Stock Price Prediction

Financial Market analyzer using LSTM

Aiden Mclean, Lei Wang, Suraj

COMP-3704: NEURAL NETWORKS AND DEEP LEARNING

Introduction:

This project aims to leverage historical data for predicting stock prices, addressing the challenging task of time series forecasting in the highly volatile financial markets. To achieve this, we will employ the Long Short-Term Memory (LSTM) model, a specialized type of Recurrent Neural Network (RNN) known for its ability to process sequential data and retain information over extended input sequences. This capability makes LSTMs particularly effective for forecasting stock prices.

The historical and real-time stock price data will be sourced using the Yahoo Finance API (yfinance). The primary focus of the project will be to predict the closing prices of selected stocks.

Initial Stage:

The following experiments have been conducted:

1. Single Layer LSTM Model: Initial model training to understand baseline performance.

2. Stacked LSTM Model without Regularization: Experimented with a deeper model.

3. Stacked LSTM Model with Regularization: Experimented with a deeper model and added L2 regularization to control overfitting.

4. Bidirectional LSTM Model: Tested a Bidirectional LSTM to improve the model's understanding of time dependencies.

5. Hyperparameter Tuning: Adjustments in learning rate, dropout rate, and batch size were also explored.

Optimization Process:

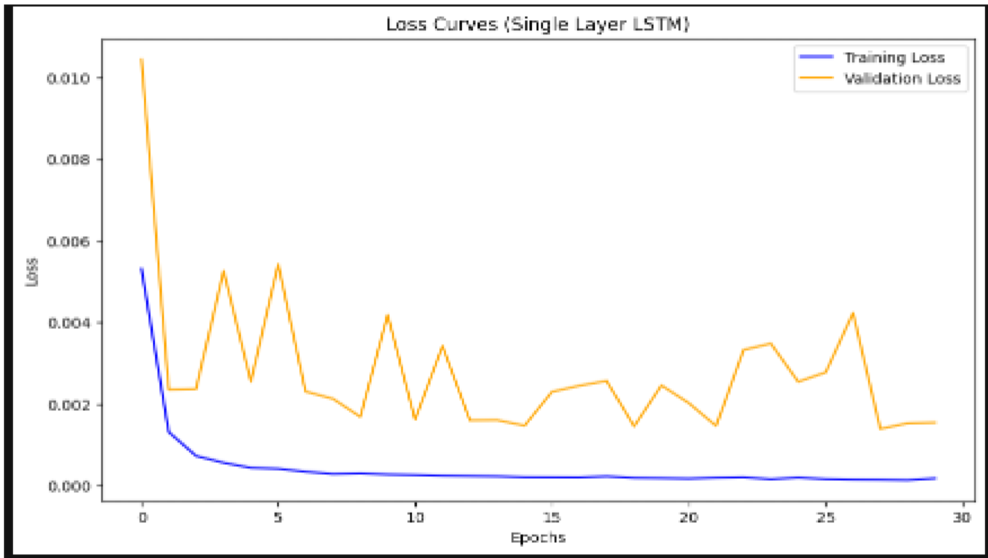
1. Overfitting: This issue was especially prevalent with deeper LSTM models, prompting the use of L2 regularization and dropout techniques to mitigate the risk of overfitting and improve generalization.

2. Slow Convergence: Certain hyperparameter combinations led to slower convergence, increasing training time and making the tuning process more complex. Addressing this required careful adjustments in learning rate and batch size.

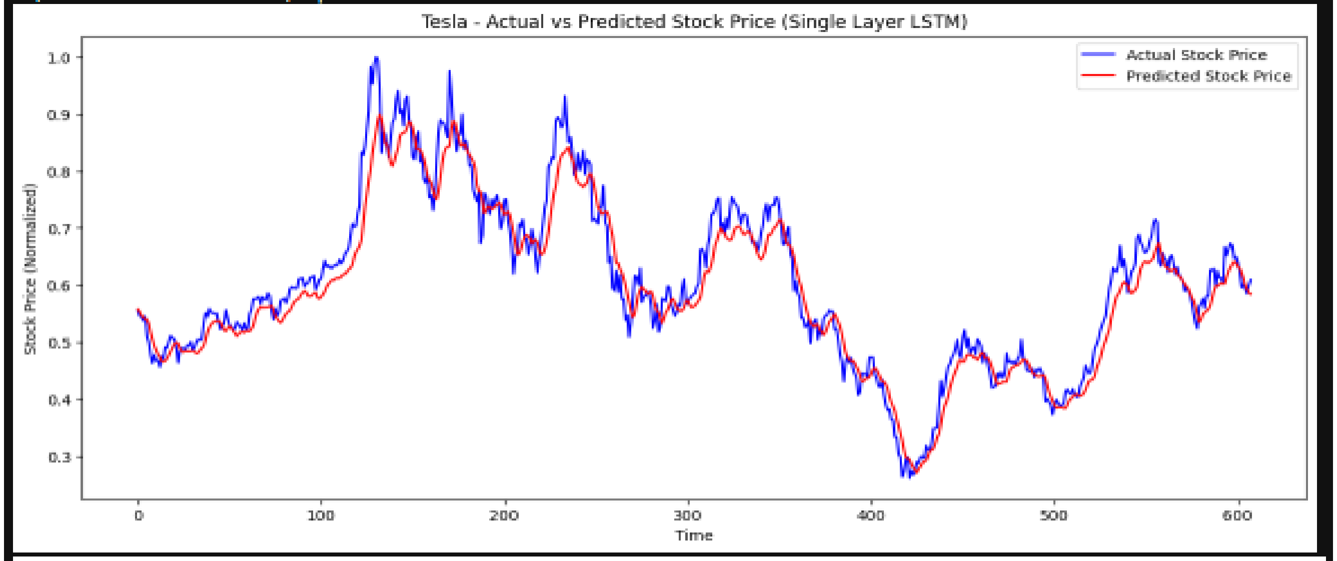
3. Data Scaling and Prediction Alignment: Ensuring consistent data scaling and proper input shapes for multi-step forecasting posed challenges. These were resolved by refining the data preprocessing approach and using robust scaling methods like MinMaxScaler.

