USING MACHINE LEARNING TO PREDICT FUTURE WEATHER FROM PAST WEATHER DATA

Arlou Astrologia

I. INTRODUCTION

The main aim of the mini project was to develop a machine learning technique which can predict future weather from past weather data.

It is important to develop such a technique as knowledge of the weather is almost essential in day-to-day life. The impact of knowing the weather can range from simply helping people determine their schedules, up to the point of saving lives. Knowing the upcoming temperature or amount of rain can help people plan whereas knowing when there will be extreme rain or snow can prevent people driving to work and potentially save lives.

Nevertheless, machine learning is essentially programming a model with the ability to learn rather than explicitly programming the model with what it's meant to learn. For instance, if the goal was to create a model which can do multiplication, one can explicitly programme the multiplication table into the model or one can programme a model with the ability to learn and then train it to be able to perform multiplication. The benefit of machine learning is that the model can make further predictions; for example, the model trained to be able to carry out multiplication can possibly be able to do division, whereas the model which was explicitly programmed will only be able to conduct multiplication.

The foundation of a machine learning model is a neural network. A neural network consists of input, output, and intermediary layers of neurons. If all the neurons in a layer are connected to all the neurons in the adjacent layers, then this layer is called a dense layer and if each layer is a dense layer then the neural network is a fully connected network [1].

For a fully connected network, the input neurons take some input value which are multiplied by some arbitrary constant, called a weight, to produce a new value. Each different connection has its respective weight value. Then, each neuron in the adjacent hidden layer sums this new value for all the input neurons and weights and this sum is added by an arbitrary constant, called a bias. After the bias is added, a non-linear function is applied to the total sum. The values after the non-linear function is applied are the input values to the neurons in this first adjacent layer. This process is repeated for all the neurons in the intermediary layers and is a complicated process, especially for a network with many neurons within many intermediary layers.

A safe and simpler way to describe neural networks would be that it consists of an input and output layer of neurons and in between these layers is a black box which consists of a non-linear function of the weights and biases; this is illustrated in Fig. 1.[2].

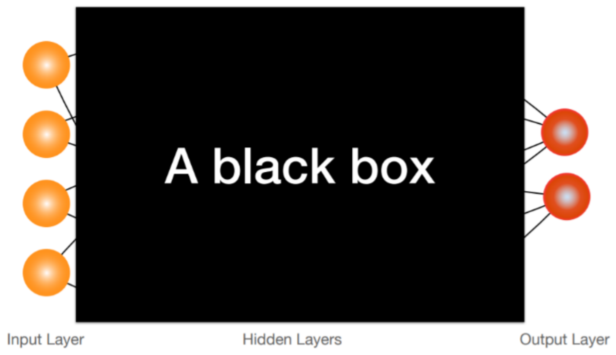

Fig. 1. Schematic of a neural network. Illustrates the neurons in the input and output layer which are represented as circles. The hidden layers can safely be assumed to just be a black box which consists of a complicated non-linear function of weights and biases.

There is another type of neural network called the Recurrent Neural Network (RNN) which is useful when trying to model something that is time dependent. One problem with using RNN's is gradient vanishing [3-4], this is problem was overcome with the development of the LSTM (Long Short-Term Memory) algorithm. The LSTM is essentially an improved version of the RNN and incorporating this into the model allows the model to learn when to store information and when to access it [4].

The values of the parameters of the model, such as the weights and biases, are randomly initialized and therefore the initial output of the model will be random. However, the model can be trained to produce a desired output through stochastic gradient descent. Stochastic gradient descent is essentially finding the values of the parameters which will minimise the loss function. The loss function is the comparison between the output of the model and the desired target; the greater the difference between the two, the greater the loss. This repeated process of calculating the output of a model, calculating the loss and adjusting the parameters of the model is the training process.

In this project, the input to the model was the weather data which comes from the GHCN (Global Historical Climatology Network) which has records of weather data over many stations. The different weather variables from this database are maximum and minimum temperature (C), precipitation (mm), snowfall (mm) and snow depth (mm). These weather variables are referred to as 'TMAX', 'TMIN', 'PRCP', 'SNOW', and 'SNWD', respectively. The weather data consists of the value of the weather variable on a given date. The data for the different weather variables for the station CA002400404, ARCTIC BAY CS, Canada, is shown in Figures 2,3,4,5 and 6.

