Linnet: Limit Order Books Within Switches

Xinpeng Hong, Changgang Zheng, Stefan Zohren, and Noa Zilberman
University of Oxford
{xinpeng.hong, changgang.zheng, stefan.zohren, noa.zilberman}@eng.ox.ac.uk

ABSTRACT

Financial trading often relies nowadays on machine learning. However, many trading applications require very short response times, which cannot always be supported by traditional machine learning frameworks. We present Linnet, providing financial market prediction within programmable switches. Linnet builds limit order books from high-frequency market data feeds within the switch, and uses them for machine-learning based market prediction. Linnet demonstrates the potential to predict future stock price movements with high accuracy and low latency, increasing financial gains.

KEYWORDS

In-network computing, machine learning, programmable switches, P4, microstructure market data, limit order books, time series prediction

1 INTRODUCTION

High frequency trading (HFT) requires executing trades at high speed, with latency measured in microseconds [5] and where every nanosecond counts. Machine learning (ML) and deep learning approaches are applied nowadays to problems arising in HFT, increasing the demand for reduced latency across all components of the trade.

Over the past few decades, equity and derivative markets have witnessed an increasing use of electronic limit order books (LOBs) [3]. Formed by unmatched limit orders with specified prices, LOBs provide real-time information for traders and support market transparency [1]. In most electronic marketplaces, bids and offers are matched based on price/time priority rules [8]. When multiple orders with the same price are queuing for execution, the earliest active order is given top trading priority. Fast response time makes the difference between being the first to last order executed.

In-network computing executes applications within network devices with low latency and high throughput [12]. Although the application of in-network computing to ML is constrained, it has been demonstrated that trained ML models can be deployed within network devices to solve classification tasks [11, 13, 16]. Therefore, in-network ML becomes a potential solution for low latency trading as well as numerous time-sensitive financial applications.

This work focuses on exploring the application of in-network ML to a typical problem in HFT: predicting future price movements from market microstructure signals. Such prediction leads to higher profitability and was shown to be feasible (ML-wise) in previous work [6]. We design and implement Linnet, an in-network application prototype for generating, keeping, and updating an LOB based on order-based market data feeds, and using it to predict future price movements. A preliminary evaluation shows that Linnet achieves minimal loss of accuracy and f1-score compared with a server-based benchmark. Linnet was implemented on a behavioral model version 2 (bmv2), and will be extended to switch-ASICs or data processing units (DPUs).

2 HIGH-LEVEL ARCHITECTURE

Linnet accelerates stock price movement prediction by constructing and updating LOBs within the programmable data plane. The high-level system architecture is shown in Figure 1: in traditional solutions, orders' information goes from the stock exchange to the traders' servers through a switch, and the LOB is constructed within the server (and its accelerators). In contrast, in Linnet the LOB construction and ML algorithm are implemented within the switch, eliminating the latency of getting to the server, and its processing latency.

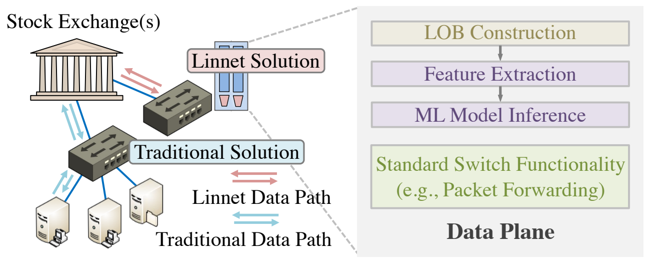

Figure 1: Architecture Design of the Linnet prototype.

NASDAQ is the world's first electronic stock market. A NASDAQ trace of market by order (MBO) data [10] is used for this study. MBO messages are used to construct an LOB within a P4-programmable network device, and then information from the LOB is used to predict stock price movement using a trained ML model, running within the same device. For training and exploration purposes, only a single stock is used at a time, where stocks are identified within MBO messages. The constructed LOB follows the principle of price/time priority. At each timestamp, features are extracted from the current state of the LOB, including different levels on both buy and sell sides, with information on both price and volume. The current bid-ask midpoint (mid-price) is calculated in order to create labels that represent the direction of price changes.

