**Assignment 3 - WEATHER TIME SERIES FORCASTING**

**REPORT**

In order to analyze time series data, we created 14 different models. Using common-sense techniques, the first model provided a baseline and produced a Mean Absolute Error (MAE) of 2.62. After that, we developed a simple machine learning model with a dense layer, which produced an MAE of 2.65, which is marginally higher. The flattening of the time series data, which eliminated the temporal context, resulted in poor performance of the dense layer model. Additionally, a convolutional model was used, but it produced subpar results because it treated every data segment equally even after pooling disturbing the sequential order of the data.

Recurrent Neural Networks (RNNs) are therefore more appropriate for time series data, as we have discovered. The ability of Recurrent Neural Networks (RNNs) to integrate information from previous steps into their current decision-making process is a crucial feature. As a result, the network can find patterns and dependencies in sequential data. The internal state of the RNN functions as a kind of memory, storing data from previous inputs and enabling it to model sequences of different lengths.

Nevertheless, the fundamental Simple RNN is frequently overly straightforward to be truly useful. Interestingly, Simple RNN has a major flaw: as the graphical representation shows, it consistently performs the worst out of all the models. The infamous "vanishing gradient problem" causes Simple RNN to struggle practically, especially in deep networks, even though in theory it should be able to retain information from all previous time steps. Because of this issue, the network is essentially untrainable. More sophisticated RNN variations, like the Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM), were created in response to this difficulty and are incorporated into Keras. Because the simple GRU model can capture long-range dependencies in sequential data, our experiments with it produced the best results of all the models data while using less computational power than long short-term memory LSTMs.

We tested six different LSTM models with different units in stacking recurrent layers (8, 16, and 32), and the model with 8 units showed the best performance. LSTMs are a well-known architecture for handling time series data effectively. We also experimented with bidirectional data presentation to improve accuracy and solve the forgetting problem, and we used recurrent dropout to prevent overfitting. Similar MAE values, which were consistently lower than the common-sense model, were displayed by all of these LSTM models.

Finally, we tried to integrate an RNN with a 1D convolution model. Nonetheless, the hybrid model produced a higher MAE of 4, most likely as a result of the convolution's inability to preserve information order. My findings suggest that simple RNNs should be avoided for time series analysis since they have trouble with the vanishing gradient issue and are unable to accurately capture long-term dependencies. Instead, consider more sophisticated RNN architectures that are intended to get over these obstacles, like LSTM and GRU. Although GRU may provide more effective results than LSTM, our experiments indicate that LSTM is a popular option for handling time series data. Hyperparameters like the number of units in stacked recurrent layers, recurrent dropout rates, and the use of bidirectional data presentation can all be tuned to improve GRU models. Additionally, since the combination of RNN and 1D convolution did not produce the best results, it is advised to concentrate on RNN architectures designed for sequential data. Convolutional methods are less appropriate for time series data analysis because they frequently cause information to be out of order.

Bottom of Form