Recurrent Neural Network for Time Series (Weather) Prediction

[redacted]
[redacted]
[redacted]
[redacted]

***Abstract*—In this report, we talk about a Recurrent Neural Network (RNN) model for a time-series weather prediction which is implemented for forecasting the temperature based on univariate meteorological data. The Mean Square Error (MSE) is used to evaluate the performance accuracy of the model.**

***Index Terms*—time-series, weather prediction, RNN, Machine Learning**

I. Introduction

Time series forecasting in Machine Learning refers to the implementation of a model that can that can accurately predict future values based on some previously observed data. A time series model can be widely used for predicting the stock market prices or for finding out the approximate retail sales for a store on a given day as well as be used for heart rate and brain monitoring and much more.
Human beings don't have the need to retrain their brains to interpret any statement. We understand the context of an article based on our prior knowledge of the words and retention power. Similarly, in the world of machines, Recurrent Neural Network or RNN is the first algorithm with an internal memory that remembers its input. This enables RNN to predict what will happen next with a great accuracy which in turn makes it ideal for machine learning problems that involve sequential data [1].
In this report, we will thus be implementing a time series model using Recurrent Neural Networks to accurately forecast the temperature for the Dallas region based on some previously gathered data.

II. Background Work

Weather data is a crucial input for a myriad of applications in the built environment, including building energy modeling and daylight analysis [2]. It is non-linear, sequential, continuous, dynamic and multi-dimensional in nature. Many people have researched about it and have built various models for weather prediction. In general, statistical models have always proven to be more accurate. Neural networks are specifically designed to address the major issues that statistical or traditional methods would find difficult to solve. Additionally, non-linear models have proven to outperform linear models.
RNN is known to specialize in sequence processing and is quite suitable for performing a weather forecast. Also, as said by Lex Fridman from MIT, RNN should be used "Whenever there is a sequence of data and that temporal dynamics that connects the data is more important than the spatial content of each individual frame."

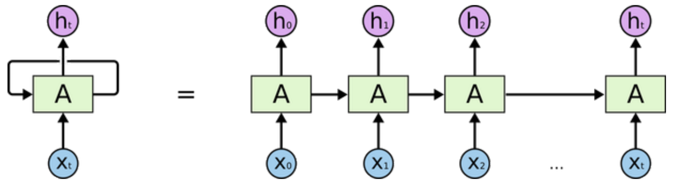


Fig. 1. Basic architecture of Recurrent Neural Networks, Source- [3]

III. Theoretical and Conceptual Study

A. Time Series

In mathematics, a time series is a series of data points indexed in time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time. Thus it is a sequence of discrete-time data [4]. Time series data can be found almost everywhere, as time is a component of just about everything perceptible.
Time series forecasting is the process of predicting future occurrences by evaluating previous trends with the assumption that future trends would be pretty similar to previous trends. It involves predicting future values by fitting models to historical data. Time series forecasting is a data-driven method for effective and efficient planning that is used to solve prediction problems with a time component. Time series forecasts are based on time series analysis, which entails methods for evaluating time series data in order to extract useful statistics and other data features. The goal of forecasting time series data is to predict how the series of observations will continue in the future. Time Series models are found to be one of the most successful methods for forecasting in situations when there is a degree of uncertainty about the future.
Weather or Temperature forecasting is a time series problem in which an estimate is made using data prior to the current time. A one-dimensional data set, as well as a multi-dimensional data set, can be used for weather forecasting. For one-dimensional data sets, temperature estimation is ideally performed using only the temperature variable.In multidimensional data set problems, in addition to temperature data, values such as pressure, wind speed, wind direction, humidity

etc. can be utilized. “Fig. 2” shows an evolution of global temperatures in a time-series representation.

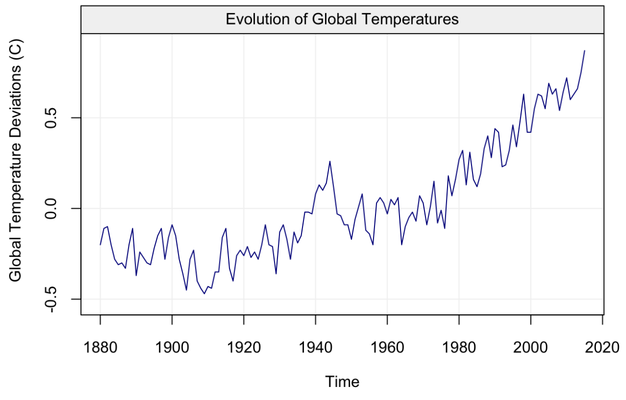


Fig. 2. Time Series Data for Global Temperature, Source- [5]

B. *Recurrent Neural Network (RNN)*

Recurrent neural networks are a type of artificial neural networks that are extensively utilized in speech recognition and natural language processing. Recurrent neural networks recognize the sequential properties of data and use patterns to forecast the next most likely outcome.

