

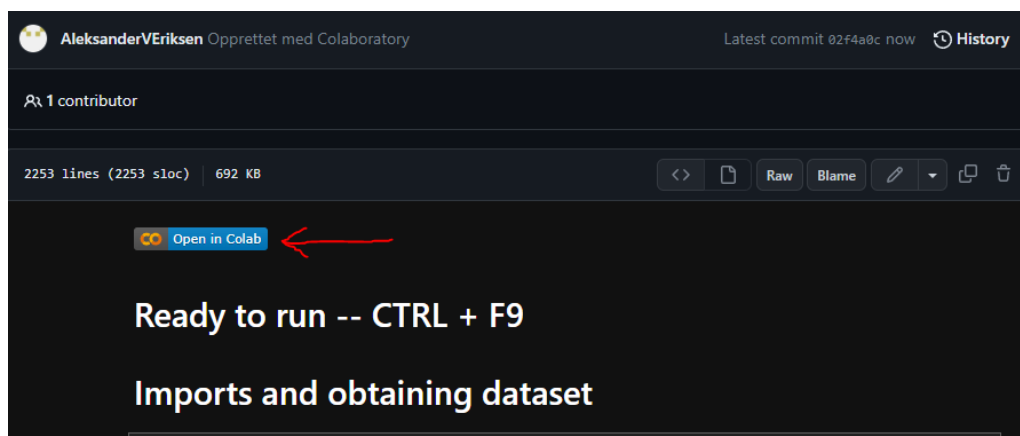
Long-short term memory: An analysis of the ML technique

By Aleksander Vanberg Eriksen

05.05.2023

GitHub link to the code:

https://github.com/AleksanderVERiksen/My_Projects/blob/main/Jupyter%20Notebook/Machine%20learning/ADV_ML_Weather_Forecasting_LSTM_Prediction.ipynb



Click *open in Colab* to run code!

Table of contents

Abstract	1
Introduction	2
The LSTM	3
The Chosen Application	4
The Articles	5
Methodology	6
Experiments	7
Conclusion	8
References	9
Figures	10

Abstract

The LSTM is an artificial neural network that deals with either sequential data or time series prediction. It is a very good model due to the memory it uses, but it can take some time to train it. Weather forecasting is one of the applications of time series predictions and will be tested on an LSTM model created in Keras with TensorFlow. Weather forecasting is the prediction of weather, which is based on the environmental variables in a given area you want to predict. Some other papers in this area have implemented Transductive-LSTM's when predicting which have given very good results. My proposed models consist of two LSTM layers and two dense layers. The first model predicts maximum air temperature for the next day, while the other model predicts minimum air temperature for the next day. The data used is from the machine learning repository, and is real data from Seoul, South Korea. Both my models resulted in decent loss, but bad accuracy. The prediction of both models were quite good but had some small errors when comparing predicted sample and the true value. The first model had an RMSE of 2.5 while the other one had an RMSE of 1.4. The MSE of model one was at 6.5 while model two had an MSE of 1.66. The R2 score of the first model was at 0.399, and the other had a score of 0.63. From these results, the second model performed the best. For this dataset, the LSTM model has an easier time predicting lower value temperature than higher temperature, which might as well be a coincidence.

Introduction

This paper will contain information and an analysis about a machine learning technique called LSTM. I will find out how it works, why it was made, and how it performs on a certain dataset with a chosen application. I will talk about similar methods that perform the same or is in the same genre of solving the same problems, compare them and provide why LSTM is better. I will investigate and mention some research articles/paper that contain information about the LSTM and different aspects of its implementation regarding my chosen implementation. Lastly, I will go into a deep analysis on how the LSTM performs regarding my results. If it runs poorly, I will use the articles to see if there are any methods that can be implemented to make it perform better, or just general adjustments. The initial metrics I will be focusing on is the accuracy of the model, the loss value, and its predicted value compared to the actual value. Lastly, I will have a conclusion based on the analysis of what I think of the LSTM model.

The LSTM

LSTM stands for Long-short term memory and is an artificial neural network that was constructed to deal with the problem of vanishing gradient that occurred in Recurrent neural network. The LSTM consist of two paths. The upper part is called the Cell state, and holds the long-term memory, while the lower part is called the Hidden state and holds the short-term memory. Both these states have an initial start value at 0 and gets updated each iteration. The LSTM's input is at the bottom, connected to the hidden state, and represent the x value. The cell states hold the encoded value of the whole data sequence, while the hidden state holds the encoded value of the input datapoint.

The way that the LSTM operate is that it is divided into three gates of the iteration. We first have the forget gate, which decide if the input value is to be forgotten or stored in the long-term memory after some calculations. The calculation is from the hidden state in conjunction with the input data. So, both the short-term memory and the input data gets feed into a Sigmoid activation function, and the output of that determines if the value shall be forgotten or not. The activation function produces a number between 0 and 1, and the closer it is to 0, the more irrelevant it becomes. So, we can think of this operation as a filter function, so that the LSTM can only focusing on getting the better long-term values to work with. Before it gets into the activation function, both the input and short-term memory value is multiplied by a weight that each have, and the result is summed together with an added bias. This will happen at each gate.

Next gate is the input gate, and this gate determines the value of the long-term memory that will go into the next gate. Here we use the same input value and short-term memory that we started with. In this gate we have two activation functions, the tanh function, and the sigmoid function. Tanh produces a number between -1 and 1 and help us increase or decrease the impact of the new long-term memory value. Then the sigmoid activation function will work as a filter once again to see if the input value is worth using for the new long-term memory. Both the result of the activation functions gets multiplied and added to the current long-term memory.

The last gate is called the Output gate and decides the new short-term memory. Here we once again use the sigmoid function on the input data for filter reasons, and the long-term memory that was calculated in the previous gate gets feed into a tanh function, and the result is multiplied by the result of the sigmoid function. Now we have a new short-term memory for the next iteration. This output can also be seen as a prediction, or target variable [1,2,3].

Advantages

1. Because of the memory it stores, LSTM is good when it comes to predicting time series or sequential data. This means that it uses the memory in the previous iteration to learn and uses it to predict the next one.
2. Good at handling large datasets because of the long-term memory.

3. Handles the problem of vanishing gradient because of the gates.

Disadvantages

1. The LSTM takes longer to train due to the gates.
2. Require more memory to train due to both long and short memory.
3. It is easy to overfit.
4. Implementing dropout is much harder.
5. Sensitive to different random weight initialization.

Applications of LSTM

Language modelling, Image captioning, Machine Translation, speech, or video, Handwrite recognition, anomaly detection, Time-series data processing, prediction, and classification [2].

Similar techniques

Recurrent neural network

Recurrent neural network is setup like a feedforward network, but it also adds a recurrent unit in the node that is at the hidden layer. This makes sure the RNN can process sequential data by using the previous value / or timestep and combining it with the inputs from the current value. Instead of gates that decide what to remember and what to forget before updating the hidden state, RNN only uses hidden states to remember information. They only have short-term memory [2,3].

Gru

Gru stands for Gated recurrent unit and is another variation of the recurrent neural network. It is like LSTM in the way that it uses gates. The gates are reset gate, update gate, and the hidden state candidate. Both the first two gates use sigmoid functions, and last only uses tanh function. The difference between those two is that since GRU has fewer gates, there are less variability in the gradient descent compared to LSTM. GRU has fewer parameters which makes it faster, and more compact. It also is more efficient at smaller datasets due to the simpler architecture [4,5,6,7].

The Chosen Application

Weather forecasting

Weather forecasting is the prediction of the weather through techniques that can read measurements of existing weather through time and decode it in a way that it is able to give predictions on future weather. This means that the techniques can understand the flow of the weather in a given area and give predictions on what's to come [8].

