Mid Point Price Prediction of Crypto High Frequency Limit Order Books

Shivam Balikondwar
MS CS NYU Courant
NYU
ssb10002@nyu.edu

Arjun Parasuram Prasad
MS CS NYU Courant
NYU
ap9334@nyu.edu

Rivujit Das
MS CS NYU Courant
NYU
rd3681@nyu.edu

Abstract

Forecasting financial midpoint prices is a crucial task for understanding market dynamics and aiding decision-making in trading strategies. Traditional classical machine learning models offer interpretable solutions but often struggle to generalize in high-dimensional, rapidly changing datasets. Deep learning methods capture complex patterns effectively but are prone to over-fitting and fail to incorporate directional trends robustly. This research introduces a novel approach that leverages neural networks alongside classical machine learning models to address these limitations. By focusing on both prediction accuracy and trend consistency, the model achieves a balance between precision and robustness.

1 Introduction

In recent years, the analysis of financial markets has increasingly leveraged machine learning and deep learning techniques to model complex patterns and trends. One key challenge in financial forecasting is the accurate prediction of **midpoint prices**, which are critical for understanding market dynamics and informing trading decisions. Traditional models often struggle with capturing both the **precision of price prediction** and the **alignment with directional trends**, especially in scenarios involving high-dimensional data and rapidly changing market conditions.

Classical machine learning approaches such as **Support Vector Regression (SVR)** and **K-Nearest Neighbors (KNN)** offer interpretable frameworks for regression tasks but are often limited in their ability to generalize well across varying scales and data complexities. On the other hand, deep learning models like **Convolutional Neural Networks (CNNs)** and **Long Short-Term Memory (LSTM) networks** have demonstrated superior performance in capturing temporal and spatial dependencies. However, these models can suffer from issues like overfitting and a lack of robustness when applied to financial time series data.

To address these challenges, this research introduces a novel approach combining **CNNs** with a custom loss function inspired by the principles of **epsilon-insensitive regression**. The proposed loss function dynamically adapts to data variability and incorporates a trend alignment penalty to ensure that the model captures both the magnitude of predictions and the directional trends in the data. This architecture aims to bridge the gap between traditional machine learning and deep learning approaches, achieving a balance between interpretability, accuracy, and scalability.

The primary goal of this research is to evaluate the performance of the proposed method against classical models like **SVR** and **KNN**, as well as deep learning architectures like **CNN + LSTM** and **CNN + SVR**. By leveraging the proposed custom loss, we demonstrate improvements in both **Mean Squared Error (MSE)** and **trend alignment**, which are critical for financial forecasting tasks. The results showcase the effectiveness of the proposed method in delivering robust predictions in high-stakes applications like financial time series forecasting.

2 Related Work

Deep learning (DL) has become essential in high-frequency stock forecasting, outperforming traditional models by learning directly from raw order book data. CNNs, LSTMs, and their combinations (e.g., CNN-LSTM) dominate due to their ability to capture complex patterns and dependencies in stock price data.

- **CNN and LSTM Models:** Tran et al. (2018), Passalis et al. (2020) and Kolm et al. [2023] showed that CNNs with attention mechanisms and LSTMs outperform standalone models for mid-price prediction.
- **Hybrid and Attention Models:** Attention-augmented networks, as used by Rahimikia et al. (2021), improve performance by focusing on relevant features.
- **DL vs. Traditional Models:** Research consistently shows DL models surpass classical approaches like linear models and decision trees in predictive accuracy (Sirignano et al., (2019); Briola et al., (2020)).

Overall, CNNs, LSTMs, and MLPs are common in the literature, with hybrid architectures often achieving the best results. Optimal model choice remains data-dependent, though DL's adaptability to raw inputs makes it well-suited for financial forecasting.

3 Dataset

3.1 Overview

The experiments were conducted using the **Ethereum Limit Order Book (LOB) dataset**, sourced from the Kaggle dataset titled "High Frequency Crypto Limit Order Book Data." This dataset spans from **April 7, 2021, to April 19, 2021**, with minute-level granularity, covering approximately 12 days.

