Week:4 ADML Assignment Transformer

Task Implementation:

For this assignment, I selected the Yahoo Stock Market dataset. The implementation of the transformer architecture can be found in the attached Python code. Despite being challenging, this task was incredibly enjoyable. However, it was also quite tough to complete.

Model Comparison:

Regarding the model-wise comparison, the initial observation that caught my attention was the significantly greater complexity and time consumption of the Transformer architecture compared to the LSTM model. While I don't have precise timing details, tasks that the LSTM could complete in under 2 minutes often took the Transformer upwards of 5 minutes. Moving on to the training and validation epochs loss analysis, a closer look at Fig. 1 and Fig. 2 reveals that the LSTM outperforms the Transformer. The Transformer, running for 50 epochs, lags behind the LSTM's performance achieved in 30 epochs. Adding to this, the Transformer also benefitted from hyperparameter analysis. Altogether, these aspects raise concerns about the efficiency of the Transformer in this context.

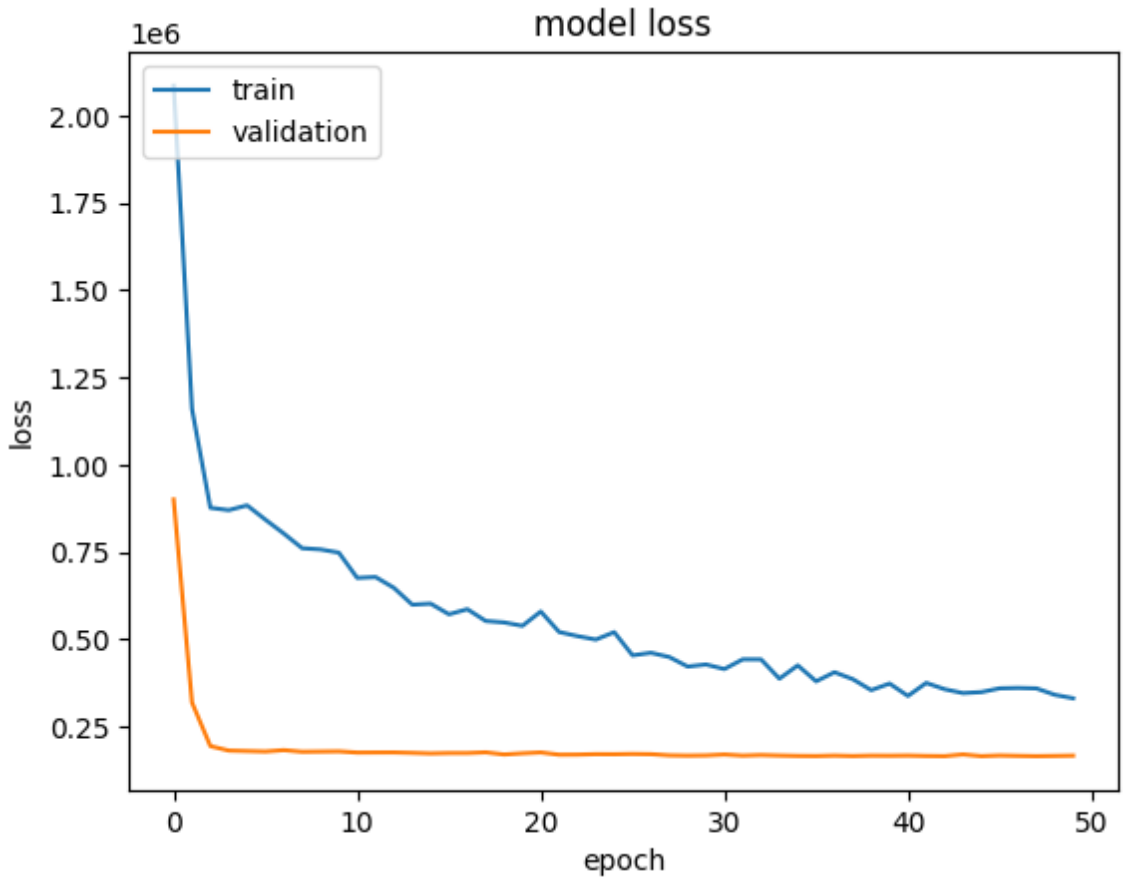

Fig.1 Transformer Train & Validation Loss

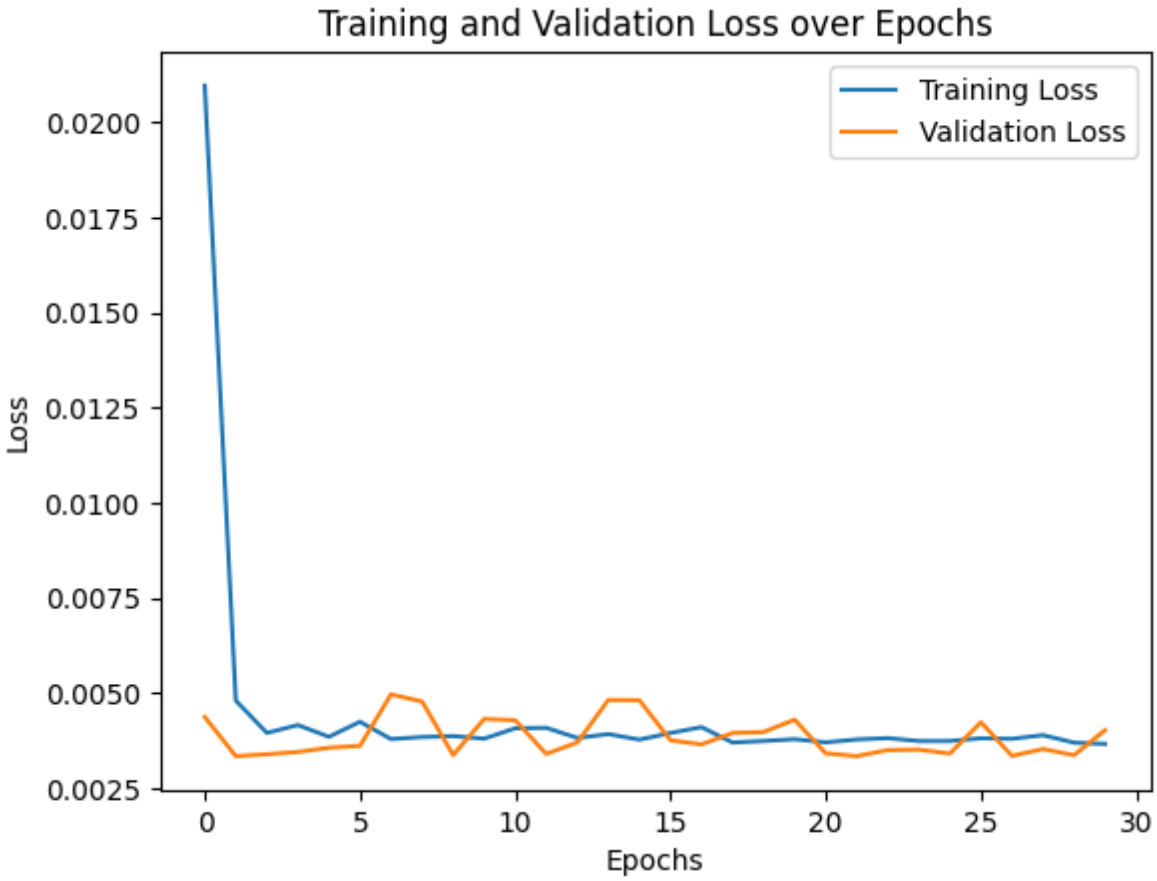

Fig.2 LSTM Training & Validation Loss

Moving on to Fig.3 and Fig.4, the initial plot depicts the daily price change. Although there may appear to be differences between Fig.3 and Fig.4, it's crucial to note that these variations cannot be solely attributed to differences in model performance. This is because the response variable for the LSTM differs from that of the Transformer. Specifically, for the Transformer, we aim to predict the closing price one day ahead. Conversely, the LSTM predicts the difference between the closing price one day ahead and the current day's closing price.

Now, turning our attention to the histogram of residuals, a substantial disparity becomes apparent between the two models. The frequency of observations in LSTM is confined to or near zero, while this is not the case for the Transformer. Transformers exhibit residual frequencies throughout, except for zero 🙁.

The last plot showcases cumulative returns, marking a turning point in our analysis. It becomes evident that, somehow, the Transformer model exhibits superior performance. However, to validate this observation, further analysis and hyperparameter tuning are required for both the LSTM and Transformer Neural Network. I was genuinely surprised to observe that, over time, the Transformer had better one-day-ahead predictions compared to the LSTM models.