The challenges

One of the key challenges with weather forecasting is that we are trying to predict something that is very unpredictable. We are looking at the atmosphere when predicting, and this can be seen as a very chaotic system. If there is a slight change in the atmosphere where we are trying to predict, it can have huge consequences in flow of what's "normal" in that area. So even though we have the data needed to try to predict, there will always be assumptions in the end since it's basically impossible to try to model the atmosphere 100% without some errors [9].

LSTM with weather forecasting

Since weather forecasting generally uses big datasets to predict to avoid errors, the LSTM is quite suitable to the fact that the model uses a long-term memory to understand the flow of the whole dataset so it can predict what to come. So, it's very good for sequential data.

Other techniques that deal with weather forecasting, and how LSTM is better.

Linear regression

Linear regression is a regression model that assumes that there is a linear relationship between the target variable, and the other dependent variables. Even though it sounds good for forecasting, the model is limited. Since Time series can be unpredictable, and LSTM contains feedback mechanisms to sort that out, LSTM is superior [10].

ARIMA Time Series Forecasting

ARIMA stands for Autoregressive Integrated Moving Average and is a popular model for forecasting. This model requires that the dataset is stationary, meaning that the mean, variance, and autocorrelation do not change over time. This means that the model works best when the data used have linear patterns and does not have complex non-linear relationships. Also, it's more preferred if the data is univariate (single variable). This is where LSTM excels as it does not care for relationship between the variables [11].

Random forest regression

Random forest regression is a method that uses different decision trees to make a prediction based on all decision trees in play. This means that it selects the best tree for prediction. Again, LSTM is superior to this technique for the fact that it can handle much more data

efficient than RF due to vanishing/exploding gradient. It also needs the data to be univariate [12].

XGBoost

Extreme Gradient Boosting and is a specific implementation of gradient boosting. It uses advanced regularization which improves model generalization and reduces overfitting [13]. As the RF model, XGBoost needs to change the dataset to supervised learning [14].

The Articles

The content

The content of the selected articles mainly focuses on the implementation of the LSTM that is used to predict weather forecasting. Some of the articles uses a hybrid version of the model to gain better accuracy.

Their relevancy to my work

They are relevant in my work since I can understand better to how I'm going to work with weather forecasting and improve my LSTM model. They show me different approaches to weather forecasting, and implementation of the model.

Summary of each article

An LSTM Short-Term Solar Irradiance Forecasting Under Complicated Weather Conditions.

Their goal

The goal of this article is to predict Solar Irradiance one hour in advance and one day in advance using LSTM so that there will be safe to operate the grid and improve the economics of the PV (photovoltaic) system, which is a system designed to supply usable solar power.

Their methodology

They implemented LSTM and used four performance metrics to evaluate the forecast model. RMSE (root means squared error), MAE (Mean absolute error), MAPE (Mean absolute percentage error) and R-Square.

RMSE measures the difference between actual values and the forecasted values. A lower value indicates better forecasting results.

MAE is the average of absolute errors. It's used to better reflect the accuracy of the forecasting value error.

MAPE reflects the ratio of error to true value in percentage.

R-Square judges whether the predictive model is fitting and reflects the prediction deviates from reality. A 0 means the model fits poorly, while a 1 means the model has no errors.

RMSE and MAE evaluates hourly forecast while MAPE evaluates the daily forecasts.

ARIMA, SVR, BPNN, CNN and RNN where also implemented to compare.

The weather is divided into Sunny, Cloudy, Mixed, and all.

Their result

Hourly forecast

Location	Model	RMSE (W/m ²)				MAE (W/m ²)				R ²			
		Sunny	Cloudy	Mixed	All	Sunny	Cloudy	Mixed	All	Sunny	Cloudy	Mixed	All
Atlanta	ARIMA	30.44	58.06	166.88	110.78	24.26	53.56	122.639	71.48	0.97	0.86	0.57	0.86
	SVR	35.14	69.49	172.53	116.45	23.33	61.62	132.58	75.84	0.97	0.78	0.49	0.84
	BPNN	31.34	80.19	174.91	112.56	23.07	65.01	134.44	74.17	0.98	0.73	0.45	0.84
	CNN	52.55	65.32	194.64	122.36	44.00	50.50	139.47	77.99	0.96	0.72	0.32	0.82
	RNN	30.10	38.69	130.66	80.57	26.28	30.65	90.55	49.83	0.99	0.94	0.70	0.92
	LSTM	28.65	27.18	68.89	45.84	22.82	19.11	53.65	31.86	0.99	0.97	0.92	0.97
New York	ARIMA	42.27	42.66	96.37	65.56	34.23	30.66	74.37	46.58	0.97	0.83	0.83	0.93
	SVR	53.87	61.49	117.38	79.23	42.49	62.36	88.064	57.64	0.96	0.45	0.74	0.90
	BPNN	59.04	69.89	123.10	88.55	43.35	51.11	95.59	63.35	0.95	0.55	0.72	0.88
	CNN	56.15	83.75	124.95	92.70	38.70	63.46	89.01	63.72	0.95	0.35	0.71	0.87
	RNN	54.93	32.49	77.08	57.77	44.81	22.85	60.24	42.64	0.96	0.90	0.89	0.93
	LSTM	32.05	29.59	56.86	41.37	22.56	22.03	45.97	30.19	0.99	0.92	0.94	0.97
Hawaii	ARIMA	35.29	121.49	174.35	133.71	26.26	94.86	139.67	90.50	0.98	0.57	0.60	0.80
	SVR	48.01	155.22	176.06	144.43	48.65	108.71	149.29	96.25	0.95	0.31	0.48	0.75
	BPNN	80.40	123.49	178.44	133.61	60.13	86.76	132.96	93.29	0.91	0.56	0.52	0.78
	CNN	42.91	159.14	195.92	147.82	35.24	109.42	146.48	97.04	0.97	0.26	0.42	0.74
	RNN	29.90	108.17	183.29	124.09	21.17	83.66	125.08	77.30	0.98	0.66	0.79	0.81
	LSTM	22.13	73.07	86.68	66.69	18.72	54.34	65.07	46.04	0.99	0.91	0.90	0.95

Figure 1 Hourly forecast RMSES MAE R²

Daily forecast

Location	Model	MAPE%				R ²			
		Sunny	Cloudy	Mixed	All	Sunny	Cloudy	Mixed	All
Atlanta	ARIMA	10.9	22.9	12.5	12.9	0.79	0.70	0.83	0.87
	SVR	11.6	23.8	17.3	14.8	0.77	0.70	0.72	0.83
	BPNN	16.2	42.1	19.7	21.0	0.76	0.34	0.54	0.59
	CNN	10.4	29.6	16.9	12.5	0.81	0.68	0.80	0.85
	RNN	5.6	19.6	7.1	9.4	0.91	0.72	0.93	0.91
	LSTM	7.3	14.9	6.8	8.0	0.93	0.86	0.94	0.92
New York	ARIMA	8.5	29.8	21.5	16.7	0.90	0.69	0.73	0.89
	SVR	11.5	36.7	21.6	20.9	0.89	0.67	0.77	0.85
	BPNN	22.5	61.6	22.1	25.9	0.54	0.22	0.82	0.78
	CNN	9.4	39.3	21.0	17.7	0.84	0.67	0.80	0.90
	RNN	7.5	27.2	25.5	14.5	0.91	0.74	0.78	0.91
	LSTM	9.3	20.1	15.3	11.1	0.90	0.85	0.93	0.95
Hawaii	ARIMA	14.4	38.7	15.4	12.5	0.80	0.53	0.76	0.77
	SVR	17.7	32.8	19.4	15.3	0.77	0.54	0.76	0.78
	BPNN	32.0	59.7	23.8	29.9	0.57	0.37	0.43	0.54
	CNN	12.2	32.1	7.3	11.1	0.83	0.54	0.92	0.70
	RNN	10.5	24.9	7.5	9.8	0.88	0.66	0.92	0.81
	LSTM	5.9	18.1	6.2	8.2	0.95	0.86	0.95	0.91

