Time Series Weather Forecasting with Different Models

In this Project, we have developed 19 models and have trained them using 50% of the data using 14 features and then validated and tested with the remaining 25% of the data for each respectively.

The first model we developed was a common-sense, non-machine learning baseline model, which will serve as a sanity check, and it will establish a baseline that we will have to beat to demonstrate the usefulness of more advanced machine learning models. This model yields a validation MAE of 2.44 and a test MAE of 2.62. a model should beat this MAEs to become the best model.

Moving further we applied this data to dense deep learning layers and convolution network models but their performances are below par. We got a test MAE of 2.70 for the deep learning model due to the flattening of the time series data, removing the temporal context. We got a test MAE of 3.08 for the Convolution model as it treats all data segments uniformly, even after pooling, which disrupted the data's sequential order.

As a result, we identified that Recurrent Neural Networks (RNNs) are more appropriate for time series data. A key feature of Recurrent Neural Networks (RNNs) is their ability to incorporate information from previous stages into their current decision-making process. This enables the network to identify relationships and patterns in sequential data. The RNN's internal state functions as a memory, keeping information from previous inputs, and allowing it to represent sequences of variable lengths. it processes sequences by iterating through the sequence elements and maintaining a state that contains information relative to what it has seen. However, basic Simple RNN can be too simple to be truly operational. Notably, Simple RNN has a major disadvantage; as seen by the graphical representation, it consistently performs the worst of all models.

While Simple RNN should be capable of preserving information from all previous time steps in principle, it often fails in practice, particularly in deep networks, due to the infamous "vanishing gradient problem". This flaw makes the network almost untrainable. In response to this issue, more complex RNN variations, such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), were created and included in Keras. Our experiments with the basic GRU and bidirectional model produced the best results of all the models, owing to its capacity to capture long-range relationships in sequential data while being more computationally efficient than LSTMs.

When we tried Simple RNN for this data the test MAE was 9.92, which was worse than Convolution network, As simple RNN was not capable of preserving information from all previous time steps due to the infamous "vanishing gradient problem". Then we tried stacking the layers in Simple RNN with different Unit sizes and numbers of stacks, and we tried with different numbers of units in each stack for 3 stacks. But still, the test MAE hasn't been reduced. The best we got was 9.90 for the model with 3 stacked layers and 3 different unit sizes in each stack. As RNNs can suffer from the vanishing gradient problem, where the gradients become very small in the lower layers, making it difficult to train the model effectively. So, Specialized RNN variants like LSTMs and GRUs are used to mitigate this issue.

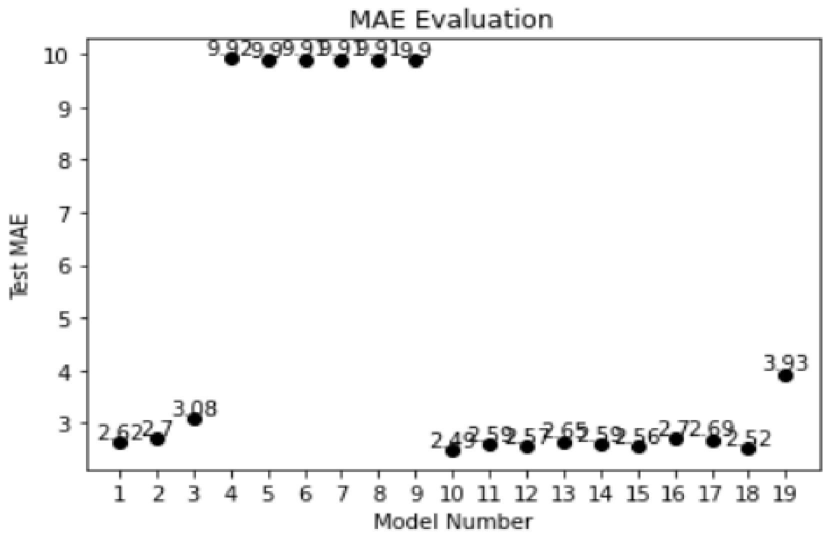
When we applied our data to GRU model the validation and test MAEs were 2.36 and 2.49 respectively, which was well under the common-sense baseline. When we tried with 2 stacked it performed better than this. We tried replacing the GRU model with LSTM and we ran seven different LSTM models with varying units in stacking recurrent layers 8, 16, and 32, and by keeping 3 stacked layers. Out of all these the model with 2 stacked layers and 8 units performed well with a test MAE of 2.56. Additionally, we used recurrent dropout to prevent overfitting. But still the model with 2 stacked layers and 8 units performed better than the remaining as this LSTM model better fits our data than the remaining models

We experimented with bidirectional data presentation to enhance accuracy and address the forgetting problem. As, It performs better in capturing contextual information and long-term dependencies, leading to improved performance in various tasks. These LSTM models all displayed similar MAE values, which were consistently lower than the common-sense model with validation and test MAE 2.38 and 2.52.

After all these, we tried the combination of 1D convnets with RNN(LSTM), but still got test MAE as 3.93, which did not even beat the common-sense baseline. As the model treats all data segments uniformly, even after pooling, which disrupted the data's sequential order.

Based on all the models, We conclude that the GRU model best fits this time series data when compared with the remaining with best test MAE of 2.49. It is recommended to avoid simple RNNs for time series analysis, as they are not so good with the vanishing gradient problem and cannot effectively capture long-term dependencies. So, consider more sophisticated RNN designs, like as LSTM and GRU, which are intended to overcome these issues.

While LSTM is a common choice for dealing with time series data, our tests indicate that GRU may produce more efficient outcomes for our dataset. So, we can say that the determination of the best model purely depends on the data we have and the hyperparameters we use in tuning it. To improve GRU models, we can try adjusting hyperparameters such as the number of units in stacked recurrent layers, and number of stacked layers, recurrent dropout rates, and the usage of bidirectional data presentation.



Various models and their test MAEs