Slow Turtle BackTesting with CRYPTEX

Lucas Raicu[1] and Ioan Raicu[5]

[1] Glenbrook South High School, Glenview, IL, USA 266520@glenbrook225.org
[2] Illinois Institute of Technology, Chicago, IL, USA iraicu@iit.edu

Abstract. When simulating financial models, fine-grained datasets significantly impact model effectiveness. In prior work, we obtained high-quality, one-second granularity data from Binance.us exchange for 153 cryptocurrency trading pairs from September 2019 to May 2024. We explored various backtesting algorithms and implemented the in Python. We simulated trading strategies against historical data to evaluate their performance and identify potential opportunities and risks. We incorporated several indicators, such as moving averages and thresholding to determine winning performance and risks. We focused on optimizing hyperparameters within these models to enhance predictive accuracy and profitability, and used concurrency to speedup backtesting on multi-core systems.

Keywords: Backtesting, Datasets, Time-series, Binance, Cryptocurrencies, Bitcoin

1 Introduction

This work investigates the premise that cryptocurrency time-series data can be used to build models to automate trading. In prior work, we obtained high-quality, one-second granularity data from Binance.us exchange for 153 cryptocurrency trading pairs from September 2019 to May 2024. [8, 9, 7] I found an interesting book titled "The Original Turtle Rules" [1]. This short book is a comprehensive trading system developed by Richard Dennis and William Eckhardt in the 1980s to prove that trading skills can be taught. The system is based on mechanical rules for markets, position sizing, entries, exits, and risk management, emphasizing discipline, consistency, and reliance on trend-following principles. Key components include fixed rules for exits and stops.

2 Proposed Backtesting Framework

To this end, we explored various backtesting algorithms and implemented a backtesting framework in Python. This framework allowed us to simulate trading strategies against historical data to evaluate their performance and identify potential opportunities and risks. We incorporated several indicators, such as moving averages and thresholding to determine winning performance and risks.

Additionally, we focused on optimizing hyperparameters within these models to enhance predictive accuracy and profitability.

We used concurrency to parallelize the computation of trading simulations by employing the concurrent.futures.ProcessPoolExecutor. Our basic backtesting function represented the core logic for performing a trading simulation using specific parameters like stop percentages, buy/sell window sizes, and profit/loss thresholds. We created a range of possible values for parameters like stop percentage, buy window, sell window, go percentage, and bad percentage, and submitted each combination of parameters as separate tasks to the process pool. This allows multiple simulations to be executed simultaneously in separate processes, leveraging multi-core CPUs for faster execution. The results from all tasks are gathered into a list with metrics like the final balance, number of trades, and the performance of the configuration. After all simulations are complete, the script evaluates the results to determine the best-performing parameter combination (e.g., highest final balance).

By using ProcessPoolExecutor, the script can handle multiple simulations concurrently, significantly reducing the overall runtime compared to a sequential approach. Each simulation runs independently, making it well-suited for parallel execution since there are no dependencies between tasks. This approach efficiently utilizes system resources and enables the script to explore a wide range of parameter combinations with linear scalability (tested up to 16-cores) compared to sequential execution.

3 Preliminary Evaluation Results

Here is an example of a backtesting simulation, after a hyperparameter search of 10K simulations were done. We used the 1W BTCUSD candlestick data from 09/2019 to 05/2024 for evaluation (see 1.

Fig. 1: BTCUSD historical weekly candlestick data from 09/2019 to 05/2024

The backtest stated with $1K$ worth of bitcoins, and ended with about $28K$ after 5 years (see Figure 2).