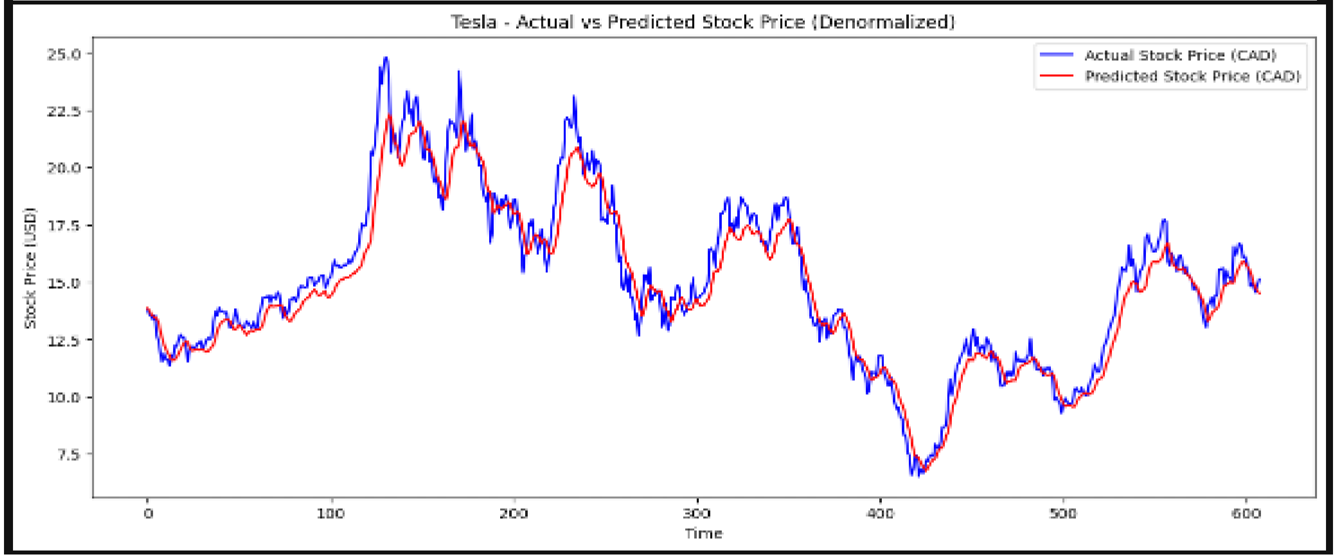
Results:

1. Single Layer LSTM

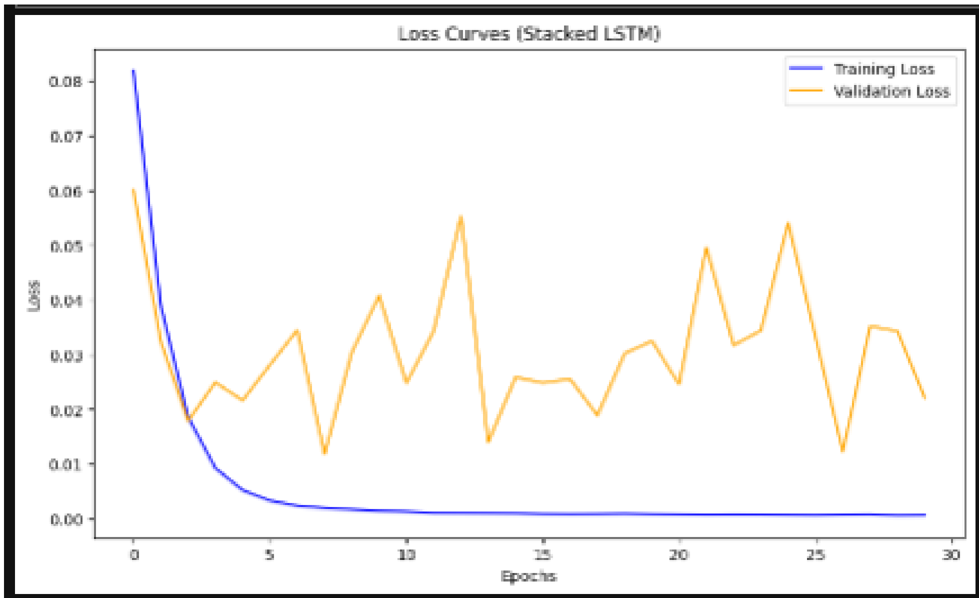


```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
```

