Using Neural Networks for Weather Prediction

Maxwell Lee

Viterbi School of Engineering, University of Southern California,

Los Angeles, California 90089, USA

(Dated: January 19, 2022)

Abstract

In this report, we will explore many different ways to forecast weather based on previous weather data. Additionally, we will look at a method to understand feature importance in our more sophisticated models. Overall, we are able to verify that our models can predict temperature quite well, both through metrics like mean absolute error, and graphically comparing our predictions to the true labels. However, we see that there are diminishing returns with the complexity of our model. Regardless of scale, we beat our baseline and simple linear models with both a Dense and Recurrent Neural Network. However, the performance between the Dense and RNN is so similar, that the increased complexity of the RNN perhaps isn't necessary for our task.

I. INTRODUCTION

For this task, we are given training and testing datsets of weather data. The data is taken at hourly intervals. Our task is to use preceding data to predict future weather. A part of our task is seeing what future window our model is useful for. We will fit our model trying to look one hour, one day, and 10 days into the future to see if performance diminishes. This task is an example of making use of time series data. Time series data is unique as there is a high level of autocorrelation. Depending on the level of autocorrelation, we may be able to achieve reasonably high levels of success with simple baseline models. Creating these baselines is incredibly important for this task, as we want to make use of the least complex model that performs well for our task. It should be noted that all results borrow heavily from [1].

II. DATA EXPLORATION

The training and testing datasets were read into a Jupyter notebook using the Pandas package. Basic statistics such as mean, minimum, and maximum show no outlier values that need to be fixed. There are however a few missing values in the testing dataset. These are filled with a forward filling methodology. The logic here is that weather data is highly correlated with the preceding hour. If we fail to record a measurement during one of our intervals, a simple and accurate replacement will likely be the preceding hour, moreso than the overall mean, median, or other similar missing value treatments. The other element of note is that weather direction is measured in degrees. This is not entirely useful, so instead in our Data Preprocessing we will convert the wind speed and direction into a 2d vector.

III. DATA PREPROCESSING

As noted previously, the first data preprocessing step was to convert wind speed and direction into a two dimensional vector, which will capture both magnitude and direction of the wind. The next preprocessing step is to translate the datetime information into a more machine friendly format. We note that both date and time are periodic, that is hour 0 (midnight) and hour 23 (11pm) are very similar. The same is true of year, where December 31 is is similar to January 31. To allow for this relationship, both date and time are converted into sin and cosine values. This will allow our models to easily account for the periodic nature we know exists.

Next, we remove the 'weather' column, a textual description of the weather from our dataset. While we could process this data, the necessary steps were felt to be unnecessarily complicated and we are able to achieve quality results without the textual column.

We then split our datasets into a training set, a validation set, and a testing set. There is no overlap between the sets, and instead of a random shuffle, the data is still read sequentially to allow us to simulate the process of receiving the previous timestep's data and using that as an input to the next hour's prediction.

The next step is to normalize the features, subtracting the mean and dividing by the standard deviation so as not to overfit a column by nature of its unit/scale.

Finally, we create a class that allows us to utilize different windows in our fitting and predictions. A window input is the timesteps used as the features in our prediction, while the output is the timesteps we are trying to predict. An example window could be using the previous hour's data to predict the next hour, or using the previous day's data to predict the next 10 days, or however you may like to configure. Because our testing set only has 10 days of data, we will not consider window outputs of larger than 10 days.

IV. ONE STEP WINDOW

We first begin with one step windows. This is using the previous hour to predict the next hour. We start with a baseline, in which we assume the next hour will be exactly the same as the previous. While this may seem naive, it is perfect for a baseline model. The weather tends to change over a broader timescale than just one hour. So while we may be able to achieve great results with a complicated model, it's important to have this baseline to make sure that our complicated model isn't just doing exactly that.

The next model we make is a simple linear model, which takes the previous hour's information and fits a linear model to the next hour's temperature. Looking at the coefficients of this model, the temperature coefficient is near 1. This indicates the model is performing a similar function to our baseline model.

We then begin to try new neural networks. All neural networks are constructed using Keras, fit with the training dataset, validated with the validation dataset, with a maximum of 20 epochs and an early stopping patience of 2. The first neural network we try is a simple Dense neural network with three layers, again operating only with the previous hour's information predicting the next hour's temperature.

We then slightly alter our window, allowing for three previous hours as input for the next hour. We again use a dense NN with three layers, this time adding a flatten step to the 3 hour input.

We then make use of a convolutional neural network. This is very similar to the multi-step dense model, however the CNN allows for there to be variance in the number of inputs, so it need not be 3 hours.

Finally, we use a RNN that allows us to open the timeframe to 24 hours and pass previous inputs to the next input and learn the weights of the passage.

