COSE474-2023F: Final Project Proposal
Building Trading Strategies of Cryptocurrency and Stock Using LSTM and Transformers

2019170611 Heeseo Park

1. Introduction

1.1. Motivation

The financial market, with its inherent fluctuation and complexity, especially in the domains of stocks and cryptocurrencies, has long presented a formidable challenge to analysts and investors. Traditional analysis methods often struggle to capture the dynamic and unpredictable nature of market prices. This gap highlights the critical need for more advanced, data-driven approaches. The rise of machine learning and deep learning techniques offers a promising avenue for addressing these challenges. This study is motivated by the potential of these technologies to bring a new level of sophistication and accuracy to financial market analysis and trading strategy development.

1.2. Problem Definition

The central problem this research addresses is the difficulty in predicting price movements in financial markets. Market dynamics are influenced by many factors, which are also very complex, making price prediction notoriously challenging. The unpredictability of price movements in stocks and cryptocurrencies not only hampers effective investment decisions but also poses a significant risk to market stability. Therefore, developing models that can accurately capture and predict these price fluctuations can be very useful.

1.3. Concise Description of Contribution

This paper contributes to the field by exploring the capabilities of Long Short-Term Memory (LSTM) models and Transformers in financial forecasting. The study investigates how LSTM, known for its effectiveness in handling time-series data, and Transformers, renowned for their efficient sequential data processing through attention mechanisms, can be utilized to develop trading strategies. The research looks at various model structures, examining the impact of different layers, including dropout and self-attention encoder layers, on the accuracy of price predictions. Additionally, it extends into binary classification and hyperparameter optimization to refine the predictive power of these models. The findings of this research provide valuable insights, paving the way for more informed and reliable decision-making in the volatile realm of financial markets.

2. Methods

2.1. Significance of Using LSTM and Transformers

- **LSTM's Role:** The Long Short-Term Memory (LSTM) models are pivotal in this study due to their unique ability to focus on long-term memories. In financial markets, price fluctuations can be influenced by data points that are significantly older, and LSTM's architecture can be effective at capturing these long-term dependencies. This characteristic is particularly crucial in understanding and predicting trends in stock and cryptocurrency markets where historical patterns play a vital role.
- **Transformers' Role:** Self-attention mechanism, which is one of the most important aspect of the Transformer, allows the model to weigh the importance of specific points in a sequence, identifying which factors most significantly influence price rises or falls. This ability to focus on relevant segments of data is essential in the context of financial markets, where certain events or indicators can have a disproportionate impact on market movements.

2.2. Novelty of the Main figure

The novelty of this research lies in the integration of LSTM layers with Transformer models tailored for financial forecasting. Traditionally, Transformers in language models first embed sentences, encode sequences into higher dimensions, and then apply self-attention mechanisms, followed by a decoder to return to the target dimension. In contrast, this study introduces a novel approach where the LSTM layer precedes the self-attention Transformer layer. This modification allows the Transformer to consider long-term memories more effectively, a critical aspect in financial time series analysis. Additionally, to address the computational intensity of regression models in predicting precise price values, the study employs a binary output approach. This approach focuses on predicting the direction of price move-

ment (rise or fall) rather than exact values. However, for training, the model uses more granular data, specifically the percentage change in prices, to ensure comprehensive learning from market dynamics.

2.3. Main Figure

The main challenge in predicting stock and cryptocurrency price movements lies in the multitude of factors influencing these markets, many of which are interrelated and complex. It's a difficult task for traditional analysis methods to consider all these variables comprehensively. This research addresses this challenge by leveraging the combined strengths of LSTM and Transformers. The LSTM layers capture long-term dependencies and trends, while the Transformers' self-attention mechanism efficiently shifts through the data to focus on the most influential factors. This integrated approach provides a more wider view of the market dynamics, enabling more accurate predictions of price movements in the notoriously unpredictable financial markets.

2.4. Reproducibility-pseudocode and Formulation

The training process involves the following steps. For each epoch:

1. Iterate over the training dataset:
   (a) Forward pass: Compute the model's predictions.
   (b) Compute the loss.
   (c) Backward pass: Compute the gradients and update model parameters.
2. Evaluate the model on the test dataset.
3. Save the model if it has the best performance.

3. Experiments

This section details the experiments conducted, including information about the dataset used, computing resources, and the experimental design and setup.

3.1. Dataset

The dataset in this study are a minute-by-minute open and close prices of Bitcoin from January 2021 to March 2023, sourced from Binance. The dataset's preparation involved the following steps:

- **Dataset Source:** Open and close prices for Bitcoin with a 1-minute timespan, obtained from Binance.
- **Preprocessing:** Each day's data was provided in individual CSV files. We utilized NumPy's `vstack` method to concatenate the open and close prices for each day into a single dataset.
- **Statistics:** The dataset focuses on a single feature, the price, recorded every minute. With 1,440 minutes in a day and spanning across 820 days, the total dataset comprises approximately 1,180,800 data points.

