AI Weatherman

Simplifying the dynamics of weather using neural networks.

Adam Kollgaard, Jared Dewey, and Daniel Miller

03.29.2023

12TH GRADE SCIENCE

ABSTRACT

Weather prediction and forecasting is useful for preparing for hurricanes, planning plant growth, and just making plans outside. Deep Learning networks have been shown to be effective in making predictions (Caruana). In this paper, we attempt to predict how a day's temperature can be predicted using a regression model of deep learning. We collect, clean, and format weather data. We then pass several days worth of data into the network in order to predict the next day's weather. We find that using more days worth of data requires more forms of data (average/minimum/maximum temperature, uv index, moon phase) to make accurate predictions while fewer days worth of data require less forms. We also found that, in general, shorter periods led to more accurate predictions with an average error under 3 degrees celsius.

INTRODUCTION

The world is a more dynamic place than ever before, and with a constantly changing environment, it sometimes seems almost impossible to plan for the future. One of the biggest challenges of planning is dealing with a constantly changing climate. For years, people have created different models, constructed devices, and created diagrams to explain how the weather works, but a new player is in town, AI. The world is taking a turn in a new direction where computer systems can accurately predict and simulate seemingly impossible situations better than humans. At this point, it seems like a no-brainer to apply AI to a problem as complicated as predicting the weather. With that being said, we are going to do the impossible, predict the weather in Nazareth, PA based on the previous day's weather using a regression model.

HYPOTHESIS

It is possible to predict the temperature of the following day using a regression model based on the weather from the previous 3 days.

MATERIALS

1. Weather Data
2. Python
3. TensorFlow
4. Google Sheets

PROCESS

A regression model is a statistical model commonly used to predict values typically based on continuous numbers. For our purposes, we will be trying to predict the following day's average temperature based on several parameters from the previous days. For example, if today is Saturday, then our regression model will be able to predict the average temperature on Sunday using data from the previous 3 days, Thursday through Saturday.

The regression model itself is a deep neural network with multiple layers and a lot of neurons. By using a neural network it is possible to use multiple data points that can help in predicting a single number, the average temperature in Celsius.

To predict the temperature, the following data points from the previous 3 days are needed: month, day, maximum temperature, minimum temperature, average temperature, feels like maximum, feels like minimum, humidity, precipitation, precipitation probability, precipitation cover, snow, wind gust, wind speed, wind direction, sea level pressure.

All of these variables are used to correlate the actual temperature. Additionally, we will try trimming our dataset by creating a reduced version that contains only some of the data points. This enables us to see if certain data points are more useful in predicting future temperatures.

PROCEDURE: Getting and Formatting Data

1. Access historical weather data from a reliable source
    a. Visual Crossing Weather was used
2. Format the data to be a CSV (comma-separated values) file
3. Remove any data that isn't numerical
    a. Ex: Descriptions of weather such as "clear" or "cloudy"
4. Convert any 3 consecutive rows to become just 1 row
    a. For example, given the rows:
        i. 1, 2, 3, 4
        ii. 5, 6, 7, 8
        iii. 9, 10, 11, 12
        iv. 13, 14, 15, 16
    b. They would become:
        i. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
        ii. 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
5. Add an additional column to show what the temperature of the day following those in the sequence is expected to be
6. Save the updated data into a new CSV file
7. Repeat steps 4 through 6 using a different amount of days merged
8. "Deep clean" data so that even fewer numerical values are present
    a. This is to test how much data is needed to accurately estimate the temperature
9. Repeat steps 4 through 7 using this new dataset