The results of all these models is provided below in Figure 1. We see that all our models beat our baseline, indicating there is useful information beyond the previous hour's temperature. The two best performing models are our simple dense model and our RNN. They perform similarly, and since the RNN is a far more complex model, the most useful model is likely the Dense model, as it would be easier to understand and implement in production.

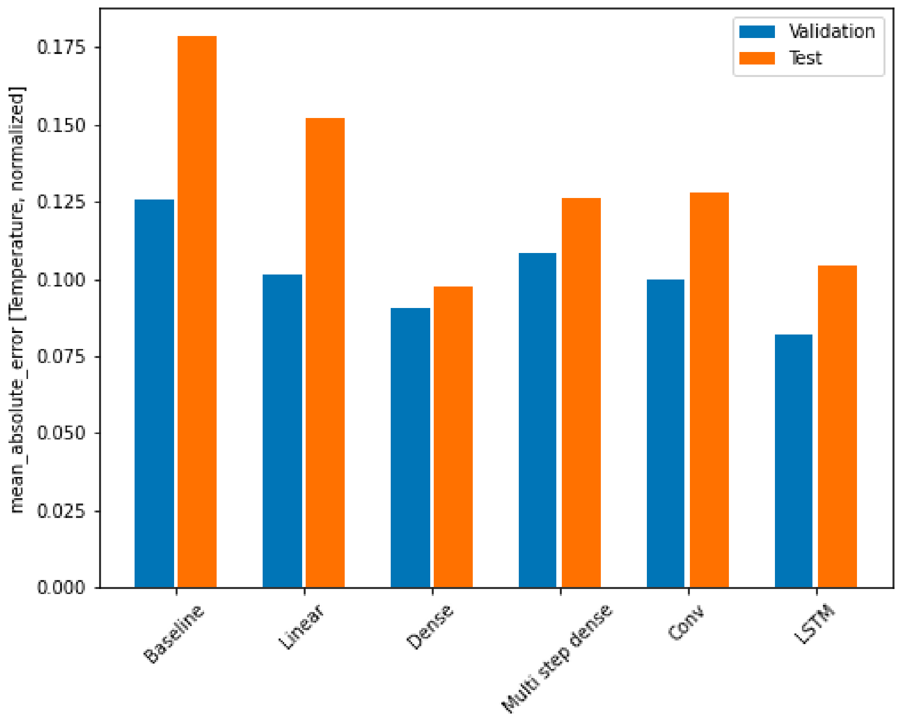

FIG. 1. Results of 1 hour predictions

V. 24 HOUR WINDOW AND 10 DAY WINDOWS

The above process is repeated for a 24 hour window. This would be a day's worth of data predicting the next day's worth of data. We start with two baseline models. The first repeats the last hour's data for the next day. This is admittedly a poor baseline, so we also include a baseline that assumes the entire previous day's data repeats.

We again create linear, dense, convolutional, and RNN models with the 24 hour window. We also create an autoregressive RNN model, which not only passes the internal state but also the prediction of the previous timesteps as inputs to the next prediction.

We see below in Figure 2 that our models beat our baseline, but barely. The assumption that the previous day repeats is only slightly worse than our complicated approaches. We also see there is virtually no difference between the linear, dense, RNN, and AR RNN models, indicating that the linear model would be sufficient for this problem as it is simple, accurate, and interpretable.