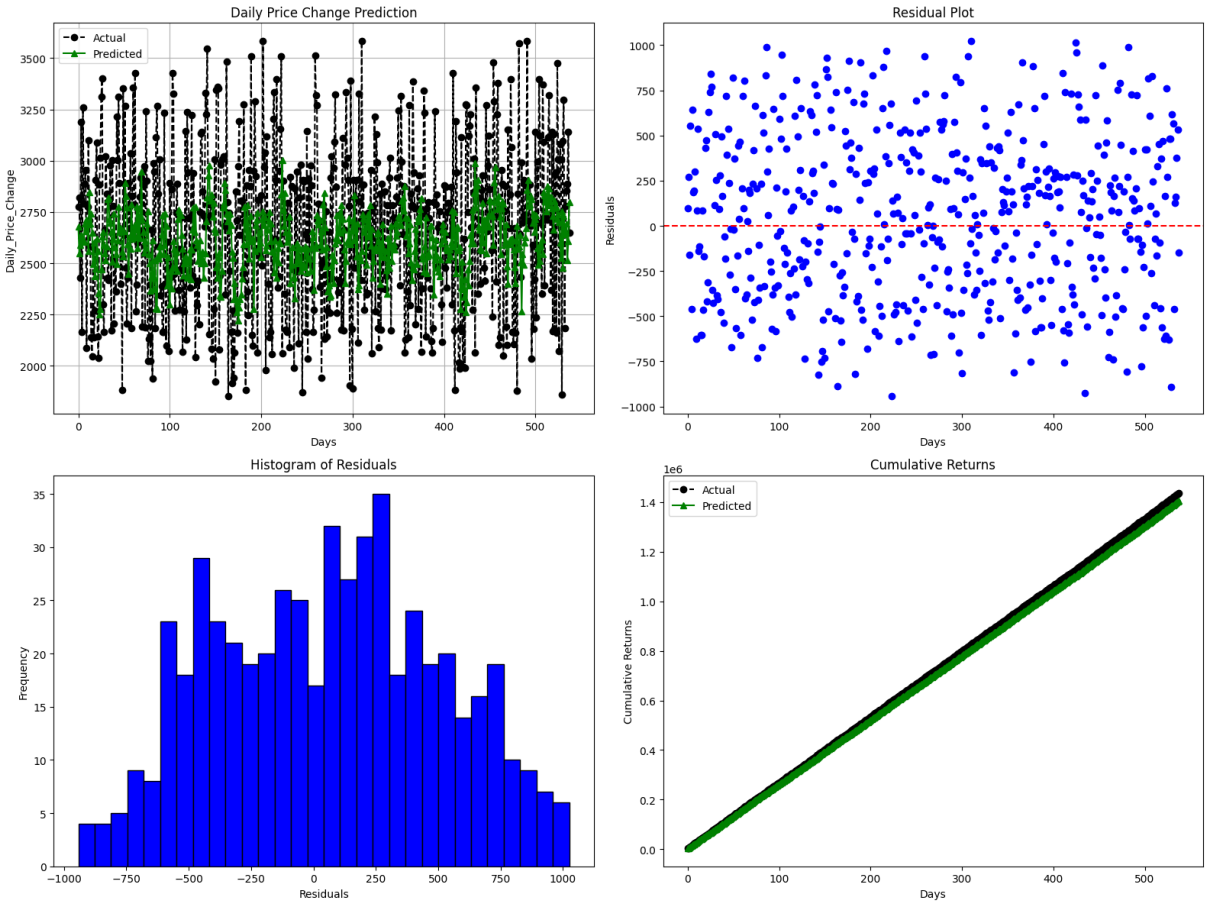

Fig.3 Transformer Plots

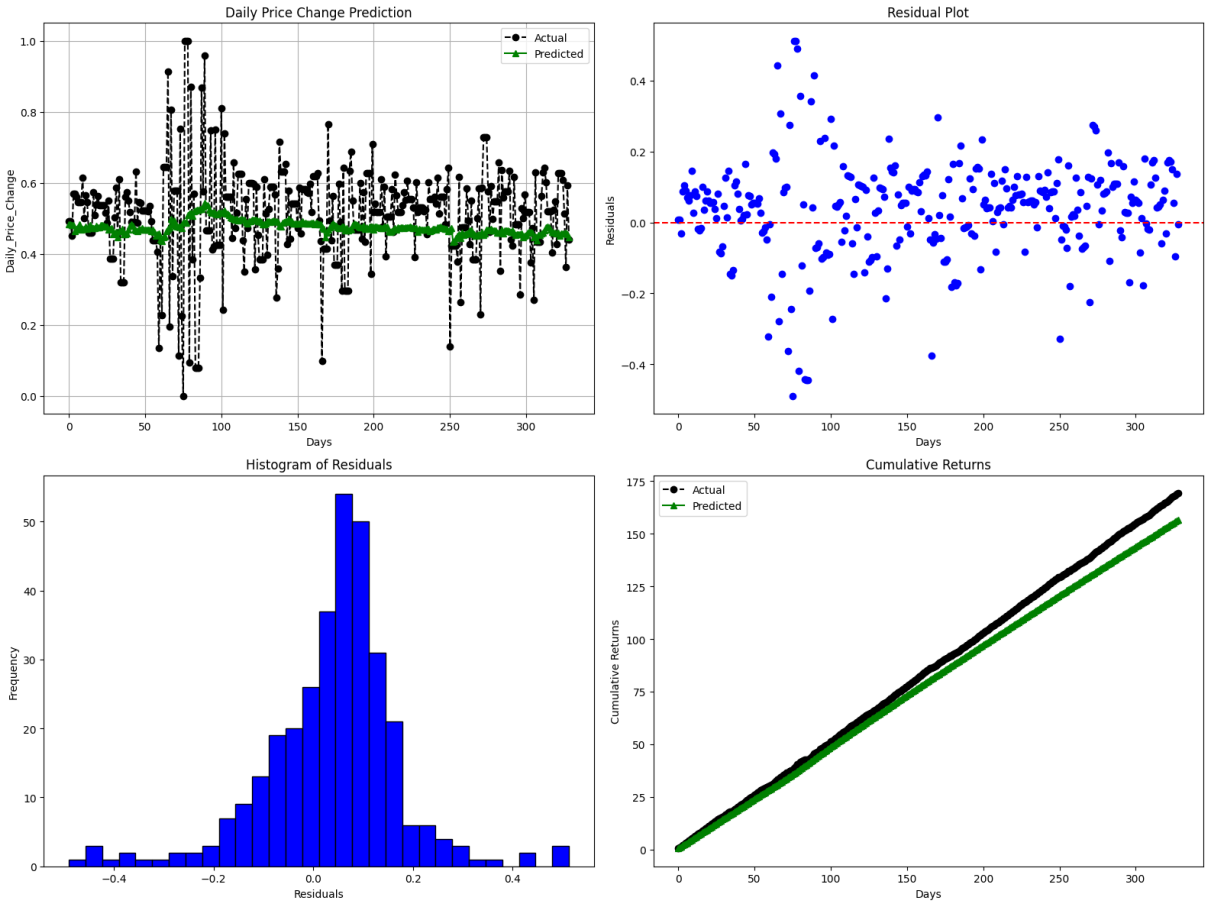

Fig.4 LSTM Plots.

Transformer Architecture:

The Transformer architecture, introduced in the paper "Attention is All You Need" by Vaswani et al. in 2017, has become a cornerstone in natural language processing (NLP) and beyond. It was primarily designed for sequence-to-sequence tasks like machine translation, but its versatility has led to its adoption in various domains.

