Quantitative Finance

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/rquf20

Deep attentive survival analysis in limit order books: estimating fill probabilities with convolutional-transformers

Álvaro Arroyo, Álvaro Cartea, Fernando Moreno-Pino & Stefan Zohren

To cite this article: Álvaro Arroyo, Álvaro Cartea, Fernando Moreno-Pino & Stefan Zohren (2024) Deep attentive survival analysis in limit order books: estimating fill probabilities with convolutional-transformers, Quantitative Finance, 24:1, 35-57, DOI: 10.1080/14697688.2023.2286351

To link to this article: https://doi.org/10.1080/14697688.2023.2286351

Quantitative Finance, 2024
Vol. 24, No. 1, 35–57, https://doi.org/10.1080/14697688.2023.2286351

Deep attentive survival analysis in limit order books: estimating fill probabilities with convolutional-transformers

ÁLVARO ARROYO[*][†][¶], ÁLVARO CARTEA[†][‡], FERNANDO MORENO-PINO[†][§][¶] and STEFAN ZOHREN[†]

[†]Oxford-Man Institute of Quantitative Finance, University of Oxford, Oxford, UK
[‡]Mathematical Institute, University of Oxford, Oxford, UK
[§]Signal Processing and Learning Group, Universidad Carlos III de Madrid, Madrid, Spain

(*Received 6 June 2023; accepted 11 November 2023; published online 4 January 2024*)

One of the key decisions in execution strategies is the choice between a passive (liquidity providing) or an aggressive (liquidity taking) order to execute a trade in a limit order book (LOB). Essential to this choice is the fill probability of a passive limit order placed in the LOB. This paper proposes a deep learning method to estimate the filltimes of limit orders posted in different levels of the LOB. We develop a novel model for survival analysis that maps time-varying features of the LOB to the distribution of filltimes of limit orders. Our method is based on a convolutional-Transformer encoder and a monotonic neural network decoder. We use *proper scoring rules* to compare our method with other approaches in survival analysis, and perform an interpretability analysis to understand the informativeness of features used to compute fill probabilities. Our method significantly outperforms those typically used in survival analysis literature. Finally, we carry out a statistical analysis of the fill probability of orders placed in the order book (e.g. within the bid-ask spread) for assets with different queue dynamics and trading activity.

Keywords: Fill probabilities; Limit order book; Optimal execution; Market making; Order placement; Survival analysis

1. Introduction

Most electronic financial exchanges use limit order books (LOBs) to organise and clear the demand and supply of liquidity in various asset classes. LOBs offer several types of orders, where *limit orders* and *market orders* are the most common types. Market orders cross the bid-ask spread and obtain immediate execution, while limit orders can be placed at various levels of the order book, and, if executed, obtain a better price than that of a market order. The price improvement of the limit order over the market order comes at a tradeoff. Market orders are immediately executed at the best prices available in the book, while limit orders rest in the LOB until they are filled by an incoming market order or they are withdrawn from the LOB; thus, there is no guarantee that a limit order will be executed. The length of time a limit order takes to get filled is known as *time-to-fill*, which can be estimated using different methods. In this work, we use *survival analysis* to estimate the time-to-fill distribution of limit orders placed at different depths of the LOB.

We propose a deep learning method to estimate survival functions from longitudinal data. Our approach uses an encoder-decoder neural network architecture based on the Transformer model (Vaswani *et al.* 2017), whose adoption in financial applications is in its early stages and has shown significant potential in various domains (Mishev *et al.* 2020, Ding *et al.* 2020, Wallbridge 2020, Wang *et al.* 2022, Lezmi and Xu 2023), and partially monotonic neural networks (Chilinski and Silva 2020). The model uses the self-attention mechanism through the encoder to summarise the most recent events over a lookback window before a limit order is placed in the order book, and employs this latent representation of the LOB to estimate the probability of the order being filled after its submission. This self-attention mechanism employs a locally-aware representation of the time series as input, which is obtained through a convolutional network. The attention mechanism and convolutional filters