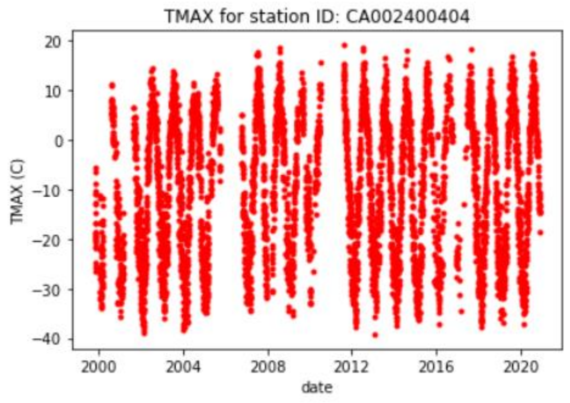

Fig 2. Recorded TMAX data for station CA002400404.

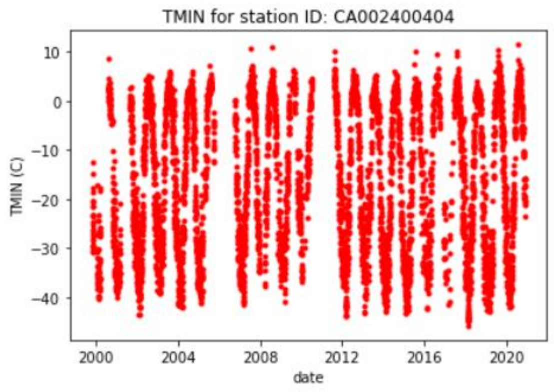

Fig. 3. Recorded TMIN data for station CA002400404.

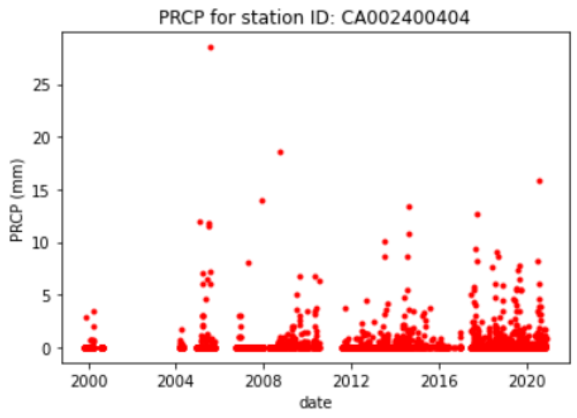

Fig. 4. Recorded PRCP data for station CA002400404.

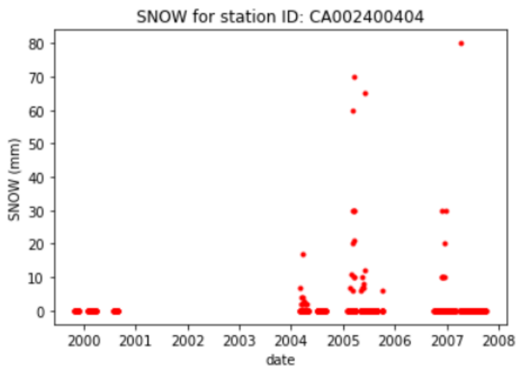

Fig. 5. Recorded SNOW data for station CA002400404.

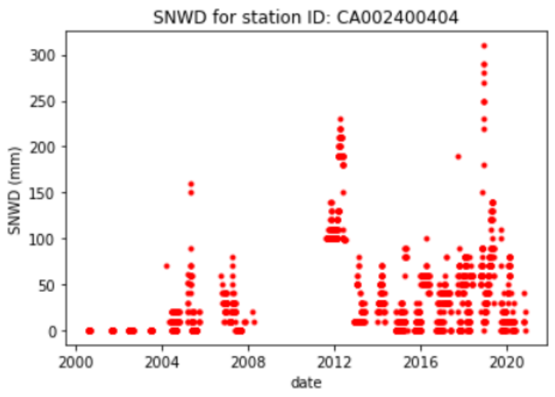

Fig.6. Recorded SNWD data for station CA002400404.

