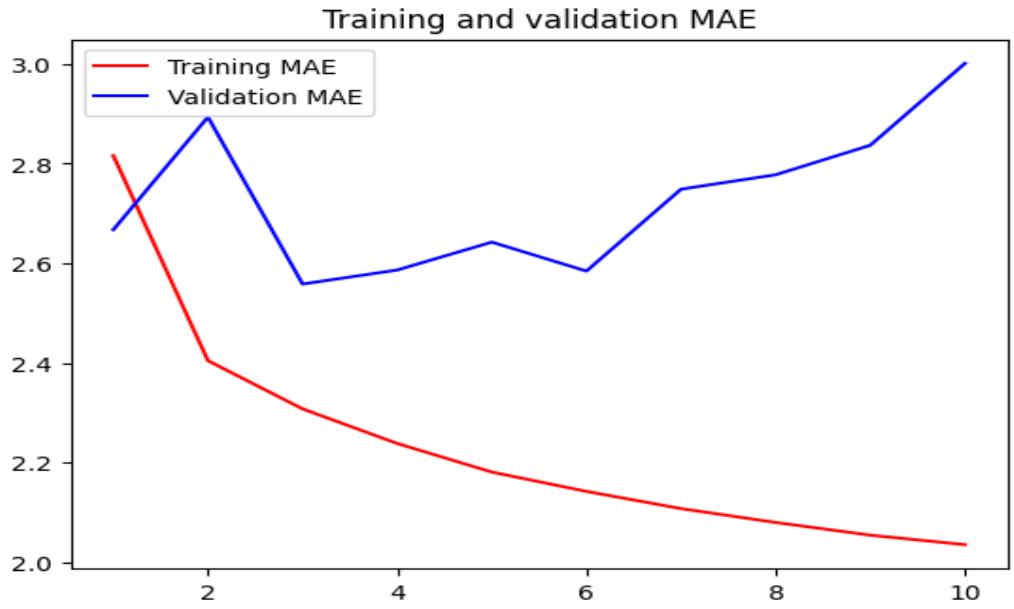
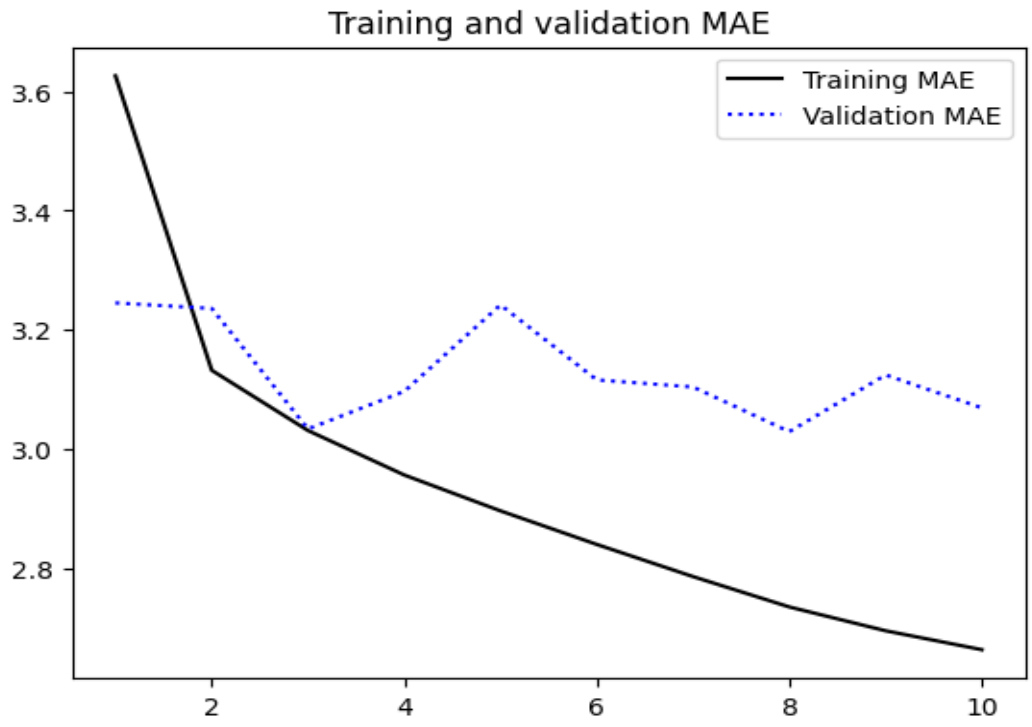
AML Assignment – 3

Goli Sai Priyanka

I built an aggregate of 14 models to analyze time series data. My starting point was a simple, common-sense approach that gave me a Mean Absolute Error (MAE) of 2.62. I thought I could do better, so I created a basic machine learning model with lots of layers. Surprisingly, it performed slightly worse, with an MAE of 2.82. And realized that my new model struggled because I had flattened the time series data and removed some contextual information. This made it hard for the model to pick up on important patterns. I also tried using convolutional modeling, but that didn't work well either. It treated each part of the data separately and used pooling, which messed up the sequential nature of our time series. In hindsight, my attempts to get fancy actually made things worse. Sometimes the simplest approaches work best, especially when dealing with time-dependent data.





Upon further reflection, I recognized that recurrent neural networks (RNNs), particularly their more sophisticated variants, are inherently better suited for time series analysis. The fundamental advantage of RNNs lies in their ability to propagate information across sequential steps, allowing them to capture temporal dependencies and patterns within the data.