3.2. Computing Resource

The experiments were conducted using the following computing resources, ensuring adequate computational power and efficiency for the deep learning tasks:

- **CPU:** 12th Gen Intel(R) Core(TM) i7-12700F, 2.10 GHz
- **GPU:** NVIDIA GeForce RTX 3070 Ti, providing accelerated computation for model training and testing.
- **Operating System:** Windows 11, 64-bit version, ensuring compatibility with the latest software and hardware drivers.
- **Deep Learning Framework:** PyTorch version 2.0.1+cu117, chosen for its flexibility and ease of use in building complex models.
- **Other Libraries:** Essential Python libraries including math, pandas (for data handling), datetime (for managing date and time data), pyplot (from matplotlib for visualization), random (for random number generation), and numpy (for numerical computations).

3.3. Experimental Design/Setup

This subsection describes the experimental design and setup, including the model architecture, training process, evaluation metrics, and specific configurations used for the experiments.

Model Architecture The model architecture integrates the LSTM and Transformer models to leverage their respective strengths in processing time series data. The architecture is composed of the following components:

- **LSTM Layer:** The LSTM (Long Short-Term Memory) layers capture long-term dependencies and sequential information in the time series data. The model includes LSTM layers with a hidden size of `lstm_hidden_size` and `num_lstm_layers`.
- **Linear Transformation:** This layer transforms the LSTM output to match the Transformer's `d_model` size, preparing the data for the Transformer Encoder.
- **Positional Encoding:** Incorporated to add information about the sequence order, crucial for the model to understand the temporal dynamics of the time series data.

- **Transformer Encoder:** Comprising `num_encoder_layers` and `n_heads` heads, the Transformer Encoder efficiently processes the sequence in parallel, using a self-attention mechanism to identify relationships between distant elements in the sequence.
- **Attention Weights:** Applied to calculate a weighted average of the encoder output, focusing on specific parts of the input sequence.
- **Output Layer:** A linear layer that generates the final prediction.

Loss Function and Optimizer The model's training process utilizes the following loss function and optimizer:

- **Loss Function:** For regression tasks, the model uses Mean Squared Error Loss (MSELoss), which is defined as:

$$MSELoss(x, y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 \qquad (1)$$

  where x represents the predicted values by the model, y is the ground truth, and N is the number of samples.
- **Optimizer:** The model employs the Adam optimizer with a learning rate defined by `learning_rate` for parameter optimization. The Adam optimizer is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments.

Training Process and Evaluation

- **Training Process**
  - Total Epochs: 100
  - Batch Size: 16
  - Learning Rate: 0.01
  - Loss Function: Mean Squared Error Loss (nn.MSELoss)
  - Optimization Algorithm: Adam Optimizer
  - Training Device: GPU (if available) or CPU
- **Evaluation Metrics**
  - Primary Metric: Mean Squared Error (MSE) between predicted and actual values

Configurations for Reproducibility

- **Random Seed**
  - Seed Value: 42
  - Applied to random, NumPy, and PyTorch
- **Model Saving and Loading**
  - Checkpoints: Saved when test loss improves
  - Filename: Contains model architecture and hyperparameters

3.4. Figures and Analysis

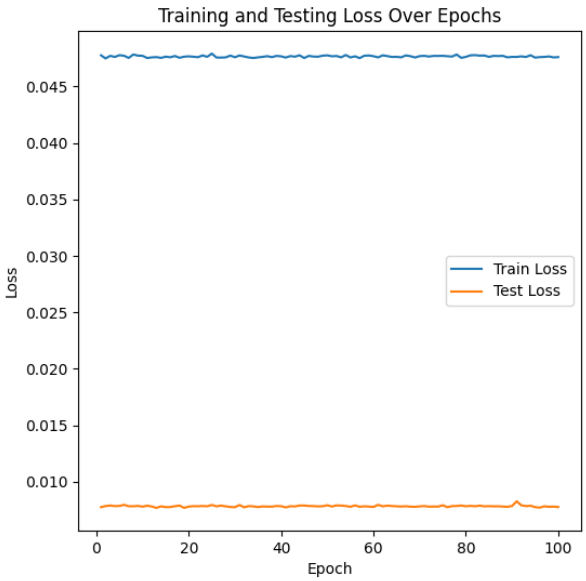

Figure 1. Training and testing loss over epochs.

As can be seen in the Figure, it cannot be concluded that the model has learned effectively. This is indicated by the lack of significant decrease in both training and test loss. The minimum value of the train loss is 0.04759564815385506 and the minimum value of the test loss is 0.007690836919380291. One interpretation could be that the complexity of the data was not sufficiently captured and learned by the model's architecture.

4. Future Direction

To enable the model to learn more effectively, one possible approach is to engineer features based on financial theories. A planned direction is to apply theories that predict future trends based on past performance, such as the Elliott Wave Theory, to train the model. This involves processing data to reflect economic indicators that could potentially influence the prediction of future price movements.

References