Bitcoin Price Prediction with Long Short-Term Memory (LSTM) Networks

Dhanish Parimalakumar
dhanishpari@gmail.com

***Abstract*—The goal of this project was to build a fully manual Long Short-Term Memory (LSTM) network utilizing only NumPy and apply it to the task of Bitcoin price prediction. This model was created from scratch, including gate equations, hidden- and cell-state updates, and backpropagation through time.**

The Dataset consisted of minute-level bitcoin OHLCV data from Kaggle, which were resampled into daily data points for 2024-2025. By using 30-day rolling windows of OHLCV inputs, the LSTM was trained to predict the next day's closing price by using mean squared error (MSE) loss, and stochastic gradient descent.

Multiple training runs of (50, 100, and 200 epochs) were conducted to explore model behavior and properties of convergence. The best performing model achieved a test RMSE of approximately 0.039 in normalized space. To examine the practical performance of the mode, a simple trading simulation was also implemented, where the model's daily predictions determined whether a hypothetical portfolio held BTC or remained in cash. The LSTM driven strategy outperformed a buy-and-hold baseline over the evaluation period, indicating that the model captured short-term directional patterns in the data.

Overall, this project demonstrates that a hand-crafted LSTM is capable of modeling financial sequences and extracting useful temporal structure without modern ML Libraries. This highlights the strengths of recurrent models for sequential predictions and also the limitations that arise when applying neural networks to noisy non-stationary financial markets.

Keywords—Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM) Networks.

I. Introduction

Forecasting financial time-series data is a difficult yet lucrative challenge due to its noisy, highly variable nature. Cryptocurrencies like Bitcoin are especially volatile, being influenced by a wide variety of external factors such as market sentiment, macroeconomic events, and liquidity shocks. Traditional machine learning models may struggle to capture these temporal dependencies since they rely on fixed-length feature representations.

Recurrent Neural Networks (RNN), and specifically Long Short-Term Memory (LSTM) models offer a principled way to model these sequential patterns by maintaining and updating a learnable hidden state over the time period chosen. Unlike networks that are classified as 'feed-forward', LSTMs utilize gating mechanisms that allow the model to retain or forget information across longer time horizons. This ensures that they are well suited for challenges such as day-to-day price forecasting, where recent historical structure is often predictive of short-term market direction.

In this project, I manually implemented an LSTM framework from scratch utilizing library tools from NumPy, not using pre-built frameworks from TensorFlow or PyTorch. The goal in mind was to evaluate whether a hand built recurrent model would be able to learn from Bitcoin's historical temporal OHLCV (Open, High, Low, Close, Volume) data and produce competitive next-day pricing predictions. The model was trained on daily bitcoin data from 2024-2025 and evaluated on out-of-sample sequences, with some additional analyses in the form of a simplified trading simulation to evaluate the models real-world effectiveness.

II. Theoretical and Conceptual Study

A. Background and Motivation

Recurrent Neural Networks (RNNs) were originally designed to model sequential data by maintaining a hidden state that evolves over time. Unlike feed-forward networks, which treat each input independently, RNNs incorporate information from previous timesteps, allowing them to represent temporal structures. This property makes RNNs well suited for tasks such as language modeling, speech recognition, and financial time-series predictions, where recent history is an important influence on future behavior.

However, classical RNNs suffer from the vanishing and exploding gradient problem, especially when they are modeling long sequences. During backpropagation through time (BPTT), gradients shrink exponentially, making it difficult for the network to learn dependencies beyond a short horizon. In financial markets, important patterns often unfold over multiple days or weeks; therefore, traditional RNNs are not ideal for capturing such long-term dynamics.

To address these limitations, Hochreiter and Schmidhuber introduced the Long Short-Term Memory (LSTM) architecture in 1997. LSTMs augment the recurrent structure with gating mechanisms designed to preserve important information over extended periods. The LSTM architecture remains foundational for modern sequence modeling and is widely used in applications that need memory and long-range time context.

Specifically for financial forecasting, LSTMs are especially attractive because markets exhibit temporal autocorrelation, momentum effects, and structural patterns that are distributed across time. A well-trained LSTM can theoretically capture relationships that simpler models may overlook.

B. LSTM Architecture and Mechanics

An LSTM cell introduces several gates that regulate how information flows through the network. These gates allow the network to selectively keep or discard information which helps overcome the vanishing gradient problem.