II. METHOD

There were three main goals in this project; one was to be able to develop a machine learning technique which could predict the weather a day in advance, another was to be able to predict the weather a year in advance and then be able to predict the weather ten years in advance.

The essence of the method was the same for each goal; it consisted of extracting the data from the GHCN dataset, converting that data into a form so that it can be used to train the model and then carrying out predictions with the model and comparing these results with the true data to judge the performance of the model.

The data for the different variables were extracted from the dataset using the 'extractdatavalues' function I defined within the code. It stores the data for a given variable in a (n,2) array where n is the amount of recorded data for that weather variable, for a given station. The first column of the array are the dates of the recordings, and the second column are the corresponding values of the weather variable on that day.

Converting the data into training form consisted of firstly transforming this two-dimensional array of data into dataframe format. After the data is in dataframe format, it was prepared in a form ready for training; this consisted of data in a sequence of length n_ts, which was the input to the model, and some data value which is an offset away, which was the target of the model. The training data was also split into a validation data set to reduce overfitting.

When trying to predict the weather one or ten years in advance, I took an extra step which was to take weekly averages of the data, a consequence of doing so was that each datapoint was a weekly average rather than a daily value; this was important to note when setting the offset value. When dealing with daily data, the offset was set to 0 and 364 (365 days in a year) when trying to train the model to predict a day and a year in advance, respectively. With weekly data, the offset was set to 51 (52 weeks in a year) and 519 when training the model to predict a year and ten years in advance, respectively.

After the data had been prepared for training, the model was trained and predictions on future weather were made with the model. These predictions were then compared to the true data to analyse the quality of the model.

The whole process was encapsulated in 2 functions to improve the efficiency of the code and these were the ‘tabulatelist’ and ‘trainingconditionlist’ functions. It was important to set a condition for training, as incorporated in the latter function, as it may be the case that there is not enough training data for a weather variable which leads to an error in the code when attempting to train such data; defining a training condition overcomes this error.

Aside from achieving the goals, another thing that was considered was the configuration of the model as this is what determines the efficiency and effectiveness of the model. Essentially, what we want is from a model is one that produces the best output with the least computing power as possible.

One thing which was considered was the number of epochs in the training, it is important when training a neural network that there are enough epochs because the training is a stochastic process, hence the loss function will not decrease linearly and will fluctuate up and down during the training, hence it is important there are sufficient epochs in the training to compensate for the fluctuations. On the other hand, increasing the number of epochs will take more computational effort and the code will take longer to run.

Another thing considered was the batchsize in the stochastic process, if the batchsize is small, then the code will run faster but the possibility that the loss function will fluctuate strongly during the stochastic gradient descent is increased; a bigger batchsize minimises this problem but takes more computational effort.

Finally, the complexity of the network was also adjusted. The greater the complexity of the network, the more potential it has to achieve better results however this is not always the case and after a certain point, increasing the complexity of the network is just a waste of computational effort.

III. Results & Analysis

The quality of the model which was developed to try to predict the weather a day in advance is illustrated in Fig. 7.

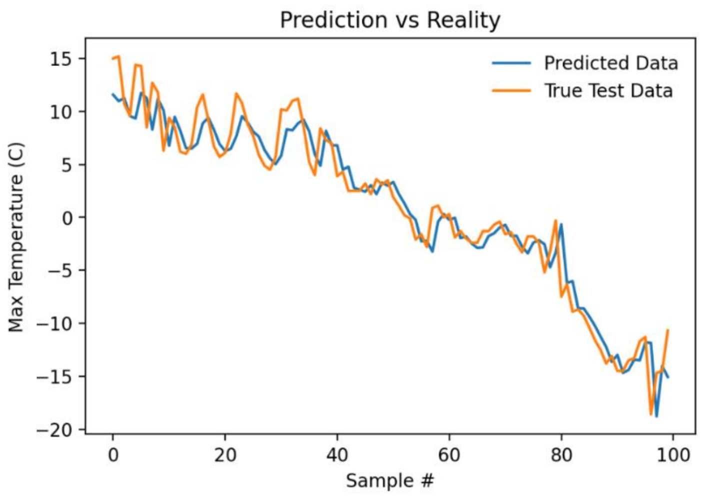

Fig. 7. Plot of model prediction of TMAX data in blue and the true data in orange. The model was trained to predict 1 day in advance. The training data exposed to this model was the TMAX daily data from station CA002400404 and was prepared with an offset of 0 and n_ts=7. The epochs in the training was 10 and the batchsize=64.

