Colin Greeley, 1156718

CptS 315, Final Project

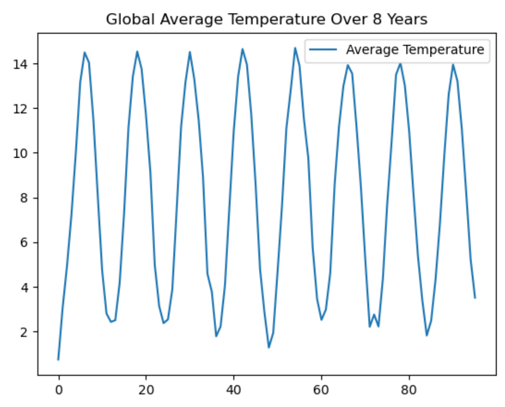
Introduction

The data mining task that I have chosen for this project is creating a recurrent neural network to predict global temperatures at any reasonable time scale. The motivation for this project mostly came from by interest in recurrent machine learning models since I have not worked with them before. Given that recurrent models work on sequential data, I wanted to find a time series data set that best suited my interests. That is when I found a climate change data set from Kaggle which has a record of global temperature data sampled at once a month from 1850 to 2015. From here I was curious if I could create a recurrent model that was able to fit the function that is representative of climate change, meaning that it could accurately predict monthly changes in temperature as well as long stretches since climate change is very gradual.

One of the biggest challenges for me during this project was tuning the hyper parameters of the neural network. There was a significant amount of trial and error for finding an appropriate neural network architecture in terms of layer count, layer size, and layer type. Relative to the specifications of training a recurrent neural network, I was able to implement lookback and delay correctly but I did not get steps working. This means that I was only able to change the sample rate of the dataset, I could only create samples at the default rate which was one sample per month. As a result of the training process, the neural network was able to predict the monthly temperature of any time period given at least two months' worth of data samples.

Data Mining Task

The goal of this process was to create a recurrent neural network that would be able to accurately predict global temperatures for an arbitrary time period, given previous temperature data. The temperature dataset was composed of monthly sample containing the date, average temperature, maximum temperature, minimum temperature, ocean temperature, and all of their uncertainties. In total, there are three independent neural networks, all with the same input data but each is classifying average, max, and min temperatures, respectively. The input for the network would be a sequence of data samples to represent temperature history. The output of each network would also be a sequence of data points representing each month's temperature in the future.



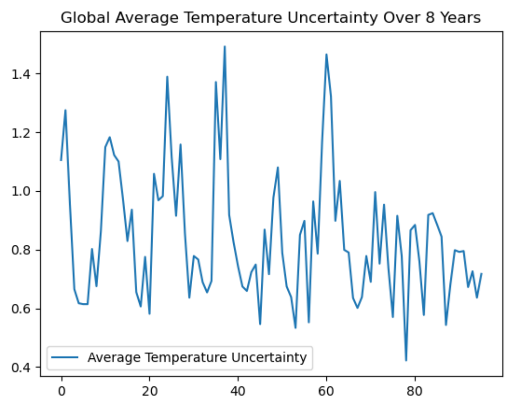


Figure 1.1

Figure 1.2

The output of the neural networks was much better than I was initially expecting it to be. Given only two consecutive months of data, the models were able to predict any amount of time samples slightly higher than 80% accuracy (accuracy being calculated from Mean Averaged Error). Any number of months added to the input of the model only improves the accuracy a small amount for each month, converging around 89%.

Going into this project, I had some knowledge of what recurrent models were capable of, but the main question I had before this project was how far into the future could such a model accurately predict? The answer I found is that it can predict accurately pretty far into the future. I assume that this is majorly due to the fact that there is not much variance in the pattern throughout the dataset. The sequences that we are trying to fit a function to are roughly in the shape of a sine wave as seen in Figure 1.1.

Technical Approach

The incentive behind using a recurrent neural network for this problem is that they are very good at dealing with sequential data. This is because each recurrent layer in the neural network uses the output of a previous state as the input for the current state on top of the data being given for the current state. Each previous state in the neural network is influencing every future state for all states in a sequence, this cycle allows the neural network to have a memory of each state in the input history which is very helpful in making predictions for sequential data.

During the trial and error process for finding the best neural network architecture, I tried using RNN, GRU, LSTM, and Bidirectional layers to see what yielded the best results. The layers type that ended up working the best for this problem was the LSTM (Long-Short Term Memory) layer. Although for the most part, the evaluation metrics were mostly the same.