3.2 Size and Features

The dataset contains over **17,700 samples**, each described by more than **150 features**. Key attributes include:

- **spread**: Difference between the highest bid price and the lowest ask price.
- **buys and sells**: Indicators of trading activity.
- **bids_distance and ask_distance**: Distances of bids and asks from the midpoint price.
- **ask_market_notional and bids_market_notional**: Volumes of traded asks and bids.

3.3 Feature Selection and Transformation

To streamline the analysis, we selected significant features such as **ask_price, bid_price, ask_volume, and bid_volume**, renamed according to their depth levels in the LOB hierarchy (e.g., **bid_volume0**, **ask_volume0**, etc.).

The bid and ask prices, provided as percentage distances from the midpoint price, were transformed into actual prices using:

$$\text{Bid Price} = \text{Midpoint} + (\text{Midpoint} \times \text{bids\_distance}), \quad \text{Ask Price} = \text{Midpoint} + (\text{Midpoint} \times \text{ask\_distance}).$$

3.4 Depth Levels

Two depth levels were considered for modeling:

- **5 levels**
- **10 levels**

These levels represent the hierarchy of bid and ask values, providing a structured view of liquidity and price dynamics.

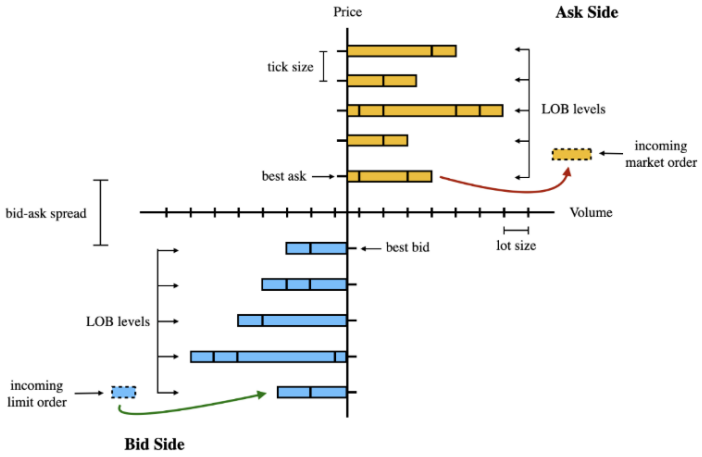


Figure 1: Ask and Bid Chart Denoting A Sample Limit Order Book

	time	midpoint	ask_volume0	ask_volume1	ask_volume2	bid_volume0	bid_volume1	bid_volume2	ask_price0	a
0	2021-04-07 11:33:49.861733+00:00	1965.845	20005.570312	3234.250000	2.000531e+04	86711.492188	9650.309570	29679.500000	1965.85	
1	2021-04-07 11:34:49.861733+00:00	1969.645	36921.578125	29840.800781	1.003755e+06	29539.800781	1031.329956	2736.439941	1969.97	
2	2021-04-07 11:35:49.861733+00:00	1975.595	13438.839844	3873.080078	6.975460e+03	1278.369995	2453.889893	31211.480469	1975.74	
3	2021-04-07 11:36:49.861733+00:00	1969.335	2028.510010	31570.119141	2.204419e+04	5555.390137	1969.010010	224564.453125	1969.43	
4	2021-04-07 11:37:49.861733+00:00	1970.965	1163.010010	1163.020020	3.769630e+03	1032.089966	4995.910156	29558.550781	1971.21	

Figure 2: Snapshot of our dataset after pre-processing

3.5 Order Flow Imbalance (OFI)

Order Flow Imbalance (OFI) is a key feature to gauge market sentiment:

- **Positive OFI**: Indicates bullish sentiment with higher buying pressure.
- **Negative OFI**: Indicates bearish sentiment with higher selling pressure.

OFI incorporates changes in price and volume for bid and ask levels. The computation formulas are as follows:

$$\text{Bid Order Flow (bid\_of)} = \begin{cases} \text{bid\_volume}, & \text{if bid\_price} > \text{bid\_price\_prev}, \\ \text{bid\_volume} - \text{bid\_volume\_prev}, & \text{if bid\_price} = \text{bid\_price\_prev}, \\ -\text{bid\_volume}, & \text{otherwise.} \end{cases}$$

$$\text{Ask Order Flow (ask\_of)} = \begin{cases} -\text{ask\_volume}, & \text{if ask\_price} > \text{ask\_price\_prev}, \\ \text{ask\_volume} - \text{ask\_volume\_prev}, & \text{if ask\_price} = \text{ask\_price\_prev}, \\ \text{ask\_volume}, & \text{otherwise.} \end{cases}$$

The final Order Flow Imbalance is computed as:

$$\text{OFI} = \text{bid\_of} - \text{ask\_of}.$$

3.6 Significance of OFI

The OFI metric enhances the model by:

- Adding interpretive market sentiment analysis.
- Providing an additional predictive dimension to complement price and volume data.
- Distinguishing between bullish and bearish market trends.

