Deep Hawkes Process for High-frequency Market Making

Kumar, Pankaj

Document Version
Final published version

Published in:
Journal of Banking and Financial Technology

DOI:
10.1007/s42786-024-00049-8

Publication date:
2024

Journal of Banking and Financial Technology
https://doi.org/10.1007/s42786-024-00049-8

ORIGINAL ARTICLE

Deep Hawkes process for high-frequency market making

Pankaj Kumar[1]

Received: 23 June 2021 / Accepted: 15 February 2024

Abstract
High-frequency market making is a liquidity-providing trading strategy that simultaneously generates many bids and asks for a security at ultra-low latency while maintaining a relatively neutral position. The strategy makes a profit from the bid-ask spread for every buy and sell transaction, against the risk of adverse selection, uncertain execution and inventory risk. We design realistic simulations of limit order markets and develop a high-frequency market making strategy in which agents process order book information to post the optimal price, order type and execution time. By introducing the Deep Hawkes process to the high-frequency market making strategy, we allow a feedback loop to be created between order arrival and the state of the limit order book, together with self- and cross-excitation effects. Our high-frequency market making strategy accounts for the cancellation of orders that influence order queue position, profitability, bid-ask spread and the value of the order. The experimental results show that our trading agent outperforms the baseline strategy, which uses a probability density estimate of the fundamental price. We investigate the effect of cancellations on market quality and the agent's profitability. We validate how closely the simulation framework approximates reality by reproducing stylised facts from the empirical analysis of the simulated order book data.

Keywords High-frequency trading · Deep Hawkes process · Deep learning · Agent-based models · Market making

1 Introduction

Technological innovations and regulatory initiatives in the financial market have led to the traditional exchange floor being displaced by the electronic exchange. The electronic exchange is a fully automated trading system programmed to incisively enforce order precedence, pricing and the matching of buy and sell orders. Each order's pricing, submission and execution is performed using sophisticated algorithmic trading strategies, which account for 85% of the equity market's trading volume [40]. High-frequency trading (HFT, or high-frequency trader), a subset of algorithmic trading, is characterised by exceptionally high speeds, minuscule timeframes and complex programs for initiating and liquidating positions [51]. The critical discussion on the role of HFT in a fragmented market has been reignited after the Flash Crash of 6 May 2010 [25]. This systemic intra-day anomaly only lasted for a couple of minutes, but temporarily wiped away a trillion dollars in market value. The analysis of agents resolved transaction level data in the E-mini by [25] also looks at the behaviour of market makers, whose inventory dynamics remain stationary in conditions of fluctuating liquidity. Even though the market design of E-mini has no high-frequency market maker liability, unlike equity markets, this seminal paper [25] gave a boost to research aimed at understanding high-frequency market making or other liquidity-providing strategies in an algorithmic trading setting.

Market making is a liquidity-providing trading strategy that quotes numerous bids and asks for a security in anticipation of making a profit from a bid-ask spread, while maintaining a relatively neutral position [6]. The high-frequency market making strategy can be characterised as subset of HFT that uses latency, at a scale of nanoseconds, to trade in a fragmented market [35]. The growing literature reports that the market makers provide quality liquidity, improve market quality, contribute to price efficiency and have a positive but moderate welfare effect [3, 25, 35]. However, there is another strand in the literature that argues that the quality of liquidity is deceptive. The orders are characterised as phantom liquidity, which quickly disappears before

✉ Pankaj Kumar
pk.mpp@cbs.dk

[1] Copenhagen Business School, Porcelænshaven 18B, 2.152, 2000 Frederiksberg, Denmark

Published online: 26 April 2024

other market participants can access it. The optimal design of market making strategies is therefore an important question for practical applicability, market design and security exchange regulations.

The research on market making spans numerous disciplines, including finance [1, 2, 5, 15, 17, 20, 44], agent-based modeling [7, 12, 47, 59], and artificial intelligence [14, 26, 53]. Inspired by seminal work of [20] and its mathematical formulation [2], the quintessential research in finance considers market making as a stochastic optimal control problem. In a simplistic setting, the market is modelled as a stochastic process, in which market makers try to maximise the expected utility of their profit and loss under inventory constraints [17]. In parallel to inventory-based models, [15] proposed information-based models, in which market makers face adverse selection risk emerging from informed traders. The unrealistic assumptions placed on market models to mathematically extract the market maker's asset pricing forces researchers to look beyond stochastic optimal control approaches.

Market making has also been extensively investigated in agent-based modelling (ABM, or agent-based model) literature [12, 59]. The ABMs in market making evolved from zero intelligence to an intelligent variant by incorporating order book microstructure for order placement, execution and pricing policy. For example, [41] reinforced the zero-intelligence market maker model with order arrival following mutually exciting Hawkes processes. The Hawkes process has been exhaustively used in an empirical estimation and calibration of market microstructure models deemed essential for designing optimal market-making strategies [19, 38, 41]. In these models, the arrival rate of orders is not dependent on the state of the limit order book. However, the empirical results suggest the existence of feedback loop between order arrival and the state of the limit order book, together with self- and cross-excitation effects, for which current models fail to account [16, 38]. In addition, the Hawkes process constrains the parametric specification for conditional intensity, which limits the model's eloquence. To tackle the parametric specification problem, [34] proposed the Neural Hawkes process (NHP), in which the Hawkes process is generalised by calculating the event intensities from the hidden state of a long short-term memory (LSTM). Despite the success of the NHP in natural language processing [34], the facile LSTM architecture might be inadequate when it comes to modelling noisy, asynchronous order book events.

In recent years, deep learning has made significant inroads into high- frequency finance. The Convolutional Neural Network (CNN) architecture and its variants were used to model price-formation mechanisms using order book events as input [10, 52, 55, 57]. However, the CNN architectures are not sophisticated enough to capture self- and cross-excitation effects in the limit order book (LOB) [60]. The deep long-short term memory (DLSTM) architecture performs a hierarchical processing of complex order book events, and as such is able to capture the temporal structures of LOB [49, 50]. The architecture of the DLSTM is same as the previously introduced LSTM, apart from the fact that it involves multiple LSTM layers stacked on top of each other. The DLSTM is efficient in performing the hierarchical processing of noisy order book data which has also complex spatial-temporal dependency between order book events and highly non-linear. The literature supports our claim that only LSTM will not be able to capture that robust pattern [49, 50]. Additionally, the multivariate asynchronous order book data have non-normal conditional distributions which would be not that easy to learn using LSTM only.

However, training the DLSTM model directly through stochastic gradient descents, initialised with random parameters, may have led to the backpropagation algorithm being trapped within multiple local minima [50, 58]. To circumvent the aforementioned limitations, the literature proposed an unsupervised pre-training of each layer and a stacking of many convolutional layers [58]. Other tailored heuristics developed after trial and error methods to train a deep neural network relatively easily includes Naive initialization, LeCun initialization, Kaiming initialization, layer-sequential unit-variance (LSUV) initialization, Finite width networks and Dynamical isometry [54, 58]. We use Stacking Denoising Autoencoders (SDAEs) together with DLSTM to resolve the random weight initialisation problem in base architecture [50]. SDAEs are quite effective at filtering out noisy and non-linear order-level data at minuscule resolutions [29]. Additionally, the above discussed tailored initialization heuristics require the tuning the scaling of the random initial point whose complexity increases as the network becomes deep. Also, It would be hard to directly extend the derivation of initialisation to general non-linear activations [54]. As stated extensively in [58], we follow the training process for SDAE same as earlier work.

The exemplary predictive performance of deep-learning models has encouraged researchers to augment order book data with agent-based artificial market simulation, for the purpose of investigating algorithmic trading strategies [31]. The success of the model is dependent on the simulation framework of the financial market being close to realism. However, algorithmic trading research is still waiting for market simulators that could be used for developing, training, and testing algorithms in a manner similar to classic Atari 2600 games simulator [36]. In this paper, we develop realistic simulations of the financial market and use them to design a high-frequency market making agent using the Deep Hawkes process (DHP). The DHP models the streams of order book events by constructing a self-exciting multivariate Hawkes process and a limit order state process,

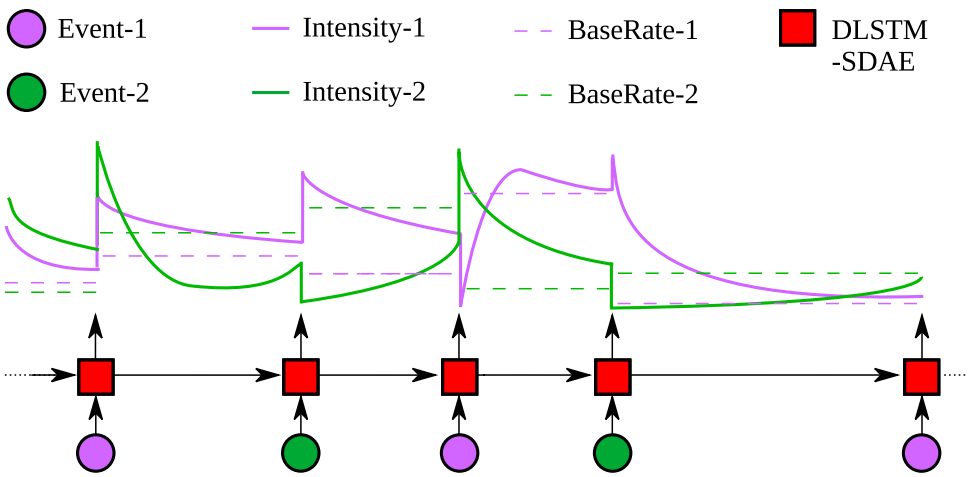


Fig. 1 Snapshot of order book event stream from the DHP. A DLSTM-SDAE takes the sequence of previous order book events (circles) to have hidden state representation (squares) of the top layer in stacked LSTM architecture, which in turn gives future intensities (solid lines) eventually approaching the base rate (dashed line). Here, an order book event (Event-1) excite itself but inhibits other (Event-2). Similarly, the order book event (Event-2) excites itself, and excites or inhibits the earlier (Event-1) as per the counting process. The sudden jump in intensity depicts immediate effects. The example is taken from NHP [34]

which are coupled and interact with each other. Based on a long stream of high-frequency transaction-level order book data for the different events (e.g. buy, sell, cancel, etc), the high-frequency market makers use DHP to accurately predict every held-out event.

This paper is the first to incorporate DHP into the market making strategy, which allows feedback loops between order arrival and the state of the limit order book, together with self- and cross-excitation effects. We extend the neurally self-modulating multivariate point process [34] to the deep framework by stacking SDAE with DLSTM, resulting in DLSTM-SDAE. The SDAE resolves the problem associated with weight initialisation, multiple local minima and ultra-noisy order book data that the stacked recurrent network fails to address. Our approach outperforms the NHP in predicting the next order type and its time. The gained predictive power helps agents to outperform the benchmark market making strategy, and uses a probability density estimate of the fundamental price. We outline our contribution as follow. We designed a multi-asset simulation framework that is scalable and can augment markets of substantial size. The framework is built on realistic market architecture, interface kernels, a matching engine and the Financial Information eXchange (FIX) protocol [46]. We are first to introduce a feedback loop between order arrival and the state of the order book using DHP in the high-frequency market making setting. However, the key limitation of our framework is its inability to take account of the basket events, external events and actions-reactions of multiple agents trading in the markets. We investigate the predictive and trading performance of the agents with the benchmark. We explore the effect of cancellation on order queue position, agent's profitability, bid-ask spread, and value of order relative to queue position, in order to verify the existing empirical findings [11, 37].