Figure 2 Daily forecast MAPE% and R²

Monthly forecast

Location	Model	MAPE%											
		Jan.	Feb.	Mar.	Apr.	May	Jun.	Jul.	Aug.	Sep.	Oct.	Nov.	Dec.
Atlanta	ARIMA	15.6	11.3	12.4	10.4	10.5	5.8	9.2	9.8	11.7	11.0	8.9	17.6
	SVR	18.6	15.6	15.5	12.3	14.7	11.9	8.0	12.1	12.3	15.0	10.4	20.2
	BPNN	26.9	24.6	20.5	18.2	10.2	12.6	12.2	14.9	12.5	18.5	22.9	41.2
	CNN	15.4	10.9	11.2	9.8	11.6	10.1	12.7	14.6	8.0	11.4	9.2	21.3
	RNN	11.3	11.6	7.8	7.9	7.8	7.9	7.1	7.0	8.9	10.5	9.5	15.9
	LSTM	10.6	10.6	7.8	8.2	7.7	5.1	6.0	5.9	9.5	10.3	8.2	13.4
New York	ARIMA	28.2	30.7	15.8	13.0	13.2	12.2	11.6	11.6	12.7	22.6	33.7	28.3
	SVR	35.9	35.4	23.0	17.1	14.1	16.4	15.3	12.9	11.3	28.0	40.4	25.1
	BPNN	33.2	37.5	33.5	28.4	17.3	20.5	13.5	18.1	23.8	22.2	54.9	20.3
	CNN	27.8	24.8	18.8	13.4	10.2	15.8	12.7	12.0	11.5	21.9	22.9	25.7
	RNN	26.0	26.5	21.5	10.7	14.9	12.6	9.7	10.7	12.5	24.0	36.0	23.6
	LSTM	20.3	20.3	18.5	9.6	7.8	7.9	8.1	7.4	10.4	19.3	22.7	19.6
Hawaii	ARIMA	9.5	9.5	10.8	10.0	15.3	13.2	13.3	12.4	14.7	13.4	19.9	13.7
	SVR	13.2	12.6	12.8	11.5	20.9	18.8	17.5	15.4	15.6	16.4	14.1	16.6
	BPNN	21.5	19.4	35.5	35.9	31.7	40.9	30.3	36.6	30.1	22.6	22.6	32.3
	CNN	6.8	8.1	10.2	11.1	11.2	14.7	13.2	14.3	11.3	10.5	11.3	13.7
	RNN	6.7	7.9	9.8	9.5	9.5	9.2	9.5	12.1	10.4	10.2	11.0	11.6
	LSTM	3.2	3.5	5.9	9.4	10.4	7.5	7.3	9.3	10.2	6.1	10.3	10.3

Figure 3 Monthly forecast MAPE%

The advantages/disadvantages with this study

This study had very good results regarding the use of LSTM, even though it struggled with cloudy weather and when the solar irradiance where low [15].

Transductive LSTM for time-series prediction: An application to weather forecasting.

Their goal

The goal of this article is to implement an LSTM which uses Transductive to predict weather forecasting. It is used since it then can exploit the local information in time-series prediction. This means that the samples in the test point vicinity are considered to have higher impact on fitting the model.

Their methodology

They used a dataset from Weather Underground website that contains real measurements for cities such as Brussel, Antwerp, Liege, Amsterdam, and Eindhoven. They used TensorFlow to implement the LSTM model. They compared the Transductive LSTM with a regular LSTM to see performance. To create the T-LSTM, they altered the cost function of the LSTM such that the samples about the test point have higher impact in the objective function.

Their result

The study resulted in that the T-LSTM outperforms the LSTM in many cases.

Nov/Dec

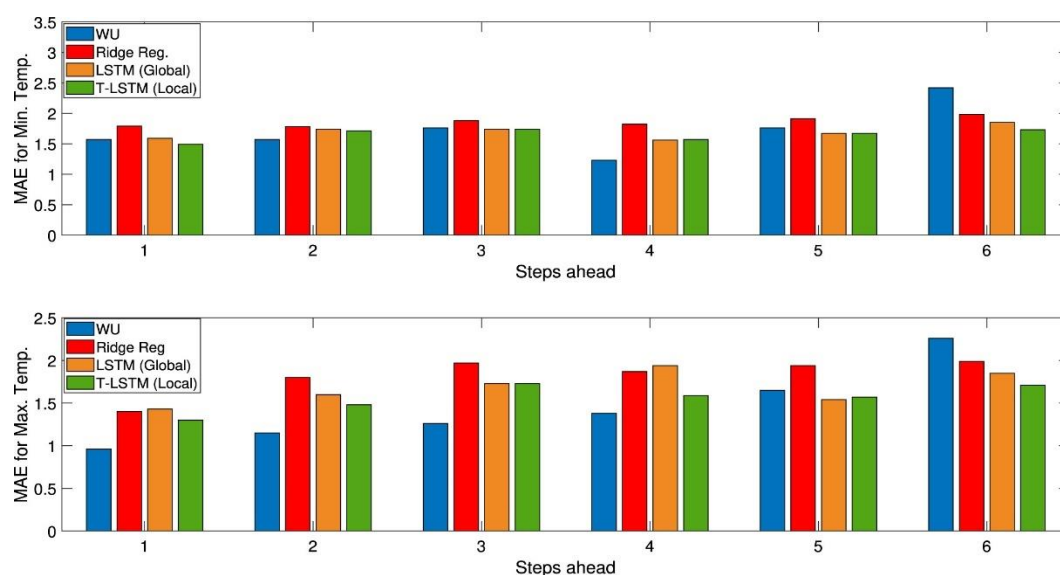


Figure 4 MAE for min/max temp NOV/DEC

Apr/May

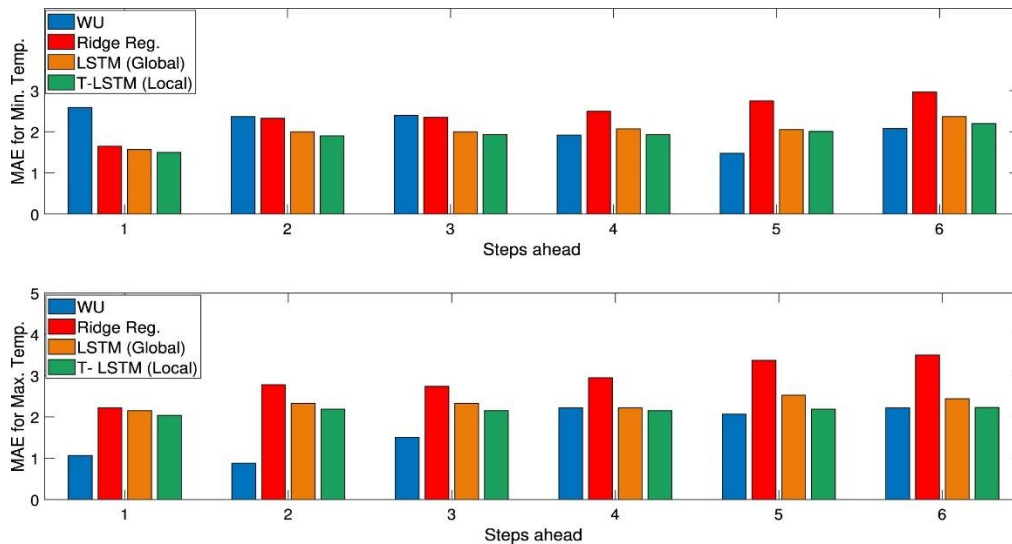


Figure 5 MAE for min/max temp APR/MAY

The advantages/disadvantages with this study

But even though the T-LSTM had better results, the LSTM is still a very competitive approach [16].