PROCEDURE: Making the Network

1. Upload the datasets and remove any unwanted data.
    a. Test for both sets, one set containing all the original data points and another set that has the unwanted data points removed.
    b. One dataset contains data points from 3 days, another for 5 days, and the final for 7 days. Each variant contains another reduced version.
2. Separate the features from the labels.
    a. 80% of the dataset will become a feature, which is a data point used during training, while 20% will become a label which is a data point used during testing.
3. Normalize the data so it is easier for the network to process.
    a. Normalization involves converting each input variable into a distribution between -1 and 1 centered around 0. This enables all variables to share the same scaling making it easier for the model to train.
4. Construct the model using 2 layers with 128 nodes each, ReLU activation, Adam Optimization, batch size of 64, and a variable number of epochs.
    a. The larger datasets use 196 nodes in order to ensure proper training results.
    b. The batch size and the number of epochs can differ depending on the size of the inputs used in the dataset. Use 500 epochs at 128 batch size for 3 days, 50 epochs at 64 batch size for 5 days, and 20 epochs at 64 batch size for 7 days.
5. Run the model using TensorFlow and record the metrics using MSE(Mean Squared Error) and MAE(Mean Absolute Error).

PROGRAM

weather.py > ...

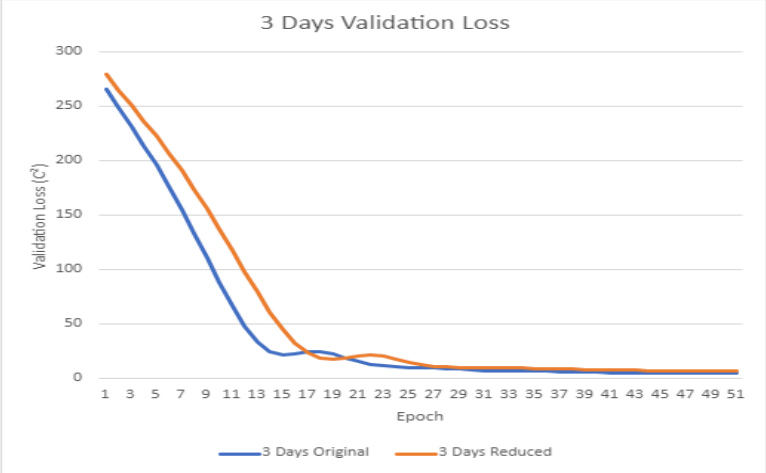
```python
import pandas as pd
import numpy as np

np.set_printoptions(precision = 3, suppress = True)

import tensorflow as tf
from tensorflow import keras
from tensorflow.python.keras import layers
from tensorflow.python.keras.callbacks import History

# Dataset

filePath = 'C:/Users/kollg/OneDrive/Documents/Code/Python/TensorFlow/ScienceFair/day_three_reduced.csv'
column_names = ['tempmax1', 'tempmin1', 'temp1', 'feelslikemax1', 'feelslikemin1', 'feelslike1',
                'humidity1', 'precip1', 'precipprob1', 'precipcover1', 'snow1', 'windgust1',
                'windspeed1', 'winddir1', 'sealevelpressure1', 'severerisk1', 'moonphase1', 'month1', 'day1',
                'tempmax2', 'tempmin2', 'temp2', 'feelslikemax2', 'feelslikemin2', 'feelslike2',
                'humidity2', 'precip2', 'precipprob2', 'precipcover2', 'snow2', 'windgust2',
                'windspeed2', 'winddir2', 'sealevelpressure2', 'severerisk2', 'moonphase2', 'month2', 'day2',
                'tempmax3', 'tempmin3', 'temp3', 'feelslikemax3', 'feelslikemin3', 'feelslike3',
                'humidity3', 'precip3', 'precipprob3', 'precipcover3', 'snow3', 'windgust3',
                'windspeed3', 'winddir3', 'sealevelpressure3', 'severerisk3', 'moonphase3', 'month3', 'day3',
                'expected']

raw_dataset = pd.read_csv(filePath, names = column_names)
dataset = raw_dataset.copy() # Check what the point of this is
dataset = dataset.dropna()

# Seperate training/testing data

train_dataset = dataset.sample(frac = 0.8, random_state = 0)
test_dataset = dataset.drop(train_dataset.index)

train_features = train_dataset.copy()
test_features = test_dataset.copy()

train_labels = train_features.pop('expected')
test_labels = test_features.pop('expected')

# Normalization

normalizer = tf.keras.layers.Normalization(axis = -1)
normalizer.adapt(np.array(train_features))
normalizer.mean.numpy()

first = np.array(train_features[:1])
normalizer(first).numpy()

# Building the model

model = tf.keras.Sequential([
    normalizer,
    tf.keras.layers.Dense(128, activation = tf.nn.relu),
    tf.keras.layers.Dense(128, activation = tf.nn.relu),
    tf.keras.layers.Dense(1)
])

model.compile(loss = tf.keras.losses.mean_squared_error, optimizer = tf.keras.optimizers.Adam(0.001), metrics = ['mse', 'mae'])

history = model.fit(
    train_features,
    train_labels,
    validation_split = 0.2,
    verbose = 2,
    epochs = 500,
    batch_size = 128
)

model.evaluate(test_features, test_labels, verbose = 2)

# Displaying Results
test_predictions = model.predict(test_features).flatten()

error = (test_predictions - test_labels) * (9/5) # Converts to Fahrenheit

tf.print(error)
```