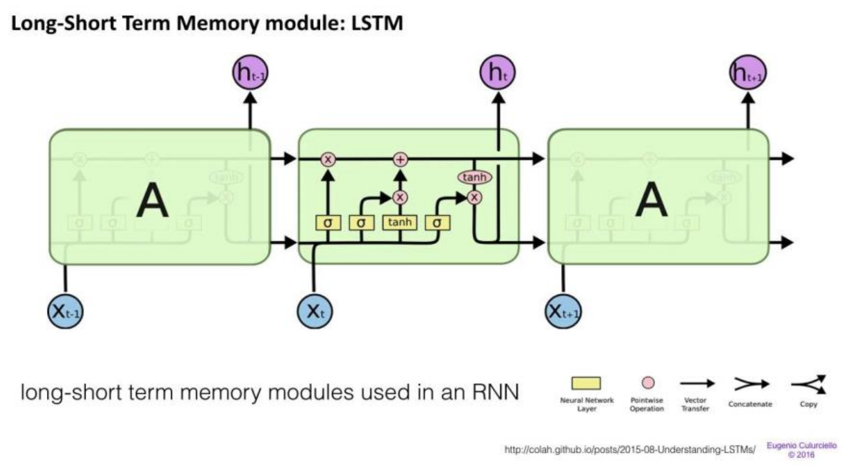

Long-Short Term Memory module: LSTM
long-short term memory modules used in an RNN
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy
http://colah.github.io/posts/2015-08-Understanding-LSTMs/
Eugenio Culurciello © 2016

Figure 2.1

The reason why the LSTM layers may have preformed better if because they are designed to only remember key parts of the past rather than the entirety of it. This should not have made much of a difference for the temperature data because it patten is mostly the same everywhere. Although, there may have been key information in the history of the temperature uncertainties that allowed the LSTM to perform better. Figure 3.1 below shows the model summary of the model created for this project.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, None, 32)          5248
_________________________________________________________________
dropout (Dropout)            (None, None, 32)          0
_________________________________________________________________
lstm_1 (LSTM)                (None, 32)                8320
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense (Dense)                (None, 36)                1188
=================================================================
Total params: 14,756
Trainable params: 14,756
Non-trainable params: 0
```

Figure 3.1

Figure 3.2, code for creating the neural network