DWFH: An improved data-driven deep weather forecasting hybrid model using Transductive Long Short-Term Memory (T-LSTM).

Their goal

The goal of this article is to create a LSTM and T-LSTM model for weather forecasting and compare those two.

Their methodology

They use a dataset called The Jena climate dataset that contained 14 features for the weather forecasting. They implemented the T-LSTM and LSTM with Keras and TensorFlow.

Their result

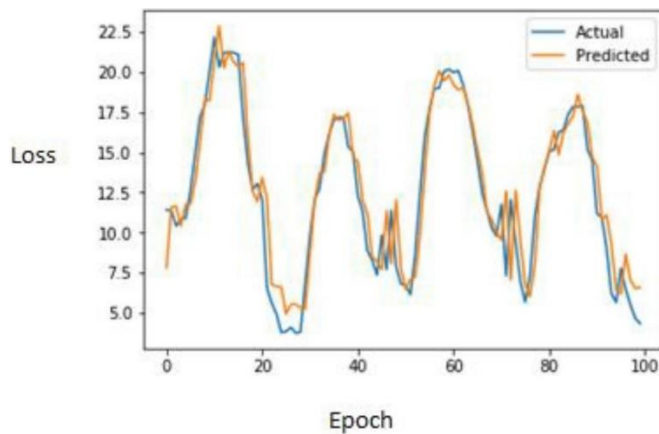


Figure 6 Loss value in epochs, pred vs actual

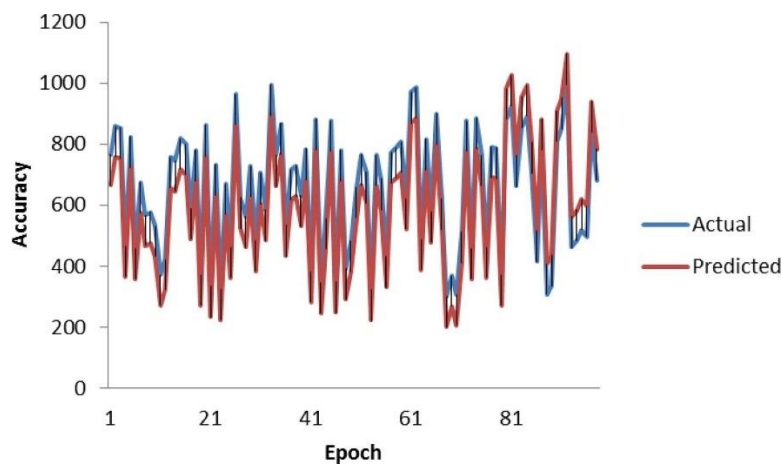


Figure 7 Acc value in epochs, pred vs actual

The advantages/disadvantages with this study

We can see that the hybrid approach T-LSTM is very good [17]

Application of LSTM for short term fog forecasting based on meteorological elements.

Their goal

Use the LSTM framework for fog forecasting.

Their methodology

Data from Anhui Metrological Bureau of China. They have employed about 3 years of meteorological data to use on the LSTM. They also experiment with different values to find the optimal parameters for the model. They experimented with number of layers in the LSTM, number of cells in the LSTM, and the number of layers in the fully connected layer. They split the dataset into four different ones.

Their result

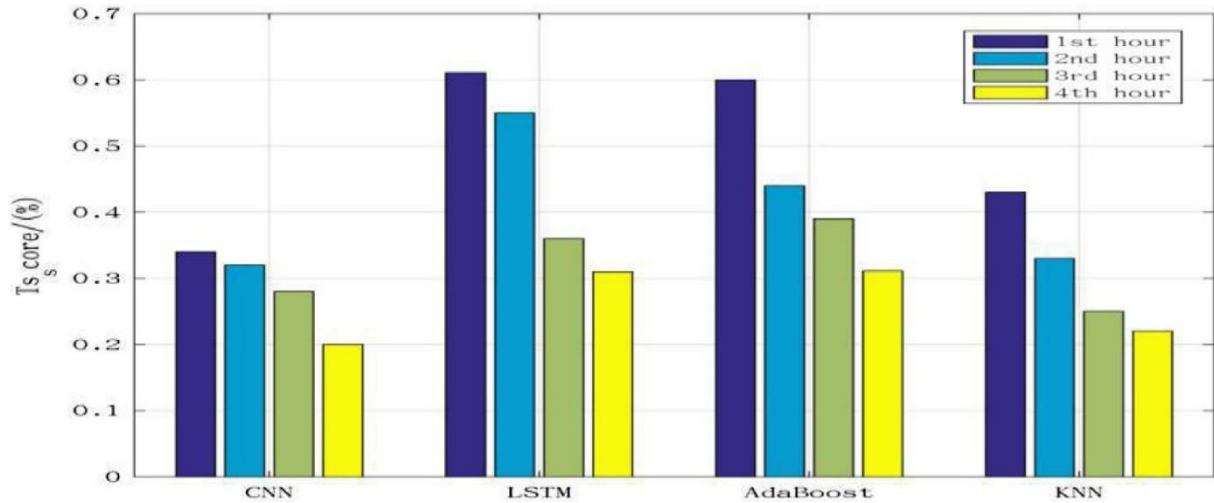


Figure 8 TS score comparison with different ML techniques

The advantages/disadvantages with this study

The study resulted in that the LSTM is better than the comparable models for short-term fog forecasting [18].

Methodology

The environment

For my implementation, I used Keras with TensorFlow to create an LSTM model using the google colab environment.

The dataset

I've used a dataset called Bias correction of numerical prediction model temperature forecast, which is a dataset that contains variables from Seoul, South Korea in the summer from 2013 to 2017. The dataset is divided into stations around Seoul, and I've decided I wanted to focus on station 1 when it comes to prediction. The variables used are Present_Tmax (current maximum air temperature), Present_Tmin (current minimum air temperature), lat (latitude), lon (Longitude), DEM (Elevation), Slope, Solar radiation (Daily incoming solar radiation), Next_Tmax (Next day maximum air temperature) and Next_Tmin (Next day minimum air temperature) [19]. This is regarded as a regression problem since I need to predict the value of the next minimum and maximum value air temperature for the next day.

Data cleaning

The dataset was used in google colab by copying the path of the download from the machine learning repository and past it into a data frame using the *pandas* library, by converting it from a csv file using the *pandas* function `read_csv`. I then checked if there were any null values and remove the rows that had it. I then proceeded to reset the index and inspect the columns to see if I needed to use a label encoder so that the model would have an easier time understanding the dataset, or to just remove irrelevant column features. Since the dataset had

predictions from its own weather station LDAPS, I decided to remove them so that I could make my model only rely on the raw data. I also decided to remove *Date* as It's not relevant to the prediction since the weather values are mostly random anyway. I then extracted all the values that weather station 1 had in the dataset to use in the prediction and later analysis of the dataset.

Transforming and preprocessing

Since the LSTM works using timesteps, I needed to convert the dataset from a 2d tensor to a 3d tensor. The air temperature values in the dataset went from row to row, which means that each air temperature value got predicted based on the columns in the same row it was in. This concluded in that I chose to use time steps 1 when transforming. I also kept two target variables *Next_Tmin* and *Next_Tmax*. Furthermore, I preprocessed the data using the library *sklearn.preprocessing* and imported *MinMaxScaler* to that each feature in the dataset would be scaled between 0 and 1. This is to ensure that the model have an easier time learning the dataset.