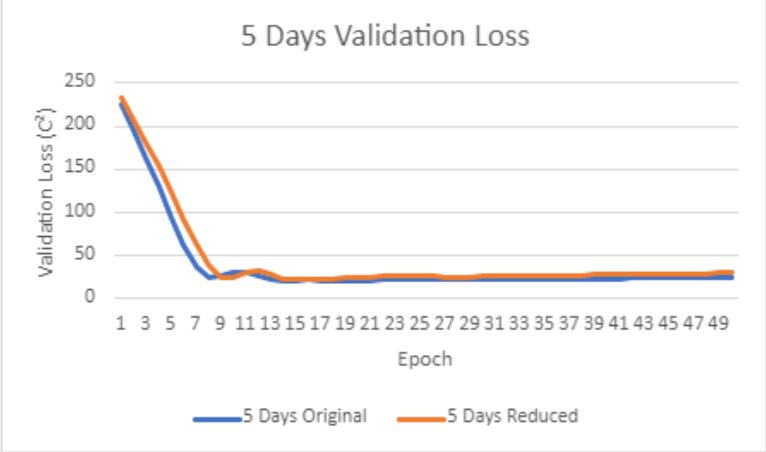
DATA

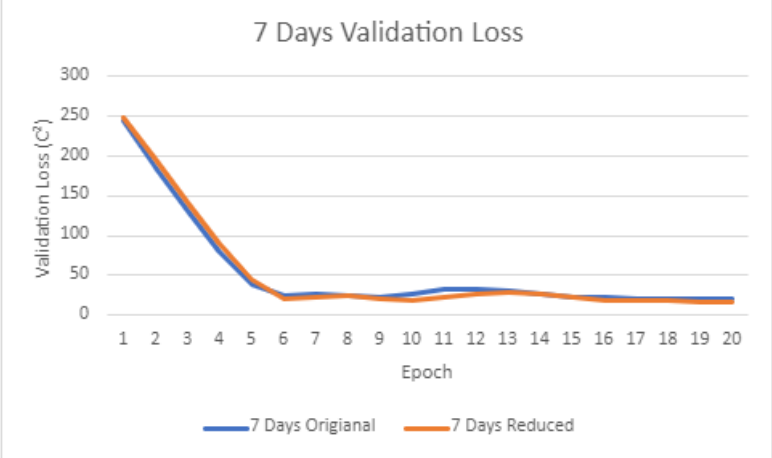
	3 Days	3 Days (Reduced)	5 days	5 days (Reduced)	7 days	7 days (Reduced)
MSE*	4.01 (7.22)	2.63 (4.73)	15.99 (28.78)	18.77 (33.79)	19.68 (35.42)	21.12 (38.02)
MAE**	1.48 (2.66)	1.17 (2.11)	2.83 (5.10)	3.22 (5.80)	3.50 (6.30)	3.69 (6.64)

Lower is better; Celsius(Fahrenheit)
**Mean Squared Error*. The average difference between the predicted value and the actual value squared. Used as the loss function.
***Mean Absolute Error*. The average absolute difference between the predicted value and the actual value.

