**Siu King Sum**

📁 **Complete Source Code and Model Output** — is available on GitHub:
🔗 https://github.com/siiiiiiiiii/LSTM-with-Attention-for-Log-Return-Prediction/tree/main

# LSTM with Attention for Log Return Prediction

Price forecasting in financial markets has always been a challenging task, in order to effectively capture the underlying patterns of price fluctuations and improve prediction accuracy, this report presents a financial time series forecasting model, based on Long Short-Term Memory (LSTM) neural networks combined with an Attention mechanism. After undergoing experimental design and evaluation, the model demonstrates significant performance improvements over traditional naive baseline models and avoids overfitting during training. Through this project, I gained a deep understanding of the importance of overfitting control and establishing proper baselines, while also developing stronger risk awareness and analytical logic when facing real-world financial market volatility.

## Data Acquisition and Preprocessing Methods

This project begins by using `yfinance` to download historical price data for Google (GOOG) from 2015 to 2025, covering the open, high, low, close prices, and trading volume. To enhance data stability, the project uses the daily log return of the closing price as the prediction target, applying log transformation and differencing to the raw prices. A 7-day moving average (MA7) is introduced as an extra feature to help the model better capture price trends.

## Data Splitting and Feature Normalization

To avoid data leakage when building the financial time series forecasting model, the dataset is split in chronological order into a training set (the first 80%) and a test set (the remaining 20%). I then apply `MinMaxScaler` to normalize the data, ensuring that all features are on a similar scale, which improves model convergence stability. At this stage, special care is taken to fit the scaler using only the training set to prevent the test data from being influenced by the training data.

## Building the LSTM Model with an Attention Mechanism

## The Importance of Long Short-Term Memory (LSTM) Networks

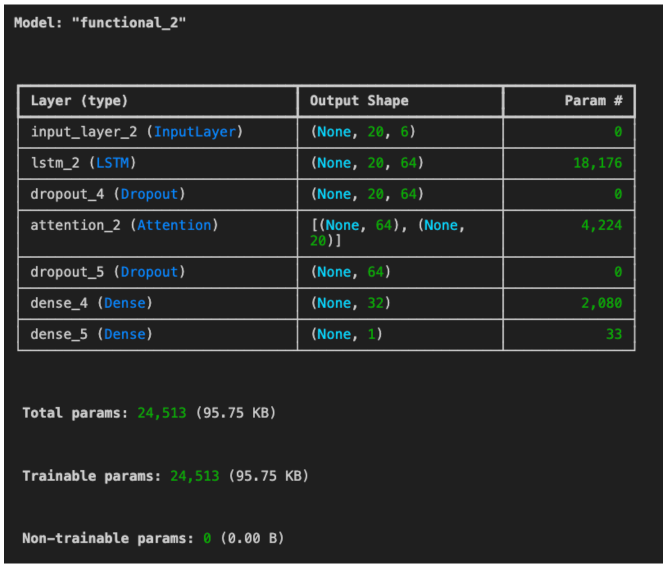
LSTM is a type of recurrent neural network (RNN) particularly well-suited for handling time series problems, as it can effectively capture both short-term and long-term dependencies in data. Unlike traditional RNNs, LSTM overcomes the vanishing gradient problem, making it useful for analyzing volatile and uncertain data.

## Role of the Attention Mechanism

However, using LSTM alone in long-term time series forecasting may sometimes overlook the varying importance of different pieces of information within the sequence. To address this limitation, I incorporate an Attention mechanism on top of the LSTM model (Figure 1). The Attention layer enables the model to automatically learn and assign different weights to features at each time step, allowing it to focus more effectively on the most relevant information for prediction. This enhances both the interpretability and predictive performance of the overall model.

**Figure 1**

*Attention mechanism on top of the LSTM model*



```
Model: "functional_2"
```

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer_2 (InputLayer) | (None, 20, 6) | 0 |
| lstm_2 (LSTM) | (None, 20, 64) | 18,176 |
| dropout_4 (Dropout) | (None, 20, 64) | 0 |
| attention_2 (Attention) | [(None, 64), (None, 20)] | 4,224 |
| dropout_5 (Dropout) | (None, 64) | 0 |
| dense_4 (Dense) | (None, 32) | 2,080 |
| dense_5 (Dense) | (None, 1) | 33 |