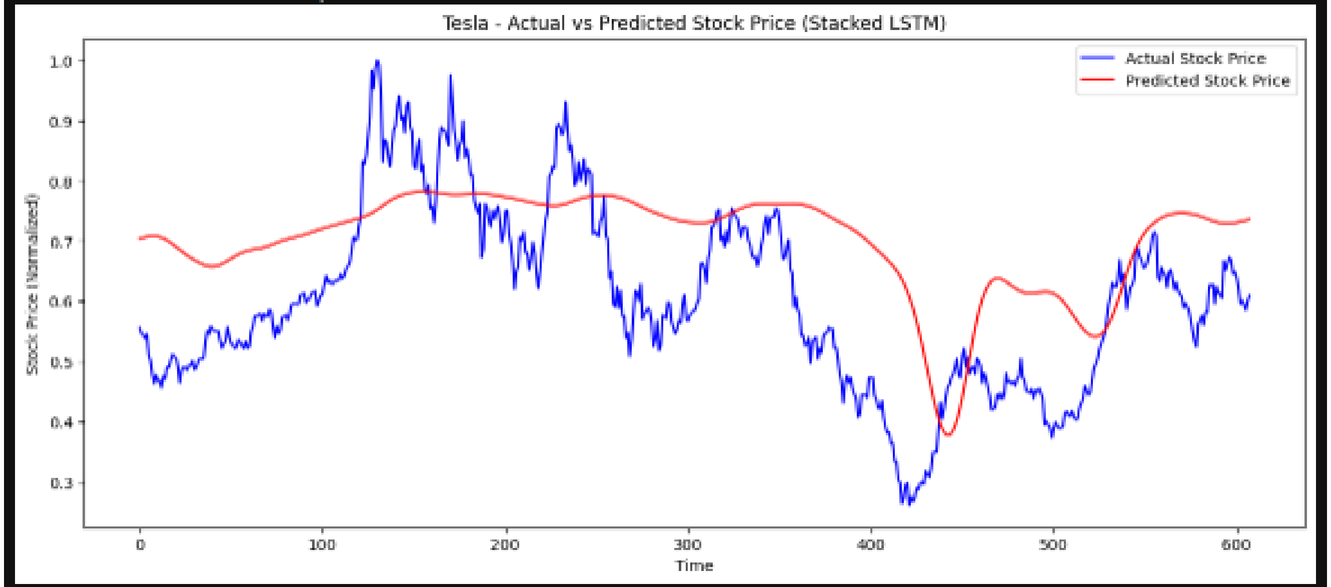


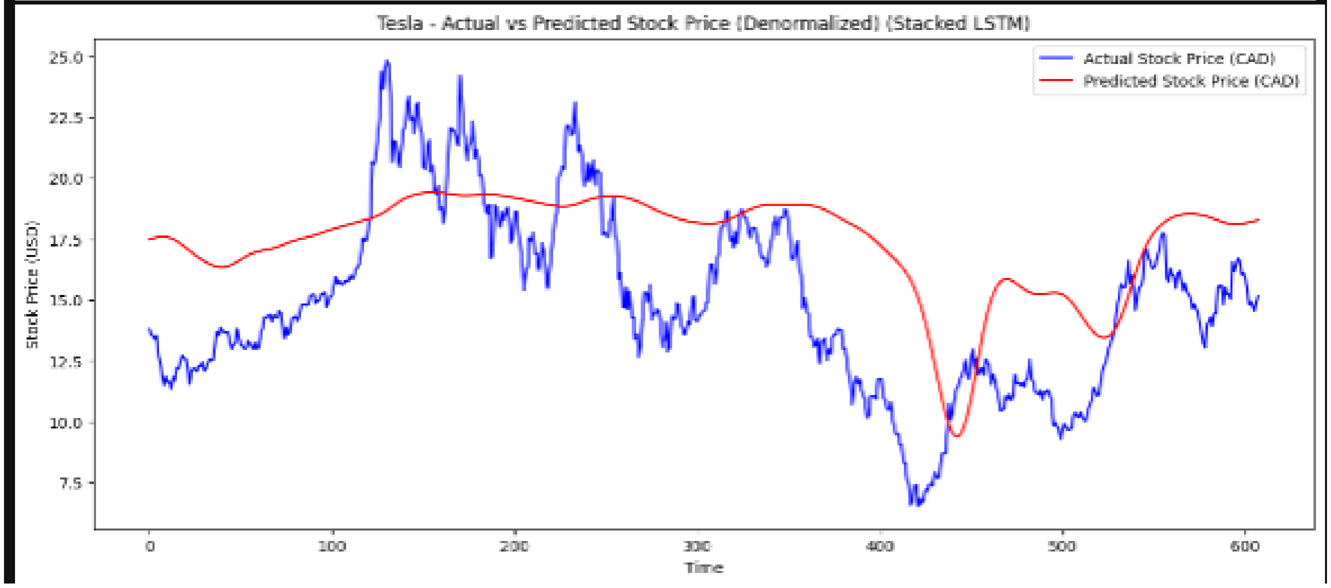


2. Stacked LSTM with L2 Regularization:

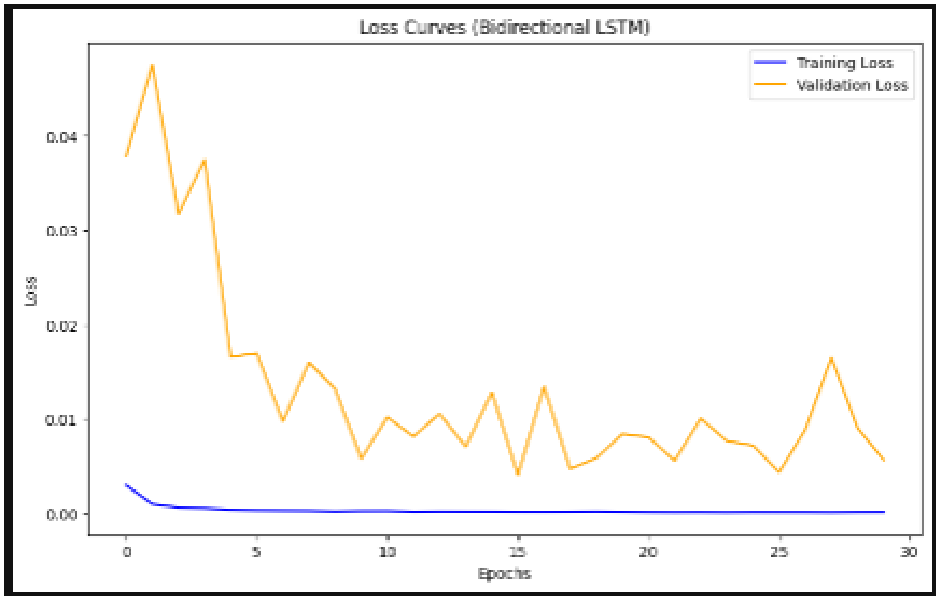


```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
```