3 LIMIT ORDER BOOKS IN LINNET

LOBs are implicitly derived from MBO data [14]. The typical structure of MBO data includes the fields timestamp, type, ID, side, size and price of an order. In Figure 2 Step ❶, "Type" refers to the type of instruction (to add, cancel, or update an order) and "Side" shows whether an order is a bid (buy) order or an ask (sell) order. Only add-order messages are currently used to update an LOB in our prototype. Figure 2 Step ❷ demonstrates an exemplary slice of an LOB. It includes two types of orders residing in the bid side and ask side. While a bid order is used to buy an asset at or below a specified price, an ask order does the opposite, selling an asset at or above a given price [15]. A mid-price sits at the midpoint of the highest bid price and the lowest ask price. Figure 2 Step ❸ illustrates how an LOB is updated with MBO messages. In the shown workflow, ask orders are used as an example, and bid orders are handled by the algorithm in a similar way.

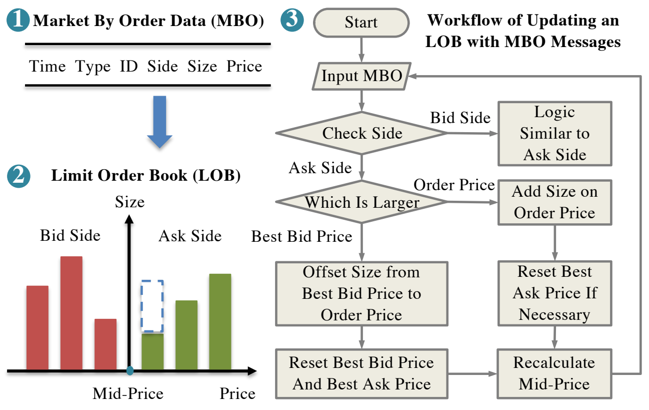
❶ Market By Order Data (MBO)
Time Type ID Side Size Price
❷ Limit Order Book (LOB)
Size
Bid Side
Ask Side
Mid-Price
Price
❸
Start
Workflow of Updating an LOB with MBO Messages
Input MBO
Check Side
Bid Side
Logic Similar to Ask Side
Ask Side
Which Is Larger
Order Price
Add Size on Order Price
Best Bid Price
Offset Size from Best Bid Price to Order Price
Reset Best Ask Price If Necessary
Reset Best Bid Price And Best Ask Price
Recalculate Mid-Price

Figure 2: ❶ Market by order (MBO) data fields, ❷ graphical representation of a limit order book (LOB), and ❸ workflow of updating an LOB with MBO messages.

One of the biggest challenges of implementing in the data plane the update process of an LOB is that P4 inherently does not support loops. Consequently, there is a trade-off between accurate extraction of high-level features from the LOB and resource consumption within switches. In our prototype's implementation, if any block of code needs to be executed a number of times, loop unrolling is used. This prevents excessive resource overhead in the switch without a significant loss of prediction accuracy.

4 PRELIMINARY EVALUATION

NASDAQ's Historical TotalView-ITCH sample data feeds [10] are reconstructed and filtered using [2], with the trace used to test Linnet. Planter, a framework for in-network ML deployment [17, 18], provides the ML substrate, with server-side benchmarks based on Scikit-learn [9]. Given their wide use in similar tasks over the past decades, naive Bayes (NB), decision trees (DTs), random forests (RFs), and extreme gradient boosting (XGB) are chosen for stock price movement prediction. The features used for ML model training are the volumes at the first three levels of an LOB, as well as the mid-price. Since financial data is highly stochastic with low signal-to-noise ratio, a smoothing labelling method is applied as part of the training to produce more consistent signals [7]. The synthetic minority over-sampling technique (SMOTE) is used to solve the class-imbalance problem [4]. Taking three different stocks (COST, NVDA, and ASML) as an example, Table 1 shows the preliminary evaluation results in terms of accuracy (ACC) and f1-score (F1).

Among all the models for these three stocks, the average accuracy loss of Linnet is 1.7% compared to the benchmark while the average f1-score loss is 4.5%. Linnet achieves the same accuracy as the benchmark in some cases. Given that the size of the models mapped into the programmable switch was limited, Linnet's performance is promising.

	COST				NVDA				ASML			
	Switch (L)		Sklearn (U)		Switch (L)		Sklearn (U)		Switch (L)		Sklearn (U)	
Model	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
NB	92.9	45.7	93.1	45.9	79.8	66.2	81.3	66.7	89.2	43.5	91.3	46.2
DT	92.5	61.5	93.4	62.6	91.8	56.2	96.2	63.1	97.0	79.9	98.7	90.0
RF	93.1	62.3	93.2	62.4	96.3	68.2	97.9	73.7	96.0	69.3	98.2	82.3
XGB	86.3	79.8	90.0	83.3	95.7	62.9	97.9	69.5	95.1	72.0	95.1	76.1

Table 1: Preliminary evaluation results (%). Linnet runs on a switch with (L)imited model size and the benchmark runs on a server with (U)nlimited model size.