[*]Corresponding author. Email: alvaro.arroyo@univ.ox.ac.uk
[¶]These authors contributed equally to this work.

provide a more informative summary of the time series, which improves the estimate of the survival function conditioned on the most recent trades. To evaluate the performance of the estimated survival function we make use of *proper scoring rules* (Gneiting and Ranjan 2011, Avati *et al.* 2020, Rindt *et al.* 2022), which ensure we fit the true survival function and not to an incorrect proxy. This is in contrast with the common approach in the survival analysis literature where performance is typically evaluated using improper scoring rules such as time-dependant concordance. Improper scoring rules can lead to an incorrect evaluation of the model's fit because they may not faithfully reflect the accuracy of the fitted distribution; i.e. the score of the true survival distribution may be worse than that of incorrect distributions, see Rindt *et al.* (2022) for more details.

In this paper, we focus on financial applications and study LOB data from Nasdaq exchange. We use our model to predict the survival functions of orders placed at different levels of the LOB, where matching of orders is determined by price-time priority. Our results show that the proposed architecture significantly outperforms baseline off-the-shelf architectures and standard benchmarks in the survival analysis literature. These results are consistent throughout several assets with different characteristics, which speaks to the versatility of our model-free approach. We also carry out a comprehensive analysis of the fill probability of limit orders for assets with different queue dynamics and order arrival rates (of both limit and market orders). Furthermore, we provide the first statistical evaluation of the fill probability of limit orders that are placed within the bid-ask spread. Finally, we use Shapley values (Lundberg and Lee 2017) to perform an interpretability analysis and show that the model relies heavily on high-frequency information to perform the estimation, and gives less importance to slow-moving features which provide information about seasonal intraday patterns in the fill probability.

2. Literature review

Time-to-event analysis, also known as survival analysis, is widely used in various fields, including the estimation of the time-to-recovery of a patient (Laurie *et al.* 1989), clinical trials (Singh and Mukhopadhyay 2011), churn prediction (Larivière and Van den Poel 2004), and others (Ziehm and Thornton 2013, Susto *et al.* 2014, Dirick *et al.* 2017). A fundamental issue in most applications is how to relate the distribution of event times and the features (or *covariates*) describing a system. Simple parametric models, such as the *Cox proportional hazards model* (Cox 1972) and the *accelerated failure time model* (Wei 1992), are commonly used to make these connections. However, in recent years, several approaches integrate more complex deep learning models into survival analysis, particularly in the medical field. One early example of this is Faraggi and Simon (1995), who use a feed-forward neural network to extend the Cox proportional hazards model. Similar approaches are in Katzman *et al.* (2018) and Kvamme *et al.* (2019), who incorporate techniques such as dropout, which are now common in deep learning. Additionally, some popular models output a discretised survival function without imposing any assumptions on its shape or form (Lee *et al.* 2018, 2019). Subsequent work aims to improve upon these models (Wang and Sun 2022, Zhong *et al.* 2021, Hu *et al.* 2021), while Rindt *et al.* (2022) highlights the importance of using proper scoring rules in survival analysis and illustrates the shortcomings of previous approaches. Other notable works use Gaussian processes (Fernández *et al.* 2016, Alaa and van der Schaar 2017), random forests (Ishwaran *et al.* 2008), or adversarial approaches (Chapfuwa *et al.* 2018).