```
    model = Sequential()
    model.add(LSTM(32, return_sequences=True,
              input_shape=(None, data.shape[-1])))
    model.add(Dropout(0.1))
    model.add(LSTM(32, activation='relu'))
    model.add(Dropout(0.1))
    model.add(Dense(n_future, activation='linear'))
    model.compile(optimizer='adam', loss='mae', metrics=[accuracy])
```

In Figure 3.2, we can see that the input size for the neural network is the same size as the feature vector representing one sample from the dataset, meaning that the size of the history can be arbitrary. Following the input layer are two LSTM layers with a dropout rate of 0.1 to help prevent falling into a local minimum in the loss landscape. The size out the output is equivalent to the size of the future temperature sequence that we are trying to predict. Since this is a regression problem, the outputs go through a linear activation and we use Mean Averaged Error as our loss function.

Evaluation Methodology

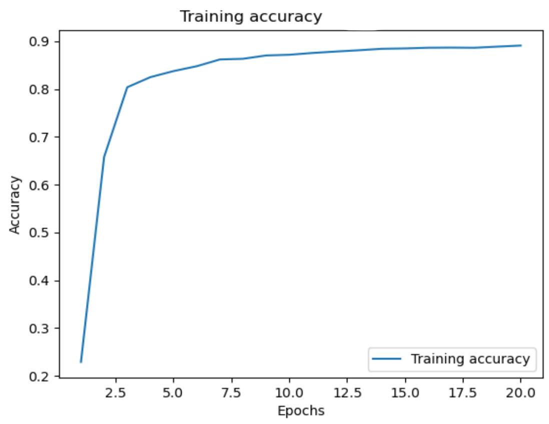
Link to original dataset: https://www.kaggle.com/berkeleyearth/climate-change-earth-surface-temperature-data

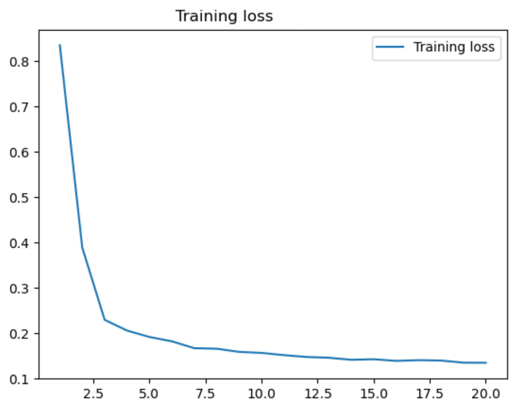
Overall, this dataset was relatively easy to use. The only major issue was that the data from 1750-1850 was useless because it only contained global average temperature so it could not be incorporated to the training set. Given monthly sample from 1850-2015 resulted in 1992 data samples to use. The data from 1850-1999 was dedicated to training while the data from 2000-2015 was used for testing. Due to the small size of the dataset sampled at one reading per year, I was unable to train a network on the yearly change in temperature.

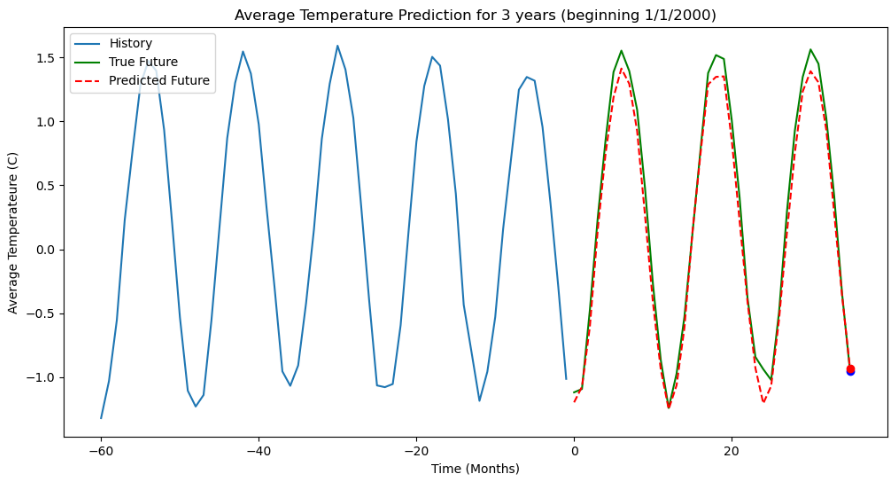
The metrics that were taken from the training and testing processes were the training loss and accuracy over time as well as the testing loss and accuracy. These statistics were measured by comparing the predicted future sequence with the true future sequence of temperatures. For the prediction plots shown below, the temperature data is standardized so that is why the values are so close to zero.