The model and train-test split

I then split the data into two train set, and two test set, were one set used *Next_Tmax* as target value and the other set used *Next_Tmin*, using the library *sklearn.model_selection* and imported *train_test_split*. The train sets contained 67% of the data, while the test sets contained 33%.

The model was made using Keras and was initially a sequential model. The first layer contains an LSTM layer with 64 units, an input shape of (1,6) which represent the timesteps and feature count, and with *return_sequence* set to True. Next layer is another LSTM layer with 32 units, an input shape of (1,6) and with *return_sequence* set to False. Then a Dense layer is inserted with 10 units, and lastly another Dense Layer with 1 unit which represent the target value, or next air temperature.

Units refers to the dimensionality of the output space of the layer. Since the LSTM have 64 units, it means that the layer has 64 cells that will give 64 output values for each of the time steps [20].

The input shape refers to the tensor of the input that will be feed to the network. For my model, the input shape is (1,6) which means that for each input that will be fed to the network, it will contain 1 array with 6 values.

The parameter *return_sequence* means that the layer will return the last output in the output sequence. This is applied to make sure the next LSTM layer get the right input value [20].

A Dense layer is a densely connected neural network layer that contains a user-defined number of neurons [21]. In my case, I use one layer with 10 neurons, and one with 1 neuron.

Training parameters

The model uses MeanAbsoluteError as its loss function, while the optimizer used is Adam with a starting learning rate at 0.001. The batch_size is set to 16, and epochs is set to 100.

The loss function measures the absolute mean error between the true value and the predicted value. This means that the return value will be the average difference between the true value and the predicted value, so a low number is ideal [22].

The optimizer Adam is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments [23]. Its job is to minimize the error in the loss function by updating the weights and biases. The learning rate is the amount of steps it takes to get to the optimal weights/biases. A high learning rate results in a big step, while a small learning rate results in a much smaller step. A stochastic gradient descent means that it will go towards the optimal result by updating the parameters one at a time in a total of the amount of data you have [24].

A batch size of 16 means that the first 16 samples will be trained at a time, while an epoch set to 100 means that the cycle of each training iteration, which is when all samples are used, will happen 100 times. So, a combination of batch size 16 and epoch 100 will result in that it will train 16 samples at a time until all samples are used 100 times.

Experiments

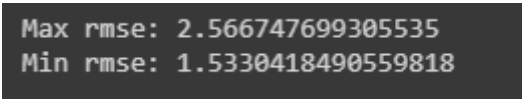
For the experiments, I looked at the accuracy of the models, the loss, the root mean squared error, mean squared error, R2 score, and plotted some visual representation on how the predicted values was in compared to the actual values.

Evaluation of the models

When completing the training of the two models, both were evaluated using their respective tests sets to see how good the models were trained. The metrics looked at was the loss and accuracy value. The model that was going to predict *Next_Tmax* landed on a loss value of 0.1316 and an accuracy of 0.0200. The other model that predicted *Next_Tmin* had a loss of 0.1042 and an accuracy of 0.0100.

Root Mean squared error

To test mean squared error, I created a prediction set for each model using their respectively test set with no target value so that they would predict the next value based on the values they got. Since I transformed the values earlier, I needed to use inverse transformation to scale them back to their original value. Furthermore I used the library *sklearn.metrics* and imported *mean_squared_error*. This method takes the true target values and the predicted values as parameters. I also changed the parameter *squared* from True to False, so that it would compute root mean squared error.



```
Max rmse: 2.566747699305535
Min rmse: 1.5330418490559818
```

Figure 9 RMSE score for max and min model.

This was the result I got, which is a decent result. The point of RMSE is that the lower it is, the better the prediction is since it measures the average error from all predicted samples towards the true samples.

Mean squared error

MSE computes the difference between the predicted and the actual value, and then squares it. This is to understand the model performance of the whole dataset, and it will highlight if there are any large errors.

```
Max mse: 6.589601685597318
Min mse: 1.6619573624973187
```

Figure 10 - MSE for min and max model

R² score

The R^2 is an overall measurement of the model's performance. It ranges from 0 to 1, where 1 is the best.

```
Max r2 score: 0.3990792764344433
Min r2 score: 0.6314254157204349
```

Figure 11- R^2 score

Prediction of next day max air temperature

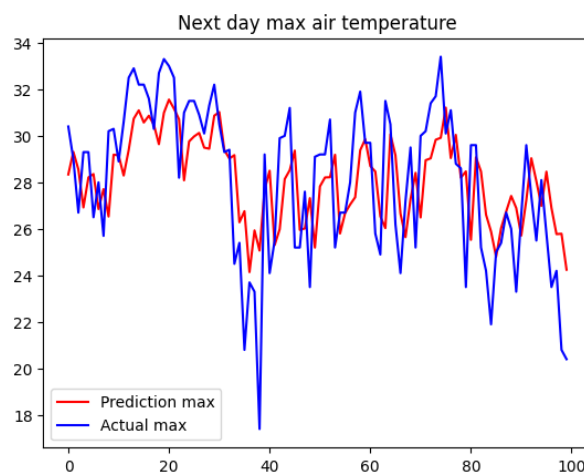


Figure 12 Max air temp - pred vs actual

This graph represents the *Next_tMax* predicted values in red, and the actual values in blue.

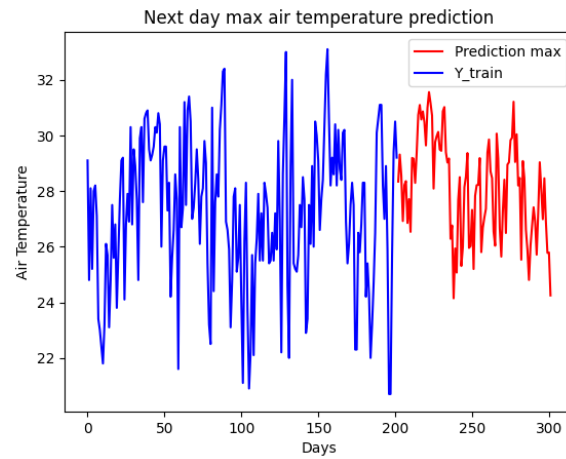


Figure 13 Predicted graph for max air temperature.

This graph represents the train values combined with the predicted values for the test values.

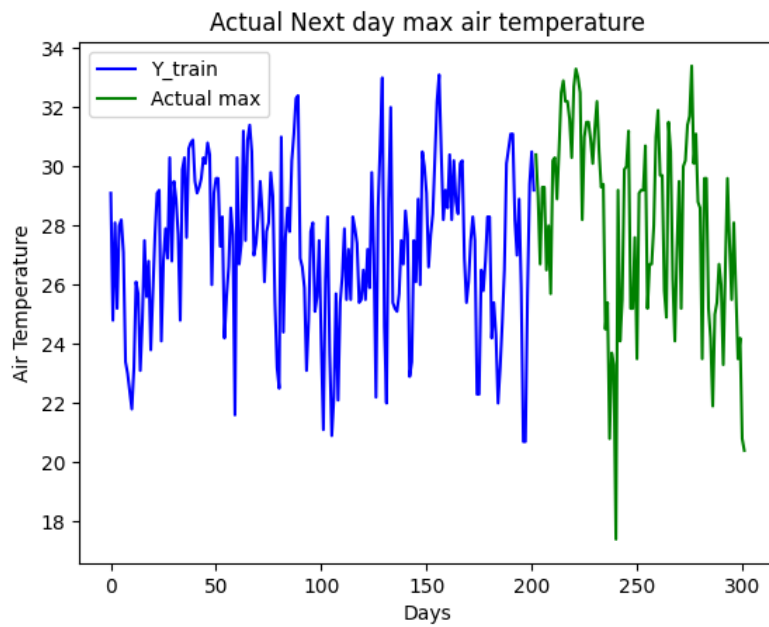


Figure 14 The actual graph for max air temp

This graph is the actual combined train and test values.