The rest of the paper is organized as follow. Section 2 explains the novel deep Hawkes process. Section 3 illustrates the multi-agent simulation framework. Section 4 elaborates on the experimental configuration. Section 5 provides the results of the experiments. Section 6 presents our conclusions.

2 Deep Hawkes process

In this section, we propose that the DHP be used to concurrently model the order book event timings and associated event types. By assimilating it into the order arrival process, the high-frequency market makers have control over the sending of different orders at specific points in time. The basic idea behind our approach is to view the conditional intensity of the Hawkes process as a nonlinear deterministic function of past history, and to use DLSTM to automatically learn a high-dimensional representation from the data. A schematic example representing an order book event sequence from DHP is shown in Fig. 1. Unlike traditional Hawkes Process [19], the base rate in DHP is not constant and shifts after each event. Additionally, the intensities drift (solid lines) are non-monotonic because the hidden states of DLSTM may decay at different rates. The hidden states have delayed response in comparison to the exponential decay [34]. Next, we discuss each component of the figure in detail.

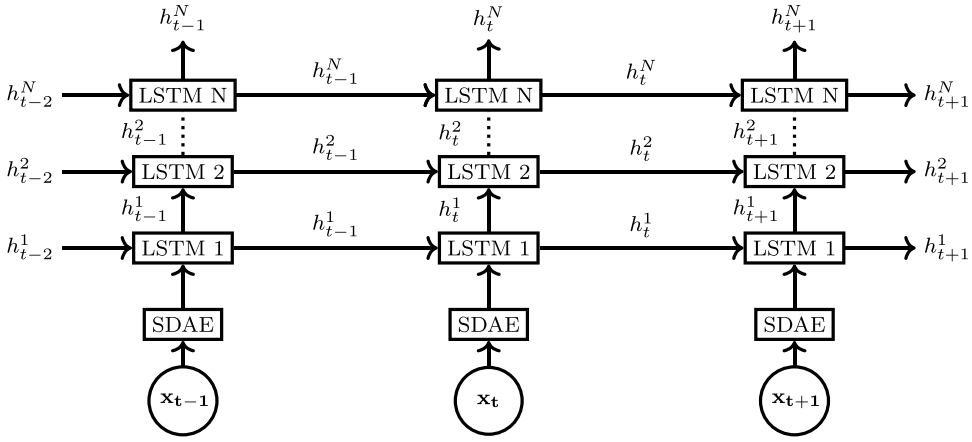


Fig. 2 The DLSTM-SDAE architecture

2.1 DLSTM-SDAE

In an ideal setting, the performance of the DLSTM model proved to be empirically better than existing contemporary statistical and deep modules. However, as the non-linear variables are modelled in such a way as to be scaled up, the overall learning of the models suffers greatly due to the back-propagation algorithm trapped within multiple local minima [50]. This may be the biggest hurdle when it comes to modelling limit order book events that comprise complex, asynchronous non-linear multivariate time series. The ultra-noisy order book data also makes processing more challenging. To circumvent the limitations of the conventional stacked LSTM model, we use the stacked denoising auto-encoder (SDAE) together with DLSTM. The SDAE enables the deep neural networks with multiple nonlinear hidden layers to learn complex features from noisy limit order book data [29, 58] and resolves the random weight initialization of LSTM's units problem in DLSTM [50]. Figure 2 shows the proposed DLSTM- based SDAE architecture for DHP. It is worth to be mentioned that, the SDAE and DLSTM are independent architecture. We only employ SDAEs to denoise the non-stationary and noisy asynchronous order book data. The pre-trained high-level hidden state, a robust representation of the order book event is then passed to DLSTM. In short, the SDAEs act as a non-linear denoiser that enhances data robustness. The DLSTM deals with the asynchronous order book data where events are high dimensional and have a complex spatial-temporal relationship. Of course, this reduces the model's interpretability and complexity.

The architecture of SDAEs can be designed by staking multiple Denoising Autoencoders (DAE) [58]. DAEs have been widely used to extract low-dimensional features of raw data by using an inbuilt network module, an encoder and a decoder. While the encoder learns the robust low-dimensional representation of the data, the decoder reconstructs the raw data with minimum reconstruction loss. First, the input $\mathbf{x_t}$ is corrupted into $\tilde{\mathbf{x}}_\mathbf{t}$ using stochastic mapping $\tilde{\mathbf{x}}_\mathbf{t} \sim \mathcal{S}_\mathcal{D}(\tilde{\mathbf{x}}_\mathbf{t} \mid \mathbf{x_t})$. Then, the autoencoder maps corrupted input $\tilde{\mathbf{x}}_\mathbf{t}$ to a hidden representation $h = f_\theta(\tilde{\mathbf{x}}_\mathbf{t})$ with encoder $f_\theta(\tilde{\mathbf{x}}_\mathbf{t}) = (W\tilde{\mathbf{x}}_\mathbf{t} + b)$. Lastly, the decoder $g_{\theta'}$ reconstruct $\mathbf{z} = g_{\theta'}(\tilde{\mathbf{x}}_\mathbf{t})$ from the hidden representation h. The parameters θ and θ' are trained using stochastic gradient descent to minimize reconstruction error measured in the squared error loss $L_2(\mathbf{x}, \mathbf{z}) = \|\mathbf{x} - \mathbf{z}\|^2$. Once mapping is learned, the high-level hidden state h is applied for training the next layer. For our model, we use Gaussian noise to corrupt the raw order book data. For detail learning procedure in SDAE, please refer to seminal paper on the subject [58].

To denoise the high-dimensional order book data, we adopt convolutional DAEs. Here, the convolutional encoder uses customized CNN architecture to reduce the spatial dimension of the order book data by increasing the depth. On the contrary, the convolutional decoder does the reverse operation. The convolutional encoder consist of two convolutional layers, **Conv.(32 filters**, **size** $= 16 \times 16$, **stride** $= 1 \times 2$), a batch normalization layer and leaky rectifying linear units an activation layer. The architecture of the convolutional decoder part is inversely symmetric to the convolutional encoder. We use two convolutional layers to integrate the convolutions effect of order book events over time and multiple order book depths.

As shown in Fig. 2, the reconstructed order book data is denoised by SDAEs layers in DLSTM-SDAE architecture. Then at time t, the denoised input $\mathbf{x_t}$ from SDAE is then passed to first layer of LSTM together with previous hidden state h^1_{t-1}. The hidden state at time t, h^1_t is calculated using recursive LSTM procedure. Its is then moved to next time step and LSTM layers. In the second layer, the hidden state h^1_t and the previous h^2_{t-1} is used to compute h^2_t and procedure repeats until last layer is complied.

2.2 Model formulation

Let $\{t_n, \varkappa_n, k_n\}_{n\in\mathbb{N}}$ be a stream of order book event, where t_n are times of occurrence of an event, its component $\varkappa_n$, and

their corresponding mark $\{k_n\}_{n\in\mathbb{N}}$ in $k_n := \{1, \ldots, K\}$. Then, the probability that the next event occurs at time t_n is of type k_n is $\mathbb{P}\{(t_n, \varkappa_n, k_n) \mid \mathcal{H}_n, (t_n - t_{n-1})\}dt$. We are interested in the model to predict next event stream $\{t_n, \varkappa_n, k_n\}$ given a past history of event k_n, evaluate its likelihood and simulate the next event stream by learning from the past event stream. Equation (1) shows the associated intensity function of the DHP with relaxed positivity constraints:

$$\lambda_k(t) = f_k(\mathbf{w}_k^\top \mathbf{h}(t)) \tag{1}$$

The hidden state $\mathbf{h}(t)$ is updated from the memory sell $\mathbf{c}(t)$ as in Eq. (2).

$$\mathbf{h}(t) = \mathbf{o_n} \odot \phi(c(t)) \text{ for } t \in (t_{n-1}, t_n] \tag{2}$$

The life of interval $(t_{n-1}, t_n]$ is determined by the next event k_n at t_n, DLSTM reads $\{t_n, \varkappa_n, k_n\}$ and updates the current memory cells $\mathbf{c}(t)$ to $\mathbf{c}_{n+1}(t)$, associated with hidden state $\mathbf{h}(t_n)$. The other parameters of the DLSTM are recursively updated according to the Eq. (3).

$$\begin{aligned}
\mathbf{i}_{n+1} &= \sigma\big(\mathbf{W}_{xi}\mathbf{x}_n + \mathbf{W}_{hi}\mathbf{h}(t_n) + \mathbf{W}_{ci}\mathbf{c}(t_n) + \mathbf{b}_i\big),\\
\mathbf{f}_{n+1} &= \sigma\big(\mathbf{W}_{xf}\mathbf{x}_n + \mathbf{W}_{hf}\mathbf{h}(t_n) + \mathbf{W}_{cf}\mathbf{c}(t_n) + \mathbf{b}_f\big),\\
\bar{\mathbf{c}}_{n+1} &= \phi\big(\mathbf{W}_{xc}\mathbf{x}_n + \mathbf{W}_{hc}\mathbf{h}(t_n) + \mathbf{b}_c\big),\\
\mathbf{c}_{n+1} &= \mathbf{f}_{n+1} \odot \mathbf{c}(t_n) + \mathbf{i}_{n+1} \odot \bar{\mathbf{c}}_{n+1},\\
\hat{\mathbf{c}}_{n+1} &= \hat{\mathbf{f}}_{n+1} \odot \hat{\mathbf{c}}(t_n) + \hat{\mathbf{i}}_{n+1} \odot \bar{\mathbf{c}}_{n+1},\\
\mathbf{o}_{n+1} &= \sigma\big(\mathbf{W}_{xo}\mathbf{x}_n + \mathbf{W}_{ho}\mathbf{h}(t_n) + \mathbf{W}_{co}\mathbf{c}(t_n) + \mathbf{b}_o\big),\\
\bar{\bar{\mathbf{s}}}_{n+1} &= f\big(\mathbf{W}_{xd}\mathbf{x}_n + \mathbf{W}_{hd}\mathbf{h}(t_n) + \mathbf{W}_{cd}\mathbf{c}(t_n) + \mathbf{b}_d\big),
\end{aligned} \tag{3}$$

where $\mathbf{x}_n$ is nth input vector represented by one hot encoding of new order book event k_n; the activation functions $\sigma(x)$ / $\phi(x)$ are sigmoid / hyperbolic tangent function, respectively; $\mathbf{W}_{AB}$ (e.g., $\mathbf{W}_{ci}$) is the weight matrix from the memory cell to input gate vector; $\mathbf{b}_B$ denotes the bias term of B with $B \in \{i, f, c, o, d\}$, $\bar{\bar{\mathbf{s}}}$ is exponential decay parameter and $f(x) = s\log(1 + \exp(x/s))$, $s > 0$ is scaled soft plus function. As it can be seen in the Eq. (3), the parameters are updated using the hidden state $\mathbf{h}(t_n)$ at time t_n, succeeding its decay over interval $t_n - t_{n-1}$ rather previous hidden state. The memory cell $\mathbf{c}(t)$ on the interval $(t_{n-1}, t_n]$ follows power-law distribution decaying from $\mathbf{c}_{n+1}$ to $\hat{\mathbf{c}}_{n+1}$ and defined as:

$$\mathbf{c}(t) = \hat{\mathbf{c}}_{n+1} + \big(\mathbf{c}_{n+1} - \hat{\mathbf{c}}_{n+1}\big)\Big(\big(t - t_n\big)^{-\bar{\bar{\mathbf{p}}}_{n+1}}\Big) \text{ for } t \in (t_n, t_{n+1}] \tag{4}$$

The DHP, with novel discrete update of stacked LSTM state, allows the model to capture a delayed response, fits non-interacting event pairs, and copes with partially observed event streams. Mei and Eisner [34] discuss in detail these benefits of the neural version of the model. In order to ensure mathematical tractability, we have illustrated parameter updates for one of the layers of stacked LSTM, but this can be easily extended to deep architecture. For example, the hidden state $\mathbf{h}^b$ at block b in stacked LSTM is recursively computed from $b = 1:N$ and $t = 1:T$ using $\mathbf{h}_t^b = \sigma(\mathbf{W}_{h^{b-1}h^b}\mathbf{h}_t^{b-1} + \mathbf{W}_{h^bh^b}\mathbf{h}_{t-1}^b + \mathbf{b}_h)$.