An LSTM cell contains four key components:

1. **Forget Gate**
   a. Decides which information should be removed from the previous cell state.
   b. Takes the previous hidden state and the current input and outputs a value between 0 and 1 for each cell dimension.
   c. A value near 1 means "keep this information" and a value near zero means to "forget.".
2. **Input Gate:**
   a. Determines how much new information can enter a cell state.
   b. Works with a "candidate" update, to control how much new content will be written in.
3. **Candidate State (Proposed Update)**
   a. Generates a vector of potential new information based on the current input and previous hidden state.
   b. Values range from -1 to 1 due to the tanh activation function.
   c. This represents the "new memory" that may be added.
4. **Cell State Update:**
   a. The cell state is updated by combining:
      i. information the forget gate chooses to keep
      ii. new information determined by the input gate the candidate state
   b. This is the core mechanism that allows LSTMs to maintain long term memory while staying flexible for new data.
5. **Output Gate:**
   a. Determines what part of the cell state should be exposed as the hidden state for the current lineup.
   b. Produces the hidden state, which is passed to the next timestep and used for the model's output.

Summary of LSTM Behavior:

Across each timestep, the LSTM:

1. Decides what past information to keep.
2. Decides what new information to add.
3. Updates its internal memory.
4. Produces an output based on the updated memory.

Together, these mechanisms make LSTMs extremely effective at modeling temporal relationships, where the importance of past events depends on the context, making them highly applicable to financial pricing data.

C. Manual Implementation Using NumPy

In this project, the entire LSTM architecture was implemented from scratch using NumPy, without frameworks such as TensorFlow or PyTorch. This required explicitly coding:

- all gate computations
- hidden and cell state updates
- forward unrolling across 30 timesteps
- backpropagation through time (BPTT)
- gradient computation for every parameter matrix
- and stochastic gradient descent (SGD) updates

Since LSTMs rely on sequential computations, gradients have to be backpropagated over every timestep. This exposes numerical instability and makes debugging more challenging than in feed-forward networks. Implementing BPTT manually allowed for a deeper understanding of how recurrent models learn and how each gate contributes to controlling temporal information.

D. Applicability to Financial Time-Series

Financial markets exhibit several characteristics that align well with LSTMs:

- **Short-term temporal dependencies**: Bitcoin often displayed patterns related to recent pricing movements
- **Non-linear dynamics**: Market behavior is rarely linear, and LSTMs can model non-linear transformations using tanh and sigmoid activations
- **Noise tolerance through memory**: The forget gate will filter out noisy fluctuations while retaining meaningful longer-term structure

- **Multivariate Inputs (OHLCV)**: Price and volume provide complementary signals, and LSTMs are able to integrate them naturally.

Because the model receives sequences of 30 days of OHLCV data, it can learn patterns such as trend continuation, volatility compression, or unusual volume spikes that may precede movement in the next day's closing price.

III. Results and Analysis

This section evaluates the performance of the manually implemented LSTM model across multiple training configurations and assesses its effectiveness in forecasting Bitcoin's next-day closing price. Three training durations were examined: 50, 100, and 200 epochs to better analyze the convergence behavior, prediction stability, and overall generalization of the out of sample data. Additional analyses were conducted using visualization of predicted vs actual prices and through a simplified trading simulation.

A. Training Behavior and Convergence

The training loss curves for the 50, 100, and 200 epoch models all show a consistent downward trend, confirming that the manual LSTM implementation successfully learned temporal patterns in the OHLCV data. The loss decreased most rapidly during the first 15-20 epochs and then the reductions became progressively smaller. The pattern is expected for models trained with stochastic gradient descent particularly when using small batch sizes and manually computed gradients.

Minor oscillations in the loss curve were observed in all models likely due to the absence of batch averaging, the noise present in Bitcoin price data, and the numerical instability inherent in a manually coded BPTT. Despite these challenges, all of the models reached a stable convergence region, demonstrating that a hand built LSTM architecture is indeed functional and optimizable without advanced frameworks.

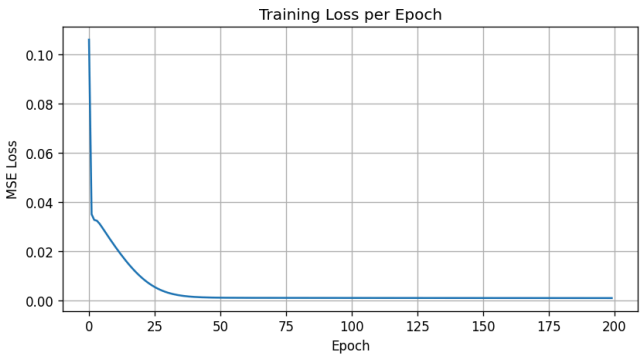

Fig. 1. Training loss curve for the manual LSTM across all epochs

B. Quantitative Performance Metrics

A summary of the test-set performance for each training duration is shown below:

Epochs	Test MSE	Test RMSE	Notes
50	0.002234	0.04727	Underfits slightly, higher residual variance
100	0.001776	0.04214	significant improvement, smooth convergence
200	**0.001535**	**0.03918**	**best performance, diminishing returns afterwards**

This progression demonstrates steady improvement as training time increases, with the largest gains occurring between 50 and 100 epochs. The 200-epoch model achieves the strongest performance, though the decrease in error beyond 100 epochs is relatively modest.