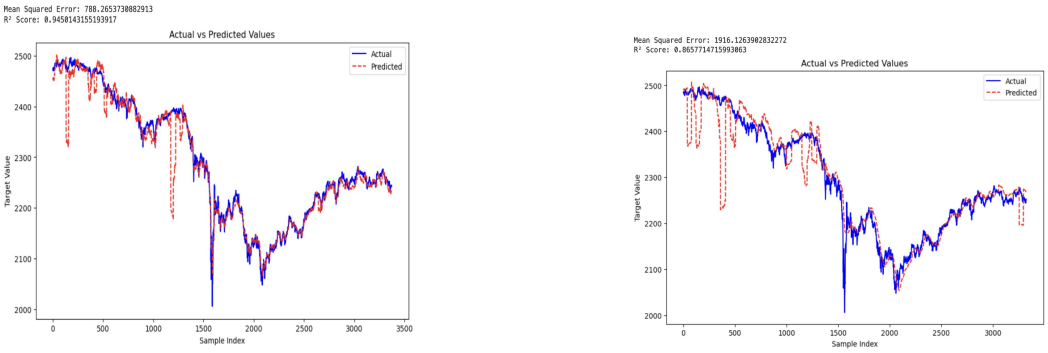
4 Method

We train an array of classical and deep learning models on the dataset for a fixed window size of 100 historical records. The input vector fed into these models is a normalized single-dimensional vector of size $100 \times$ (number of features per record). Each record's features consist of information from the top 5/10 buy bid and sell bid levels, along with optional derived features such as *Order Flow Imbalance (OFI)*.

The specific constructs of each of the models used are described below:

1. **KNN**: The KNN model is implemented using the `KNeighborsRegressor` class from scikit-learn. It serves as a baseline classical machine learning model.
2. **SVR**: A simple Support Vector Regressor (SVR) is employed using the `SVR` class from scikit-learn. Since, as shown in Figure 3, SVR captures the trend appropriately. Hence, we extend this approach by combining SVR with CNNs for improved feature extraction.
3. **LSTM**: The LSTM model is a deep learning architecture designed to capture sequential dependencies in the data. It consists of three LSTM layers, followed by dense layers and dropout layers for regularization.
4. **CNN + LSTM**: This hybrid deep learning model combines 10 Convolutional layers attached to a single Long Short-Term Memory (LSTM) layers followed by a dense layer to extract spatial and temporal features. CNNs capture spatial patterns from the Limit Order Book, while LSTMs model sequential dependencies.
5. **MLP**: A Multi-Layer Perceptron (MLP) regressor is employed, consisting of three fully connected layers with ReLU activation. It serves as a simple neural network baseline for regression.
6. **CNN + SVR** : To improve both prediction accuracy and trend alignment, we extend the SVR approach by integrating it with CNNs. The CNN is used for feature extraction, and the SVR provides the regression output. This model is particularly suited for capturing both magnitude and directional trends in the data.

All models are trained with a fixed window size of 100 and bid levels $\in \{5, 10\}$. The model with the lowest Mean Squared Error (MSE) and the highest R^2 score is taken as the best-performing model for these experiments.



(a) SVR with Level = 3

(b) SVR with Level = 10

Figure 3: Comparison of SVR performance at different levels.

4.1 Novel Architecture and Loss Function

Since the MSE for CNN and original SVR was significantly lower than CNN and LSTM combined, we defined a simple neural network architecture and integrated the custom loss function to further enhance prediction robustness and decrease model computational costs. This neural network, consisting of a single Dense layer performing a linear transformation, mimics the behavior of SVR. By replacing the standard loss function with our custom loss, the model achieves epsilon-insensitive regression while incorporating trend alignment and adaptability through a dynamically scaled margin.