We can see from the graph that the model predictions are of good quality; the model predicts the general trend of the data and picks up all of the fluctuations in the data as well. However, the predicted and true fluctuations differ in amplitude slightly and the timing of the predictions also differs to the true data

One problem with this approach was that we do not know if the model was trained to predict the next day TMAX. The data has 'holes' in it, in other words the station may not have data for TMAX on a given day. Therefore, when training the data, the input data and target data may not necessarily be 7 consecutive days and the following day, respectively. For instance the station can record data for Monday to Saturday, then the next record is the following Monday and Thursday. In this scenario, the input data would be the Monday-

Saturday and the following Monday and the offset data would be the following Thursday data.

Therefore, we cannot exactly say that the model is trained over a weeks' worth of TMAX data to predict the TMAX of the following day; a more accurate statement of what was done is that the model has been trained over 7 days of TMAX data to predict the TMAX of some day in the future.

The quality of the model which was developed to try to predict the weather a year in advance is illustrated in Figures 8,9,10 and 11.

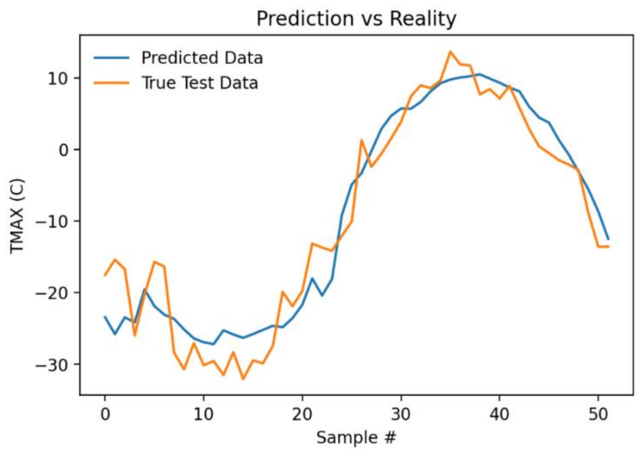

Fig. 8. Plot of model prediction of TMAX data in blue and the true data in orange. The model was trained to predict 1 year in advance. The training data exposed to this model was the weekly averaged data from station CA002400404 and was prepared with an offset of 51 and n_ts=8. The epochs in the training was 20 and the batchsize was the nearest integer after diving the number of datapoints, for this variable, by 4.

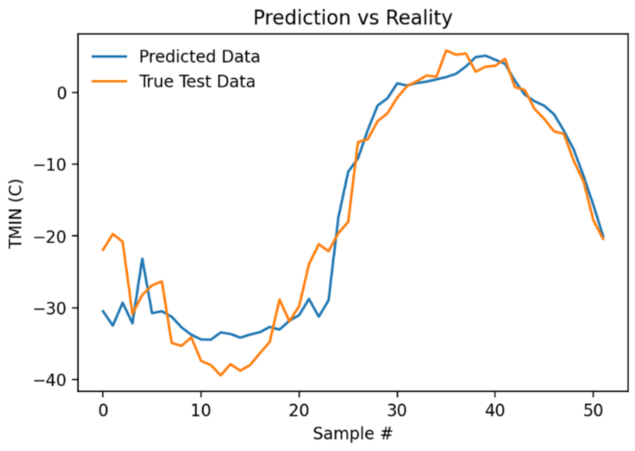

Fig. 9. Plot of model prediction of TMIN data in blue and the true data in orange. The model was trained to predict 1 year in advance. The training data exposed to this model was the weekly averaged data from station CA002400404 and was prepared with an offset of 51 and n_ts=8. The epochs in the training was 20 and the batchsize was the nearest integer after diving the number of datapoints, for this variable, by 4.

