**Assignment 3 Time Series Report**

**Venkata Suresh Naradasu**

**Siddharth Yendra**

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 2.70 that was 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 since it treated every data segment equally—even after pooling—disturbing the sequential order of the data.

Our conclusion was that time series data is a better fit for Recurrent Neural Networks (RNNs). Recurrent Neural Nets (RNNs) are crucial because they may use information from previous steps to guide current decisions. In sequential data, this allows the network to find patterns and dependencies. An RNN can model sequences of different lengths because of its internal state, which functions as a kind of memory and retains information from previous inputs. But often, the fundamental Simple RNN is too straightforward to be very useful. The graphical representation indicates that Simple RNN consistently performs the worst out of all the models, which is a notable downside. Although the renowned "vanishing gradient problem" causes Simple RNN to struggle operationally, particularly in deep networks, it should be able to retain knowledge from all prior time steps theoretically. Due to this issue, the network is essentially untrainable. More sophisticated RNN variations, including 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 basic GRU model is more computationally efficient than LSTMs and can capture long-range dependencies in sequential data, our experimentation with it produced the best results of all the models.

LSTMs are a well-known architecture for handling time series data efficiently. We tested six different LSTM models with varied numbers of units in stacked recurrent layers (8, 16, and 32), and the model with eight units performed the best. Recurrent dropout was also used to avoid overfitting, and bidirectional data presentation was tested to improve accuracy and solve the forgetting issue. Similar MAE values, which were consistently lower than the common-sense model, were displayed by all of these LSTM models.

At last, we tried to integrate an RNN with a 1D convolution model. The greater MAE of 3.79 obtained by this hybrid model, however, is probably attributable to the convolution's shortcomings in preserving the information's order. Simple RNNs are advised against using them for time series analysis due to their inability to capture long-term dependencies and their difficulties with the vanishing gradient problem, based on my observations. LSTM and GRU are examples of more sophisticated RNN designs that are intended to address these issues. Though our trials indicate that GRU might provide more effective outcomes, LSTM is a common option for processing time series data. Hyperparameters including the number of units in stacked recurrent layers, recurrent dropout rates, and the usage of bidirectional data presentation can all be tuned to improve GRU models. Additionally, since 1D convolution plus RNN did not produce the best results, it is recommended that you concentrate on RNN designs designed for sequential data. Convolutional techniques are less appropriate for time series data analysis since they frequently upset the information's order.