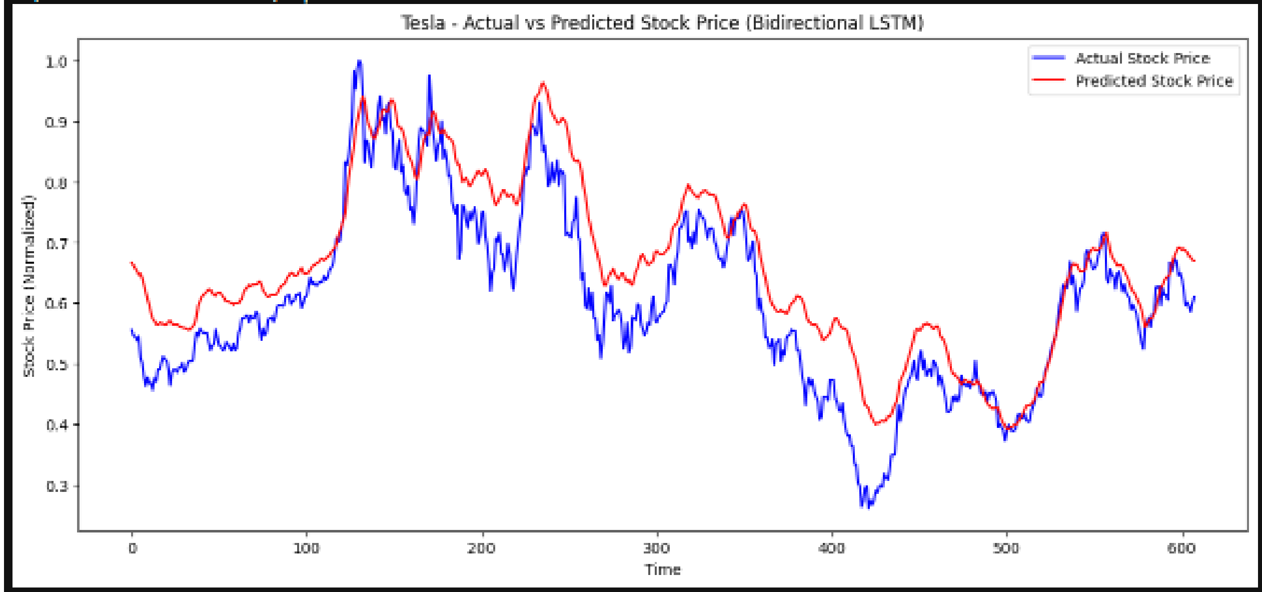


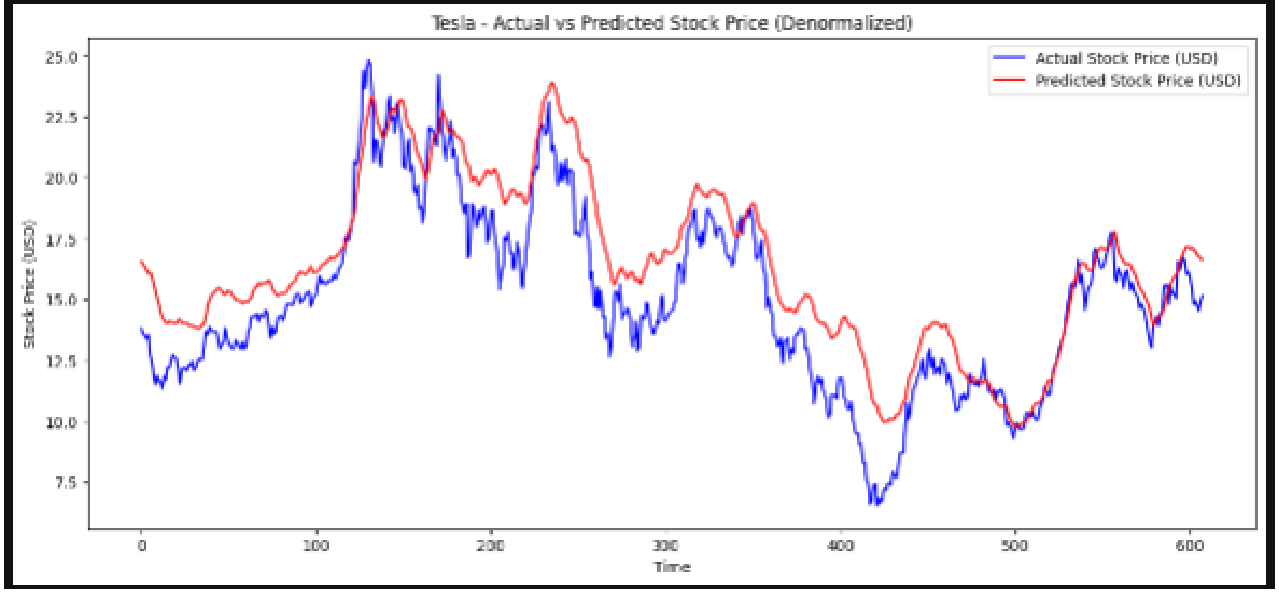


3. Bidirectional LSTM Model:



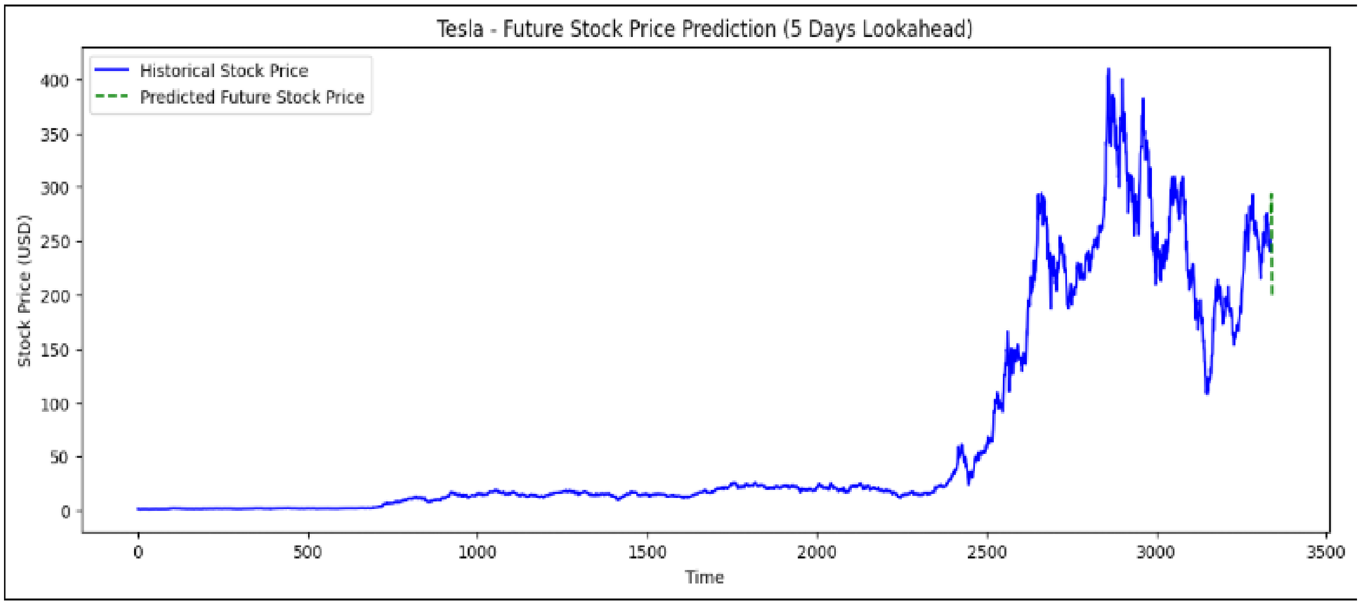
```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
```





Challenges and Solutions:

The challenges encountered in the First stage primarily revolved around ensuring the model's performance on unseen data. While the model seemed to perform well during training and testing, it struggled to accurately capture and reflect the stock price movements in new or unseen data. This limitation highlighted the model's difficulty in generalizing effectively to unpredictable market conditions and accurately predicting price trends.



The green line at the end shows the predicted stock price 5 days in the future and it looked something like this which was clearly wrong.

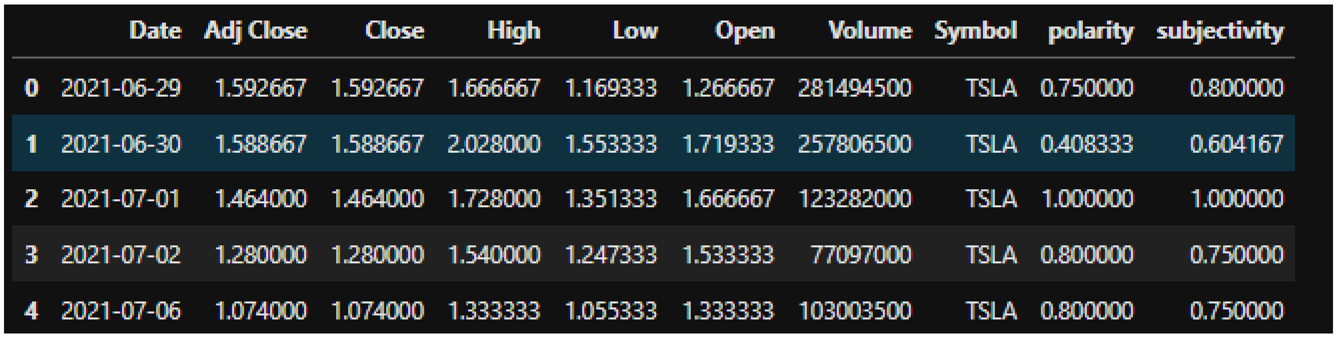
Middle Stage:

In This Stage We Practiced with using sentiment analysis by analysing tweets from Elon Musk.

Data:

We got the data from Kaggle which packs all the tweets from Elon Musk from 2010 to 2023

We used TextBlob to analyze tweets from Elon musk which give us two more features to try with that is polarity and subjectivity.



	Date	Adj Close	Close	High	Low	Open	Volume	Symbol	polarity	subjectivity
0	2021-06-29	1.592667	1.592667	1.666667	1.169333	1.266667	281494500	TSLA	0.750000	0.800000
1	2021-06-30	1.588667	1.588667	2.028000	1.553333	1.719333	257806500	TSLA	0.408333	0.604167
2	2021-07-01	1.464000	1.464000	1.728000	1.351333	1.666667	123282000	TSLA	1.000000	1.000000
3	2021-07-02	1.280000	1.280000	1.540000	1.247333	1.533333	77097000	TSLA	0.800000	0.750000
4	2021-07-06	1.074000	1.074000	1.333333	1.055333	1.333333	103003500	TSLA	0.800000	0.750000

Experimentation and Results:

We attempted to build a model using this data but found it to be ineffective, as there was no discernible correlation between his tweets and the share prices. Despite extensive analysis and testing, the patterns in the data did not indicate any significant influence of his social media activity on market performance. This suggests that external factors or other variables might play a more dominant role in driving share price movements, rendering the data less relevant for predictive modeling in this context.