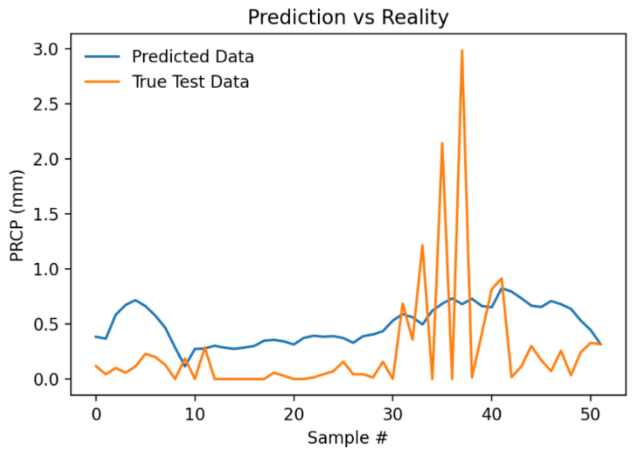

Fig. 10. Plot of model prediction of PRCP data in blue and the true data in orange. The model was trained to predict 1 year in advance. The training data exposed to this model was the weekly averaged data from station CA002400404 and was prepared with an offset of 51 and n_ts=8. The epochs in the training was 20 and the batchsize was the nearest integer after diving the number of datapoints, for this variable, by 4.

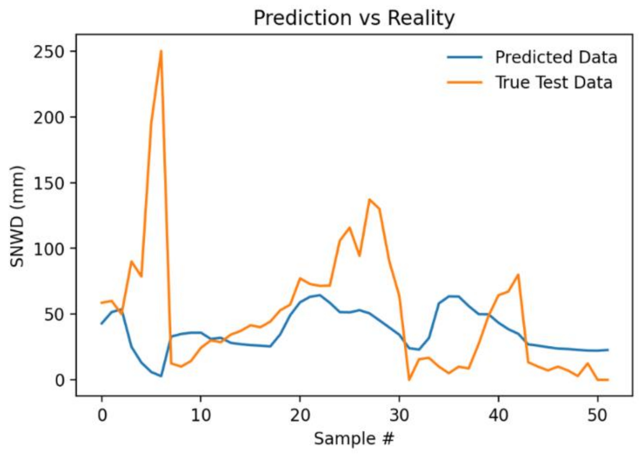

Fig. 11. Plot of model prediction of SNWD data in blue and the true data in orange. The model was trained to predict 1 year in advance. The training data exposed to this model was the weekly averaged data from station CA002400404 and was prepared with an offset of 51 and n_ts=8. The epochs in the training was 20 and the batchsize was the nearest integer after diving the number of datapoints, for this variable, by 4.

There was insufficient weekly data to be able to train a model for SNOW and hence did not achieve any predictions for this variable.

It can be observed that the model has been able to identify the general trend of the TMAX and TMIN data and a few of the fluctuations in the data, however, still has plenty of room for improvement.

We can see that the model does not predict as well for the PRCP and SNWD variables as they do for TMAX and TMIN. The model somewhat predicts the trend in the data however not at a good quality and does not pick up the fluctuations well either.

We also see that the model prediction of TMAX a year in advance is significantly better when trained over weekly data rather than daily data. The model prediction of TMAX a year in advance when being trained over daily data is shown in Fig. 12.

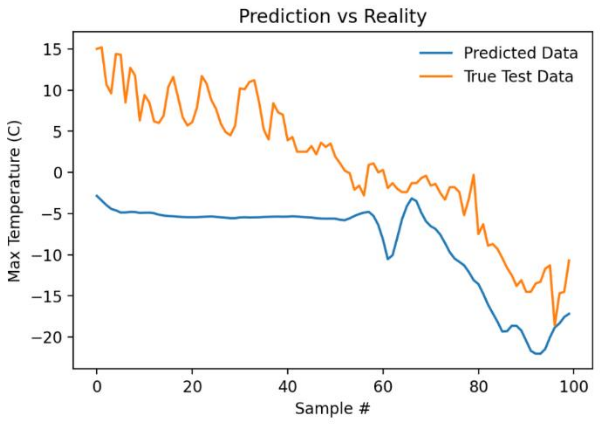

Fig. 12. Plot of model prediction of TMAX data in blue and the true data in orange. The model was trained to predict 1 year in advance. The training data exposed to this model was the TMAX daily data from station CA002400404 and was prepared with an offset of 364 and n_ts=30. The epochs in the training was 10 and the batchsize=64.