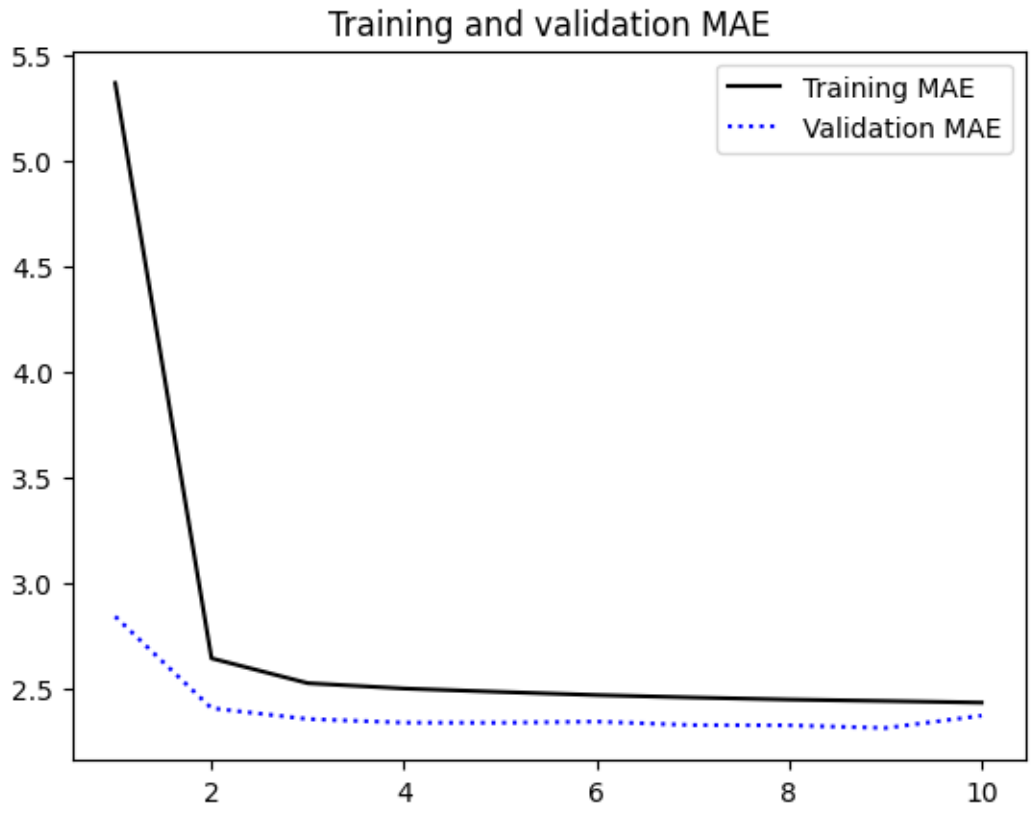
The architecture of RNNs, with their internal state acting as a dynamic memory, enables them to process sequences of variable length. This characteristic is particularly advantageous when dealing with time series data of inconsistent duration or frequency. However, I soon discovered that the basic RNN, while conceptually straightforward and often recommended for novices, falls short in practical applications. Its performance consistently ranked at the bottom of my model ensemble; a fact starkly illustrated by the comparative visualizations I generated. The simplicity of the vanilla RNN comes at a significant cost, manifesting in its limited ability to capture long-term dependencies and its susceptibility to issues like vanishing gradients. This realization led me to explore more advanced architectures like LSTMs and GRUs, which address many of the shortcomings of the simple RNN while retaining its core benefits for sequential data processing.

During my investigation into recurrent neural networks (RNNs), I came across a few fascinating problems and discoveries. The infamous "vanishing gradient" issue is the reason why the fundamental RNN design frequently fails in practice, while having the theoretical ability to preserve knowledge from all prior timesteps. This problem is most noticeable in networks built to handle longer sequences, making them practically untrainable. More advanced versions, such Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM) networks, were created by researchers in order to overcome these constraints. These more complex designs, which are

easily accessible in frameworks such as Keras, are meant to lessen the drawbacks of their more basic forerunners.

In my comparative analysis, the GRU model emerged as a standout performer. Its ability to capture long-range dependencies in sequential data, coupled with its computational efficiency relative to LSTMs, consistently yielded superior results across my test suite. Delving deeper into LSTM architectures, I experimented with six distinct configurations, varying the number of units in the recurrent layers (8, 16, and 32). Interestingly, the model with 8 units demonstrated the most refined performance. LSTMs have earned their reputation as robust solutions for time series analysis, and my findings corroborated this.

In an effort to further enhance accuracy and address potential overfitting, I incorporated bidirectional data flow and implemented dropout regularization. These LSTM variants consistently outperformed the baseline model, achieving lower Mean Absolute Error (MAE) values. This journey through various RNN architectures underscored the importance of thoughtful model selection and configuration in time series analysis. While more complex models often hold promise, finding the right balance between sophistication and practicality proved crucial in achieving optimal results.



Finally, I conducted an experiment combining a recurrent neural network (RNN) with a one-dimensional convolutional model. However, this hybrid approach yielded a suboptimal mean absolute error of 3.75, likely due to issues with maintaining proper data sequencing.

Based on my findings, I advise against using basic RNNs for time series analysis. These models are sensitive to the vanishing gradient issue and have difficulty capturing long-range interdependence. Rather, I suggest looking at more complex RNN architectures like Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) networks that are intended to deal with these issues. While LSTM is a popular choice for time series data, my experiments suggest that GRU may offer superior performance. To optimize GRU models, I suggest fine-tuning hyperparameters like bidirectional information flow, the number of hidden layers, and the unit count in stacked layers.

Given that the RNN-1D convolution hybrid did not produce optimal results, I suggest focusing on RNN architectures specifically tailored for sequential data. Convolutional approaches often distort temporal information, making them less effective for time series analysis.