The proposed loss function consists of the following components:

Dynamic Epsilon Margin

Instead of using a fixed tolerance margin ϵ, we dynamically scale ϵ based on the variability (standard deviation) of the true values:

$$\text{dynamic\_epsilon} = \epsilon \cdot \text{std}(y_{\text{true}})$$

This approach adapts the tolerance margin to datasets with varying scales, providing consistent performance across diverse conditions.

Baseline Loss (Epsilon-Insensitive Loss)

The baseline loss penalizes errors that exceed the dynamic epsilon margin:

$$\text{baseline\_loss} = \max(0, |y_{\text{true}} - y_{\text{pred}}| - \text{dynamic\_epsilon})$$

Predictions within the margin ($|y_{\text{true}} - y_{\text{pred}}| \leq$ dynamic_epsilon) incur no penalty, allowing small deviations to be ignored.

Direction-Sensitive Penalty

To enforce trend alignment, we add a direction-sensitive penalty. Relative changes in the true and predicted values are computed as:

$$\Delta y_{\text{true}} = y_{\text{true}} - \text{mean}(y_{\text{true}}), \quad \Delta y_{\text{pred}} = y_{\text{pred}} - \text{mean}(y_{\text{pred}})$$

The angles of these changes are derived using the arctangent function:

$$\theta_{\text{true}} = \tan^{-1}(\Delta y_{\text{true}}), \quad \theta_{\text{pred}} = \tan^{-1}(\Delta y_{\text{pred}})$$

The angular difference between true and predicted trends is calculated as:

$$\text{angular\_diff} = \frac{|\theta_{\text{true}} - \theta_{\text{pred}}|}{\pi/2}$$

The direction-sensitive penalty is proportional to the angular difference and the error magnitude:

$$\text{direction\_loss} = \mu \cdot \text{angular\_diff} \cdot |y_{\text{true}} - y_{\text{pred}}|$$

Total Loss

The total loss is the combination of the baseline loss and the direction-sensitive penalty:

$$\text{total\_loss} = \text{mean}(\text{baseline\_loss} + \text{direction\_loss})$$

This ensures that the model prioritizes accurate predictions both in value and in trend alignment, making it well-suited for tasks where directional trends are crucial.

Benefits of the Proposed Loss

We observed that this approach outperformed the CNN + original SVR and achieved the highest accuracy among all tested methods, including both classical and deep learning approaches. While the integration of the custom loss function introduces non-convexity, which can lead to challenges in optimization compared to the convex optimization used in classical SVR, the flexibility and adaptability of the proposed method allow it to capture complex patterns in the data. This makes it particularly well-suited for scenarios where the alignment of trends is as important as minimizing prediction errors.

5 Experiments

5.1 Evaluation Metrics

To evaluate the performance of the proposed model, we utilize two standard regression metrics: **Mean Squared Error (MSE)** and **Coefficient of Determination (R^2)**.

The **MSE** measures the average squared difference between the predicted (y_{pred}) and true values (y_{true}). It captures how close the predictions are to the actual values, with lower values indicating better performance. In the context of this project, MSE quantifies the model's precision in predicting the midpoint values. Mathematically, it is expressed as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_{\text{true},i} - y_{\text{pred},i})^2$$

where n is the number of samples.

The **R^2 score**, also known as the coefficient of determination, evaluates the proportion of variance in the true values explained by the model's predictions. It ranges from 0 to 1, where values closer to 1 indicate a better fit. For this project, R^2 helps assess how well the model captures the trends in the midpoint data. Its formula is given by:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_{\text{true},i} - y_{\text{pred},i})^2}{\sum_{i=1}^{n}(y_{\text{true},i} - \bar{y}_{\text{true}})^2}$$

where $\bar{y}_{\text{true}}$ is the mean of the true values.

These metrics together provide a comprehensive understanding of the model's accuracy (via MSE) and its ability to explain the variability in the data (via R^2).