C. Prediction Results

Predicted values were compared against ground-truth prices for the entire test period. The full-range plot shows that the LSTM captures the overall structure of the price trajectory, including longer upward and downward movements, while smoothing out some of the sharpest fluctuations. The smoothing effect is typical in recurrent models trained on noisy data, as the networks learned general directional patterns but reacts less strongly to isolated volatility spikes.

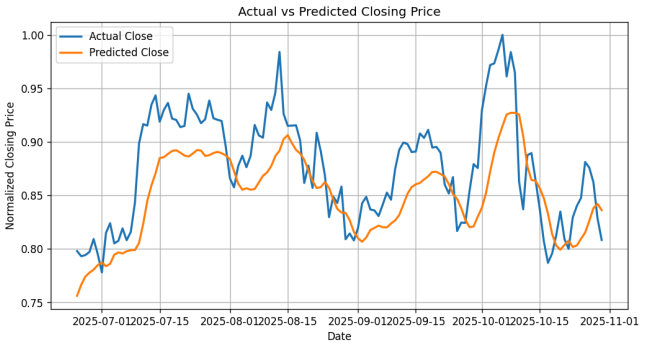

Fig. 2. Predicted vs. actual Bitcoin daily close for the full test period

A zoomed in view of the final 30 days provides a clearer demonstration of short-term predictive

behavior, The model tracks local direction changes reasonably well, correctly anticipating most small rises and dips, but tends to underestimate sharp upward jumps and overestimate abrupt falls. These patterns reflect the limitations of purely historical inputs. The LSTM cannot anticipate external shocks or sudden liquidity events that drive rapid price movements.

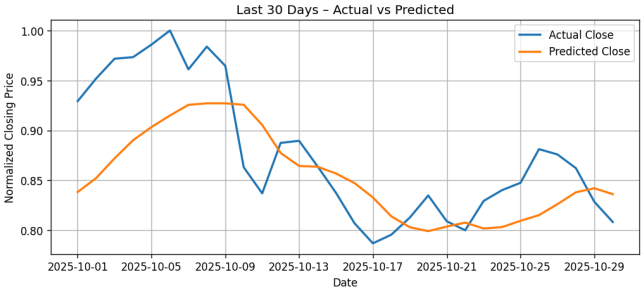

Fig. 3. Zoomed predictions for the last 30 days of the test set

D. Trading Simulation

To evaluate whether the model's predictions provided actionable value beyond numerical accuracy, a simple trading simulation was conducted. The strategy operated under the following rules: if the predicted next-day closing price was higher than the current close, the model held bitcoin, otherwise it moved completely to cash. No leverage, slippage, or transaction costs were included, and the initial capital was set to $10,000.

The resulting equity curve is shown in figure 4. The buy-and-hold portfolio (orange line) exhibits the typical volatility of Bitcoin, with several sharp rises and equally sharp drawdowns. In contrast the LSTM-based strategy (blue line) shows a noticeably smoother trajectory. Instead of participating in every major price swing, the model stays flat during predicted downturns, effectively sidestepping several significant drawdowns seen in the buy-and-hold curve, especially in the steep decline in mid October.

Although the LSTM strategy occasionally lags behind buy-and-hold during rapid rallies, it consistently avoids large losses and maintains a steadier upwards progression. By the end of the evaluation period, the model driven strategy reached a final value of $12,957.70, compared to $10,082.67 for buy-and-hold. This indicates that even a simple directional trading rule based on the LSTM's predictions captured enough short term structure to outperform a simple naive benchmark.

It is important to emphasize that this result does not imply real-world profitability. Transaction costs, spreads, partial fills, and market impact would reduce performance, and the strategy assumes perfect next day execution at closing prices. Despite these challenges, the simulation offers evidence that the model learned meaningful directional signals in the data and was able to translate that into implemented trading outcomes.