RNNs are derived from the Feed-Forward Neural Networks (FNN) with repeated units that feed the outputs of units as inputs in the following step. Whereas the information in a Feed-Forward Neural Network flows only in one direction - from the input layer to the output layer via the hidden layers. The data travels in a straight line through the network, never looping through the same node twice as shown in “Fig. 3”.

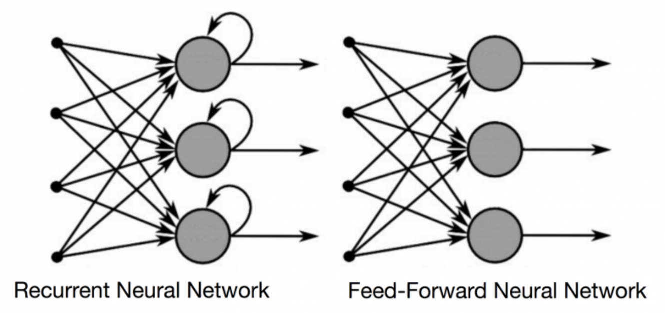


Fig. 3. RNN vs FNN

Feed-forward neural networks have no recollection of the information they receive and are poor predictors of what will happen next. A feed-forward network has no concept of time order because it only analyzes the current input. Recurrent neural networks rely on prior stages as input to the future state and to recall information from the past. As a result, RNNs are suitable for processing sequences of data. Traditional or Convolutional Neural Networks work with only fixed-size inputs and fixed-size outputs while RNN can accept variable-length sequences as inputs and outputs. They learn by updating the weights over time in order to train the model, which requires computing the gradients of the loss with respect to the weights matrix.

The image “Fig. 1” depicts a basic architecture of RNN being spiralled into a full network. Which simply means, it is repeating the same network layer structure for the entire sequence. At time step t, Xt is the input. Xt is an N-dimensional vector. At time step t, A is the hidden state. It’s the network’s ”internal memory”. It is calculated using the previous hidden state and the current step’s input. Represented by At= f (W.Xt + U.At-1). W and U are weights for input and previous state input value, respectively. f is the non-linearity that is applied to the sum to produce the final cell state.

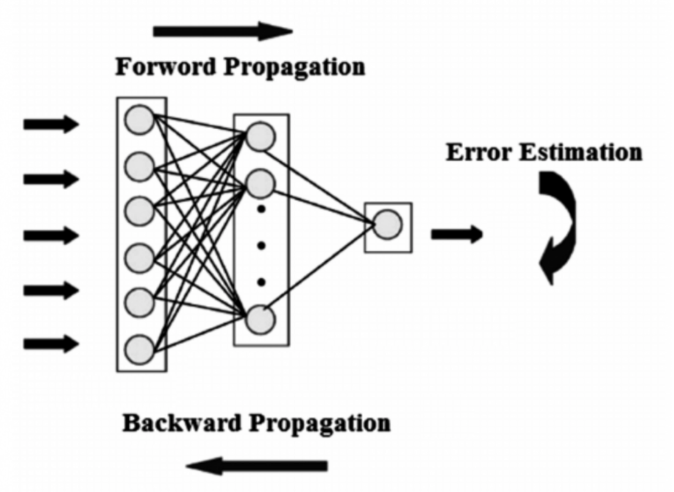


Fig. 4. Propagation in RNN, Source - [1]

RNN is made up of three layers: Input Layer, Hidden Layer and Output Layer. RNN is based on a recursive formula, as shown in equation (1).

$$\mathrm{S}_{\mathbf{t}} = \mathrm{F}_{\mathbf{w}}(\mathrm{S}_{\mathbf{t}-1}, \mathrm{X}_{\mathbf{t}}) \tag{1}$$

In the above formula, S_t refers to the next hidden state. To calculate this value the function F_w is used which depends on the same weight matrix W at each time step involved in the computation. This recursive formula uses the previous hidden state value S_{t-1} along with the input at the current state X_t.