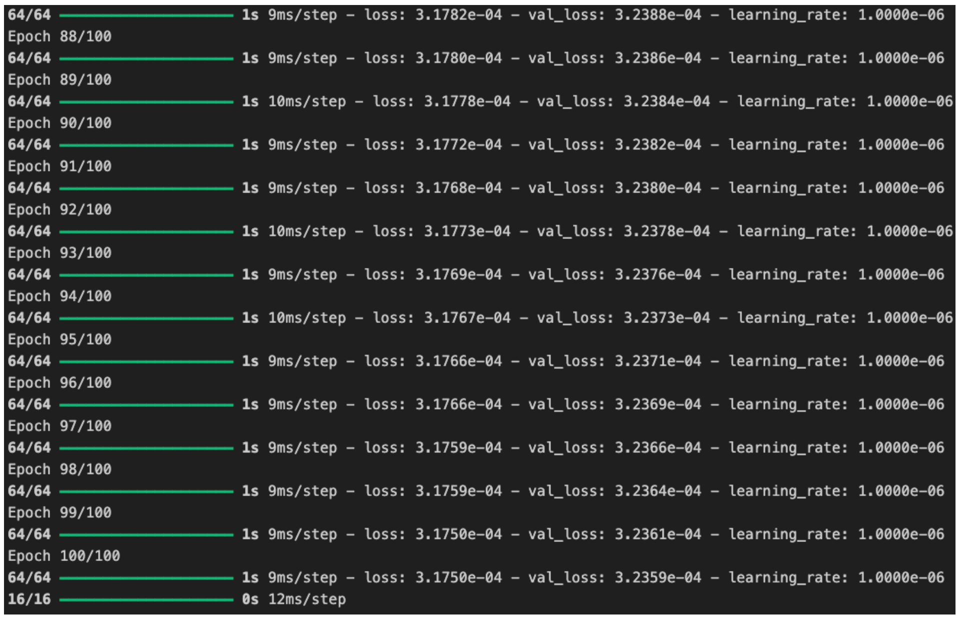
```
Total params: 24,513 (95.75 KB)

Trainable params: 24,513 (95.75 KB)

Non-trainable params: 0 (0.00 B)
```

## Model Training Process and Overfitting Prevention Strategies

The model was trained over 100 epochs (Figure 2). To prevent overfitting, I adopted the following three strategies: First, I added Dropout layers and L2 regularization to the model architecture to control model complexity. Second, I applied an Early Stopping mechanism, which halts training if the validation loss does not improve for five consecutive epochs. Third, I used the `ReduceLROnPlateau` callback to dynamically reduce the learning rate if the validation loss stagnates for a prolonged period.

**Figure 2**

*Model Trained for 100 Epochs*

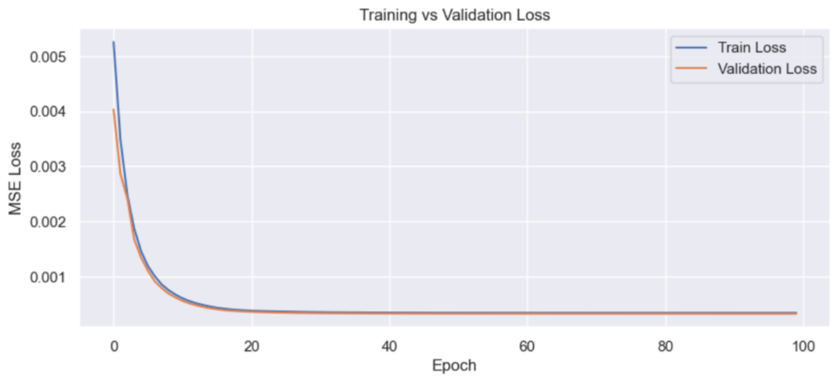


```
64/64 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 3.1782e-04 - val_loss: 3.2388e-04 - learning_rate: 1.0000e-06
Epoch 88/100
64/64 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 3.1780e-04 - val_loss: 3.2386e-04 - learning_rate: 1.0000e-06
Epoch 89/100
64/64 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 3.1778e-04 - val_loss: 3.2384e-04 - learning_rate: 1.0000e-06
Epoch 90/100
64/64 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 3.1772e-04 - val_loss: 3.2382e-04 - learning_rate: 1.0000e-06
Epoch 91/100
64/64 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 3.1768e-04 - val_loss: 3.2380e-04 - learning_rate: 1.0000e-06
Epoch 92/100
64/64 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 3.1773e-04 - val_loss: 3.2378e-04 - learning_rate: 1.0000e-06
Epoch 93/100
64/64 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 3.1769e-04 - val_loss: 3.2376e-04 - learning_rate: 1.0000e-06
Epoch 94/100
64/64 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 3.1767e-04 - val_loss: 3.2373e-04 - learning_rate: 1.0000e-06
Epoch 95/100
64/64 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 3.1766e-04 - val_loss: 3.2371e-04 - learning_rate: 1.0000e-06
Epoch 96/100
64/64 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 3.1766e-04 - val_loss: 3.2369e-04 - learning_rate: 1.0000e-06
Epoch 97/100
64/64 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 3.1759e-04 - val_loss: 3.2366e-04 - learning_rate: 1.0000e-06
Epoch 98/100
64/64 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 3.1759e-04 - val_loss: 3.2364e-04 - learning_rate: 1.0000e-06
Epoch 99/100
64/64 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 3.1750e-04 - val_loss: 3.2361e-04 - learning_rate: 1.0000e-06
Epoch 100/100
64/64 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 3.1750e-04 - val_loss: 3.2359e-04 - learning_rate: 1.0000e-06
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step
```

From the training results, I observed that the training loss and validation loss almost completely overlapped throughout the process, indicating that overfitting was well controlled and the model maintained strong generalization ability(Figure 3).

