**AML Assignment – 3 Sundeep Rachuri**

In order to analyse time series data, I developed a total of 14 models. I began with a straightforward, commonsense method that yielded a Mean Absolute Error (MAE) of 2.62. I made a simple, multi-layered machine learning model since I felt I could do better. Unexpectedly, it did marginally worse, with an MAE of 2.82. and discovered that I had flattened the time series data and eliminated some contextual information, which was why my new model was having trouble. As a result, the model had trouble identifying significant trends. Convolutional modelling was another method I attempted, but it too didn't perform well. The sequential character of our time series was disrupted by its use of pooling and independent treatment of each component of the data. In retrospect, I think that trying to be fancy actually made matters worse. The most straightforward methods are sometimes the most effective, particularly when working with time-dependent data.

A graph with a line and a line

Description automatically generated

A graph of training and validation

Description automatically generated

After giving it some more thought, I realised that recurrent neural networks (RNNs), especially their more advanced varieties, are naturally more appropriate for time series analysis. RNNs’ primary benefit is their capacity to transfer information between successive steps, which enables them to identify patterns and temporal connections in the data.

RNNs can handle sequences of varied length thanks to their architecture, which uses their internal state as a dynamic memory. When working with time series data of irregular duration or frequency, this feature is quite helpful. Though conceptually simple and frequently suggested for beginners, I quickly learnt that the basic RNN is inadequate in real-world applications. The comparative visualisations I created clearly demonstrated how frequently its performance placed at the bottom of my model ensemble. The vanilla RNN's simplicity comes at a high price, as evidenced by its weak capacity to identify long-term dependencies and its vulnerability to problems such as vanishing gradients. This insight prompted me to investigate more sophisticated architectures, such as LSTMs and GRUs, which mitigate numerous drawbacks of the basic RNN while maintaining its essential advantages for sequential data processing.

I encountered some intriguing issues and findings when researching recurrent neural networks (RNNs). Although the basic RNN design has the potential to retain information from all previous timesteps, it typically fails in practice due to the notorious "vanishing gradient" problem. Networks designed to handle longer sequences are particularly affected by this issue, which renders them very impossible to train. To get around these limitations, researchers developed more sophisticated versions, such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs). The purpose of these more intricate designs, which are readily available in frameworks like Keras, is to mitigate the shortcomings of their simpler predecessors.

According to my comparison investigation, the GRU model performed exceptionally well. It continuously produced better results across my test suite because of its capacity to identify long-range dependencies in sequential data and its computational efficiency in comparison to LSTMs. In order to learn more about LSTM architectures, I experimented with six different setups, changing the number of recurrent layer units (8, 16, and 32). It's interesting to note that the model with eight units performed the best. My results supported the reputation that LSTMs have built up as reliable time series analysis tools.

I used dropout regularisation and bidirectional data flow to try to improve accuracy even further and deal with any overfitting. The Mean Absolute Error (MAE) values of these LSTM variants were consistently lower than those of the baseline model. The significance of careful model selection and design in time series analysis was highlighted by this exploration of different RNN architectures. Even though more intricate models frequently have potential, attaining the best outcomes required striking the correct balance between complexity and usefulness.

A graph of training and validation

Description automatically generated

Lastly, I experimented with integrating a one-dimensional convolutional model with a recurrent neural network (RNN). However, because of difficulties in maintaining appropriate data sequencing, this hybrid strategy produced a mean absolute error of 3.75, which was less than ideal.

My research leads me to recommend against time series analysis using simple RNNs. These models struggle to capture long-range interdependence and are susceptible to the vanishing gradient problem. Instead, I recommend examining more intricate RNN architectures designed to address these problems, such as Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) networks. Although LSTM is frequently used for time series data, my research indicates that GRU might perform better. I recommend adjusting hyperparameters such as the number of hidden layers, the unit count in stacked layers, and bidirectional information flow to optimise GRU models.

Since the RNN-1D convolution hybrid failed to give in the best results, I recommend concentrating on RNN designs designed especially for sequential data. Convolutional methods are less useful for time series analysis since they frequently distort temporal information.