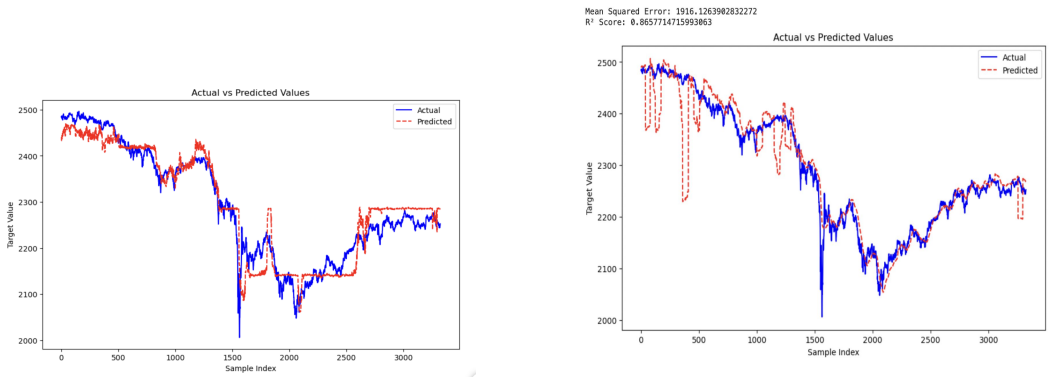
5.2 Emperical Results

As per Table 1, in the case of classical models, we observe that among the tested approaches, **SVR** achieves a lower Mean Squared Error (MSE) compared to **KNN**. Additionally, SVR demonstrates a better ability to capture the trend in midpoint prices, which is critical for evaluation. However, as the window size increases, SVR tends to perform poorly, likely due to the increased complexity and reduced effectiveness of its quadratic programming-based optimization.

For deep learning approaches, the proposed **CNN with the custom loss function** achieves the lowest MSE, outperforming all other methods. Specifically:

- The CNN with the custom loss achieves a lower MSE than the **CNN + SVR**.
- The **CNN + SVR** performs better than the **CNN + LSTM** model.

This demonstrates the effectiveness of the proposed custom loss function in improving both prediction accuracy and trend alignment.



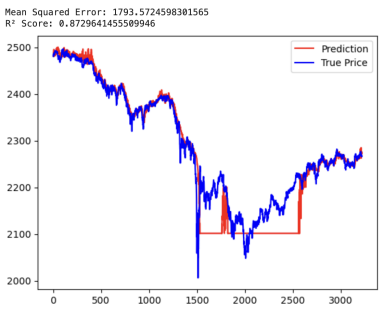
(a) Trend captured by KNN

(b) Trend captured by SVR

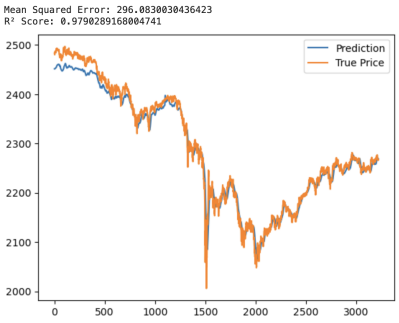
Figure 4: Trends captured by classical machine learning models.

Model	Bid Level	Metric	With OFI	Without OFI
K Nearest Neighbours	5	MSE	1478.71	1494.34
		R^2	0.895	0.894
	10	MSE	1920.86	1736.49
		R^2	0.863	0.877
Support Vector Regression	5	MSE	2109.95	1983.86
		R^2	0.85	0.85
	10	MSE	1125.17	948.72
		R^2	0.920	0.932
Multilayer Perceptron	5	MSE	178.248	440.3573
		R^2	0.9873	0.9688
	10	MSE	1793.572	188.411
		R^2	0.8729	0.9866
Long Short-Term Memory	5	MSE	250.70	350.15
		R^2	0.981	0.973
	10	MSE	335.52	446.91
		R^2	0.976	0.968
CNN and LSTM	5	MSE	88.85	35.20
		R^2	0.993	0.997
	10	MSE	45.21	36.60
		R^2	0.996	0.997
CNN and SVR	5	MSE	39.23	34.381
		R^2	0.994	0.997
	10	MSE	35.67	33.56
		R^2	0.994	0.997
CNN with Custom SVR	5	MSE	28.92	28.79
		R^2	0.9674	0.997
	10	MSE	29.12	28.05
		R^2	0.973	0.998

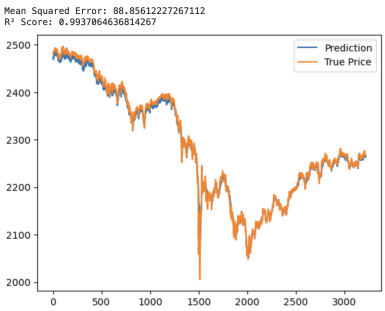
Table 1: Comparison of MSE and R^2 for Models with and without OFI across Bid Levels.