5 CONCLUSION AND FUTURE WORK

This paper presents the application of in-network ML for time-sensitive financial trading. Linnet, a prototype for building and updating limit order books in the data plane, is designed and deployed on programmable network devices. The preliminary evaluation shows that Linnet enables accurate feature extraction, keeping high prediction accuracy compared with the benchmark. In future work, Linnet will be extended to more types of programmable network devices under more complex real-world scenarios, and will be evaluated for latency and resource overhead.

Acknowledgements This work was partly funded by VMWare and we acknowledge support from Intel. Stefan Zohren thanks the Oxford-Man Institute of Quantitative Finance for financial support.

REFERENCES

[1] Shmuel Baruch. 2005. Who benefits from an open limit-order book? *The Journal of Business* 78, 4 (2005), 1267–1306.
[2] Martino Bernasconi-De-Luca, Luigi Fusco, and Ozrenka Dragić. 2021. martinobdl/ITCH: ITCH50Converter. https://doi.org/10.5281/ZENODO.5209267
[3] Charles Cao, Oliver Hansch, and Xiaoxin Wang. 2009. The information content of an open limit-order book. *Journal of Futures Markets: Futures, Options, and Other Derivative Products* 29, 1 (2009), 16–41.
[4] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.
[5] Michael A Goldstein, Pavitra Kumar, and Frank C Graves. 2014. Computerized and high-frequency trading. *Financial Review* 49, 2 (2014), 177–202.
[6] Michael Kearns and Yuriy Nevmyvaka. 2013. Machine learning for market microstructure and high frequency trading. *High Frequency Trading: New Realities for Traders, Markets, and Regulators* (2013).
[7] Adamantios Ntakaris, Martin Magris, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. 2018. Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods. *Journal of Forecasting* 37, 8 (2018), 852–866.
[8] Christine A Parlour and Duane J Seppi. 2008. Limit order markets: A survey. *Handbook of financial intermediation and banking* 5 (2008), 63–95.
[9] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.
[10] NASDAQ OMX PSX. 2014. NASDAQ OMX PSX TotalView-ITCH 5.0. (2014). http://www.nasdaqtrader.com/content/technicalsupport/specifications/dataproducts/PSXTVITCHSpecification_5.0.pdf
[11] Davide Sanvito, Giuseppe Siracusano, and Roberto Bifulco. 2018. Can the network be the AI accelerator?. In *Proceedings of the 2018 Morning Workshop on In-Network Computing*. 20–25.
[12] Yuta Tokusashi, Huynh Tu Dang, Fernando Pedone, Robert Soulé, and Noa Zilberman. 2019. The case for in-network computing on demand. In *Proceedings of the Fourteenth EuroSys Conference 2019*. 1–16.
[13] Zhaoqi Xiong and Noa Zilberman. 2019. Do switches dream of machine learning? toward in-network classification. In *Proceedings of the 18th ACM workshop on hot topics in networks*. 25–33.
[14] Zihao Zhang, Bryan Lim, and Stefan Zohren. 2021. Deep learning for market by order data. *Applied Mathematical Finance* 28, 1 (2021), 79–95.

[15] Zihao Zhang, Stefan Zohren, and Stephen Roberts. 2019. Deeplob: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing* 67, 11 (2019), 3001–3012.

[16] Changgang Zheng, Zhaoqi Xiong, Thanh T Bui, Siim Kaupmees, Riyad Bensoussane, Antoine Bernabeu, Shay Vargaftik, Yaniv Ben-Itzhak, and Noa Zilberman. 2022. IIsy: Practical In-Network Classification. *arXiv preprint arXiv:2205.08243* (2022).

[17] Changgang Zheng, Mingyuan Zang, Xinpeng Hong, Riyad Bensoussane, Shay Vargaftik, Yaniv Ben-Itzhak, and Noa Zilberman. 2022. Automating In-Network Machine Learning. *arXiv preprint arXiv:2205.08824* (2022).

[18] Changgang Zheng and Noa Zilberman. 2021. Planter: seeding trees within switches. In *Proceedings of the SIGCOMM'21 Poster and Demo Sessions*. 12–14.