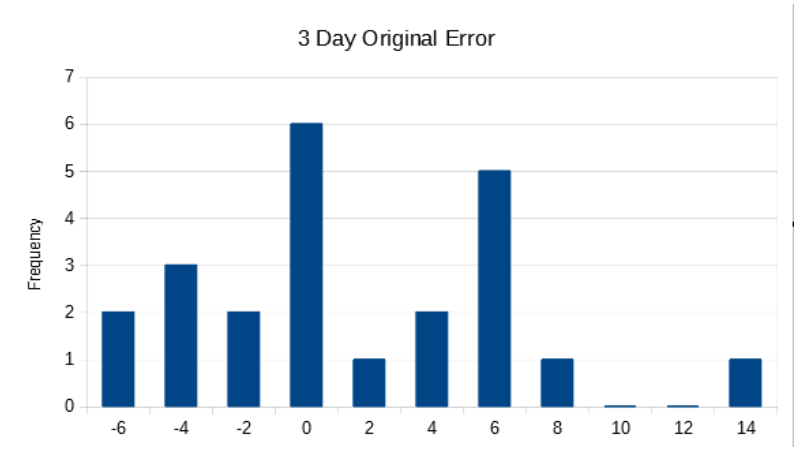
GRAPHS (Validation Loss)

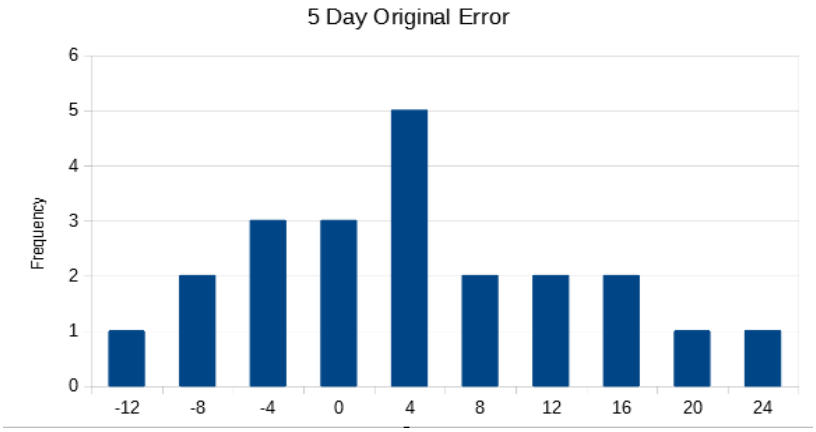


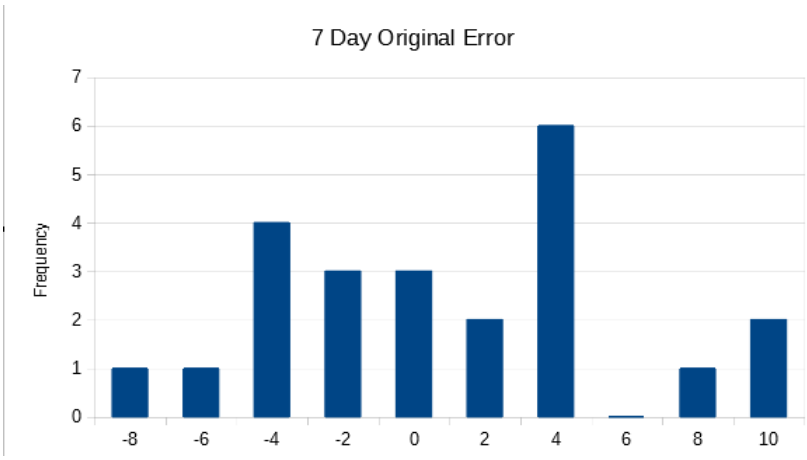




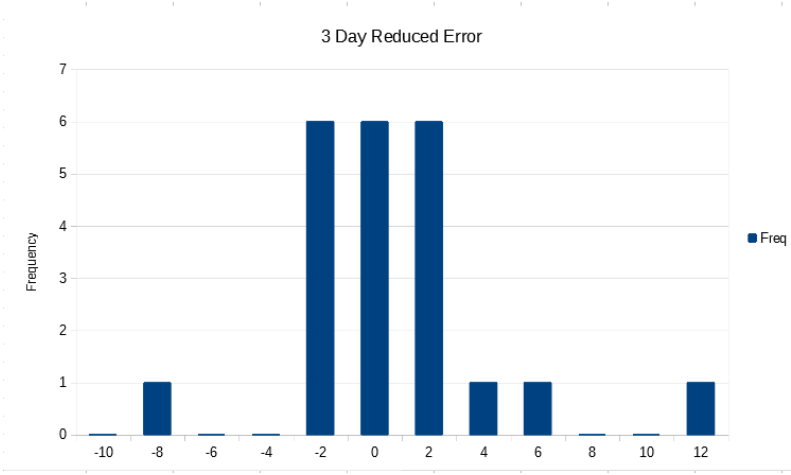
GRAPHS (Original Data Set Error)

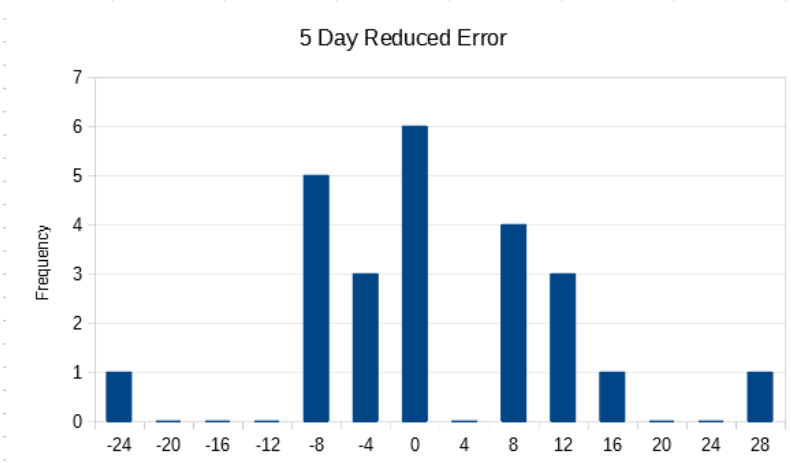


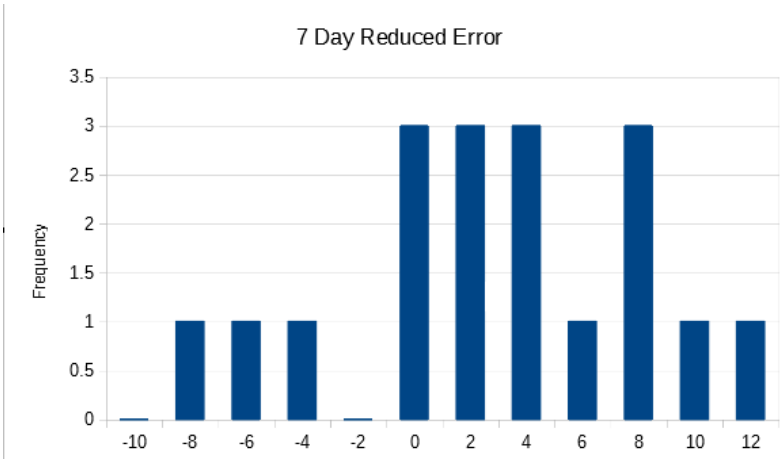




GRAPHS (Reduced Data Set Error)







RESULTS

In total, 12 main data points were gathered representing 3 different lengths of time: 3 days, 5 days, and 7 days. Furthermore, each length of time was divided into two categories based on the amount of input data present. The reduced version does not contain any of the following data: feels like minimum and maximum temperature, precipitation cover, wind direction, sea level pressure, severe risk, and moon phase. However, both categories contained the following data: minimum and maximum temperatures, humidity, precipitation, precipitation probability, snow, wind gust, windspeed, month, and day.

Each of these 6 possible combinations was then measured using two different statistical methods one being mean squared error, MSE and the other being mean absolute error, MAE.