Final Stage:

Up to this stage, after conducting extensive research, we realized that share prices are highly volatile and influenced by numerous factors beyond historical price data. With this understanding, we decided to explore a different approach, aiming to account for the multifaceted nature of price movements and incorporate additional variables that might better capture the complexity of market dynamics.

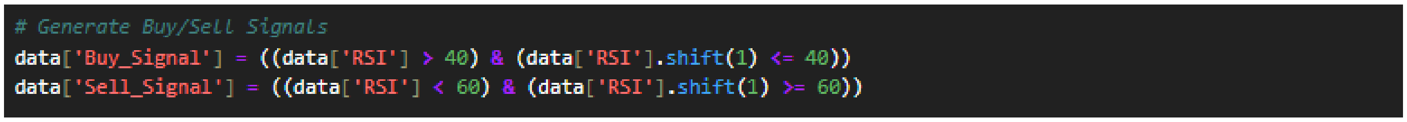
We started so by adding the technical indicators in the data:

1. **EMAF:** Stands for Exponential Moving Average - Fast.
   It is calculated using a short period (length=20), making it a faster-moving average that reacts more quickly to price changes.
2. **EMAM:** Stands for Exponential Moving Average - Medium.
   It uses a medium-length period (length=100) and moves more slowly than the fast EMA, providing a smoother trend line.
3. RSI helps identify overbought and oversold conditions, signaling potential reversals or entry/exit points.
4. MACD combines trend and momentum, providing reliable signals for trend reversals and strength.
5. ATR measures volatility and helps set appropriate stop-loss levels or assess risk.
6. Bollinger Bands help identify breakout opportunities and assess volatility.
7. VWAP is a crucial benchmark for intraday trading, showing where most of the trading volume occurred.

BIG Change:

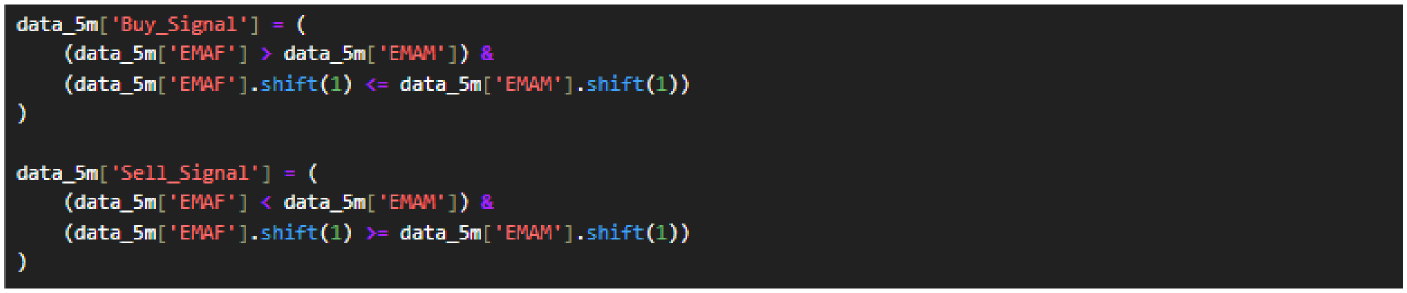
Added Additional Features, Buy Signal and Sell Signal. Based on RSI and Moving Averages.

For one day data



```python
# Generate Buy/Sell Signals
data['Buy_Signal'] = ((data['RSI'] > 40) & (data['RSI'].shift(1) <= 40))
data['Sell_Signal'] = ((data['RSI'] < 60) & (data['RSI'].shift(1) >= 60))
```