Prediction of next day min air temperature

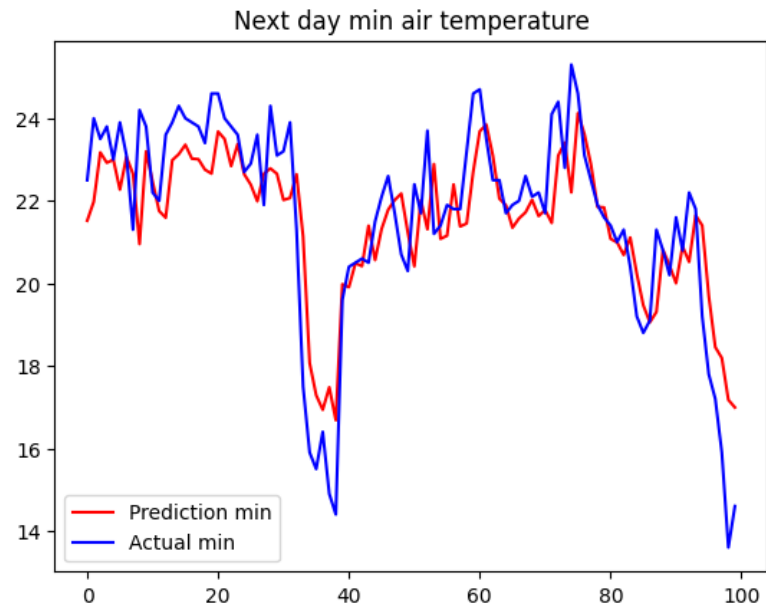


Figure 15 Next day min temperature – pred vs actual

This graph represents the predicted *Next_Tmin* values compared to the actual values.

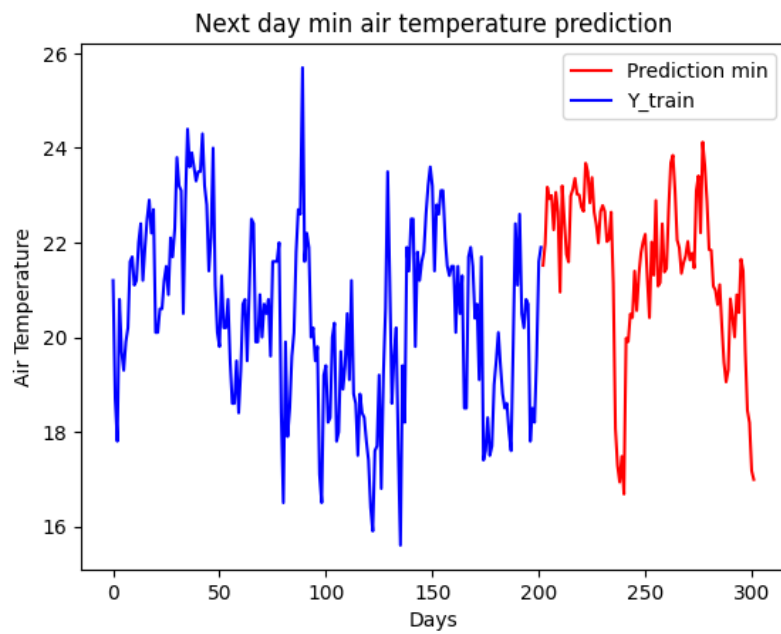


Figure 16 Next day min air temp graph

This graph contains the train values combined with the predicted values.

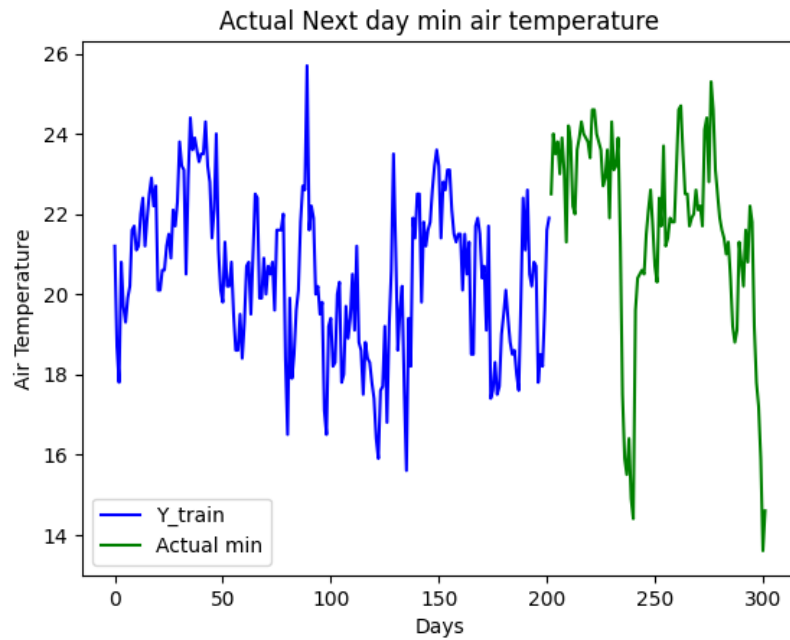


Figure 17 Actual Next day min air temp

This graph is the actual values of train and test.

Analysis

This is the part where I will analyze the results of the experiments with the combination of how the models are set up, and what parameters are included in the training of the models. I will also compare the models to some of the articles I've read to see if there are any improvements to the models.

The Model

For the model setup, this is how the neural network can be seen visually.

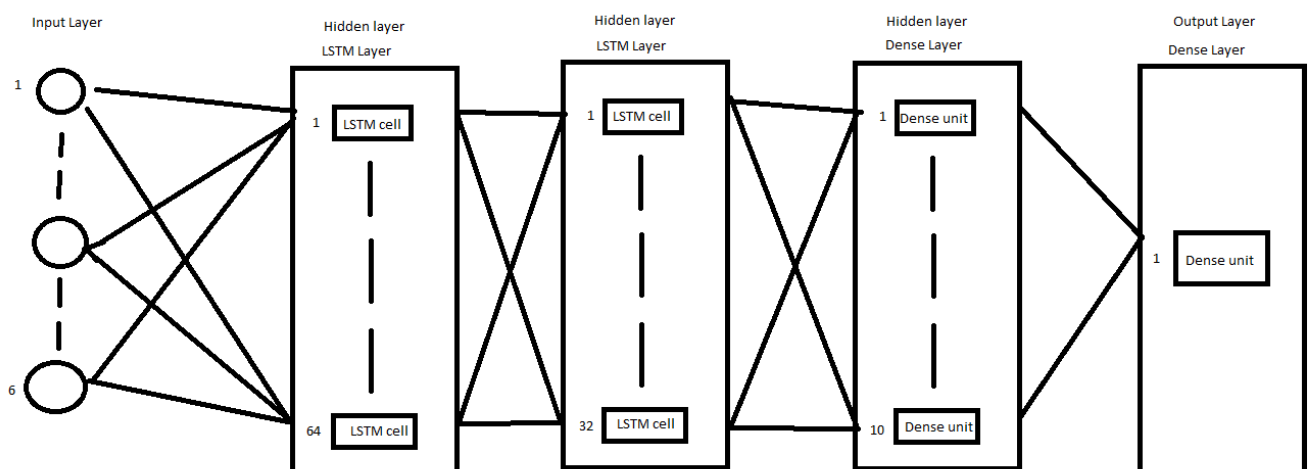


Figure 18 - A simple overview of the model structure

The model works by first feeding a sample of the dataset with a feature space of 6. This is why the input layer consists of 6 neurons. Each of the neurons are connected to the next layer, which is the hidden layer. The hidden layer consists of two LSTM layers, and one dense layer. The connections between the input layer and first LSTM layer also have weights and biases attached to them and will be updated when training. The start value of these are random.