In quantitative finance, Cho and Nelling (2000) assume that the shape of the survival function of filltimes follows a Weibull distribution. The authors track limit orders throughout the trading day, considering cancellations as right-censoring, to determine the fill probability. If a limit order is matched against multiple liquidity taking orders, the total time-to-fill is the weighted average of the individual time-to-fills, where the size of each execution is the weight. The authors treat partial fills as fills of orders that were initially sent with a reduced volume. Lo *et al.* (2002) use the generalised gamma distribution to model the filltimes and use the accelerated failure time model to incorporate market features. The authors establish separate models for time-to-completion, time-to-first fill, and time-to-cancellation. They discuss hypothetical limit orders, first proposed by Handa and Schwartz (1996), to study limit-order execution times as the first-passage time to the limit price boundary (see Appendix 1 for more details). Cartea *et al.* (2015) and Guéant (2016) derive optimal trading strategies with exponential fill rates that do not depend on time and depend on the distance between the price level in the limit order and the midprice in the book. Maglaras *et al.* (2022) use recurrent neural networks to predict the fill probability of limit orders, which is a widespread approach in the survival analysis literature. To train their model, they use hypothetical limit orders that are placed in the book and kept at a fixed price throughout the trading day, even if the price moves unfavourably over time. Their work benchmarks its results with the AUC-ROC score, which is an improper scoring rule in survival analysis; hence the assessment of model fit may not be correct, see Section 4.

In this work, we introduce a Transformer-based architecture that outperforms previous benchmarks in terms of proper scoring rules. Furthermore, we evaluate and present several approaches to generate training data, including repositioning hypothetical limit orders and predicting the fill probability directly from the limit orders observed in the LOB.

The remainder of the paper is organised as follows. Section 3 provides an overview of limit order books. Section 4 introduces the survival analysis and discusses proper scoring rules. Next, Section 5 presents a statistical analysis of the fill probabilities. Section 6 presents our neural network model, and Section 7 presents our results and an interpretability analysis.

3. Limit order books

The two most important order types are *market* and *limit* orders. A *market* order is used to buy or sell a given quantity at the best available price. A *limit* order, on the other

Table 1. Example of message data from the LOB. Time is measured as seconds from midnight. Price is dollar price times 10 000.

Time	Event Type	Order ID	Size	Price	Direction
34200.000841	1	24 974 777	100	1 381 900	1
34200.000841	1	24 974 809	1447	1 383 100	−1
34200.003940	1	24 978 469	100	1 381 900	1
34200.010366	1	24 986 889	100	1 381 900	1
34200.023144	1	25 002 805	100	1 381 800	1

Note: First five messages on 3 October 2022 for AAPL ticker.

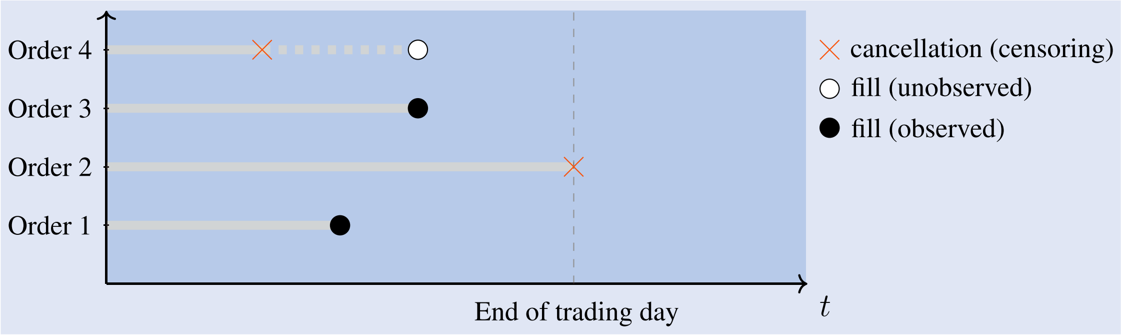


Figure 1. Events that can occur after the submission of a limit order.

hand, is an instruction to buy or sell a given quantity at a given price. Market orders guarantee immediate execution, given sufficient liquidity, while limit orders remain in the order book until they are filled or cancelled. The list of pending limit orders is stored in the LOB until they are matched or cancelled. Here, whenever we refer to market orders, we assume that these are either *fill-or-kill* (FoK) market orders, which are to be executed immediately in their entirety (within a predetermined price range) or cancelled entirely, or *immediate-or-cancel* (IoC) market orders, which are to be executed immediately (entirely or partially) at a predetermined price range (i.e. without walking the book). Further, we assume that limit orders are *good for day* (DAY) limit orders, which expire at the end of the trading day if not filled or when they are cancelled.