2.3 Feedback loop exploration

The empirical results [16, 38] indicate the existence of feedback loop between the order flow and the shape of the LOB, together with the self- and cross-excitation effects. In order to efficiently capture this feedback effect in high-dimensional parameter space, we infuse the feedback loop exploration process into the DLSTM-SDAE architecture, as discussed in Fig. 2. In connection with designing the deep network architecture and appropriate regularisation, we take into consideration that the network can automatically explore distinct feedback loops for different types of events and their codependency. For example, the feedback effect of market buy and sell orders on price, volume and the bid-ask spread of LOB.

Consequently, we design a fully connected deep network in which each neuron represents an LOB state or feedback effect of the preceding layer, in order to automatically explore the feedback loop. Furthermore, the neurons in the same layer are partitioned into $\mathfrak{B}$ blocks to take into account different combinations of feedback loops. The corresponding regularisation is incorporated into the loss function and described by $\mathfrak{L} = \min_{\mathbf{W}_{xB}} \mathfrak{L} + \lambda_1 \sum_{B\in S} \|\mathbf{W}_{xB}\|_1 + \lambda_2 \sum_{B\in S} \sum_{\mathfrak{b}=1}^{\mathfrak{B}} \left\|\mathbf{W}_{xB,\mathfrak{b}}{}^T\right\|_{2,1}$, where $\mathfrak{L}$ is the loss function of the DLSTM, and other two terms are feedback loop regularization applied to each block in the network [61]. $\mathbf{W}_{xB} \in \mathbb{R}^{N_N \times K_J}$ is weight matrix, with number of neurons N_N and inputs dimension K_J. The S characterizes the set of gates and cell in LSTM neurons for each block in DLSTM. Lastly, the $\|\mathbf{W}\|_{2,1} = \sum_i \sqrt{\sum_j w_{i,j}^2}$ is a structural ℓ_{21} norm. The loss function was solved by using Adaptive Moment Estimation (Adam) [24]. Adam optimization is an augmentation to stochastic gradient descent that is memory efficient, extremely insensitive to hyperparameters, works with sparse gradients, is appropriate for non-stationary objectives, and learns the learning rates itself on a per-parameter basis. It is well suited for highly noisy and/or sparse-gradient order book data.

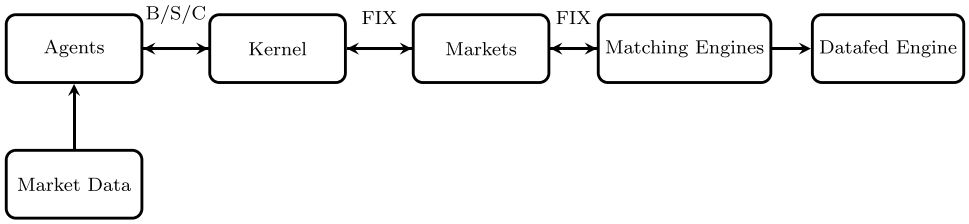


Fig. 3 The multi-asset market simulator. The simulator accounts for multiple heterogeneous agents trading in multi-asset markets. The agents trained themselves on the various market data and places buy/sell/cancellations (B/S/C) orders to the market via kernels, which use the FIX protocol for order transactions, processing and execution. The orders are matched using priority mechanism algorithms, e.g. *price-time*. All the simulated order book records including placing, transactions and execution are then stored in the datafed engine

2.4 Parameters estimation

Given a collection of sequence of order book events $\mathcal{S}$, $\{t_n, \varkappa_n, k_n\}_{n\in\mathbb{N}}$, the the log-likelihood of the model can be expressed as the sum of the log-intensities of lapsed event minus an integral of the aggregate intensities over the whole interval observed till T:

$$\mathcal{L} = \sum_{n:t_n\leq T} \log \lambda_{k_n}(t_n) - \int_{t=0}^{T} \lambda(t)dt, \tag{5}$$

The parameter (k, t) is estimated maximizing $\mathcal{L}$ using Adam [24] and Monte Carlo methods [34]. The frequently used thinning algorithm adapted from multivariate Hawkes process is then used to sample random sequence from the model.

3 Multi-agent simulation framework

In this section, we describe the important components of multi-agent-based modelling, including environment (market simulator), agent ecology (trading strategies) and reward (profit and loss), to study the behaviour of high-frequency market making agents whose strategies employ Deep Hawkes processes.

3.1 Market simulator

In this paper, we have designed a multi-asset market simulator from scratch, which is scalable to markets of substantial size. The asynchronous event-based interface is built over realistic market architecture, interface kernels, a matching engine and the FIX protocol—an open electronic communications protocol standard used to carry out trades in electronic exchanges. The market architecture consolidates the communication interface, market and matching engine. Figure 3 outlines key components of market architecture and their interaction from a high-level perspective. The agents connect to the market via a kernel that hosts order management details. This acts as a transmission channel between agents and markets, thereby providing the extreme throughput and lowest latency for order transactions. It also throttles the amount of transactions, as per the market requirement. As such, there is a guarantee of fairness between agents waiting to place orders. All the communication between kernels and market happens through FIX protocols. The markets represent an information interchange, in which heterogeneous agents communicate through kernels for order transactions, processing and execution according to the matching engine, as per the financial instruments. The markets respond to order status by sending an execution report that covers the period from start to market reset event. This provides opportunities for agents to tweak the parameters in their trading strategies after every trading period if required. At the core of the market simulator are several *matching engines* for different financial instruments. Each *matching engines* matches bids and asks to execute trades in specific instruments. The orders are matched using *price-time* priority mechanisms. In this context, among bids or asks, the matching algorithms give priority to orders with the highest or lowest price. The ties are broken by giving preference to orders with the earliest submission time compared to other orders. Our simulator uses innovative kernels and FIX protocols which brings it close to realistic market. The simulator can be scaled to any number of markets, assets class as well as any number of agents. However, in the present work,the agents trade a single asset in an equity market.

3.2 Market ecology

In this paper, we adapt the market ecology from the different strands of academic literature [25, 28, 32, 33, 42, 45, 56].

3.2.1 Deep Hawkes market makers

The order book events in the securities market are stochastically excited or impeded by a pattern in the past event

streams. The market makers are interested in learning the distribution and structure of order book events stream to accurately predict the next order (limit orders, market orders, cancellations, etc.) together with an associated labels (price, volume, etc.). Given a stream of order book events $\{t_n, \varkappa_n, k_n\}_{n\in\mathbb{N}}$, the market makers calculate the probability that the next event occurs at time t_n is of type k_n and its probability density conditioned on the history of events $\mathcal{H}_n$ by $\lambda(t)dt = \mathbb{P}\{(t_n, \varkappa_n, k_n) \mid \mathcal{H}_n, (t_n - t_{n-1})\}$ and $f_n(t) = \lambda(t)\exp\left(-\int_{t_{n-1}}^{t} \lambda(\tau)d\tau\right)$. To predict the time and the next event having minimum loss without information about the time t_n, we choose $\hat{t}_n = \int_{t_{n-1}}^{\infty} t f_n(t)dt$ and $\hat{k}_n = \arg\max_k \int_{t_{n-1}}^{\infty} \frac{\lambda_k(t)}{\lambda(t)} f_n(t)dt$. The associated intensity function for calculating the next order book events is the same as DHP as described in Eq. 1.

The deep hawkes market maker (DHMM) place orders at a specified depth relative to the mid-price, $\overline{p_t}$. At each time step t, the DHMM agent's pricing mechanism is given by $p_t^{a,b} = \overline{p_t} + \sum_{i=1}^{\overline{\delta}} i \cdot \mathbf{J}_t^{i,u} - \sum_{i=1}^{\overline{\delta}} i \cdot \mathbf{J}_t^{i,d}$, where $\overline{p_t}$ is the mid price at time t, $\mathbf{J}^{i,u}$ is the number of upward jumps with i ticks, and $\mathbf{J}^{i,d}$ is the number of downward jumps with i ticks between 0 and t, $i = 1, \ldots, \overline{\delta}$. The intensities of $\mathbf{J}^{i,u}$ and $\mathbf{J}^{i,d}$ are $\lambda_{k,u}(t)$ and $\lambda_{k,d}(t)$, respectively, $\lambda_{k,i}(t) = f_{k,i}(\mathbf{w}_{k,i}^{\top}\mathbf{h}(t)), \quad k = u, d$. The parameters of the above are calculated as discussed under model formulation in Sect. 2.2.

Most of the quantitative finance research into the high-frequency market making problem is based on the assumption of constant order size [22]. However, the empirical analyses suggests that the order sizes have striking statistical distribution at different timescales [30, 39, 48]. The limit order size follows $\mathfrak{q}$-Gamma distribution [39]. The market maker's willingness to sell or buy specified quantities of securities is defined as $q_{lt}^{a,b} = [\Gamma(\alpha, \beta)]_{q_{min}}^{q_{max}} \cdot \left(\frac{I_t \pm \bar{I}}{\bar{I}}\right)$, where I_t is inventory at time t, $\bar{I}$ maximum inventory, and $\Gamma(\alpha, \beta)$ is $\mathfrak{q}$-Gamma distribution is described as $\Gamma(\alpha, \beta; q) = \frac{1}{Z}\left(\frac{q}{\alpha}\right)^{\beta}\left[1 - (1-\mathfrak{q})\frac{q}{\alpha}\right]^{\frac{1}{1-\mathfrak{q}}}$, $Z = \int_0^{\infty}\left(\frac{q}{\alpha}\right)^{\beta}\left[1 - (1-\mathfrak{q})\frac{q}{\alpha}\right]^{\frac{1}{1-\mathfrak{q}}} dq$.