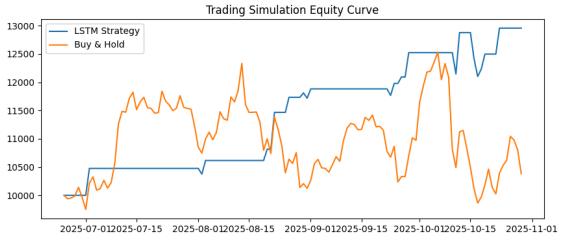

Fig. 4. Trading simulation equity curve comparing LSTM strategy vs. buy-and-hold

E. Interpretation and Limitations

Across all configurations, the manually implemented LSTM demonstrated effective learning and achieved low prediction error relative to the normal price scale. The model was able to capture short-term direction structure and perform positively in a basic trading simulation, however there are a few key limitations that need to be acknowledged.

- Bitcoin prices exhibit non-stationary dynamics that may degrade model performance over time.
- OHLCV sequences alone cannot account for external shocks, macroeconomic factors, or sentiment driven movements.
- Manual gradient computation introduces numerical imprecision that can affect training stability.
- The trading simulation omits realistic friction such as fees or slippage which would reduce actual returns.

Despite these limitations, the results seem to indicate that even a simple made-from-scratch LSTM model can learn meaningful temporal patterns in cryptocurrency markets and can make reasonably accurate next-day predictions.

IV. Conclusion and Future Work

A. Conclusion

The goal of this project was to manually implement a Long Short-Term Memory (LSTM) network using only NumPy and apply it to the task of predicting Bitcoin's next-day closing price. By resampling minute-level data into daily OHLCV features and constructing 30-day input sequences, the model successfully learned meaningful temporal dependencies within the dataset. Training results demonstrated clear convergence across 50-, 100-, and 200-epoch configurations, with performance improving steadily as training duration increased. The best model achieved a normalized test RMSE of

approximately 0.039, indicating that the handcrafted LSTM was able to closely track short-term price movements.

Prediction visualizations showed that the model captured the general structure and daily direction of Bitcoin's price, though it tended to smooth out sharp volatility spikes, an expected outcome given the level of noise in financial series. A simplified trading simulation further demonstrated that the model's directional forecasts carried practical signals, as the strategy based on LSTM predictions outperformed a buy-and-hold benchmark over the evaluation window. While this result is not indicative of real-world tradability, it shows that the model successfully extracted actionable short-term structure from the data.

Overall, the project illustrates that a manually implemented LSTM, without any deep learning libraries can effectively model sequential financial data. The work highlights both the strengths of recurrent architectures for time-series forecasting and the challenges of applying such models to noisy, non-stationary markets such as Bitcoin.

B. Future Work

Several potential improvements and extensions could enhance the performance and realism of this project:

1. **Expanded Feature Set:**
   Incorporating technical indicators (RSI, MACD, moving averages), volatility measures, or order-book features may provide additional predictive signals beyond raw OHLCV.
2. **Longer or Adaptive Lookback Windows:**
   Different market regimes may require longer historical context; experimenting with 60-, 90-, or variable-length sequences could strengthen the model's ability to handle long-term dependencies.
3. **More Advanced Architectures:**
   Stacked LSTMs, GRUs, or attention-based models (such as Transformers) could capture richer temporal relationships and reduce the smoothing behavior observed in predictions.
4. **Hyperparameter Optimization:**
   Automated methods such as grid search or Bayesian optimization could help identify optimal learning rates, hidden sizes, and regularization strategies.
5. **More Realistic Trading Simulation:**
   Adding transaction costs, slippage, position sizing rules, or risk constraints would allow for a more accurate assessment of whether predictive improvements translate into financial gains.
6. **Stability Improvements in Manual Training:**
   Techniques such as gradient clipping or mini-batch training could reduce noise in gradient updates and produce smoother convergence.

Exploring these directions could significantly improve the overall predictive power of the model and lead to more reliable financial forecasting results.

V. References

[1] Kaggle, *"Bitcoin Historical Data (Zielak),"* https://www.kaggle.com/datasets/mczielinski/bitcoin-historical-data

[2] Wikipedia Contributors, *"Long short-term memory,"* Wikipedia, https://en.wikipedia.org/wiki/Long_short-term_memory

[3] GeeksforGeeks, *"Introduction to Long Short-Term Memory (LSTM),"* https://www.geeksforgeeks.org/deep-learning/deep-learning-introduction-to-long-short-term-memory/

[4] NVIDIA Developer, *"Discover LSTM Networks,"* https://developer.nvidia.com/discover/lstm