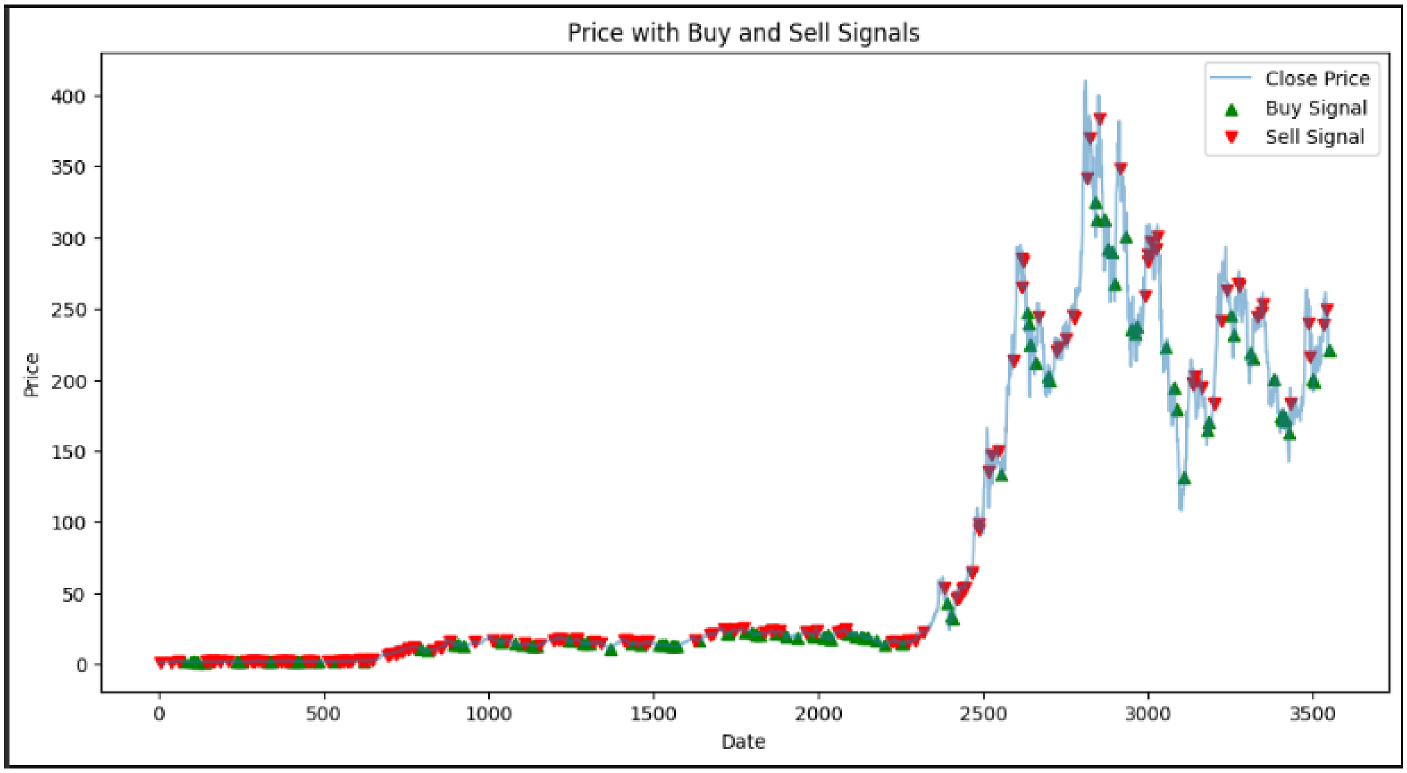
For Five Minute data



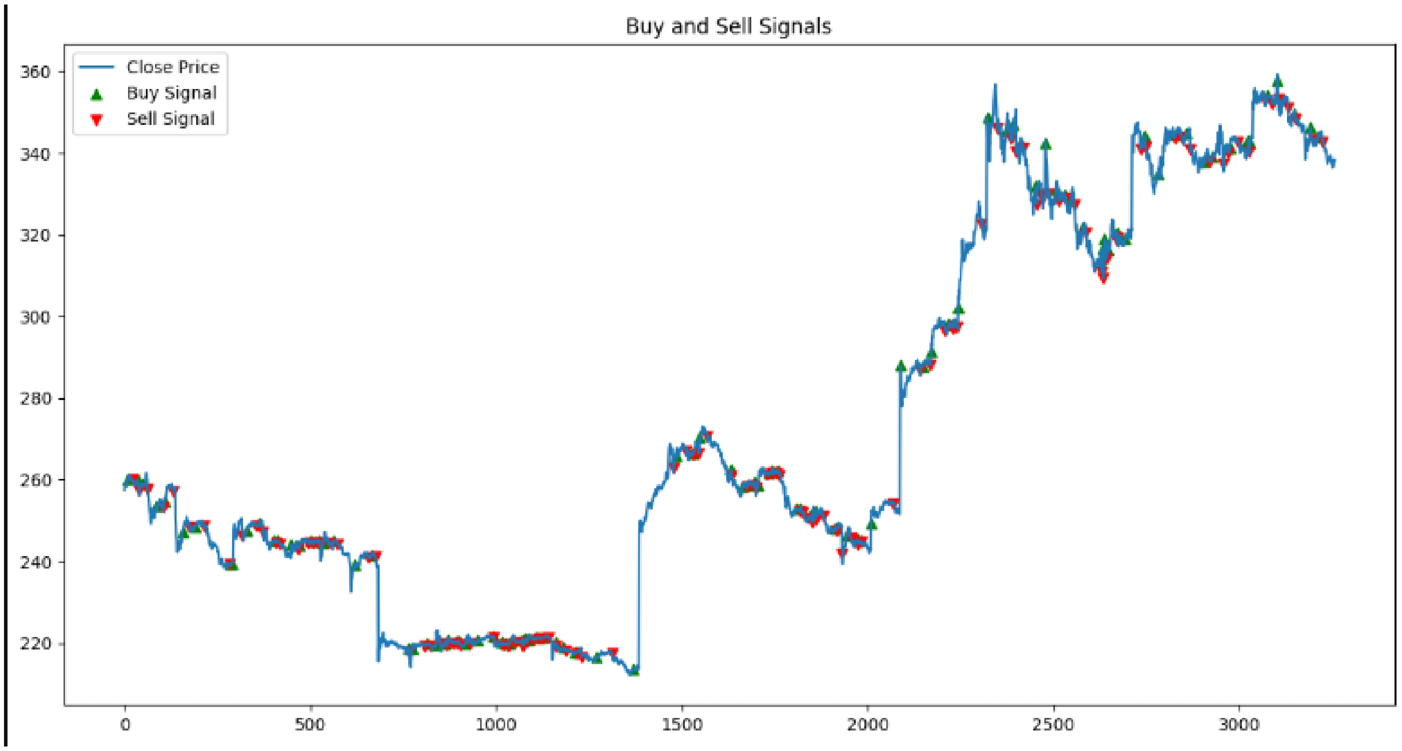
```python
data_5m['Buy_Signal'] = (
    (data_5m['EMAF'] > data_5m['EMAM']) &
    (data_5m['EMAF'].shift(1) <= data_5m['EMAM'].shift(1))
)

data_5m['Sell_Signal'] = (
    (data_5m['EMAF'] < data_5m['EMAM']) &
    (data_5m['EMAF'].shift(1) >= data_5m['EMAM'].shift(1))
)
```

<u>Visualising These Signals:</u>

For One Day Data of Tesla



For Five Minute Data of Tesla



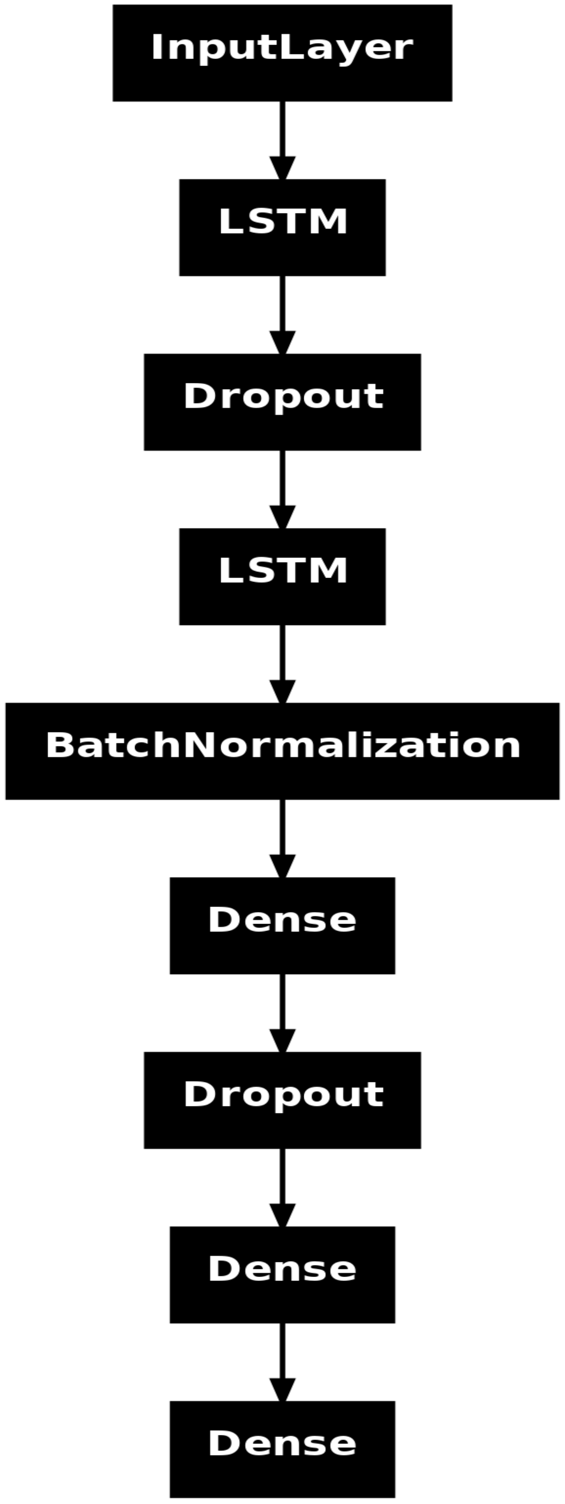
As you can see, we will now shift our focus to predicting these signals instead of directly predicting the price. This approach makes more sense, as the signals are likely to capture underlying patterns and market behaviors more effectively, providing a more robust framework for analysis and decision-making.