The Training

When the LSTM receives the input values, each of the LSTM units in the neural network gets an input vector of size 6 and works towards decoding the sequence of the whole dataset by updating the long-term memory and uses the short-term memory to understand the current input vector. So, since I have initialized time steps to be 1, the LSTM cells will go one iteration before sending the output to the next LSTM layer. In the next LSTM layer, each cell receives their input as the output of the last LSTM layer 64 times for one timestep for 16 batch size in one cycle. This can be seen as a 3D tensor with shape (16, 1, 64). This is because the parameter *return_sequence* is set to True. It is very important to do this since the next LSTM layer expects a 3D tensor as well. This layer works the same as the last expect that it has 32 units instead of 64, and it has the parameter *return_sequence* set to False. This is because the next layer is a Dense layer that expects a 2D tensor. So, for this layer, the output will be of shape (16, 32) and will go as input to the Dense layer. The Dense Layer is just a fully connected network with 10 units and is used to better update the weights and make the model learn more about the inputs. The output of this layer will be (16,10) and is connected to the last layer, which is the output layer. The output layer also is a Dense Layer, but only with one unit. This is because the value we are prediction is a single value, the air temperature.

When looking at the training loss and accuracy for the models, the loss was decent while the accuracy was very low. Both models had little to no change at all during training for the accuracy, while the loss started somewhat low and had a big jump at first then had small increments in reduction. The reason the accuracy is so low is that the model is solving a regression problem, and the accuracy expects perfect predictions which is very difficult to do. The metric I can use instead of accuracy is the R2 score which measures the score of a regression models.

The point of R2 score is that the higher it is towards 1, the better the regression model is. From the training of the models, the R2 score of the model that predicts *Next_Tmax* got a score of 0.399 which is not so good, while the model that predicts *Next_Tmin* got a score of 0.63 which is quite decent. When combining the result from the Root Mean Squared Error at figure 9, both the models should give quite decent results with low errors towards its true value.

The results

For the model that predicts *Next_Tmax*, its predicted values compared to the actual values are quite decent. The predicted curve compared to the actual curve holds the same flow with some errors to the actual value. The same can be said about the other model. If the whole curve is compared, the predicted values are more compact than the actual values. Both models struggle to stretch down and up to reach the maximum value and minimum value when predicting.

Comparing the articles

If we compare the article *an LSTM short-term Solar Irradiance Forecasting Under Complicated Weather Conditions*' implementation with mine, their implementation consisted of only one LSTM layer using 50 units, and a Dense layer with 1 unit. They included dropout at 0.1, so that 10% of the training data would be discarded so that the model wouldn't overfit. They also used ReLU as activation function instead of the tanh function. Their RMSE value where much larger than mine, but they got a good R2 score at 0.92. This is most likely because they used a larger dataset to train the model than I did.

The next article *Transductive LSTM for time-series prediction: an application to weather forecasting* used a dataset with eighteen features compared to the six features I used. Their implementation of the LSTM's was also in Tensorflow, and they tuned their parameters to find the most optimal values. They tuned number of neurons and regularization, and used tanh as the activation function, and Adam as the optimizer. They used the metric MSE instead to evaluate the model.

The article *DWFH: An improved data-driven deep weather forecasting hybrid model using Transductive Long Short Term Memory (T-LSTM)* implemented an LSTM model with an input layer, one LSTM, a dense layer with activation function and dropout, and a dense output layer. The LSTM layer used 30 units, the dense layer used 256 units, and the output layer used 1 unit. Their dataset used 14 features, and where all numerical. They used MSE, Accuracy, Precision, Recall and F1Score to evaluate the model. Their model got a loss at around 0.2 which is roughly what I got, and they also predicted temperature. Even though their loss value was close to mine, their accuracy was much higher.

The article *Application of LSTM for short term fog forecasting based on meteorological elements* used four different datasets ranged from a total of 48000 – 3 000 000 samples. They used those datasets to parameter tune the model to find the optimal LSTM layers, cells, and Dense layers. They decided the parameter values based on the metrics Precision, F1-score, Accuracy and TS-score. Their overall result was decent. They compared their implementation to other techniques and got a slightly better score.

Improvements

Since I used a minor chunk of the dataset I found, the improvements I can see being applied is to use more data when training the model. The LSTM is better when using a big dataset, so the fact that I didn't take advantage of that might be one of the issues that it didn't predict as accurate as it should. Some other issues I could think of is that the feature range was too small for the model to learn accurately, so that is why the model had some errors. To improve this, I would need to use a different dataset with much more features, so that the model would have an easier time to learn the weather in that area. Most of the models from the articles I used implemented dropout to their model to prevent overfitting, which is something I could have used. I also could have implemented some sort of parameter optimizer on the model to gain better parameter values, since I am not quite confident that the parameters that I have chosen are the most optimal.

Conclusion

From the experiments and results, the LSTM is a very nice model to use for prediction of the future using sequential data. It works very fast, and produced feasible results even though the R2 score and RMSE could have been better. It is a very easy model to use even though some datasets might need prepping so that it can be feed to the model. When dealing with a regression problem, the metric accuracy might not be the best metric as its quite difficult to predict the exact value of a future temperature based on the fact that the weather in general is quite random in its nature even if we have a good baseline of variables to use. The metrics that is preferable to use are RMSE, MSE, and R^2 score which computes the error from predicted value towards the true value. The model that predicts minimum air temperature had an easier time predicting the values, than the other one that predicted maximum air temperature. The first model had a high MSE compared to the second model, which implies that the model might not understand how the features in the dataset can affect the max air temperature as good as it can with the minimum air temperature. So it might be that the LSTM have an easier time predicting lower values than higher ones, but it might as well be a coincidence or the way the data is represented in the dataset.

References

- [1] Rian Dolphin, LSTM Networks | A Detailed Explanation, Oct 21, 2020, *Towards Data Science*, Medium, Date accessed: 02.02.2023, <https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9>
- [2] Algoscale, What is Long Short-Term Memory (LSTM)?, February 8 2022, *algoscale*, Date accessed: 02.02.2023, <https://algoscale.com/blog/what-is-long-short-term-memory-lstm/>
- [3] Y. Duan, Y. L.V. and F. -Y. Wang, "Travel time prediction with LSTM neural network," *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Rio de Janeiro, Brazil, 2016, pp. 1053-1058, doi: 10.1109/ITSC.2016.7795686.
- [4] Saul Dobilas, GRU Recurrent Neural Networks – A Smart Way to Predict Sequences in Python, Feb 21.2022, *Towards Data Science*, Medium, Date accessed: 02.02.2023, <https://towardsdatascience.com/gru-recurrent-neural-networks-a-smart-way-to-predict-sequences-in-python-80864e4fe9f6>
- [5] R. Fu, Z. Zhang and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), Wuhan, China, 2016, pp. 324-328, doi: 10.1109/YAC.2016.7804912.
- [6] S. Yang, X. Yu and Y. Zhou, "LSTM and GRU Neural Network Performance Comparison Study: Taking Yelp Review Dataset as an Example," *2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*, Shanghai, China, 2020, pp. 98-101, doi: 10.1109/IWECAI50956.2020.00027.