Challenges with Recurrent Neural Networks -

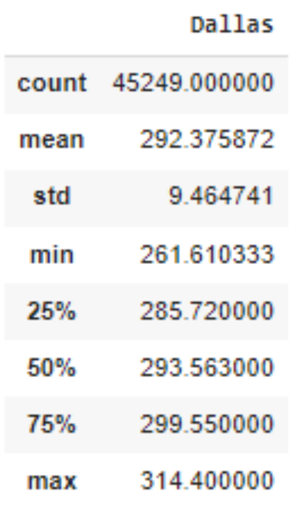
Recurrent Neural Networks generally have to deal with 2 major issues - Exploding Gradient and Vanishing Gradient. The gradient represents the errors that the neural network makes while training. When the gradients begin to explode, the neural network becomes unstable and is unable to learn from training data.

IV. Model Implementation

A. Data Set

The data set used for the experiment contains approximately 5 years of hourly measured data of various weather attributes, such as temperature, humidity, air pressure, etc. This data is available for 30 US and Canadian Cities, as well as 6 Israeli cities [6]. To begin with, we used the hourly temperature data collected during the years 2013 to 2018 for the Dallas region in the United States of America.

The table in “fig. 5” describes the data set used. The image “fig. 6” shows the preliminary visualization of the data, i.e., the hourly temperature plot for the Dallas region recorded from 2013 to 2018.



	Dallas
count	45249.000000
mean	292.375872
std	9.464741
min	261.610333
25%	285.720000
50%	293.563000
75%	299.550000
max	314.400000

Fig. 5. Data Set

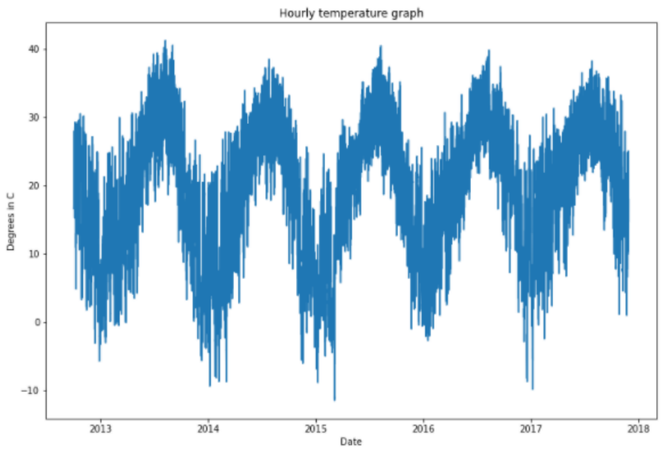


Fig. 6. Dallas - Hourly Temperature

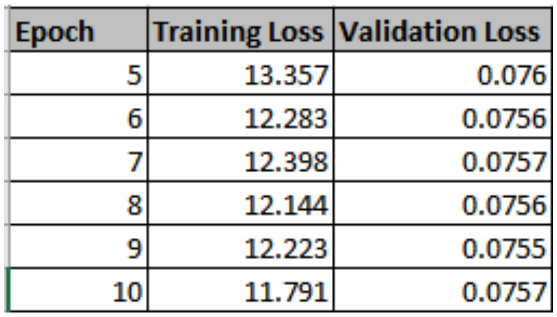
B. Implementation

To implement the RNN model from scratch, we used the pandas and numpy library in the python programming language. We set aside 20 records as our validation data. Our model takes in the input sequence, processes it through a hidden layer of 100 units, and produce a single valued output. We used a learning rate of 0.0001 over 10 epochs. For the network, we defined three weights: U is the weight matrix for weights between input and hidden layers, V is the weight matrix for weights between hidden and output layers, and W is the weight matrix for shared weights in the RNN layer (hidden layer). We have used the Sigmoid function as our activation function which is used in hidden layer. After defining the model, we did a forward pass through our RNN model and calculated the squared error for the predictions for all records in order to get the loss value. We did the same thing for calculating the loss on validation data. Then to actually start training the model, we performed a forward pass to calculate the errors and a backward pass to calculate the gradients and update them.
In the forward pass, we multiplied the input with the weights between input and hidden layers. Then passed it through a sigmoid activation function. Then we calculated the gradients at each layer, and backpropagate the errors. We have used truncated back propagation through time (TBPTT), then we updated the weights with the gradients of weights calculated. Finally to get predictions, we did a forward pass through the trained weights.