Results and Discussion

The following graphs are based on the three neural networks for average, maximum, and minimum temperatures given the past 5 years to predict the next 3.



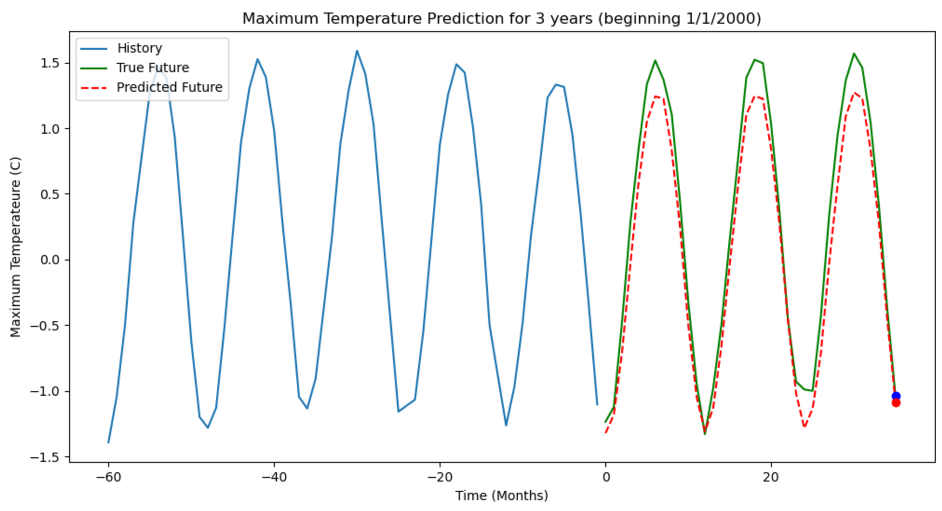




Test loss: 0.0881

Test accuracy: 0.8961

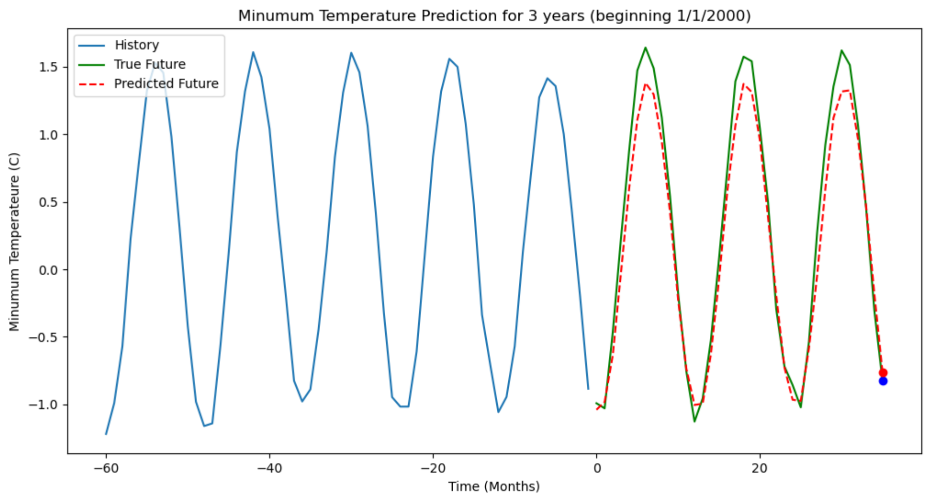
Prediction difference at 3 years: 0.083



Test loss: 0.1940

Test accuracy: 0.8145

Prediction difference at 3 years: 0.0198

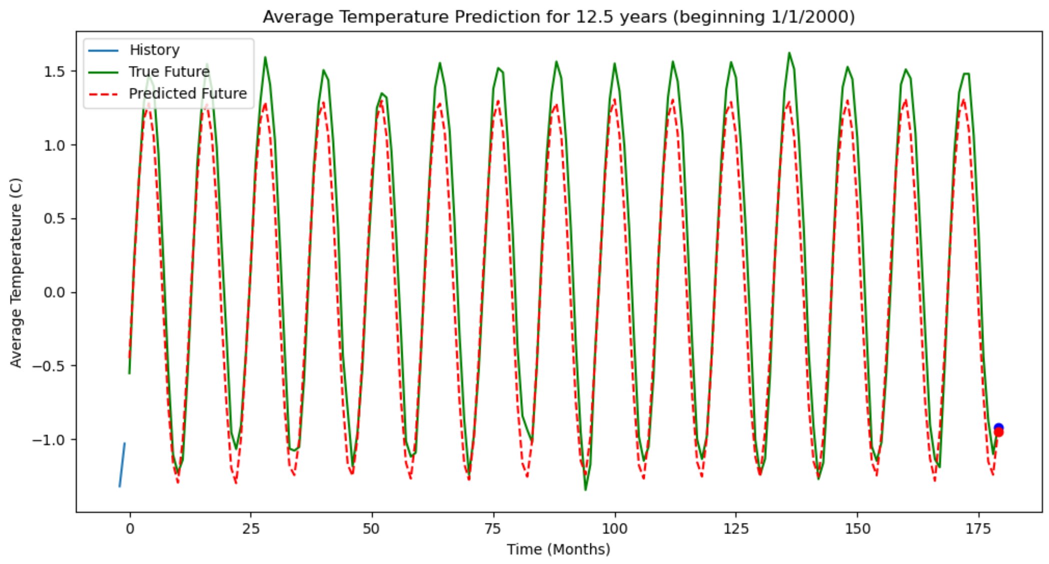


Test loss: 0.1329

Test accuracy: 0.8099

Prediction difference at 3 years: 0.1246

Given only two sample from the two consecutive months before 1/1/2000



Test loss: 0.2143

Test accuracy: 0.8228

Prediction difference at 12.5 years: 0.0809

As seen above, the models had no problem making predictions very far into the future relative to the amount of data given for the prediction. This works be cause with only two data samples to work from, the model has enough information about the data sequence to get the slop at that point in time, giving the rest of the sequence. With this intuition, we can see why the model cannot make accurate predictions with only one data sample. As a matter of fact, the model converges on 50% accuracy when only given one data sample to make predictions.

Lessons Learned

Overall, I have a much more in depth understanding of recurrent neural networks and where they are applicable. Along with that, I understand how to work with sequential data much better and how to break it up into training/testing data and labels.

Acknowledgments

(1) "Time Series Forecasting; TensorFlow Core." TensorFlow, www.tensorflow.org/tutorials/structured_data/time_series.

(2) Kosandal, Rohan. "Weather Forecasting with Recurrent Neural Networks." *Medium*, Analytics Vidhya, 5 Jan. 2020, www.medium.com/analytics-vidhya/weather-forecasting-with-recurrent-neural-networks-1eaa057d70c3.

(3) Chollet François. Deep Learning with Python. Manning Publications Co., 2018.