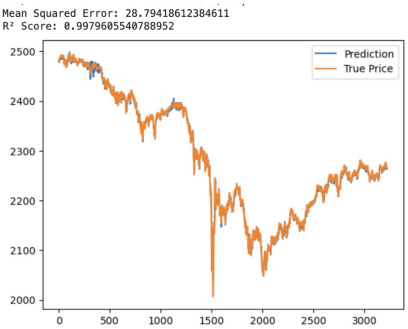
(a) Trend captured by MLP



(b) Trend captured by LSTM



(c) Trend captured by CNN + LSTM



(d) Trend captured by CNN + SVR with custom loss

Figure 5: Trends captured by deep learning models.

6 Conclusion and Future Work

In this research, we explored the efficacy of classical and deep learning models for financial midpoint price prediction, utilizing the Ethereum Limit Order Book (LOB) dataset. Among classical models, **SVR** demonstrated superior performance over **KNN**, effectively capturing trends in the data. However, its performance degraded with increasing window size, highlighting the limitations of traditional machine learning approaches in handling high-dimensional time series data.

In the deep learning domain, the **CNN with the custom loss function** outperformed all other models, including **CNN + SVR** and **CNN + LSTM**, achieving the lowest Mean Squared Error (MSE) value of **28.05** and highest R^2 score of **0.998**. This result demonstrates the effectiveness of the proposed custom loss function in capturing both prediction magnitude and directional trends. The hybrid **CNN + LSTM** model also showed promise, leveraging spatial and temporal features, but it lagged behind due to its higher computational complexity and potential overfitting.

The integration of **Order Flow Imbalance (OFI)** as a feature further enhanced model interpretability, enabling the detection of bullish and bearish market trends. Although the inclusion of OFI did not result in a drastic improvement in overall accuracy, we observed a slight reduction in MSE and better trend alignment. This suggests that OFI provides complementary information that improves model robustness.

Future Work

As a potential extension, we propose incorporating an **attention mechanism** on the OFI feature to further exploit its predictive power. An attention network could dynamically assign weights to the OFI values across different time windows, allowing the model to focus on critical market sentiment variations. This approach can help the model better capture subtle patterns in market activity and improve both accuracy and trend detection.

To implement this, the OFI column can be passed through an attention layer within the **CNN + SVR** or **CNN + LSTM** architectures. The attention scores can highlight relevant time steps where OFI has a significant impact, which are then used as additional inputs to the prediction layers. By enabling the model to prioritize these market sentiment shifts, we expect an improvement in both interpretability and predictive performance.

Future work will also explore the scalability of this approach to more granular datasets such as mid point price per second dataset, the inclusion of additional market features, and the development of ensemble methods to further enhance forecasting performance.

References

Dat Thanh Tran, Alexandros Iosifidis, Juho Kanniainen, and Moncef Gabbouj. Temporal attention-augmented bilinear network for financial time-series data analysis. *IEEE transactions on neural networks and learning systems*, 30(5):1407–1418, 2018.

Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Temporal logistic neural bag-of-features for financial time series forecasting leveraging limit order book data. *Pattern Recognition Letters*, 136:183–189, 2020.

Petter N Kolm, Jeremy Turiel, and Nicholas Westray. Deep order flow imbalance: Extracting alpha at multiple horizons from the limit order book. *Mathematical Finance*, 33(4):1044–1081, 2023.

Eghbal Rahimikia and Ser-Huang Poon. Machine learning for realised volatility forecasting. *Available at SSRN*, 3707796, 2021.

Justin A Sirignano. Deep learning for limit order books. *Quantitative Finance*, 19(4):549–570, 2019.

Antonio Briola, Jeremy Turiel, and Tomaso Aste. Deep learning modeling of limit order book: A comparative perspective. *arXiv preprint arXiv:2007.07319*, 2020.