One striking feature of equity markets is the existence of short- lived limit orders that are modified or cancelled once every 50 milliseconds [11]. The limit order cancellation is an important characteristic of market making strategies that are related to expected profit, bid-ask spread and order queue position. We model cancellation sizes as follows $q_{ct}^{a,b} = \left[P_c(q;Q)\right]_{q_{min}}^{q_{max}} \cdot \left(\frac{I_t \pm \bar{I}}{\bar{I}}\right)$, where $P_c(q;Q)$ is truncated geometric distribution [30]. The LOB is represented as $[Q_{-i} : i = 1, \ldots, L]$ and $[Q_i : i = 1, \ldots, L]$ with corresponding quantities q_i. The truncated geometric distribution is defined as $P_c(q;Q) = \mathbb{P}[q|Q] = \frac{p_c^0(1-p_c^0)^{q-1}}{1-(1-p_c^0)^Q} 1_{\{q\le Q\}}$.

Finally, the market order follows a mixture of truncated geometric distribution and the dirac delta distribution [30]. The market order size that a market maker is willing to buy or sell is described as:

$$
\begin{aligned}
q_{mt}^{a,b} &= \left[P_m(q;Q)\right]_{q_{min}}^{q_{max}} \cdot \left(\frac{I_t \pm \bar{I}}{\bar{I}}\right), \\
P_m(q;Q) &= \theta_0 \frac{p_m^0(1-p_m^0)^{q-1}}{1-(1-p_m^0)^Q} 1_{\{q\le Q\}} \\
&\quad + \sum_{k=1}^{\lfloor\frac{Q-1}{5}\rfloor} \theta_k 1_{\{q=5k+1\}} + \theta_\infty 1_{\{q=Q, Q\neq 5n+1\}}.
\end{aligned} \tag{6}
$$

The parameters $\{p_c^0, p_m^0, \theta_0, \theta_k, \theta_\infty\}$ are estimated using a maximum likelihood method. The details of estimation and calibration can be retrieved from the [30]. The market orders are used to clear the unexecuted inventory at the end of trading.

3.2.2 Probabilistic market makers

To ensure fair competition with DHMM and incorporate the existing state-of-the-art, we include a probabilistic estimate-based benchmark strategy [12] adapted to our simulation framework. In this market making strategy, the agent attempts to track the fundamental price of securities by maintaining a probability density estimate of the fundamental price. The probabilistic market makers (PMM) intent to sell or buy q unit of security at time t for price $p_t^{a,b}$ in a market populated with uninformed, informed and noisy informed agents. Let us assume that the fundamental price of the security at time t is f_t, ξ be the fraction of informed agents and the probability of buy or sell orders by the uninformed agents is ζ. The noisy informed agents assumes that the price of securities follow normal distribution $p_t = f_t + \mathcal{N}_s(0, \sigma_n^2)$. Whereas the fundamental price of security evolves according to a jump process. The order book event defines the jump and prices follow normal distribution. The PMM ask and bid prices at time t are then defined as:

$$
\begin{aligned}
p_t^{a,b} &= \frac{1}{P_{Buy,Sell}} \sum_{f_t=f_{min}}^{f_t=p_t^{a,b}} \left[\left((1-\xi)\zeta + \Pr(\mathcal{N}_s(0,\sigma_n^2) \gtrsim (p_t^{a,b} - f_t)) \right) f_t \Pr(f = f_t) \right] \\
&+ \frac{1}{P_{Buy,Sell}} \sum_{f_t=p_t^{a,b}+1}^{f_t=f_{max}} \left[\left((1-\xi)\zeta + \Pr(\mathcal{N}_s(0,\sigma_n^2) \lesssim (f_t - p_t^{a,b})) \right) f_t \Pr(f = f_t) \right] \\
P_{Buy,Sell} &= \sum_{f_t=f_{min}}^{f_t=p_t^{a,b}} \left[(1-\xi)\zeta + \Pr(\mathcal{N}_s(0,\sigma_n^2) \gtrsim (p_t^{a,b} - f_t)) \right] \Pr(f = f_t) \\
&+ \sum_{f_t=p_t^{a,b}+1}^{f_t=f_{max}} \left[(1-\xi)\zeta + \Pr(\mathcal{N}_s(0,\sigma_n^2) \lesssim (f_t - p_t^{a,b})) \right] \Pr(f = f_t)
\end{aligned}
\tag{7}
$$

where $P_{Buy,Sell}$ is a priori probability of a buy or sell order and $\mathcal{N}_s(0,\sigma_n)$ is sample from normal distribution. The bids/asks equations derivation, its approximate solutions, density estimate update, and algorithm are discussed in the benchmark paper [12]. We add layers of complexity on the benchmark algorithm by allowing the PMM to sample order or cancellations size from a normal distribution. The order cancellations size at time t determined as follows $q_{t;o,c}^{a,b} = \eta_{o,c}(\bar{p}_t - \bar{p}_{t-1}) + \mathcal{N}_s(0,\sigma_{o,c}^2), \quad 0 < \eta_{o,c} < 1.$

3.2.3 Fundamental traders

Fundamental traders decide to trade based on the presumption that the securities prices will eventually return to their basic, intrinsic or fundamental value. Therefore, they strive to buy (sell) the security when the price at time t is below (above) its fundamental value. Fundamental traders are predominantly categorised as buyers or sellers, depending on the inventory at the end of a trading day. The accumulation of directional net positions is an important element in identifying buyers or sellers, since the latter acquire sizable net positions by executing numerous small-size orders, while the former only execute a couple of large orders [25, 32]. According to the agent ecology literature, the fundamental traders assume that fundamental value of a security will follow a random walk $f_t = f_{t-1}(1+\delta_f)(1+x_t), \quad \delta_f > 0; x_t \sim \mathcal{N}(0,\sigma_x^2)$. Given last mid-price at time t, the limit order price by fundamental traders is determined by $p_t = \bar{p}_{t-1}(1+\delta_f)(1+z_t), \quad z_t \sim \mathcal{N}(0,\sigma_z^2)$. Finally, the order under fundamental traders strategy are calculated as follows $q_{t;f} = \eta_f(f_t - \bar{p}_{t-1}) + \mathcal{N}_s(0,\sigma_f^2), \quad 0 < \eta_f < 1$. The decision to buy or sell is governed by following logic $\mathcal{D}_t = \begin{cases} Buy, & q_{t;f} \geq 0 \\ Sell, & q_{t;f} < 0 \end{cases}$.

3.2.4 Chartist traders

Unlike fundamental traders, the chartist or technical trader's strategy depends on predicting future price direction based on past price movement. The chartist traders in our simulation framework use a simple trend-following strategy described in [28]. The price, order size and trade direction are described as $p_t = \bar{p}_{t-1}(1+\delta_c)(1+z_t), \quad z_t \sim \mathcal{N}(0,\sigma_c^2)$, $q_{t;c} = \eta_c(\bar{p}_{t-1} - \bar{p}_{t-2}) + \mathcal{N}_s(0,\sigma_c^2), \quad 0 < \eta_c < 1$ and $\mathcal{D}_t = \begin{cases} Buy, & q_{t;c} \geq 0 \\ Sell, & q_{t;c} < 0 \end{cases}$.

3.2.5 Noise traders

In the securities market, noise traders make trading decisions based solely on non-information. In the models, they serve as an essential proxy for randomness, no trade and no speculation. We incorporate the slightly more evolved noise or background traders from a seminal paper by [59]. The noise traders ask or bid price is determined by its fundamental private valuation and trading strategy. The fundamental value evolves according to a mean-reverting stochastic process [59]. $f_t = max\left[0, \eta_n \bar{f} + \eta_n(1 - f_{t-1}) + y_t\right], \ 0 < \eta_n < 1; y_t \sim \mathcal{N}(0,\sigma_n^2)$. The private valuation for the noise traders at time t is given by $p_v = max\left[0, d_f\right], \ d_f \sim \mathcal{N}(f_t,\sigma_v^2)$. The noise trader calculates its private value and decide to buy or sell $q_{t;n}$ order sampled from a normal distribution,$\mathcal{N}(0,\sigma_n^2)$, with equal probability of 1/2.

3.3 Reward design

Unlike traditional reward design, in which an agent's performance is assessed at the end of a trading period, we calculate the agent's instantaneous rewards at each timestep t. The reward function for the agents (j) comprises profit & loss (PnL), inventory cost (IC) and transaction cost (TC). The PnL is simple profit or loss made by the agents through buying or selling security at the exchange. Its is defined as $PnL_{t;j} = q_{t;j}^a\left(p_{t;j}^a - \bar{p}_t\right) + q_{t;j}^b\left(\bar{p}_t - p_{t;j}^b\right)$. As a agent's inventory is exposed to the volatility of the market price, we incorporate it our reward design using a term associated with inventory cost. Its given by $IC_{t;j} = I_{t;j}\left(\bar{p}_t - \bar{p}_{t-1}\right)$. Finally, we consolidate a quadratic penalty on the number of shares executed to account for transaction cost. Specifically, the transaction cost for order executed q_t^e by agent j till time t is $TC_{t;j} = \gimel\left(q_{t;j}^e\right)^2, \ 0 < \gimel < 1$. The reward function is the sum of orders bought or sold plus inventory cost less a transaction cost penalty $R_{t;j} = PnL_{t;j} + IC_{t;j} - TC_{t;j}$.

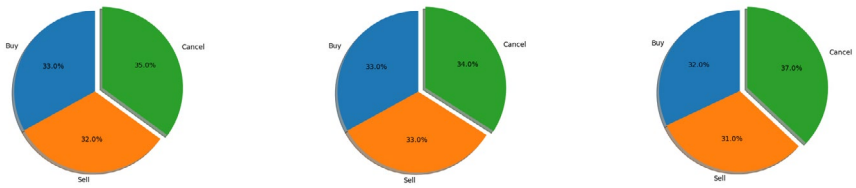
3.4 Capital allocation

The amount of currency units held by an agent is represented by capital. Prior to securities market opening in simulation

Table 1 Descriptive statistics of the orderbook data

Data	# Orderbook event token			Stream length		
	Train	Val	Test	Min	Mean	Max
Nanosecond	≈9,210,480	921,000	2,302,620	26,752	85,874	116,217
Millisecond	≈432,116	42,252	108,029	1167	3670	5216

Fig. 4 Class distribution of orderbook data



(a) Nanosecond (8 days) (b) Millisecond (8 days) (c) Millisecond (random day)

framework, every heterogeneous agents endowed with different amount of capital by a power law distribution. The agent's initial capital c_a follows a power law if it is drawn from drawn from a probability distribution $p(c_a) \propto {c_a}^{-\alpha_a}$. The α_a is referred as scaling parameter which ordinarily lies between 2 and 3 [8].

4 Experiments

In this section, we elaborate on data, its processing, performance metrics, benchmarks, training and the parameter configuration for the proposed model.

4.1 Data

We use the publicly available historical Nasdaq TotalView-ITCH 5.0 data feed sample[1] to reconstruct limit order book [21]. The reconstructed database provides tick-by-tick details of full order book depth by listing every quote and order at each price level of a specific security in Nasdaq, NYSE, and regional-listed securities on Nasdaq. The raw data feed in the binary format has a series of sequenced messages to describe the system, securities, order, and trade events at a resolution of the nanosecond scale. The event stream at nanosecond timestamp guarantees the inclusion of stochastically missing events which might increase the predictive accuracy of the Deep Hawkes model. Although the neural hawkes model [34] is expressive enough to take account of missing event stream, it makes sense to access the performance of deep hawkes model at millisecond resolution order book data as compared to nanoseconds. Nasdaq uses multiple messages to indicate the current order, trading, system, and circuit breakers event's status as discussed in technical report [43]. For mathematical tractability, we have sampled high frequency data for hundred most liquid securities over eight days from reconstructed orderbook. The extracted sample data consists of approximately a billion transaction records at nanosecond resolutions together with the possible event of limit order buy/sell, market order buy/sell, and cancellations partial/full.

