**WEATHER TIME SERIES FORCASTING**

**REPORT**

For time series data analysis, we developed a total of 14 models. The first model served as a baseline, relying on common-sense methods, and yielding a Mean Absolute Error (MAE) of 2.62. Subsequently, we created a basic machine learning model with a dense layer, resulting in a slightly higher MAE of 2.70. The performance of the dense layer model was subpar due to the flattening of the time series data, removing the temporal context. Also attempted with a convolutional model which provided poor results as it treated all data segments uniformly, even after pooling, which disrupted the data's sequential order.

Consequently, we recognized that Recurrent Neural Networks (RNNs) are better suited for time series data. An essential feature of Recurrent Neural Networks (RNNs) is their capacity to incorporate information from past steps into their present decision-making process. This enables the network to uncover dependencies and patterns within sequential data. The RNN's internal state effectively acts as a memory, retaining information from past inputs, thus allowing it to model sequences of varying lengths. However, the basic Simple RNN is often too simplistic to be genuinely practical. Notably, Simple RNN has a significant drawback: as evidenced by the graphical representation, it consistently performs the poorest among all models. While in theory, Simple RNN should be capable of retaining information from all previous time steps, it tends to struggle practically, especially in deep networks, due to the notorious "vanishing gradient problem”. This problem renders the network virtually untrainable. In response to this challenge, more advanced RNN variants, such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), were developed and are integrated into Keras. Our experimentation with the simple GRU model demonstrated best result among all the models, primarily because of its ability to capture long-range dependencies in sequential data while being more computationally efficient compared to LSTMs.

The famous architecture for effectively handling time series data is LSTMs and we ran six different LSTM models with varying units in stacking recurrent layers (8, 16, and 32), and the model with 8 units demonstrated the best performance. Additionally, we employed recurrent dropout to prevent overfitting and experimented with bidirectional data presentation to enhance accuracy and address the forgetting problem. These LSTM models all displayed similar MAE values, which were consistently lower than the common-sense model.

In the end, we attempted to combine a 1D convolution model with an RNN. However, this hybrid model yielded a higher MAE of 3.79, likely due to the convolution's limitations in maintaining the order of information. Based on my observations, it is recommended to avoid simple RNNs for time series analysis, as they struggle with the vanishing gradient problem and cannot effectively capture long-term dependencies. Instead, consider more advanced RNN architectures, such as LSTM and GRU, which are designed to overcome these challenges. While LSTM is a popular choice for handling time series data, our experiments suggest that GRU may offer more efficient results. To optimize GRU models, consider tuning hyperparameters such as the number of units in stacked recurrent layers, recurrent dropout rates, and the use of bidirectional data presentation. Furthermore, it's advisable to focus on RNN architectures tailored for sequential data, as the combination of 1D convolution and RNN did not yield optimal results. Convolutional approaches tend to disrupt the order of information, making them less suitable for time series data analysis.

Bottom of Form