The quality of the models which were developed to try to predict the weather ten years in advance is illustrated in Figures 13,14,15,16,17 and 18.

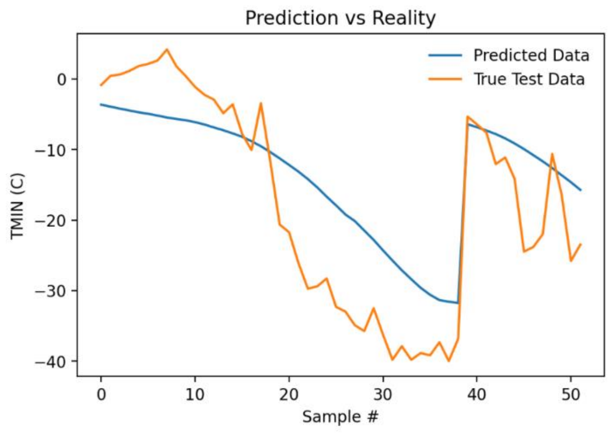

Fig. 13. Plot of model prediction of TMIN data in blue and the true data in orange. The model was trained to predict 10 years in advance. The training data exposed to this model was the weekly data from station CA002400404 and was prepared with an offset of 519 and n_ts=52. The epochs in the training was 20 and the batchsize was the nearest integer after diving the number of datapoints, for this variable, by 4.

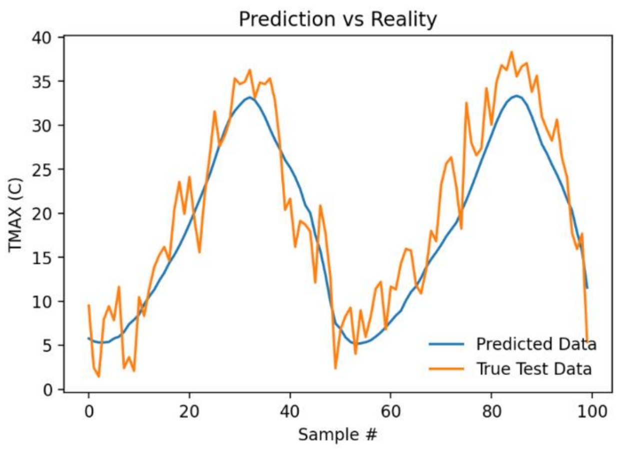

Fig. 14. Plot of model prediction of TMAX data in blue and the true data in orange. The model was trained to predict 10 years in advance. The training data exposed to this model was the weekly data from station USW00024128 and was prepared with an offset of 519 and n_ts=52. The epochs in the training was 20 and the batchsize was the nearest integer after diving the number of datapoints, for this variable, by 12.

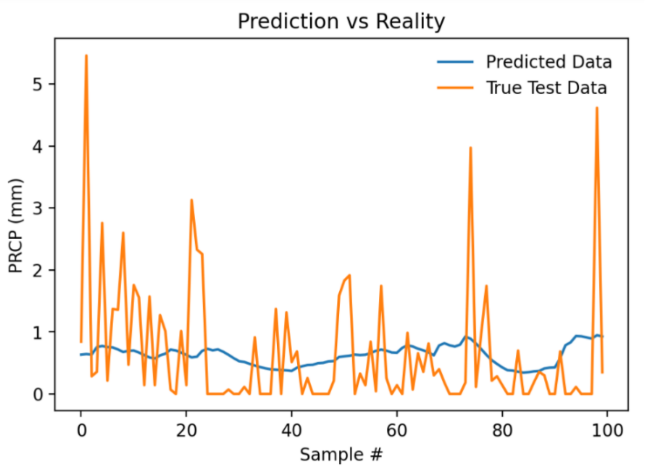

Fig. 16. Plot of model prediction of PRCP data in blue and the true data in orange. The model was trained to predict 10 years in advance. The training data exposed to this model was the weekly data from station USW00024128 and was prepared with an offset of 519 and n_ts=52. The epochs in the training was 20 and the batchsize was the nearest integer after diving the number of datapoints, for this variable, by 12.