The reconstructed orderbook data is divided into training, validation and test set. For a single security at nanosecond and millisecond resolution, the descriptive statistics are given in Table 1. The validation set is included to optimize the model's hyper-parameters while training, thus having control at over-fitting. To avoid high variance in the data set, we only record the average value over multiple splits denoted by ≈ in the Table 1.

The next step after sampling data into training, validation and test set is to understand the class distribution of the discussed sample. Deep learning algorithms are often found to perform poorly if the training dataset suffers from substantial class-imbalance [4]. We didn't encounter class-imbalance issues in our dataset and used classification error rate as a performance metric for predicting buy, sell, or cancel event classes. The class distributions of the training set at nanosecond and millisecond resolutions are given in the Fig. 4. It is evident from the figure that data is balanced and pose fewer challenges while making a prediction. It is worth mentioning that we have only taken eight days of order book data provided by NYSE. We refrain from making any big claim on the causality of the data. It is also unclear how the sample of eight days was collected.

[1] ftp://emi.nasdaq.com/ITCH/Nasdaq_ITCH/.

4.2 Performance metrics

DHMM agents use DHP to accurately predict Experiments order book events (buy, sell or cancel) and their timing. The accuracy of DHMM's predictions in terms of events and time is an important determinant of its trading profitability. The performance metrics are vital components for measuring the performance of the trained model's prediction reliability with observed test data. The widely used scale-dependent metric[23], root mean square error (RMSE) and classification error rate (ER) were used to evaluate the prediction performance of the Neural Hawkes model. Following [34], we predict each prevailed order book event stream $\{t_n, \varkappa_n, k_n\}$ from the past event stream $\mathcal{H}_n$ and evaluate prediction using RMSE and ER.

The predominant metric for evaluating the performance of the agents is profit and loss at the end of the trading period. However, this approach may be misleading, as agents are tested across heterogeneous securities, with varying pricing and liquidity structures. Alternatively, to efficiently capture spread, we use a normalised PnL (NPnL) with inventory and quadratic transaction costs. The NPnL is calculated every hour by dividing the total reward by the weighted average market spread. To take account of the small inventories maintained by the market maker,, [53] introduced the mean absolute position (MAP) metric. An extreme score under this metric indicates a risky speculative strategy, while a moderate one indicates a strategy based on a stagnant market. We record the variability for NPnL and MAP using the standard deviation and mean, respectively.

4.3 Benchmarks

We aim to evaluate the performance of the DHMM with a modified probabilistic estimate-based benchmark strategy [12]. The market making strategy is an extension of the classic information-based model [15], in which agents use the probability estimates of the fundamental price of securities to set bid and ask prices. The agents can sample limit, market or cancellation orders from the normal distribution or contradictory to a unit market order. We implement the probabilistic estimate-based strategy at the top of our simulation framework in continuous-time simulation rather than a discrete-time simulation. This provides the perfect test-bed to assess the performance of the simulation framework in extending the discrete-time mechanisms to continuous-time, where heterogeneous agents interact asynchronously.

The DHP extends the seminal NHP to the deep learning framework in a market making setting. We introduce the novel architecture to circumvent complications related to random weight initialisation, training and noisy order-level data [50]. Given that the NHP is the kernel of our proposed deep model, we evaluate the performance of market making agents that use the earlier model in their trading strategies. For comparison purposes, we use the same architecture and training mechanism as discussed in the seminal paper [34]. The neurally self-modulating multivariate Hawkes process also acts as benchmark model for evaluating DHP's performance on the prediction of order book events and in terms of time on the reconstructed limit order book data.

4.4 Training

The high-frequency marker making agents uses DHP to learns from reconstructed limit orderbook data to place bids or asks or cancels at suitable time. The learned prediction is then infused into the market making strategy to trade with the simulation framework. The agents learn the system parameters in a two-step process. Firstly, the preprocessed order book stream, n-th event, k_n is embedded into a latent space before passing into SDAE layer together with timing t_n. The deep network, consists of a stack of multiple DAEs, generate higher representation of convoluted order book events interaction. The high level denoised representations are then fed into DLSTM to predict the next order's type and the time to evaluate the loss. The DLSTM-SDAE learns the deep representation in two phases: pre-training and fine-tuning. In pre-training, a greedy layer-wise structure is used to train each layer of DAE iteratively, to form a three-layer SDAE. At the end of pre-training, a stack of three LSTMs is produced as an output of SDAE. Secondly, the parameter of DLSTM-SDAE is then fine-tuned to minimise the error in predicting events and time, using using stochastic gradient-descent and Adam optimisation algorithms. The early stopping methods used on the validation set's log-likelihood performance were also used on the held-out validation set to avoid overfitting. We also add isotropic Gaussian noise to augment generalisation in the performance of the events' classification. Table 2 lists the hyper-parameters tuned by validation set performance for the DLSTM-SDAE network architecture. The other non-LSTM parameters includes $s_n \in \mathbb{R}$ and $\mathbf{W}_n \in \mathbb{R}^D$ as discussed in Sect. 2. The market making agent using NHP uses single layer LSTM and the number of hidden nodes from a small set (64, 128, 256, 512, 1024) as described in the base paper [34]. The hyperparameters are optimized based on the performance of the validation set.

The high-frequency market making agents are trained in the simulation framework for 1000 trading days. Each trading day starts at 9:30 and lasts until 16:00. Two hundred trading days were used to fine tune the hyper-parameters using random search. We acknowledge the existence of differences between the real data and simulated data, but firmly believe that they are generated from the same mechanisms—a claim substantiated by agent-based models that reproduce stylised facts similar to empirical findings. Taking the above

Table 2 Parameters configuration

Description	Parameter/hyperparameter	Value
Total sessions size	T_{total}	1300 days
Training sample size	T_{train}	1000 days
Testing sample size	T_{test}	300 days
Total number of traders	a_t	$\approx 10^4$
Number of market makers	a_m	3
Number of fundamental traders	a_f	3000
Number of chartist traders	a_c	6000
Number of noise traders	a_n	4000
Initial capital	c_a	$\sim p(2.3) \times 10^4$
Min inventory	min Inv	$\sim -p(2.3) \times 10^5$
Max inventory	max Inv	$\sim p(2.3) \times 10^5$
Scale factor	s_n	1
LSTM weights	$\mathbf{W}_n$	$\sim \mathcal{N}(0, 0.01)$
DHMM limit order parameters	$\Gamma(\alpha, \beta)$	$\Gamma(0.07, 1.52)$
DHMM limit order cancellations parameters	$P_c(q;Q)$	0.60
Fraction of informed agents	ξ	0.33
Probability of buy/sell by uninformed agents	ζ	0.33
PMM order cancellation size parameter	$\eta_{o,c}$	0.04
Fundamental order size parameter	η_f	0.04
Chartist order size parameter	η_c	0.04
Noise price parameter	η_u	0.04
Transaction cost penalty	$\daleth$	0.06
Number of layers (DAE/LSTM)	N_L	3/3
Number of hidden unit per layer	N_H1024	
Learning rate for pretraining	α_{LPT}	0.05
Learning rate for fine-tuning	α_{LFT}	0.10
Number of pretraining epochs	ϵ_{PT}	100
Additive isotropic Gaussian noise	η_{noise}	$\sim \mathcal{N}(0, 0.50)$

into consideration, we train market makings agents using DHP and NHP for 100 trading days five times. The aim of this exercise is to synchronise the agents' learning over different sets of data generated from the same stochastic process. We then test the performance of the agents against the benchmark for 300 trading days. To ensure fair competition with market making agents, we use heterogeneous market ecology consisting of fundamental, chartist and random agents. The important parameters pertaining to the trading agents in the simulation framework are given in Table 2.

5 Results

In this section, we investigate the performance of market making agents in predicting types of order book events and their timestamps. Having learned which orders to send and at what time, we evaluate the agent's trading performance in the simulation framework. We tweak order cancellations to examine the impact on the agent's profitability and the microstructure of the order book. We then check the robustness of the model by performing sensitivity analysis. Finally, we validate our simulation framework by reproducing stylised facts with our simulated data.

5.1 Predictive performance

Given a stream of order book events $\{t_n, \varkappa_n, k_n\}_{n\in\mathbb{N}}$, the market makers seek to predict the next event type and its time. We evaluate predictive performance of $\hat{t}_n$ and $\hat{k}_n$ using RMSE and ER, respectively.To avoid getting entangled in the problem of overfitting, we divide the training set into the sub-training and validation sets. We train DHP and NHP models on the sub-training set so as to choose hyperparameters for validation set. Following the training procedure of [34], we generate the predictive performance of the market making agents on reconstructed order book data at nanosecond resolution in Fig. 5. As is evident from the figure, neither model is invariably better at predicting events or, in particular, time. It seems that the both models do not explicitly address the complex dynamics of asynchronous order book data at nanosecond resolution. The event dynamics at nanosecond

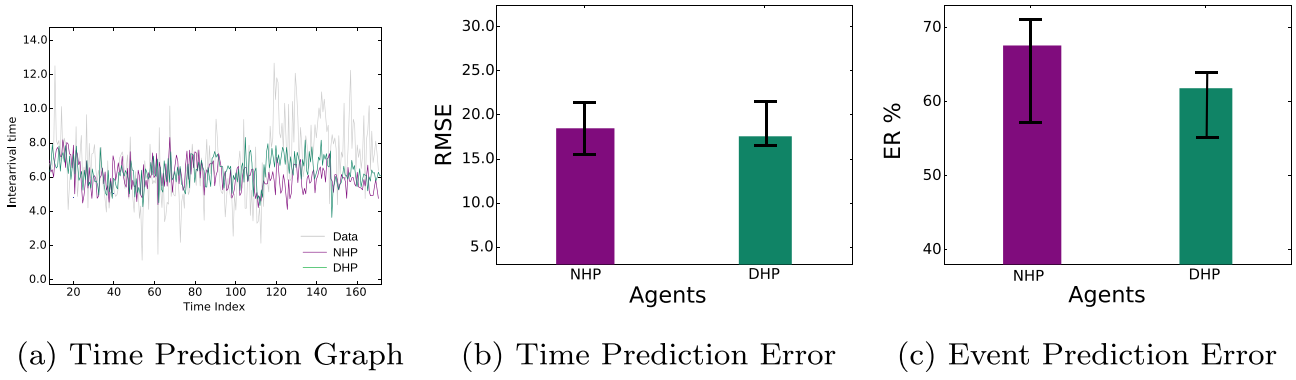


Fig. 5 Performance evaluation of high-frequency market making agents in predicting order book events and time at nanosecond resolution. The standard deviation over 10 experiments using different train-val-test sample is denoted by error bar

