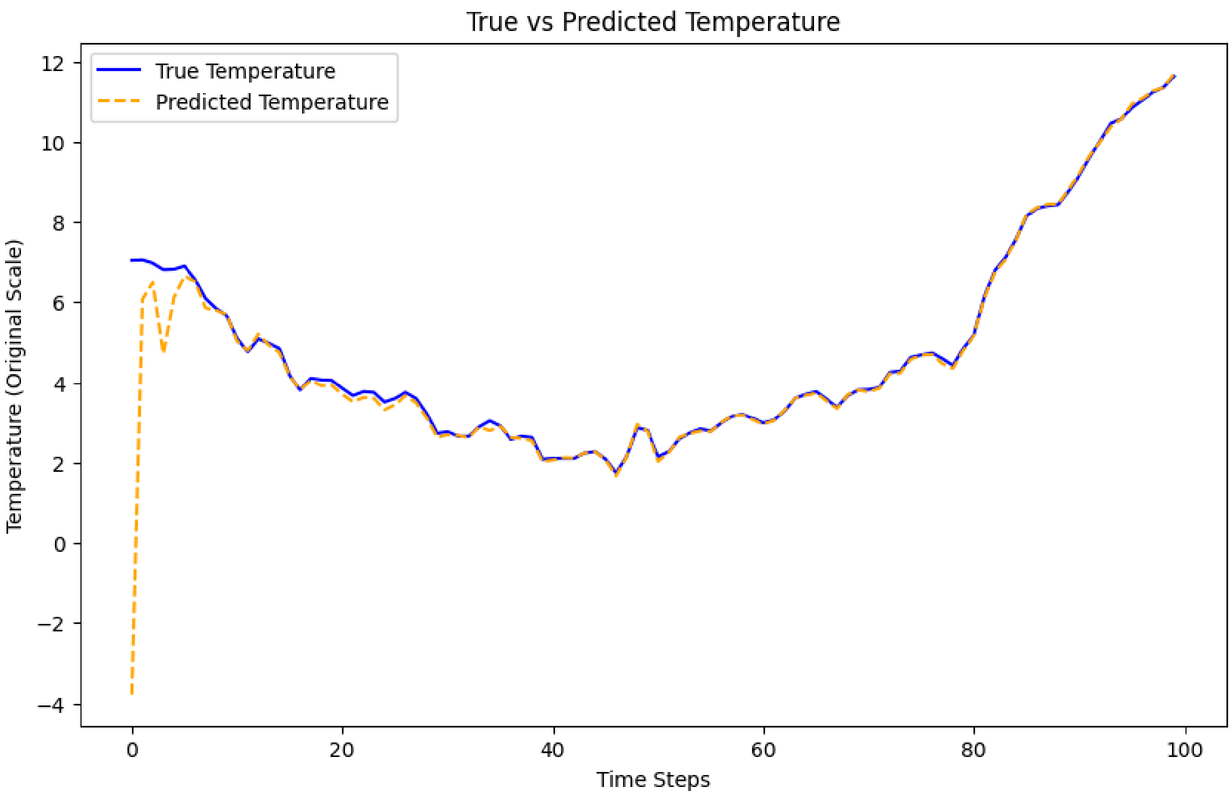
where y^(t) is the predicted temperature and x(t)\mathbf{x}(t)x(t) is the reservoir state at time t.

- **Inverse Scaling**: The predicted and true temperature values are inverse-scaled to restore them to the original temperature scale because the features and target were scaled during preprocessing.
- **Performance Metrics**: The model's performance is assessed using two common evaluation metrics: MSE and R^2 score.

```
MSE: 0.020173872221661313
```

```
R^2 Score: 0.9991891253033003
```

- Visualization: A plot of the actual and expected temperatures is created for easy comparison. This aids in evaluating how well the model represents the temperature's temporal dynamics and trends.



4 Conclusion

Final Thoughts

A Reservoir Neural Network (RNN) was used in this study to forecast air temperature using high-frequency meteorological data. Without the need for iterative training, the reservoir successfully converted input features into a high-dimensional space, capturing temporal dynamics. Ridge Regression helped the model make accurate predictions, as evidenced by its high R^2 score and low mean squared error. This method provides a strong foundation for evaluating time-series data in meteorology and other fields, demonstrating the potential of reservoir computing for effective and scalable weather forecasting.

5 References

https://www.kaggle.com/datasets/alistairking/weather-long-term-time-series-forecasting/data

https://www.nature.com/articles/s41467-021-25801-2

https://quantumcomputinginc.medium.com/neural-networks-and-reservoir-computing-0872d426a7a9