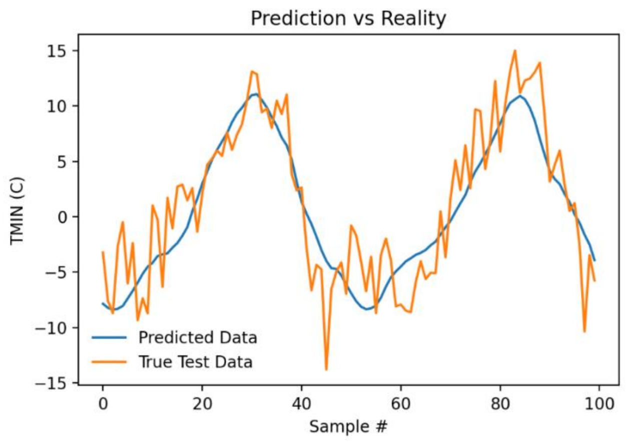

Fig. 15. Plot of model prediction of TMIN data in blue and the true data in orange. The model was trained to predict 10 years in advance. The training data exposed to this model was the weekly data from station USW00024128 and was prepared with an offset of 519 and n_ts=52. The epochs in the training was 20 and the batchsize was the nearest integer after diving the number of datapoints, for this variable, by 12.

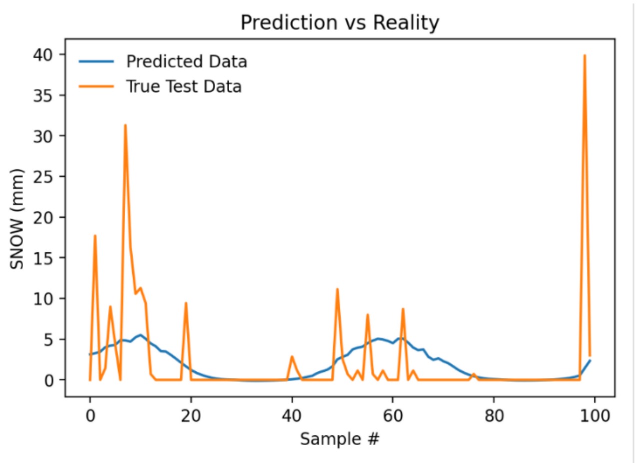

Fig. 17. Plot of model prediction of SNOW data in blue and the true data in orange. The model was trained to predict 10 years in advance. The training data exposed to this model was the weekly data from station USW00024128 and was prepared with an offset of 519 and n_ts=52. The epochs in the training was 20 and the batchsize was the nearest integer after diving the number of datapoints, for this variable, by 12.

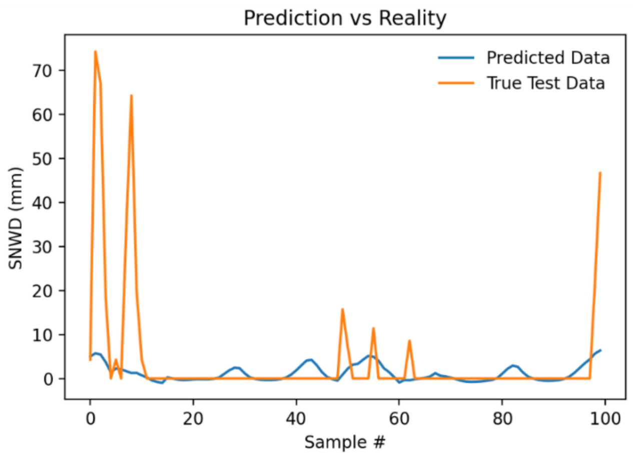

Fig. 18. Plot of model prediction of SNWD data in blue and the true data in orange. The model was trained to predict 10 years in advance. The training data exposed to this model was the weekly data from station USW00024128 and was prepared with an offset of 519 and n_ts=52. The epochs in the training was 20 and the batchsize was the nearest integer after diving the number of datapoints, for this variable, by 12.

There are two important observations we can make from Figures 13-18. One of which is made from figures 13 and 15; the difference between the two models here is that the model which produces the predictions seen in 15 is trained over the TMIN data from station USW00024128 which has multiple times more TMIN data than station CA002400404. Therefore, we can observe that the predictions of a model appears to be better when trained on a dataset with more values, hence providing evidence that the greater the quantity of training data, the better the model predictions which makes sense as there would be more data which the model can be trained on, hence leading to better training and therefore performance.