- [7] Nikolas Adaloglou, RNN cells: analyzing GRU equations VS LSTM, and when to choose RNN over Transformers, Sep 17.2020, *Towards Data Science*, Medium, Date accessed: 03.02.2023, <https://towardsdatascience.com/rnn-cells-analyzing-gru-equations-vs-lstm-and-when-to-choose-rnn-over-transformers-a28e46beef31>
- [8] Cahir, John J.. "Weather forecasting". Encyclopedia Britannica, 17 Jan. 2019, <https://www.britannica.com/science/weather-forecasting>. Accessed 6 April 2023.
- [9] Shonk, John, "Why the weather forecast will always be a bit wrong". The conversation, 22 August 2018, <https://theconversation.com/why-the-weather-forecast-will-always-be-a-bit-wrong-101547>. Accessed 6 April 2023.
- [10] R. Shah, P. Shah, C. Joshi, R. Jain and R. Nikam, "Linear Regression vs LSTM for Time Series Data," 2022 IEEE World Conference on Applied Intelligence and Computing (AIC), Sonbhadra, India, 2022, pp. 670-675, doi: 10.1109/AIC55036.2022.9848887.
- [11] S. Eskandar, "ARIMA vs LSTM: A Comparative Analysis of Time Series Forecasting for Stock Price Prediction", Medium, 2023 March 30, Date accessed: 11.04.2023 <https://medium.com/@eskandar.sahel/arima-vs-lstm-a-comparative-analysis-of-time-series-forecasting-for-stock-price-prediction-45b6fcf5ac10>
- [12] J. Brownlee, "Random Forest for Time series forecasting", Machine Learning Mastery ,2020 November 2, Date accessed 11.04.2023, <https://machinelearningmastery.com/random-forest-for-time-series-forecasting/>
- [13] I. Paliari, A. Karanikola and S. Kotsiantis, "A comparison of the optimized LSTM, XGBOOST and ARIMA in Time Series forecasting," 2021 12th International Conference on Information, Intelligence, Systems & Applications (IISA), Chania Crete, Greece, 2021, pp. 1-7, doi: 10.1109/IISA52424.2021.9555520.
- [14] J. Brownlee, "How to use XGBoost for Time Series Forecasting", Machine Learning Mastery ,2020 August 5, Date accessed 11.04.2023, <https://machinelearningmastery.com/xgboost-for-time-series-forecasting/>
- [15] Y. Yu, J. Cao and J. Zhu, "An LSTM Short-Term Solar Irradiance Forecasting Under Complicated Weather Conditions," in IEEE Access, vol. 7, pp. 145651-145666, 2019, doi:10.1109/ACCESS.2019.2946057.
- [16] Zahra Karevan, Johan A.K. Suykens, Transductive LSTM for time-series prediction: An application to weather forecasting, Neural Networks, Volume 125,2020, Pages 1-9, ISSN 0893-6080, <https://doi.org/10.1016/j.neunet.2019.12.030>. (<https://www.sciencedirect.com/science/article/pii/S0893608020300010>)
- [17] K. Venkatachalam, Pavel Trojovský, Dragan Pamucar, Nebojsa Bacanin, Vladimir Simic,DWFH: An improved data-driven deep weather forecasting hybrid model using Transductive Long Short Term Memory (T-LSTM), Expert Systems with Applications, Volume 213, Part C,2023,119270, ISSN 0957-4174, <https://doi.org/10.1016/j.eswa.2022.119270>. (<https://www.sciencedirect.com/science/article/pii/S0957417422022886>)
- [18] Kai-chao Miao, Ting-ting Han, Ye-qing Yao, Hui Lu, Peng Chen, Bing Wang, Jun Zhang, Application of LSTM for short term fog forecasting based on meteorological elements, Neurocomputing, Volume 408,2020, Pages 285-291, ISSN 0925-2312, <https://doi.org/10.1016/j.neucom.2019.12.129>. (<https://www.sciencedirect.com/science/article/pii/S0925231220304884>)

[19] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [<http://archive.ics.uci.edu/ml>]. Irvine, CA: University of California, School of Information and Computer Science.
<https://archive.ics.uci.edu/ml/datasets/Bias+correction+of+numerical+prediction+model+temperature+forecast>

Cho, D., Yoo, C., Im, J., & Cha, D. (2020). Comparative assessment of various machine learning-based bias correction methods for numerical weather prediction model forecasts of extreme air temperatures in urban areas. Earth and Space Science.

[20] Keras, ‘Keras API references’, Date accessed: 04.05.2023,
https://keras.io/api/layers/recurrent_layers/lstm/

[21] Keras, ‘Keras API references’, Date accessed: 04.05.2023,
https://keras.io/api/layers/core_layers/dense/

[22] TensorFlow, ‘losses’, Date accessed: 04.05.2023,
https://www.tensorflow.org/api_docs/python/tf/keras/losses/MeanAbsoluteError

[23] Keras, ‘Keras API references’, Date accessed: 04.05.2023, <https://keras.io/api/optimizers/adam/>

[24] Musstafa, Optimizers in Deep Learning, Mar 27.2021, ‘*MLearning.ai*’, Medium, Date accessed: 04.05.2023, <https://medium.com/mllearning-ai/optimizers-in-deep-learning-7bf81fed78a0>

Figures

[1] https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/6287639/8600701/8864021/ytu.t5-2946057-small.gif

[2] https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/6287639/8600701/8864021/ytu.t6-2946057-small.gif

[3] https://ieeexplore.ieee.org/mediastore_new/IEEE/content/media/6287639/8600701/8864021/ytu.t7-2946057-small.gif

[4] <https://ars.els-cdn.com/content/image/1-s2.0-S0893608020300010-gr9.jpg>

[5] <https://ars.els-cdn.com/content/image/1-s2.0-S0893608020300010-gr10.jpg>

[6] <https://ars.els-cdn.com/content/image/1-s2.0-S0957417422022886-gr9.jpg>

[7] <https://ars.els-cdn.com/content/image/1-s2.0-S0957417422022886-gr10.jpg>

[8] <https://ars.els-cdn.com/content/image/1-s2.0-S0925231220304884-gr5.jpg>

[9]
<https://colab.research.google.com/drive/1Ru7zrfVFmEkIbPRaFm0gdVUVxvrqmc4W?usp=sharing>

[10]
<https://colab.research.google.com/drive/1Ru7zrfVFmEkIbPRaFm0gdVUVxvrqmc4W?usp=sharing>

[11]
<https://colab.research.google.com/drive/1Ru7zrfVFmEkIbPRaFm0gdVUVxvrqmc4W?usp=sharing>

[12]
<https://colab.research.google.com/drive/1Ru7zrfVFmEkIbPRaFm0gdVUVxvrqmc4W?usp=sharing>

[13]
<https://colab.research.google.com/drive/1Ru7zrfVFmEkIbPRaFm0gdVUVxvrqmc4W?usp=sharing>

[14]

<https://colab.research.google.com/drive/1Ru7zrfVFmEkIbPRaFm0gdVUVxvrqmc4W?usp=sharing>

[15]

<https://colab.research.google.com/drive/1Ru7zrfVFmEkIbPRaFm0gdVUVxvrqmc4W?usp=sharing>

[16] An image created in paint showing the overview of the model's structure.

[17]

<https://colab.research.google.com/drive/1Ru7zrfVFmEkIbPRaFm0gdVUVxvrqmc4W?usp=sharing>

[17]

<https://colab.research.google.com/drive/1Ru7zrfVFmEkIbPRaFm0gdVUVxvrqmc4W?usp=sharing>