Initial Model:

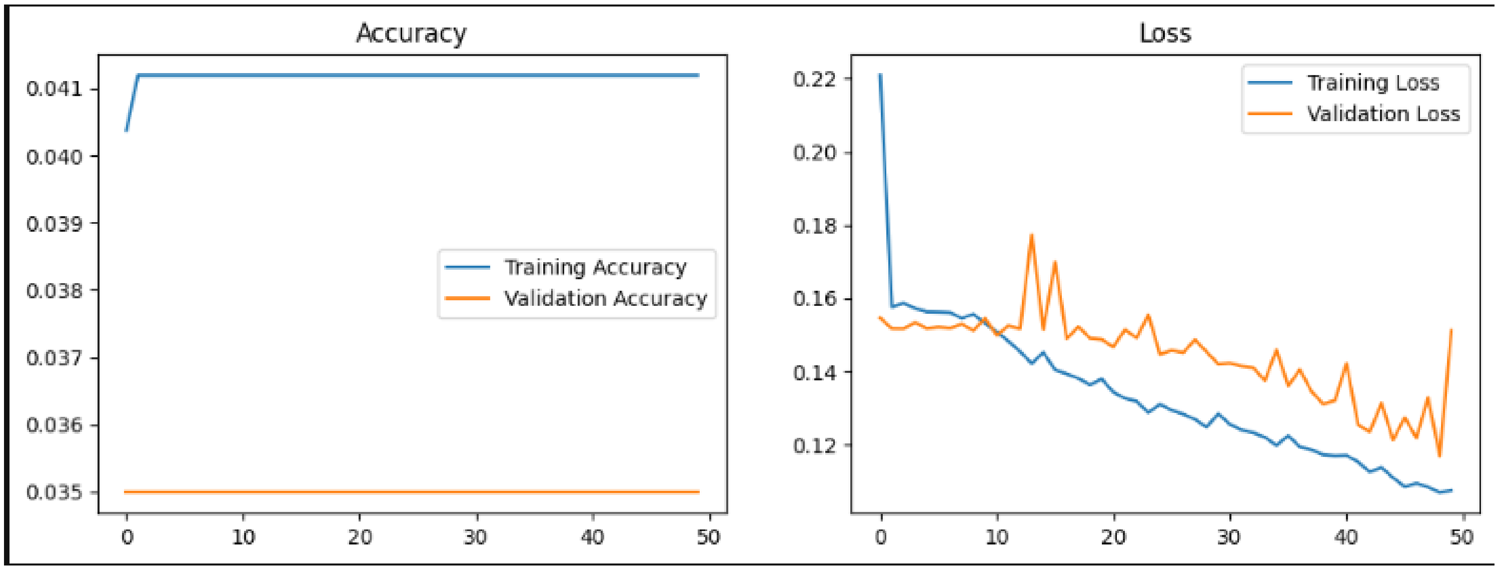
Combining both datasets to one big dataset, as there are few signals

Model With LSTM Layers and Fully Connected Dense layers:

Model Architecture:



Results:



As You can clearly see that the accuracy was not good at all and the training loss and validation loss keep getting lower

Key Observations:

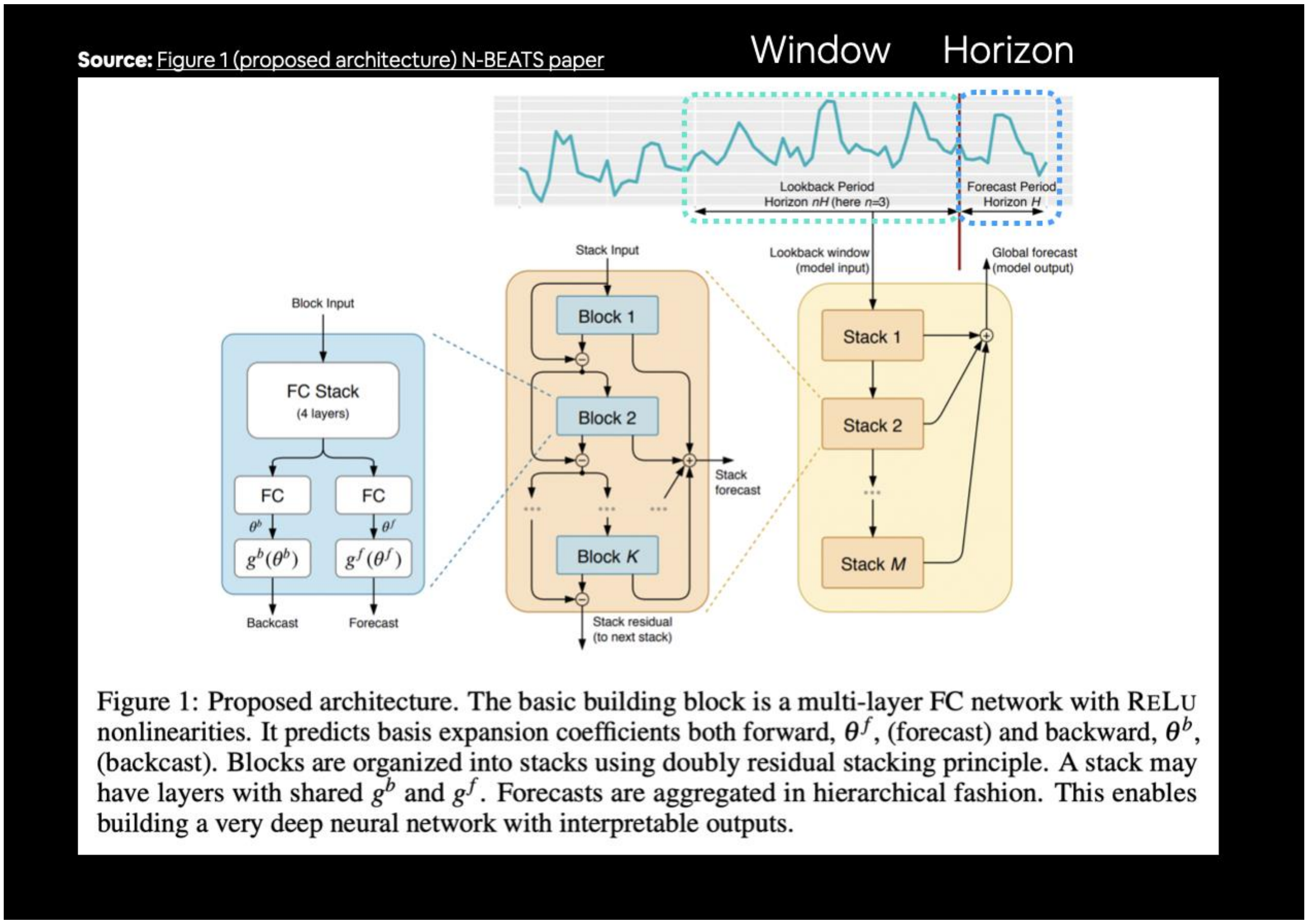
1. The flat and low accuracy for both training and validation sets suggests that the model is underfitting and may not have the capacity or the right configuration to capture patterns in the data.
2. The fluctuating validation loss indicates instability in generalization, possibly due to insufficient data, noisy input features, or the model being too simple or improperly tuned.
3. Overfitting signs are minimal, but the model appears ineffective in both learning and prediction.