The other important observation, which is also a recurring observation throughout the results is that the models do not predict the other variables as well as they do the temperature variables. A potential reason for this could be the order of the training; the reason for the low quality predictions of the other variables may be because I'm training the model on the temperature data first. Another reason could be that the other variables do not follow a pattern like TMAX and TMIN do, hence making it more difficult to train the model over these variables. Evidence suggests that the reason is likely to be the latter since I also trained a model on just the PRCP data and the results are shown in Fig. 19; we can see that there are no significant improvements to the results when training the model in this way.

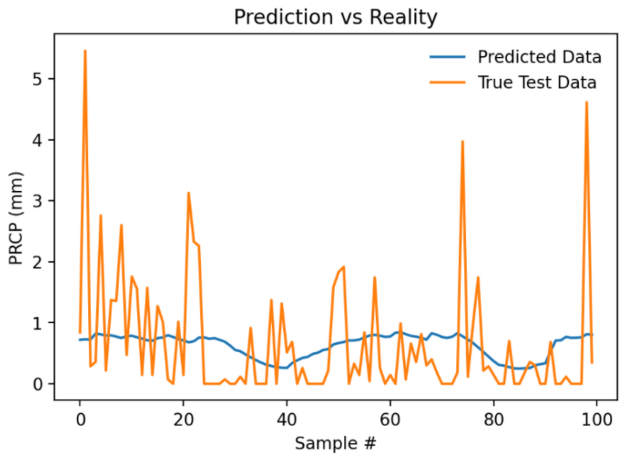

Fig. 19. Plot of model prediction of PRCP data in blue and the true data in orange. The model was trained to predict 10 years in advance. The training data exposed to this model was **only** the PRCP weekly data from station USW00024128 and was prepared with an offset of 519 and n_ts=52. The epochs in the training was 20 and the batchsize was the nearest integer after diving the number of datapoints, for this variable, by 12.

IV. CONCLUSIONS

So in conclusion, I developed a machine learning technique which consisted of a mixture of dense and LSTM layers and used stochastic gradient descent to train it over weather data from different stations.

The technique was able to predict some elements of the weather one year and ten years in advance. The machine learning technique was able to predict the TMAX and TMIN weather variables well with the quality of the predictions decreasing as the future time frame increases.

However, the technique was not able to make good quality future predictions on the PRCP, SNOW and SNWD variables which I believe is because the data of these variables do not form a strong pattern which the model can pick up.

It was also observed that the quantity of training data has an effect on the quality of predictions; we saw that when there is more training data, the quality of the model predictions improve.

One improvement I could make to the project would be to search for stations which do show a strong pattern in the data for the variables PRCP, SNOW and SNWD. I believe that if there is such a station, and there is sufficient data for training, a model can just be trained over the data from just this station and be able to predict the weather data for any other station.

REFERENCES

[1] Ryan, N. 2021. "Week1IntrotoNeuralNetworks". [Online]. UCL Moodle Resource. Available from: https://moodle.ucl.ac.uk/pluginfile.php/3414197/mod_resource/content/6/Week1IntroToNeuralNetworks.pdf [Accessed 10/01/2021]

[2] Ryan, N. 2021. "Week2NetworkTraining". [Online]. UCL Moodle Resource. Available from: https://moodle.ucl.ac.uk/pluginfile.php/3470799/mod_resource/content/2/Week2NetworkTraining.pdf [Accessed 10/01/2021]

[3] Ryan, N. 2021. "Week6_RNN". [Online]. UCL Moodle Resource. Available from: https://moodle.ucl.ac.uk/pluginfile.php/3307466/mod_resource/content/6/Week6_RNN.pdf [Accessed 10/01/2021]

[4] Huang, D. et al., 2017. Drug–drug interaction extraction from biomedical literature using support vector machine and long short term memory networks. *Information sciences*, 415-416, pp.100–109. [Online]. Available from: https://www-sciencedirect-com.libproxy.ucl.ac.uk/science/article/pii/S002002551730110X [Accessed 10/01/2022]