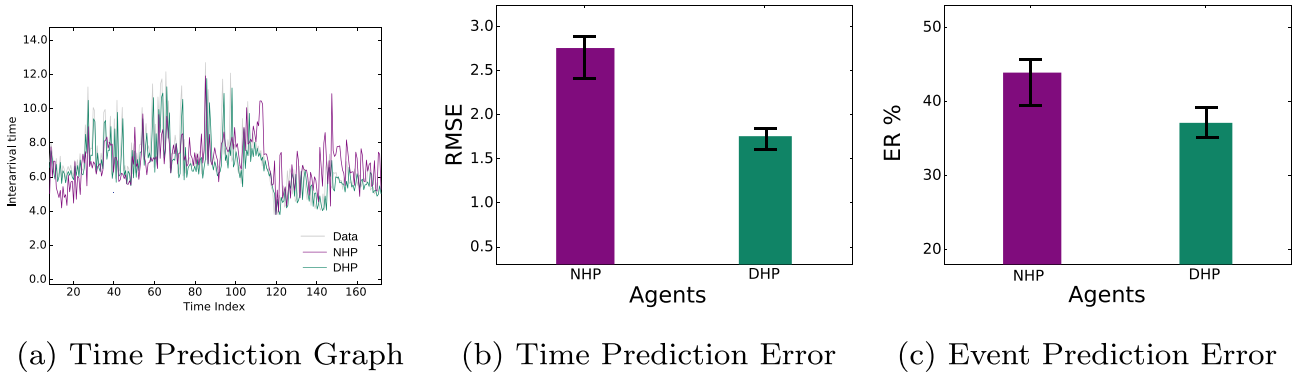


Fig. 6 Performance evaluation of high-frequency market making agents in predicting order book events and time at millisecond resolution. The standard deviation over 10 experiments using different train-val-test sample is denoted by error bar

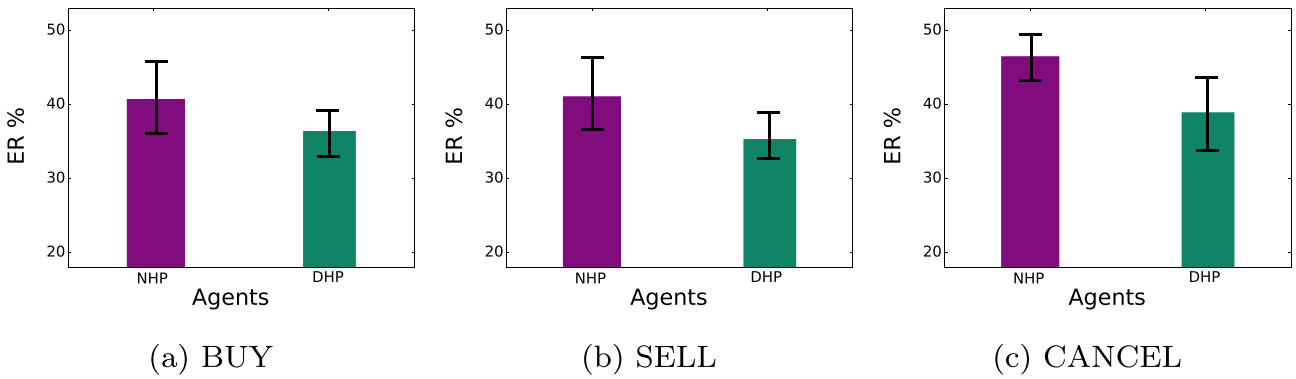


Fig. 7 Classification performance on different classes

timestamps need much more sophisticated models to filter noise and to model event interaction and non-linearity.

In Fig. 6, we evaluate the predictive performance of the high-frequency market-making agents using millisecond data sampled from the reconstructed order book. Compared to the earlier results, the DHP model's performance has drastically increased. In addition, the time prediction is consistently better than with the NHP. To further strengthen our claim, we assess the significance of DHP's predictive superiority over NHP using the Diebold-Mariano (DM) test [13]. We check whether DHP's RMSE is statistically significant over NHP's? We use a multivariate version of the test

Table 3 Mean and standard deviation on the NPnL and MAP for different market makers

Agents	NPnL [10^5]		MAP[unit]	
	Mean	Std.Dev.	Mean	Std.Dev.
DHMM	2.1	± 18.26	17	± 20
NHMM	1.1	± 4.09	4	± 6
PMM	− 1.6	± 79.55	41	± 74

using the multivariate loss differential series as discussed in the article for forecasting electricity prices [27]. The p-value of 0.002 summarizes that DHP's RMSE is significantly more accurate than NHP's RMSE.

To assess whether the high-frequency market-making agent's performance is consistent with overall classes or the other, we plot their event prediction error on the buy, sell and cancel classes in Fig. 7. The event prediction error for buy and sell are consistent with the classification performance in all classes. However, the event prediction error for the canceled class has increased. The reason for the same is attached to the stochastic nature of price which drives the order book dynamics. The order cancellations let high-frequency market-makers dynamically adjust the price in the hope of tighter spreads and better execution.

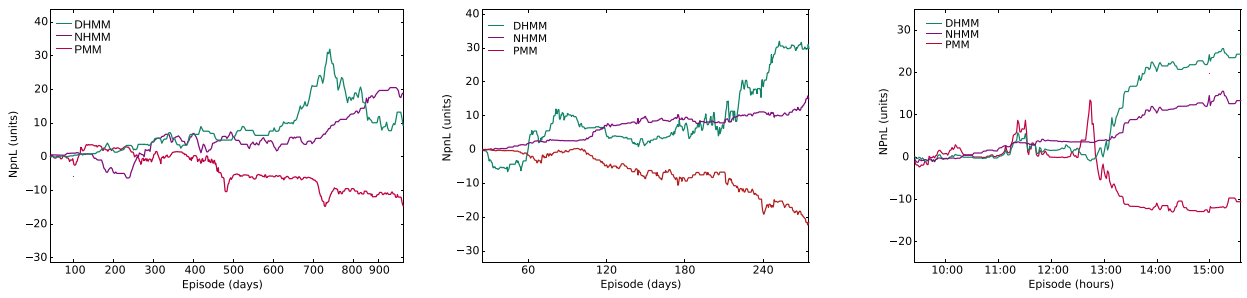
The deep model with novel architecture and pre-training module are sophisticated enough to capture excitation and feedback effects in the order book. Now the question is whether the gain in order arrival time and order type accuracy is due to the better initialization or model architecture in particular DLSTM? To do so, we compare the mean length scale in the final layer and the empirical variance of length scales across layers as a function of the network depth discussed in seminal work [18]. We find that mean square length decreases exponentially with the network depth. However, the empirical variance of length scales across layers grows approximately linearly with the network depth, confirming that the gain in classification accuracy and regression gain is due to network architecture rather than initialisation. We tested the data-dependent weight initialization technique, layer-sequential unit-variance (LSUV) initialization with our data set. The readers interested in intuition, empirical evidence and theoretical discussion of the above results should refer to the original article [18]. For brevity, we have used exported the results in our context.

The results also substantiate the claim of the NHP model regarding stochastically missing data. The order book events at millisecond timestamps theoretically omit the events at the finer time resolution, but they are generated by a different mechanisms. This is the reason why there are completely different results at the two time resolutions. The Deep Hawkes model presented here is expressive enough to learn true predictive distribution with scholastically missing events, but only if they are generated from the same mechanism. By integrating the predictive capabilities into the market making strategies, the agents trade in a simulation framework populated with heterogeneous trading strategies. In the next section, we explore the agents' trading performance.

5.2 Trading performance

The trading performance of the high-frequency market making agents is discussed in Table 3. Our simulation framework evaluates various models, including the Neural Hawkes model, the benchmark probabilistic estimate model, and our proposed Deep Hawkes model. According to the performance metrics specified in Table 3, our proposed agent using DHP (DHMM) consistently outperforms the PMM and NHMM, which suggests that the proposed trading agent benefits by learning the robust microstructure of order book data. Further, the DHP lets the agent capture the self- and cross-excitation effects of the limit order book together with a feedback loop, to place the right order at the right time. We discuss the performance of each agent in detail below.

As shown in Fig. 6, the DHMM is better at predicting the type of order and its time compared to NHMM. This is



(a) Train (b) Test (c) Random day

Fig. 8 Trading agents performance with DHMM, NHMM and PMM while training, testing and random day

Table 4 Estimated parameters for simulated orderbook data

Data	TAR (/min)	ATS (shares)	TSS (shares)	CAR (/min)	ACS (shares)	PJR (/min)	AJS (ticks)	MI	AQS (shares)
Simulated NC	2.53	3467	7664	92.72	5061	1.26	0.32	10.91	16,416
Simulated AV	2.04	4037	6901	82.21	4022	1.01	0.46	8.76	23,554
Simulated WC	5.26	6329	8083	43.71	1107	3.75	2.06	11.92	40,191
Simulated AV	4.07	5463	9147	40.31	1560	3.01	2.06	13.02	46,815

an important element of the market making strategy. The novel DLSTM-SDAE architecture allows the agents to learn the hidden representation of the noisy order book data, and therefore to place orders that add to its profitability. Furthermore, DHMM exhibits a faster convergence rate compared to NHMM, as shown in Fig. 8.

The baseline strategy used by PMM maintains a probability density estimated on the basis of fundamental price. The fundamental price evolves according to the jump process, following a normal distribution. This works in the favor of PMM, which makes more profit at the beginning, as verified in Fig. 8.Over time, however, the DHMM and NHMM learn the art of placing the right order with the right intensity. Afterwards, the profitability of the PMM fall dramatically. The PMM might perform better if it took a long position over several days, rather than trading intraday. It would be interesting to check the performance of the PMM agents with different probability density estimate conditions on the joint distribution of microstructure features.

5.3 Order cancellation effect

Massive numbers of order cancellations in a short period are a distinctive attribute of the equity market. For example, at Nasdaq Nordic, order cancellations typically account for 40% of submitted limit orders on a particular trading day. Market making strategies using limit order cancellations contribute to the market marker's profit, bid-ask spread and order queue position [11]. We study the distribution of profit, bid-ask spread and order queue position by removing the cancellations mechanism in the base simulation framework. We estimate the intrinsic value of the order relative to the queue position by applying the model developed by [37]. The agent's order queue position provides an estimate of number of orders ahead of the agent's order at a particular price. A position at the front of the queue guarantees prompt execution, higher fill rate, low latency and lower adverse selection cost. We estimate the queue position in the order book by reconstructing the limit order book from the simulated data feed.

Lets us suppose that the high-frequency market making agent places a limit order at time $t = 0$ seeking best ask price p_a which gets filled or canceled at time τ. Filling the order pays the agent p_a while cancellations pay nothing. We now describe the value of the order perceived by agents relative to the queue position as $V_t = \mathbb{E}[(p_a - p_t)\mathbb{I}_{\mathsf{FILL}} - (p - p_t)\mathbb{I}_{\mathsf{FILL}} \mid \mathcal{F}_t] = \mathsf{FP}_t(\mathsf{LSP}_t - \mathsf{ASC}_t) =$ *fill probablity*(*liquidity spread premium* − *adverse selection cost*), where $\mathsf{FP}_t \triangleq \mathbb{P}\big(\mathsf{FILL} \mid \mathcal{F}_t\big), \mathsf{LSP}_t \triangleq \big(p_a - p_t\big), \mathsf{ASC}_t \triangleq \mathbb{E}\big[(p_\tau - p_t) \mid \mathcal{F}_t, \mathsf{FILL}\big]$.