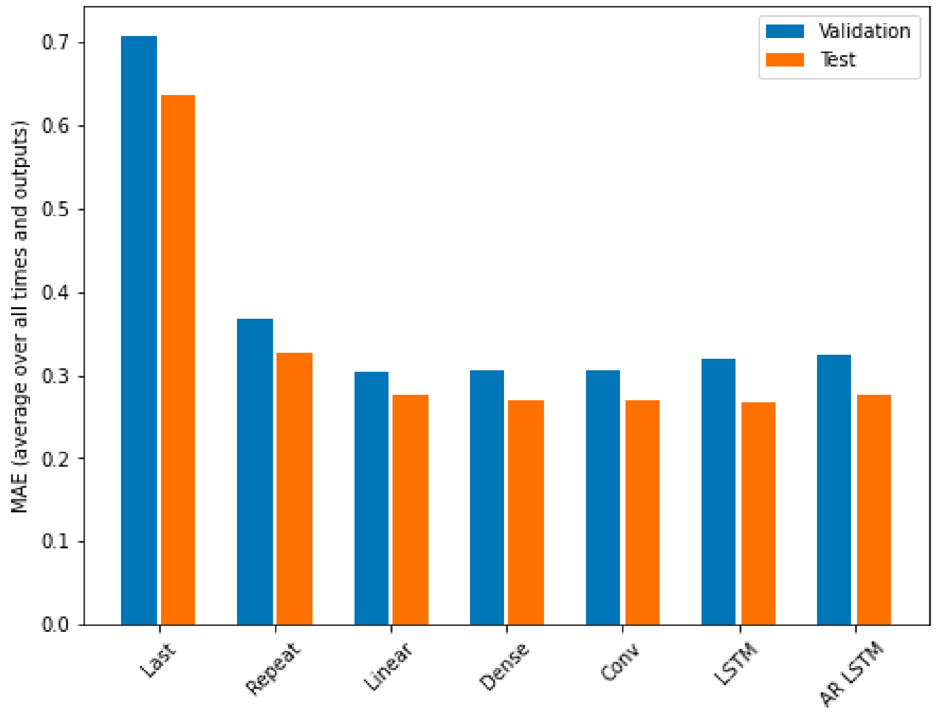

FIG. 2. Results of 24 hour predictions

Repeating this process for a 10 day window yields virtually identical results, indicating that we can have some success predicting at the 10 day level better than our baseline assumptions.

VI. FEATURE IMPORTANCE

To highlight how to obtain feature importance, I chose to obtain feature importance of the LSTM model on the 24 hour window, with the idea being this is a complex model that was reasonably successful, and does not require incredibly long to train. The methodology is that of pertubation importance. First, we have our baseline MAE. Then, for each feature, we randomly shuffle that column, retrain the model, compute it's MAE and divide by the original MAE to obtain by what percentage the MAE as a result of that feature being random.

An important caveat to note is I was unable to successful perform this with temperature. This is due to the fact that temperature is both a feature and the output. Ideally, you would only shuffle temperature when it is an input, however I was unable to devise a methodology clever enough to capture that nuance. So, this method works fine for all other features, and further work could be done to make this work for temperature, which we would believe would be the most important feature.

As shown below in Figure 3, the most important of the non-temperature features is the time of day. Year is not important likely by virtue of the training set. We only have days in November, so it's not very important to know the time of year. A larger dataset may yield a larger importance on year. It should also be noted that because the model takes in two different time and date inputs and only one is shuffled, their feature importance may be even higher.

VII. CONCLUSIONS

As we see, model complexity does not always yield greater results. We are able to create reasonably accurate models that beat our baseline assumptions with linear models and simple Dense Neural Networks. As data scientists, our preference should always be for the simplest model that yields good results, as that leaves less room for error, less time needed for training, and easier production of our models. Finally, we are able to show how pertubation importance can give us an idea of how important our features are to the output of our model. The results align with what makes the most intuitive sense, that time of day would be important to knowing the temperature. This is important as it allows us to have

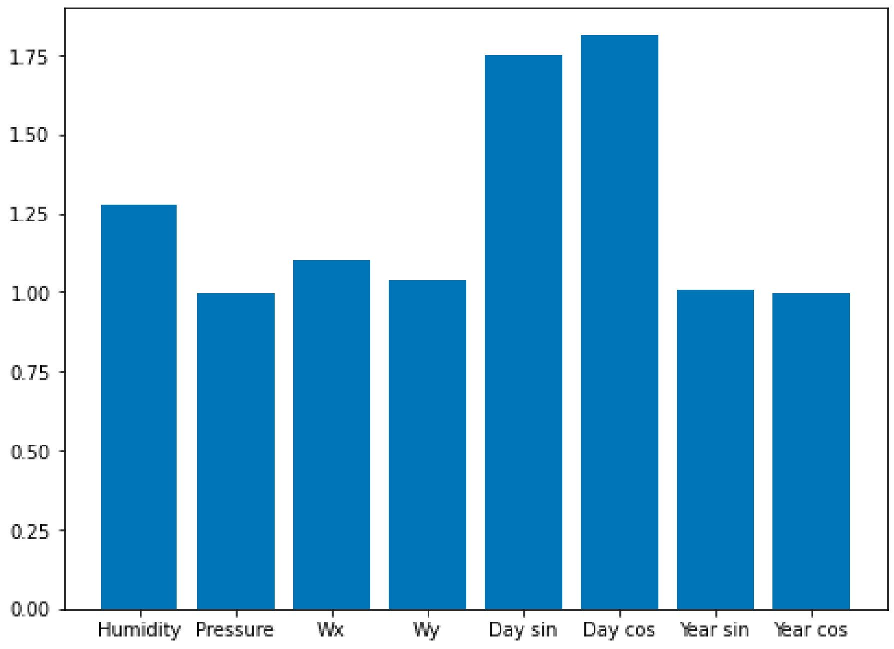

FIG. 3. Feature Importance

more confidence that our complicated models are accurately reflecting the reality we live in.

[1] Time series forecasting : Tensorflow core.