MSE is a common loss function for regression models. MSE is found by taking the average of the square of the difference between the predicted and actual values. This obtains a number that is positive in nature and represents the average error during training. Although the value from the data section was obtained from the testing dataset, MSE is typically used during training because the squaring process speeds up learning by increasing the weight of any individual error.

On the other hand, MAE is found by taking the average of the absolute value of the difference between the predicted and actual values. This obtains a number that is positive in nature and represents the average error between tested data. MAE is useful in determining the actual error represented during testing and comparing the other trials. As a result, it is more reliable to compare each dataset by using the MAE over the MSE.

In order to prove that the network was able to obtain sufficient results, the loss function depicts a downward-sloping curve that indicates that the MSE is dropping as the network trains more and more data. This is done for each of the different datasets to ensure that the network was truly “learning”.

The loss function is the MSE of different batch sizes over the course of multiple epochs. Each epoch is composed of a number of trials known as the batch size. It is customary to have the network run through multiple epochs until a desired loss is met.

Additionally, it is important to consider other statistical methods other than the mean such as the spread. The histograms provided provide a visual representation of the spread of the error found within a portion of the tested dataset. This is done for each of the possible combinations of datasets tested.

ANALYSIS

According to the data, the 3 days reduced showed the smallest error of almost 1.17°C(2.11°F). This is only marginally better than the non-reduced version of 3 days, but substantially better compared to the 5 and 7-day variants, 2.05 and 2.52 respectively. This leads to the connection that data from 3 days ago or more becomes less useful in predicting future weather data. Abstractly, this makes sense because the weather from the previous day should have more weight in predicting the next day as opposed to the weather from a week prior.

An interesting connection is how the reduced versions of the 5 and 7-day variants performed worse than their original versions. One could say that the non-reduced version had more data points to go off from making the previous days more valuable.

In contrast, the 3-day reduced performance was better than the original version. Part of this makes sense considering fewer datapoints could have made it easier for the network to pick out what was more important. In total, the reduced version had to deal with 36 datapoints compared to 57 for the original version.

Another note is on the spread of the data. Although the data tends to be centered around a specific error value there is a chance that the predicted temperature might be significantly lower or higher compared to the actual value. This is part of the overall error of the network itself. Although it tends to be fairly accurate at times it can be almost 20 degrees off. This is important to consider for real-world applications in which reliability becomes extremely important. This could be mitigated by testing a more diverse dataset.

After further analysis of the individually tested errors, there seems to be a greater range of errors during the winter as compared to the spring and summer months which seem to have a smaller error. This could be due to a number of factors such as the changing seasons or even global warming. Either way, the results show a clear indication that the average temperature is most easily predicted using only some data points from the previous 3 days.

CONCLUSION

Although the weather has a lot of variability and chance there are still patterns that arise. A lot of these patterns are even predictable without the need for computers. For example, most people might expect it to rain when it gets cloudy, or to snow during the winter months. However, the ability required to measure the specific temperature on a future day is not shared by most people which is where AI comes in.

Our network was able to successfully predict the average temperature for the following day based on specific knowledge of the previous 3 days with an error of only a couple of degrees Fahrenheit. This degree of precision enables us to further push the boundaries of computer science and meteorology. Soon, the future might have even more accurate predictions leading months into the future.

REFERENCES

1. "Basic Regression: Predict Fuel Efficiency," *TensorFlow,* https://www.tensorflow.org/tutorials/keras/regression
2. Caruana, et al. "Sub-Seasonal Forecasting With a Large Ensemble of Deep-Learning Weather Prediction Models," *Journal of Advances in Modeling Earth Systems.* 25 June, 2021.
3. Hill, Aaron and Schumacher, Russ. "Ai and Machine Learning are Improving Weather Forecasts, but They Won't Replace Human Experts." *Colorado State University.* 26 May, 2022.
4. TensorFlow Guide, *TensorFlow* https://www.tensorflow.org/tutorials/keras/regression
5. *Visual Crossing Weather*, https://www.visualcrossing.com/