To empirically calibrate the above model, we take the same parameters used by [37]. These are exponential order size distribution, trade arrival rate (TAR), average trades size (ATS), trade size in the stan-dard lot (TSS), cancellation arrival rate (CAR), average cancellation size (ACS), price jump arrival rate (PJR), average jump size (AJS), market impact (MI) and average queue size (AQS). The trade size is identified as the limit order or market order, contrary to aggressive market orders as described by [37]. Table 4 specifies the estimated parameters for simulated data with no cancellation mechanisms (Simulated NC), without cancellation mechanisms (Simulated WC) and an average (Simulated AV) over 21 days. The paper itself provides more detail regarding the parameters, calibration and model fitting [37].

Table 4 shows that an absence of cancellation mechanisms at the high-frequency market maker's end leads to a drastic increase in the average queue size. The decrease in the cancellation rate increases the queue size, which affects the high-frequency market maker's profitability, bid-ask spread and market impact. The value of the order as a function of the queue position, bid-ask spread, and the agent's profit efficiently captures the claim illustrated by the data in Fig. 9. The wider bid-ask spread when agent's are unable to cancel the limit order in Fig. 9d has negative effect on the profitability (Fig. 9e) as compared to scenarios with cancellations (Fig. 9a, b). As stated in the above model, the value of an order that is not filled is zero. Figure 9f shows that an increase in queue length decreases the probability of execution, and therefore the value. The value of the order becomes flat, as the queue length is extremely large. Our results are consistent with the findings of [11] when investigating the determinants of order cancellations. It is difficult to infer causal relationships between cancellations and market microstructure variables based on artificially created scenarios, but this approach nonetheless it paves the way for future investigation using order level data.

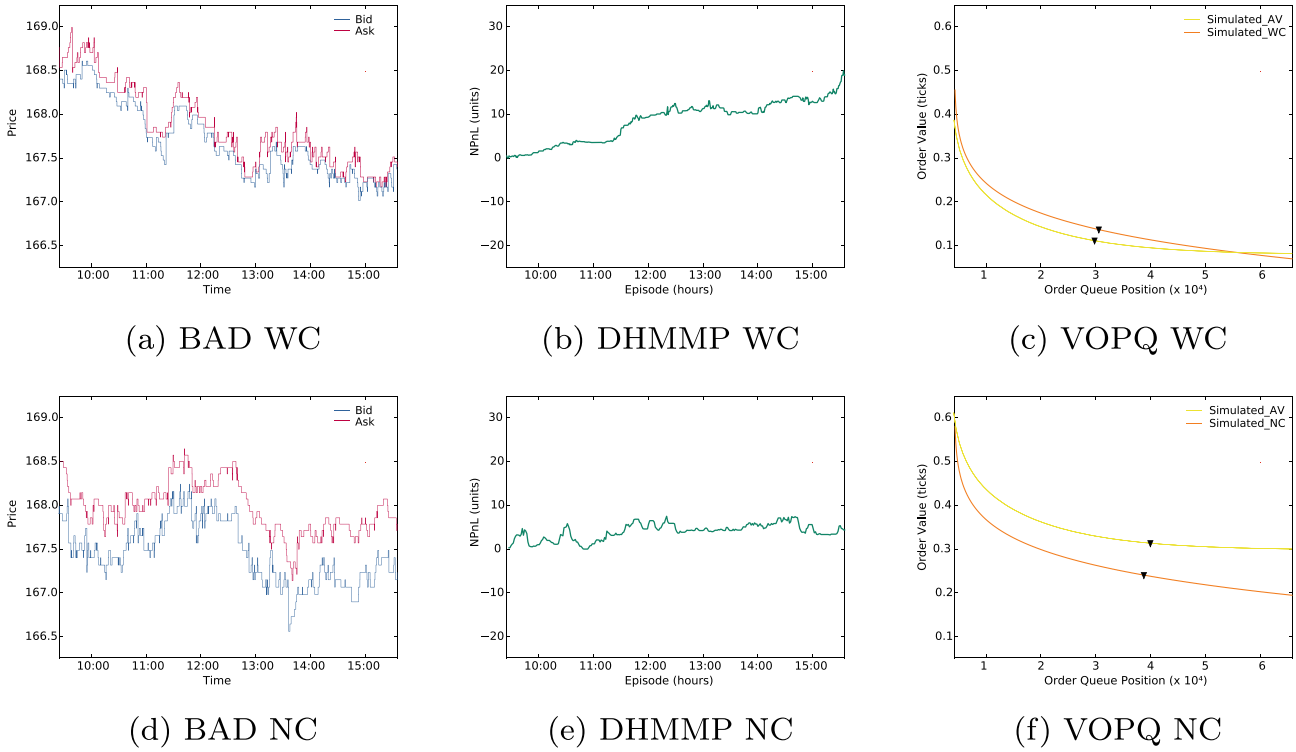


Fig. 9 Effect of limit order cancellations on the market. The top row (marked WC) represents the distribution when the market maker's agents can cancel the limit orders. The bottom row (marked NC) represents a situation with no cancellations. BAD is intraday bid-ask distribution, DHMMP is profit distribution of DHMM over the trading day, and VOPQ is the value of the orders relative to queue position. The average queue length on a particular trading day is represented by a black triangle

Table 5 Sensitivity to the number of hidden units and Gaussian noise

	# Number of layer 1	# Number of layer 2	# Number of layer 3
Number of hidden unit per layer	(64, 128, 256, 512, 1024)	(64, 128, 256, 512, 1024)	(64, 128, 256, 512, 1024)
RMSE	(4.6, 4.4, 4.1, 4.1, 4.0)	(3.5, 3.0, 3.0, 2.5, 2.5)	(2.0, 2.0, 2.0, 1.5, 1.5)
ER	(55.7, 55.7, 55.7, 50.4, 54.0)	(47.2, 44.4, 44.1, 44.1, 44.1)	(37.5, 37.0, 35.0, 35.0, 34.0)
Gaussian noise	(0.10, 0.20, 0.30, 0.40, 0.50)	(0.10, 0.20, 0.30, 0.40, 0.50)	(0.10, 0.20, 0.30, 0.40, 0.50)
RMSE	(7.2,5.9, 4.3, 5.4, 6.5)	(4.1,3.9, 3.0, 4.0, 4.6)	(2.2,2.2, 2.0, 2.1, 2.1)
ER	(60.2,55.9, 50.1, 60.0, 63.5)	(55.6,52.2, 49.1, 55.3, 67.5)	(37.2,37.9, 37.3, 37.1, 37.2)

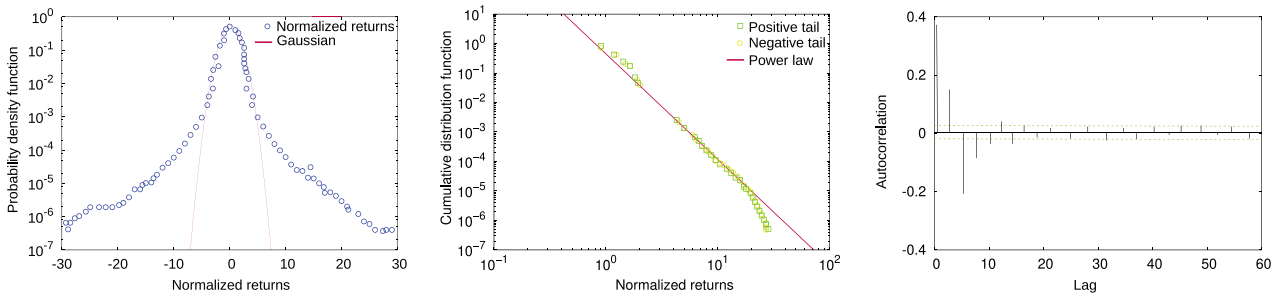
The DLSTM-SDAE used in our model has 3 DAE layers and 3 LSTM layers. In performing sensitivity analysis, we fix the 3 LSTM layers and change only the DAE layer

5.4 Sensitivity analysis

We perform sensitivity analysis on the proposed model by changing the hyperparameters—specifically, the number of layers, the number of hidden units per layer and the noise level. The aim is to confirm the robustness of the model, rather than overfitting parameters. Table 5 shows the performance of our proposed model when there is an increase in the number of layers, hidden units and noise level. The model is robust to the noise level at the optimal choice of numbers of layers, parameters and hidden units.

5.5 Validation

The validation of the trading simulation framework is performed by measuring how successfully the simulation's output exhibits persistent empirical patterns in the order book data. Such empirical patterns are common across various markets and instruments, and even timescales are often classified as"stylised facts" [9]. We present a nominal set of stylised facts, reproduced from the empirical analysis of simulated order book data, as shown in Fig. 10.



(a) PDF (b) CDF (c) ACF

Fig. 10 Stylised facts reproduced from simulated order book data. All graphs were generated on the basis of $\Delta t = 1$ ms

Let $p(t)$ be the price of a security at time t. Given a timescale Δt , we define log return at Δt as $r(t, \Delta t) = \ln p(t + \Delta t) - \ln p(t)$. The cumulative distribution (CDF)of returns is given as $F_{\Delta t}(x) = \mathbb{P}[r(t, \Delta t) \leq x]$. The derivative of the earlier gives probability density function (PDF) $F'_{\Delta t} = f_{\Delta t}$, empirically estimated for normalized simulated return, as illustrated in Fig. 10a. The cumulative distribution of return follows power law $F_{\Delta t} \sim |r|^{-\alpha}$ with $2 < \alpha < 5$. In Fig. 10b, the positive tail $F^{+}_{\Delta t}(x) = \mathbb{P}[r(t, \Delta t) \geq x]$ and the negative tail $F^{-}_{\Delta t}(x) = \mathbb{P}[r(t, \Delta t) \leq x]$ of cumulative distribution shown as yellow circles and green squares exhibit power law as denoted by the red line with $\alpha = 2.8$. In Fig. 10c, we show the absence of the autocorrelation of price change, defined as $\rho(\tau) = Corr(r(t, \Delta t), r(t + \tau, \Delta t))$. The autocorrelation function (ACF) drastically decays to zero in few lags.

6 Conclusions

We have developed a market making strategy that takes account of the feedback loop between the order arrival and the state of the LOB, with self- and cross-excitation effects, while placing an order in our realistic simulation framework. The strategy was designed by integrating the self-modulating multivariate Hawkes process with DLSTM-SDAE. The data-driven approach performed adversely in relation to predicting the next order type and its timestamp when fitted to reconstructed order book data at nanosecond resolution. When trained with millisecond resolution data, it outperforms NHP in prediction tasks and benchmark market making strategies in trading performance. We have demonstrated that extending the DHP in a market making setting accomplished better performance when validating empirical claims about the effect of cancellation on the determinants of order size.

6.1 Limitations

Applying DHP in an equity market has its own set of challenges and limitations. There are two important challenges/limitations we can think of in this context. In the current setting, an artificial high-frequency market-making, the agents can insert or suppress order book events (e.g. buy/sell, cancellations, etc) and observe the price change. The DHP is then used to infer the causal effect between actions and order book events. However, in the equity market populated with high-frequency market makers and other market participants, multiple agents with their own action space, observe each other actions or events to maximize the future trading profit or loss. This in turn distorts the learning curve of the agents to uncover the actions/events which influence the future reward. Next is the reliance of DHP based on Deep LSTM on maximum likelihood estimates which involve intractable integrals that need to be approximated. While modeling buy/sell/cancellations for stocks, there is a high probability that the models based on DHP will be able to capture long-term dependencies in the order book.