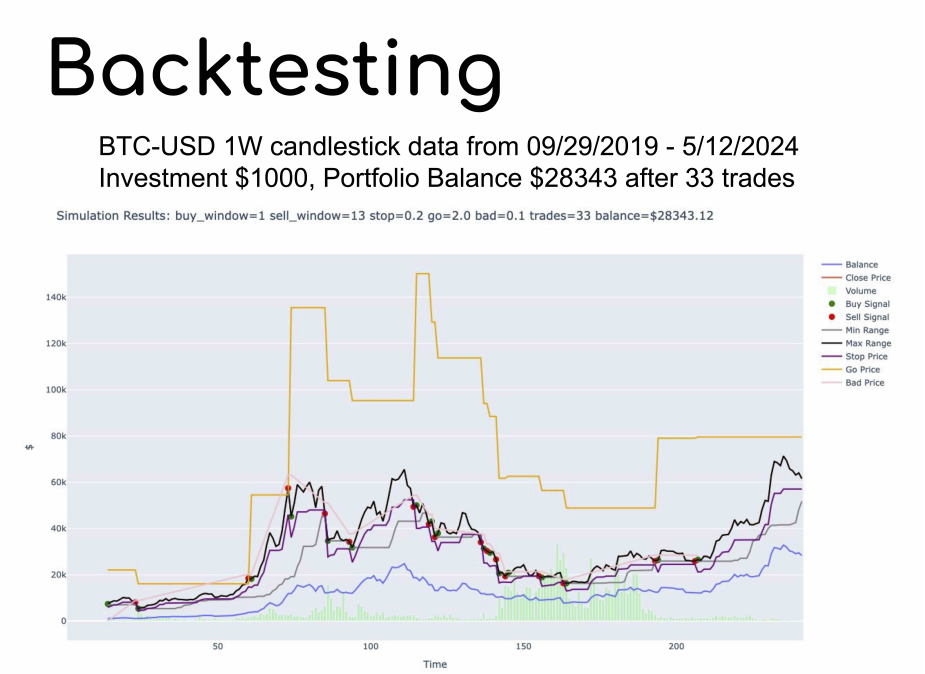

Fig. 2: Backtesting Example for weekly candlestick data for BTCUSD data from 09/2019 to 05/2024

We would like to test for millions of parameter combinations. Furthermore, we have over 100 trading pairs. Finally, we can explore finer grained data, which will significantly increase the cost of each simulation with the current implementation.

4 Future Work

There are a number of improvements we could make on this project, given more time. The concurrency used seems to be limited to a few hundred backtest simulations per second. Reading in the literature about the Parsl parallel runtime system [2], we believe we might be able to push several thousands simulations per second if we adopt Parsl and we have large enough systems (in terms of cores) and small enough simulations (e.g. daily candlesticks usually have very short runtimes).

We could investigate additional indicators, such as moving Relative Strength Index (RSI) [12] and Bollinger Bands [4], to develop and refine additional trading strategies. I would also like to implement performance metrics like Sharpe Ratio [10] and win-rate tracking to assess each strategy's viability. Furthermore, I believe I could integrate machine learning models, including long short-term memory networks (LSTMs) [6] and large language models (LLM) [11], to predict price movements and inform trading decisions, especially if I consider additional information such as news articles [5] and social media posts [3] in addition to historical price information.

The final stage of the project would be to incorporate real-time data streams and expand the project by deploying the framework into a real trading environment to test its performance in a dynamic, live market setting.

References

[1] Original turtles. http://www.tradingblox.com/originalturtles/ (2025), accessed: January 12, 2025

[2] Babuji, Y., Woodard, A., Li, Z., Katz, D.S., Clifford, B., Kumar, R., Lacinski, L., Chard, R., Wozniak, J.M., Foster, I., et al.: Parsl: Pervasive parallel programming in python. In: Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing. pp. 25–36 (2019)

[3] Bollen, J., Mao, H., Zeng, X.J.: Twitter mood predicts the stock market. Journal of Computational Science **2**(1), 1–8 (2011), examines how social media can inform financial predictions.

[4] Bollinger, J.: Bollinger on Bollinger Bands. McGraw Hill Professional (2002), comprehensive resource on Bollinger Bands.

[5] Ding, X., Zhang, Y., Liu, T., Duan, J.: Deep learning for event-driven stock prediction. In: Proceedings of the 24th International Conference on Artificial Intelligence. pp. 2327–2333. AAAI (2015), deep learning techniques for stock market prediction using news data.

[6] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997), introduced the LSTM architecture for sequence prediction.

[7] Raicu, L., Donisa, S., Ciobanu, L., Nguyen, L., Raicu, I.: Fine-grained bitcoin historical dataset 2019-2023. https://www.kaggle.com/datasets/iraicu/fine-grained-bitcoin-historical-dataset-2019-2023 (2023), accessed: August 5, 2023

[8] Raicu, L., Donisa, S., Ciobanu, L., Nguyen, L., Raicu, I.: Cryptex: fine-grained cryptocurrency datasets exploration. Greater Chicago Area Systems Research Workshop (GCASR) (2024)

[9] Raicu, L., Donisa, S., Ciobanu, L., Nguyen, L., Raicu, I.: Cryptocurrency data. http://crypto.cs.iit.edu/datasets/ (2025), accessed: January 12, 2025

[10] Sharpe, W.F.: The sharpe ratio. Journal of Portfolio Management **21**(1), 49–58 (1994), introduced the Sharpe Ratio for performance evaluation.

[11] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, , Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017), introduced the Transformer model, the basis for LLMs.
[12] Wilder, J.W.: New concepts in technical trading systems. Trend Research (1978), introduced the Relative Strength Index (RSI).