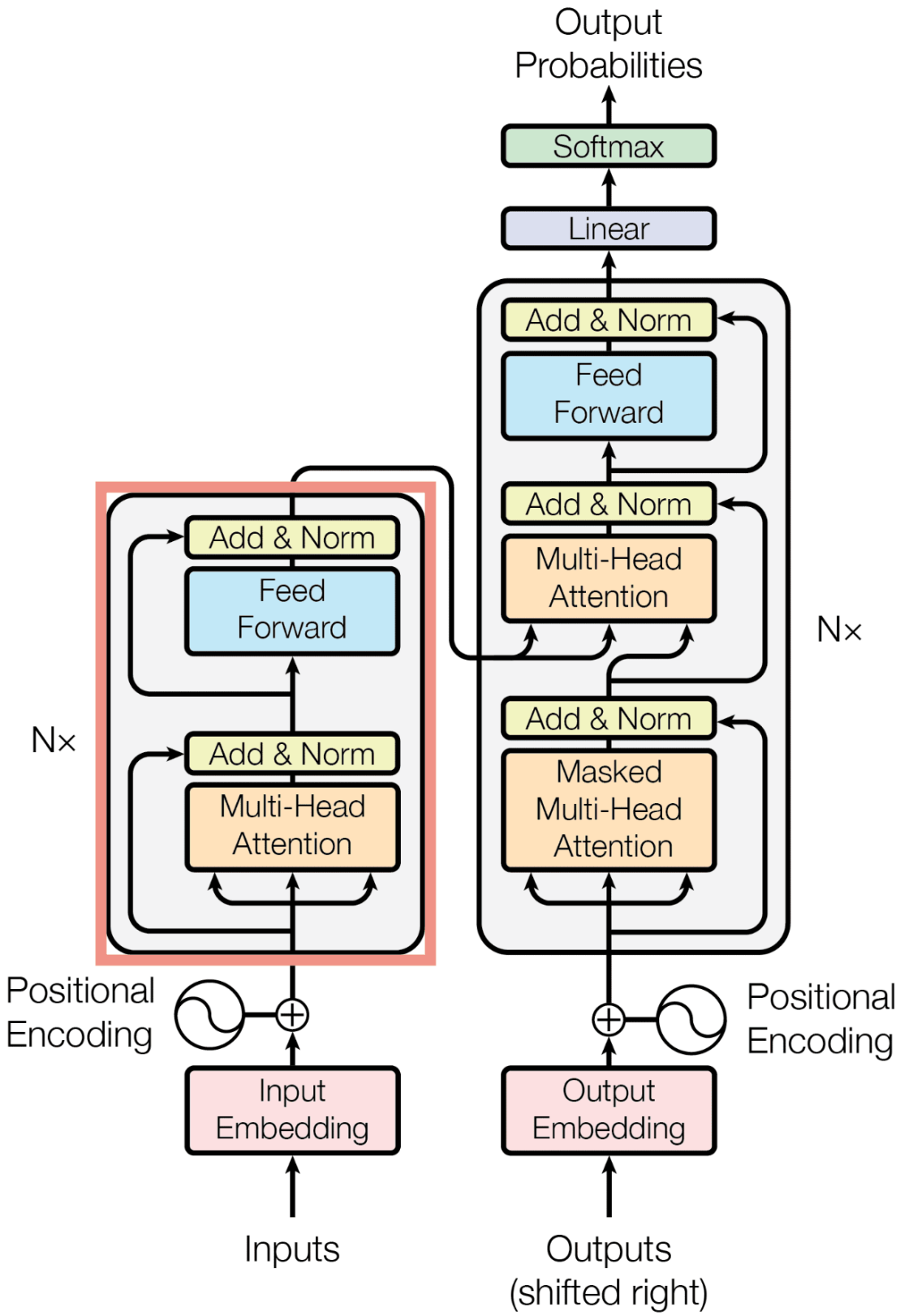

Fig.5 Transformer Architecture.

Key components of the Transformer architecture include:

- **Attention Mechanism:**

  The core innovation of the Transformer is the self-attention mechanism, which enables the model to weigh the importance of different parts of the input sequence when making predictions. This mechanism allows the model to consider all positions in the input sequence simultaneously.

- **Encoder-Decoder Structure:**

  The Transformer architecture consists of an encoder and a decoder. The encoder processes the input sequence, and the decoder generates the output sequence. Each encoder and decoder layer has its own set of parameters.

- **Multi-Head Attention:**

  To capture different aspects of the input sequence, the self-attention mechanism is extended into multiple heads. Each head focuses on a different subset of relationships in the data, providing a more nuanced understanding.

- **Positional Encoding:**

  Since the Transformer lacks inherent sequential information, positional encodings are added to the input embeddings to provide the model with information about the relative or absolute position of each token in the sequence.

- **Feedforward Neural Networks:**

  Each encoder and decoder layer includes a feedforward neural network. These networks process the information from the attention mechanism and contribute to the final predictions.

- **Layer Normalization and Residual Connections:**

  Each sub-layer (e.g., attention, feedforward) in both the encoder and decoder is followed by layer normalization and a residual connection. These helps stabilize training and enable the network to learn more effectively.

- **Transformer Block:**

A Transformer block is the basic building unit, consisting of a multi-head self-attention layer, feedforward layer, and layer normalization. Stacking multiple Transformer blocks forms the full model.

Hyper Parameter Analysis:

For the Transformer model, I adjusted three different hyperparameters: Num_heads, ff_dim, and num_transformer_blocks. The optimal values I found for them were 6, 4, and 2, respectively. I iterated through these values for 50 epochs to determine the best possible combination. Although it might not seem apparent on the surface, hyperparameter tuning impacts accuracy at a minuscule level, and this can make a significant difference, especially in longer-term tasks such as stock market forecasting.

Conclusion:

In conclusion, the Transformer architecture is relatively complex compared to LSTM, and it requires a considerable amount of time for model generation. However, with meticulous hyperparameter fine-tuning and optimal data, we can confidently achieve accurate time-series predictions.