C. Evaluation

The Mean Square Error (MSE) evaluation metric is used to calculate the error rate for the model. The error rate indicates how poorly the model has performed. Since the data is preprocessed before using to remove all outliers, the Mean Square Error provides equivalent results as that from the Root Mean Square Error (RMSE) metric. The prediction is the most accurate when the MSE value is low.

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



Epoch	Training Loss	Validation Loss
5	13.357	0.076
6	12.283	0.0756
7	12.398	0.0757
8	12.144	0.0756
9	12.223	0.0755
10	11.791	0.0757

Fig. 7. Model Results

V. Results And Analysis

We evaluated the performance of our RNN prediction model based on the root mean square error (RMSE) and execution time with loss function that represent mean square error (MSE). These parameters have an impact on the performance the model. We have used the last 20 records of the data set as validation data. When tested with validation data the model gives a RMSE score of 0.075 for an epoch of 10. This was found to be the best set of hyper-parameters when tested using various combinations.

The table in "Fig. 7" shows the accuracy for various epochs. From this we can say that with an increasing epoch the accuracy proved to be better for both training and validation. The plot in "Fig. 8" represents the Training Prediction accuracy achieved for the RNN model. Whereas, the plot in "Fig. 9" represents the Validation or the Testing accuracy achieved for the RNN model.

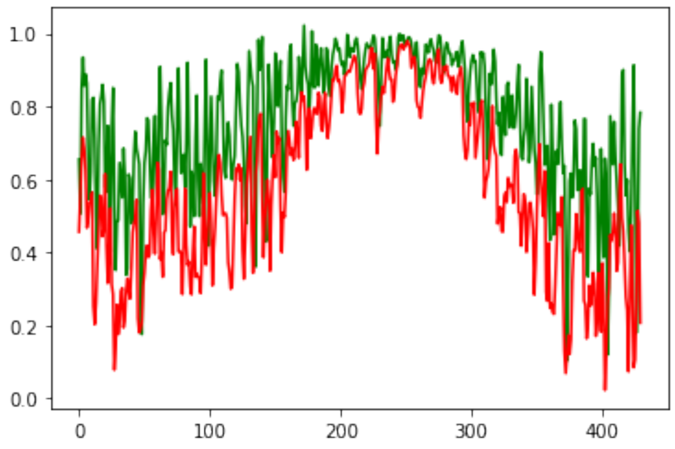


Fig. 8. Training Predictions

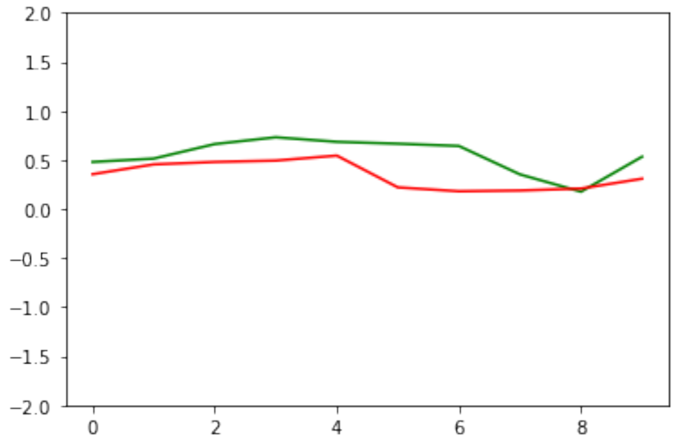


Fig. 9. Validation

VI. Conclusion and Future Work

In this paper, we predicted the weather using one-dimensional or uni-variate data for only one particular city, that is we only used the temperature reading of Dallas city for 5 consecutive years. This model can further be advanced to make predictions using multivariate data which would take into considerations factors like - temperature, pressure, humidity, weather description, wind speed and wind direction. This would further enhance the accuracy of the prediction as all the above mentioned factors combined affect the weather of a region. The model can also be extended to forecast weather for multiple cities.
In our model, we make a short-term weather forecasting using RNN. This can be enhanced further for making a long-term forecast using the Long-Short Term Memory (LSTM) approach.

References

[1] https://builtin.com/data-science/recurrent-neural-networks-and-lstm
[2] https://www.sciencedirect.com/science/article/pii/S0360132321000160
[3] pydeeplearning.weebly.com/blog/basic-architecture-of-rnn-and-lstm
[4] https://en.wikipedia.org/wiki/Timeseries
[5] https://smac-group.github.io/ts/basic-elements-of-time-series.html
[6] https://www.kaggle.com/selfishgene/historical-hourly-weather-data