ASSIGNMENT:3 TIME SERIES DATA

REPORT

Working with a weather timeseries dataset from the Max Planck Institute for Biogeochemistry weather station in Jena, Germany, will be our task. Throughout several years, 14 distinct variables (including temperature, pressure, humidity, wind direction, and so on) were recorded every 10 minutes and included in this dataset. The subset of data we will download is restricted to the years 2009–2016, however the original data dates back to 2003.

We start by creating a baseline model that isn't machine learning. It will act as a safety net and set a standard that we will need to surpass in order to show the value of more advanced machine learning models. In this instance, it is acceptable to infer that the temperature timeseries is both periodical with a daily period and continuous (the temperatures tomorrow are probably going to be quite similar to the temperatures today). Therefore, it makes sense to consistently forecast that the temperature in 24 hours would be the same as it is now. A validation MAE of 2.44 degrees Celsius and a test MAE of 2.62 degrees Celsius are attained using this common-sense baseline. Thus, you will be off by two and a half degrees on average if you always believe that the temperature will be the same in the future as it is now.

Following the baseline model, we develop a number of models, test the dataset using those models, and document the outcomes, which are shown in the table below.

MODEL	DENSE UNITS	VALIDATION MAE	TEST MAE	LOSS
Baseline		2.44	2.62	
Basic machine learning model	16	2.75	2.79	12.47
1D convolution	16	3.5	3.38	18.53
Simple LSTM	16	2.43	2.56	10.79
Simple RNN	16	9.8	9.93	151.4
Stacked RNN	16	9.8	9.90	151.12
Simple GRU	16	2.35	2.51	10.09
LSTM with dropout	16	2.39	2.56	10.71
Stacked GRU with dropout	32	2.30	2.46	9.96
LSTM dropout	8	2.46	2.67	11.95
LSTM dropout	16	2.47	2.61	11.15
LSTM dropout	32	2.59	2.60	10.83
LSTM dropout	64	2.87	2.68	11.50
Bidirectional RNN	16	2.50	2.61	11.12
1D convolution and LSTM with dropout	16	3.89	3.83	23.39

After that, we developed a simple machine learning model with a dense layer of 16 units, which produced an MAE of 2.79 that was slightly larger. The flattening of the time series data, which eliminated the temporal context, resulted in poor performance of the dense layer model. Additionally, we developed a 1D convolutional model, but it produced disappointing outcomes with 3.38 test MAE since it treated every data segment equally even after max-pooling disturbing the sequential order of the data.

Recurrent neural networks are a class of neural network topologies created especially for this use case. The Long Short-Term Memory (LSTM) layer is one of the most well-liked of them. Hence, we started with simple LSTM model which yielded the MAE of 2.56 which is closer and better to the baseline MAE. After that, we build the model using simple and stacked RNN whose performance was poor with test MAE of 9.93 and 9.90 respectively. More specifically, Simple-RNN has a significant problem: while in theory it should be able to remember inputs seen several timesteps before at time t, in practice it is unable to learn such long-term dependencies. This is caused by the vanishing gradient problem, which has an effect similar to that of many-layered non-recurrent networks (feedforward networks) in which a network eventually becomes untrainable as more layers are added. Fortunately, there are several recurrent layers accessible in Keras besides SimpleRNN. There are two more that were made to deal with these problems: LSTM and GRU.

We'll then go through a few of RNNs' advanced capabilities, which can help us get the most of the deep learning sequence models.
To overcome the problem of overfitting in the above LSTM model we add recurrent dropout layers, we yield the MAE of 2.56 which is not too bad. Now, that we are no longer overfitting, we are increasing the networks capacity to hit the bottleneck performance. For a change we used GRU instead of LSTM with recurrent layers and yielded the MAE of 2.46 which is far better than the baseline.

After this we experimented by adjusting the units as 8,16,32,64 in stacked recurrent layers with layers LSTM and noted that the MAE's obtained were not so efficient and they were almost similar and higher than the common baseline model.

Now, we used another advanced model which is bidirectional RNN which gave the test MAE of 2.61 that means it did not perform as well as the simple LSTM layer because the antichronological half of the network is known to do far worse on this job (again, because the recent past matters considerably more than the distant past, in this case), all of the predictive capacity must originate from the chronological half of the network. Concurrently, the antichronological half doubles the network's capacity and pushes the overfitting process forward significantly.

At last, we use the combination of 1D convolution model and the RNN with LSTM regularized layers, which yielded the MAE of 3.83 which is greater than the baseline most likely as a result of the convolution's difficulties in preserving information order.

SUMMARY:

- In terms of capturing temporal dependencies, LSTM and GRU demonstrated favorable outcomes. For sequential data tasks such as temperature prediction, these should be used instead of simple RNNs.
- Overfitting in RNNs can be avoided with the use of strategies like recurrent dropout, which enhances generalization.
- In stacked recurrent layers, adding more units didn't always result in better performance and could even cause overfitting. Mix the capacity to generalize with the complexity of the model.
- Bidirectional RNNs might not be appropriate for situations where the recent past is more significant than the distant past. Make sure the model architecture fits the requirements of the assignment.
- Even while GRU and LSTM are frequently employed, experimenting with other designs, such attention mechanisms or hybrid models, might produce better outcomes.
- To enhance model performance, make sure the data is properly preprocessed, taking into account things like scaling, normalization, and feature engineering.

It is possible to significantly improve the temperature prediction model's performance above the baseline and produce forecasts that are more accurate by taking these suggestions into account and experimenting with other architectures and methodologies.