**Figure 3**

*Training vs Validation Loss*



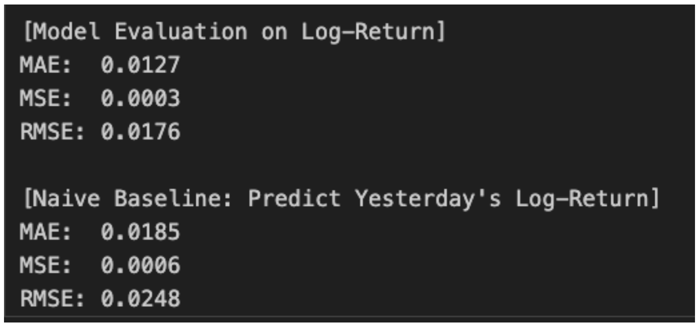
## Model Evaluation and Performance Analysis

### Comparison with Naive Baseline Predictions

After the complete training process, I conducted a evaluation of the model's performance on the test dataset, using Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) as key metrics. On the test set, the model achieved an MAE of **0.0127** and an RMSE of **0.0176**, showing a significant improvement over the simple but informative Naive Baseline (which predicts today's return as equal to yesterday's, RMSE = **0.0248**). These results indicate that the model effectively captures underlying trends in financial market prices and provides more accurate predictions than the baseline approach.

**Figure 4**

*Model vs. Naive Baseline on Log-Return Prediction*

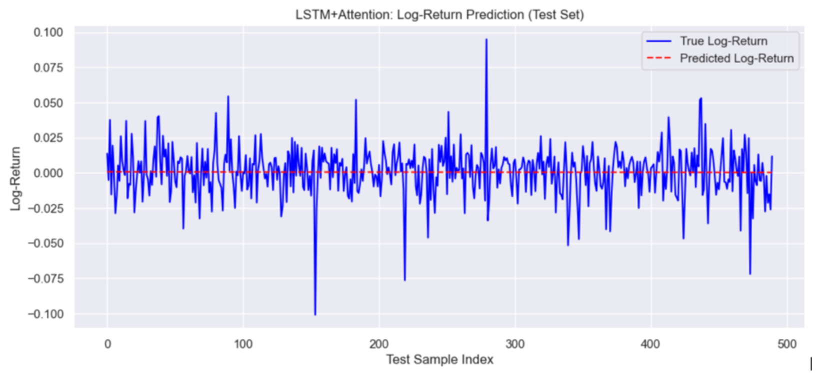

```
[Model Evaluation on Log-Return]
MAE:  0.0127
MSE:  0.0003
RMSE: 0.0176

[Naive Baseline: Predict Yesterday's Log-Return]
MAE:  0.0185
MSE:  0.0006
RMSE: 0.0248
```

### Visualization and Analysis of Results

To evaluate the model's actual predictive ability on the test set, I visually compared the predicted log returns with the true log returns (Figure 5). As shown in the chart, although the predicted curve (red dashed line) does not perfectly capture every detail of market fluctuations, it generally follows the overall trend direction of the actual market and maintains a stable range of variation with relatively small prediction errors. This is consistent with the quantitative metrics, where the model achieved MAE = 0.0127 and RMSE = 0.0176 —significantly better than the Naive Baseline's RMSE = 0.0248—indicating higher accuracy and stability from a statistical perspective.

**Figure 5**

*LSTM + Attention: Log-Return Prediction (Test Set)*



However, a closer inspection of the graph and prediction distribution reveals notable limitations when the model encounters extreme market conditions. The predicted values are mostly concentrated around zero, making the model ineffective at capturing large upward or downward movements. This can hinder the identification of key market turning points in real trading decisions. Additionally, although the model outputs continuous log-return values, it lacks clear directional discrimination—meaning it cannot reliably indicate whether prices are rising or falling. For instance, in test samples where the actual log return exceeds ±0.05, the model's predicted values remain within ±0.01, suggesting a conservative prediction tendency and insufficient sensitivity to turning points.

## Conclusion and Personal Reflections

This project built a reliable financial forecasting model using LSTM with Attention. It trained smoothly in under 100 epochs with no overfitting, as shown by closely matched training and validation losses. On the test set, it achieved an MAE of 0.0127 and RMSE of 0.0176—clearly better than the Naive Baseline (RMSE = 0.0248). This shows the model's strong accuracy for real-world forecasting.

Through this project, I not only gained hands-on experience, but also developed a deeper understanding of two critical concepts in financial modeling.

First, preventing overfitting is a core issue. In this project, by integrating techniques such as Dropout, L2 regularization, EarlyStopping, and learning rate adjustment with `ReduceLROnPlateau`, I ensured the model remained highly stable and convergent throughout training, successfully mitigating the risk of overfitting to the training data.

Second, establishing an appropriate baseline for comparison is essential for scientifically evaluating a model's effectiveness. Without a proper benchmark, it is impossible to objectively determine whether a model truly possesses predictive capability. By using a Naive model as the baseline in this project, I deeply realized that evaluation is not just about whether the model performs well numerically—it's about comparing it against reasonable alternatives.