6.2 Future work

Our modelling approach is still far from inferring causality, but does pave the way exploring a range of diverse research avenues. The most important and immediate of these are listed below. Apply more advanced pre-processing, architecture, and learning algorithms to filter out ultra-noisy order book data at nanosecond timestamps. Explore an intensity-free approach for Hawkes processes with learning mechanisms other than the maximum likelihood approach. Embedding DHP within the reinforcement learning framework to learn the optimal policy. Extend the model to the deep reinforcement learning framework, in which the agent's trading action and reward from the simulator are asynchronous stochastic events characterised by marked multivariate Hawkes processes. Extract the agent's trading algorithm parameters

directly from the order book data , rather than from random seeds or empirical literature.

Funding Open access funding provided by Copenhagen Business School.

Data availability The study utilizes a readily accessible historical dataset sourced from Nasdaq TotalView-ITCH 5.0, the established Nasdaq data feed that showcases comprehensive order book depth for Nasdaq market participants. This data is obtainable for download from https://github.com/Nasdaq/data-link-python.

References

1. Ait-Sahalia Y, Saglam M (2017) High frequency market making: implications for liquidity. SSRN Electron J, Available at SSRN: https://ssrn.com/abstract=2908438
2. Avellaneda M, Stoikov S (2008) High frequency trading in a limit order book. Quant Finance 8:217–224
3. Brogaard J, Hendershott T, Riordan R (2014) High-frequency trading and price discovery. Rev Financ Stud 27(8):2267–2306
4. Cao K, Wei C, Gaidon A, Arechiga N, Ma T (2019) Learning imbalanced datasets with label-distribution-aware margin loss. In: Proceedings of the 33rd international conference on neural information processing systems, volume 32 of NIPS'19. Curran Associates, Inc., Red Hook, p 11
5. Cartea Á, Jaimungal S, Ricci J (2014) Buy low, sell high: a high frequency trading perspective. SIAM J Financ Math 5(1):415–444
6. Chakraborty T, Kearns M (2011) Market making and mean reversion. In: Proceedings of the 12th ACM conference on electronic commerce. Association for Computing Machinery, New York, pp 307–314
7. Chao Y, Yao C, Ye M (2018) Why discrete price fragments U.S. stock exchanges and disperses their fee structures. Rev Financ Stud 32(3):1068–1101
8. Clauset A, Shalizi CR, Newman MEJ (2009) Power-law distributions in empirical data. SIAM Rev 51(4):661–703
9. Cont R (2001) Empirical properties of asset returns: stylized facts and statistical issues. Quant Finance 1(2):223–236
10. Cont R, Sirignano J (2019) Universal features of price formation in financial markets: perspectives from deep learning. Quant Finance 19(9):1449–1459
11. Dahlström P, Hagströmer B, Norden LL (2018) Determinants of limit order cancellations. In: General equilibrium and disequilibrium models of financial markets eJournal, microeconomics, pp 1–57
12. Das S (2005) A learning market-maker in the Glosten–Milgrom model. Quant Finance 5(2):169–180
13. Diebold F, Mariano R (2002) Comparing predictive accuracy. J Bus Econ Stat 20(1):134–144
14. Ganesh S, Vadori N, Xu M, Zeng H, Reddy P, Veloso M (2019) Reinforcement learning for market making in a multi-agent dealer market
15. Glosten L, Milgrom P (1985) Bid, ask and transaction prices in a specialist market with heterogeneously informed traders. J Financ Econ 14(1):71–100
16. Gonzalez F, Schervish M (2017) Instantaneous order impact and high-frequency strategy optimization in limit order books. Market Microstruct Liq 03(02):1850001
17. Guéant O, Lehalle C-A, Fernandez-Tapia J (2013) Dealing with the inventory risk: a solution to the market making problem. Math Financ Econ 7:477–507
18. Hanin B, Rolnick D (2018) How to start training: The effect of initialization and architecture. In: Proceedings of the 32nd international conference on neural information processing systems, NIPS'18. Curran Associates Inc., Red Hook, pp 569–579
19. Hawkes A (2018) Hawkes processes and their applications to finance: a review. Quant Finance 18(2):193–198
20. Ho T, Stoll H (1981) Optimal dealer pricing under transactions and return uncertainty. J Financ Econ 9(1):47–73
21. Huang R, Polak T (2011) Lobster: limit order book reconstruction system. SSRN Electronic J, Available at SSRN: https://ssrn.com/abstract=1977207
22. Huang W, Lehalle C-A, Rosenbaum M (2015) Simulating and analyzing order book data: the queue-reactive model. J Am Stat Assoc 110(509):107–122
23. Hyndman R, Koehler A (2006) Another look at measures of forecast accuracy. Int J Forecast 22(4):679–688
24. Kingma D, Ba J (2015) Adam: a method for stochastic optimization. In 3rd international conference on learning representations, ICLR 2015,San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings
25. Kirilenko A, Kyle A, Samadi M, Tuzun T (2017) The flash crash: high frequency trading in an electronic market. J Finance 72(3):967–998
26. Kumar P (2020) Deep reinforcement learning for market making. In: Proceedings of the 20th international conference on autonomous agents and multiagent systems, AAMAS '20. International Foundation for Autonomous Agents and Multiagent Systems, Richland, pp 1892–1895
27. Lago J, Marcjasz G, Schutter B, Weron R (2021) Forecasting day-ahead electricity prices: a review of state-of-the-art algorithms, best practices and an open-access benchmark. Appl Energy 293:116983
28. Leal SJ, Napoletano M, Roventini A, Fagiolo G (2016) Rock around the clock: an agent-based model of low- and high-frequency trading. J Evolut Econ 26:49–76
29. Li Y, Zheng W, Zheng Z (2019) Deep robust reinforcement learning for practical algorithmic trading. IEEE Access 7:108014–108022
30. Lu X, Abergel F (2018) Order-book modeling and market making strategies. Market Microstruct Liq 4(01n02):1950003
31. Maeda I, deGraw D, Kitano M, Matsushima H, Sakaji H, Izumi K, Kato A (2020) Deep reinforcement learning in agent based financial market simulation. J Risk Financ Manag 13(4):1–17
32. Mankad S, Michailidis G (2013) Discovering the ecosystem of an electronic financial market with a dynamic machine-learning method. Algorithmic Finance 2(2):151–165
33. McGroarty F, Booth A, Gerding E, Chinthalapati VLR (2019) High frequency trading strategies, market fragility and price spikes: an agent based model perspective. Ann Oper Res 282(1):217–244
34. Mei H, Eisner J (2017) The neural Hawkes process: a neurally self-modulating multivariate point process. In: Proceedings of the 31st international conference on neural information

processing systems, NIPS'17. Curran Associates Inc., Red Hook, pp 6757–6767
35. Menkveld A (2013) High frequency trading and the new market makers. J Financ Markets 16:712–740
36. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
37. Moallemi CC, Yuan K (2017) A model for queue position valuation in a limit order book. Technical report, Columbia Business School Research Paper No, pp 17–70
38. Morariu-Patrichi M, Pakkanen MS (2022) State-dependent Hawkes processes and their application to limit order book modelling. Quant Finance 22(3):563–583
39. Mu G-H, Chen W, Kertész J, Zhou W-X (2009) Preferred numbers and the distributions of trade sizes and trading volumes in the Chinese stock market. Phys Condens Matter 68:145–152
40. Mukerji P, Chung C, Walsh T, Xiong B (2019) The impact of algorithmic trading in a simulated asset market. J Risk Financ Manag 12:68
41. Muni Toke I (2011) "Market making" in an order book model and its impact on the spread. In: Abergel F, Chakrabarti BK, Chakraborti A, Mitra M (eds) Econophysics of order-driven markets: proceedings of Econophys-Kolkata V. Springer-Milan, Milan, pp 49–64
42. Musciotto F, Marotta L, Piilo J, Mantegna R (2018) Long-term ecology of investors in a financial market. Palgrave Commun 4:92
43. Nasdaq (2020) Nasdaq totalview-itch 5.0. http://www.nasdaqtrader.com/content/technicalsupport/specifications/dataproducts/NQTVITCHSpecification.pdf. Accessed 21 July 2020
44. O'Hara M, Oldfield GS (1986) The microeconomics of market making. J Financ Quant Anal 21(4):361–376
45. Paulin J, Calinescu A, Wooldridge M (2019) Understanding flash crash contagion and systemic risk: a micro-macro agent-based approach. J Econ Dyn Control 100:200–229
46. Pledereder R, Smith D, Chunn L (2003) System with methodology for improved transmission of financial information. US 2003/0167223 A1
47. Preis T, Golke S, Paul W, Schneider J (2006) Multi-agent-based order book model of financial markets. Europhys Lett (EPL) 75(3):510–516
48. Rambaldi M, Bacry E, Lillo F (2017) The role of volume in order book dynamics: a multivariate Hawkes process analysis. Quant Finance 17(7):999–1020
49. Sagheer A, Kotb M (2019) Time series forecasting of petroleum production using deep LSTM recurrent networks. Neurocomputing 323:203–213
50. Sagheer A, Kotb M (2019) Unsupervised pre-training of a deep LSTM-based stacked autoencoder for multivariate time series forecasting problems. Sci Rep 9:19038
51. SEC (2014) Equity market structure literature review part II: high frequency trading. U.S. Securities and Exchange Commission, Staff of the Division of Trading and Markets, pp 63–103
52. Sirignano J (2019) Deep learning for limit order books. Quant Finance 19(4):549–570
53. Spooner T, Fearnley J, Savani R, Koukorinis A (2018) Market making via reinforcement learning. In: Proceedings of the 17th international conference on autonomous agents and multiagent systems, AAMAS '18. International Foundation for Autonomous Agents and Multiagent Systems, Richland, pp 434–442
54. Sun R (2019) Optimization for deep learning: theory and algorithms. CoRR. Retrieved from arxiv:1912.08957
55. Tashiro D, Matsushima H, Izumi K, Sakaji H (2019) Encoding of high-frequency order information and prediction of short-term stock price by deep learning. Quant Finance 19(9):1499–1506
56. Toth B, Eisler Z, Lillo F, Kockelkoren J, Bouchaud J, Farmer J (2012) How does the market react to your order flow? Quant Finance 12(7):1015–1024
57. Tsantekidis A, Passalis N, Tefas A, Kanniainen† J, Gabbouj M, Iosifidis A (2017) Using deep learning to detect price change indications in financial markets. In: 2017 25th European Signal Processing Conference (EUSIPCO), pp 2511–2515
58. Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol P-A (2010) Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J Mach Learn Res 11:3371–3408
59. Wah E, Wright M, Wellman M (2017) Welfare effects of market making in continuous double auctions. J Artif Int Res 59(1):613–650
60. Zhang Z, Zohren S, Roberts S (2019) DeepLOB: deep convolutional neural networks for limit order books. IEEE Trans Signal Process 67(11):3001–3012
61. Zhu W, Lan C, Xing J, Zeng W, Li Y, Shen L, Xie X (2016) Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks. In: Proceedings of the thirtieth AAAI conference on artificial intelligence, AAAI'16. AAAI Press, pp 3697–3703