This model performed very poorly, which can clearly be attributed to the unbalanced classes in our dataset. Specifically, the dataset contains significantly fewer buy and sell signals compared to no-signal instances, leading to a class imbalance issue. This imbalance skews the model's learning process, causing it to prioritize the majority class (no signal) while failing to effectively capture patterns related to the minority classes (buy and sell signals).

Experimentation and Challenges:

After further research, we decided to explore the use of pre-trained models, which were beyond the scope of our course but seemed promising enough to warrant a try. Specifically, we implemented an algorithm called **N-BEATS** (Neural Basis Expansion Analysis for Time Series), a cutting-edge approach known for its effectiveness in time series forecasting. This model leverages deep learning to automatically extract features from time series data, eliminating the need for extensive feature engineering and making it well-suited for capturing complex patterns in the dataset. Despite being advanced, we aimed to assess its potential in addressing the challenges posed by our unbalanced data and the highly volatile nature of the signals.

N-BEATS (Architecture)



Source: Figure 1 (proposed architecture) N-BEATS paper

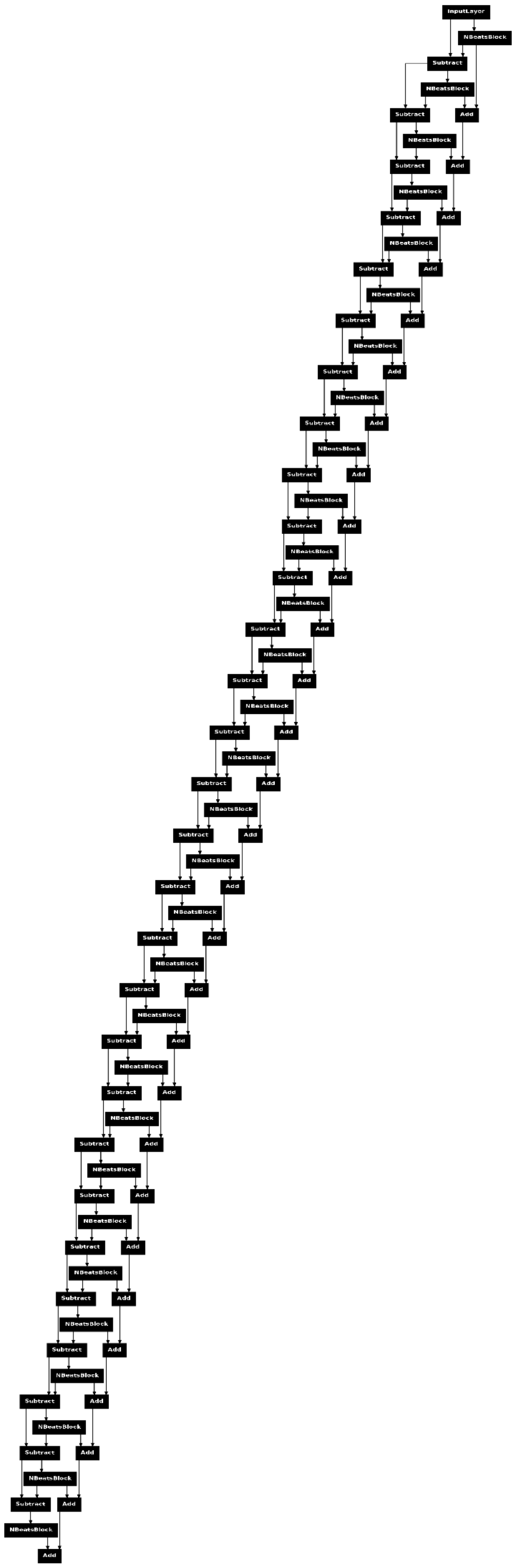
Figure 1: Proposed architecture. The basic building block is a multi-layer FC network with ReLU nonlinearities. It predicts basis expansion coefficients both forward, θ^f, (forecast) and backward, θ^b, (backcast). Blocks are organized into stacks using doubly residual stacking principle. A stack may have layers with shared g^b and g^f. Forecasts are aggregated in hierarchical fashion. This enables building a very deep neural network with interpretable outputs.

N-BEATS (Neural Basis Expansion Analysis for Time Series) is a state-of-the-art deep learning algorithm designed specifically for time series forecasting. Unlike traditional methods that require extensive feature engineering or rely on domain-specific knowledge, N-BEATS adopts a fully data-driven approach, learning patterns and relationships directly from the raw data. It is built on a stack-based architecture, where each stack comprises, multiple blocks containing fully connected neural networks. These blocks operate in a forward and backward fashion to model both trend and seasonality components in time series data. N-BEATS is highly versatile and can be applied to univariate or multivariate time series without requiring specialized preprocessing. Its flexible architecture and ability to capture complex, non-linear relationships make it a powerful tool for forecasting tasks across various domains, including finance, energy, and retail.

Modelling:

Here We have created and replicated the famous NBEATS algorithms. With the source code available on TensorFlow.

Below is the model architecture visualized using TensorFlow's ***plot_model*** function.


InputLayer
NBeatsBlock
Subtract
Add

Challenge:

The biggest challenge with this model is the training process, as it is highly complex and requires a significant amount of computational power, which we currently lack. Due to these limitations, we regretfully have to halt our project at this stage.

Conclusion:

In conclusion, this project served as a valuable learning experience in applying advanced machine learning techniques to the challenging domain of stock price prediction. While the results did not yield the expected predictive accuracy, the journey provided several key insights:

1. **Data Quality and Class Imbalance**:
   The project highlighted the critical importance of balanced datasets in machine learning. The severe imbalance between buy/sell signals and no-signal data emphasized the need for techniques such as oversampling, under sampling, or synthetic data generation to enhance model performance.
2. **Model Complexity vs. Computational Constraints**:
   We learned that while deeper and more complex models like N-BEATS have immense potential, their training requires substantial computational resources. This limitation underscored the importance of leveraging cloud-based solutions or simplified models tailored to available resources.
3. **Feature Engineering and Signals**:
   Transitioning from direct price prediction to signal prediction proved to be a more meaningful approach. Incorporating technical indicators like RSI, MACD, and VWAP enriched the dataset, enabling a deeper understanding of market dynamics and their potential predictive power.
4. **Role of Pre-trained Models**:
   Exploring state-of-the-art algorithms such as N-BEATS demonstrated the benefits of using pre-trained architectures. These models, although computationally demanding, provided a glimpse into the power of advanced methodologies for time series forecasting.
5. **Iterative Experimentation and Adaptability**:
   The iterative nature of the project reinforced the value of adaptability and continuous experimentation. Each stage of the project revealed new challenges and opportunities, driving us to refine our approach and improve our understanding.

While the immediate goals of the project were not fully realized, the insights gained have laid a strong foundation for future work. We plan to address the computational and data challenges, explore additional model architectures, and continue refining our methodology to make meaningful strides in the domain of stock price prediction.