The matching engine of most exchanges follows a price-time priority rule where orders are first ranked according to their price and then ranked according to the time they arrived into the exchange; with earlier orders given priority at the top of the price-level queue. The LOB consists of two sides; the ask side containing all the sell limit orders and the bid side containing all the buy limit orders.

A snapshot of the LOB at time m is described by the vector

$$x_m = \left\{p_a^l(m), v_a^l(m), p_b^l(m), v_b^l(m)\right\}_{l=1}^{L},$$

where $p_a^l(m), v_a^l(m), p_b^l(m), v_b^l(m)$ denote the ask price, ask volume, bid price, and bid volume for price level $l \in \{1, \ldots, L\}$ at time m, which is typically measured in microseconds after midnight. Here, L denotes the price level cuttoff. The matrix $\mathbf{x} \in \mathbb{R}^{T\times 4L}$ contains the discrete-time dynamics of the LOB from time m to $m - M$. In the remainder of our work, we use Lobster message data, which provides information on events that update the state of the order book in the Nasdaq stock exchange. For instance, messages to post or cancel orders, to provide the direction (buy or sell), and volume of orders. As an example, Table 1 shows the first five messages sent to the book of AAPL on 1 October 2022.

4. Survival analysis

The *event time* $T_l \in \mathbb{R}_{\geq 0}$ is a positive valued random variable that describes the filltime of a limit order placed at level l of the order book. Our objective is to predict event times by conditioning on a set $\mathbf{x} \in \mathbb{R}^p$ of market features. All events are subject to right-censoring, i.e. the filltime may not be observed. When a limit order is cancelled or reaches the end of the trading day without being filled, it is considered a censored event. We consider a set of N observations of the triplet $(\mathbf{x}_i, z_i, \delta_i)$, where $\delta_i = \mathbb{1}\{z_i = t_i\}$ is an indicator function that is equal to zero if the event is censored and one otherwise, where t_i is the observation of the filltime and z_i and $\mathbf{x}_i$ are the observed event timeand the observed market features up to the instant of order submission, respectively. We use the triplet to estimate the *survival function* $S_{T_l}(t \mid \mathbf{x}) = \mathbb{P}\{T_l > t \mid \mathbf{x}\}$ (Figure 1).†

This survival function computes the probability that a limit order posted at level l will not be filled before time t. The link between the survival function $S_{T_l}(t \mid \mathbf{x})$ (which represents the probability of the order not being filled before a particular time) and the cumulative density function (CDF), $F_{T_l}(t \mid \mathbf{x})$, of the event time is given by $S_{T_l}(t \mid \mathbf{x}) = 1 - F_{T_l}(t \mid \mathbf{x})$. Thus, the density function is $f_{T_l}(t \mid \mathbf{x}) = -\frac{d}{dt} S_{T_l}(t \mid \mathbf{x})$, which describes the probability of an order being filled in a particular amount of time after submission. Further, the *hazard rate*, which indicates the propensity that an order will be filled after time t, is

† This is an instance of *survival regression*, which is equivalent to survival analysis but conditioning on a set of features $\mathbf{x}$.

given by

$$h_{T_l}(t\,|\,\mathbf{x}) = \frac{f_{T_l}(t\,|\,\mathbf{x})}{1 - F_{T_l}(t\,|\,\mathbf{x})}.$$

It suffices to obtain one of the previous functions to derive the remaining three:

$$\begin{aligned} S_{T_l}(t\,|\,\mathbf{x}) &= \mathbb{P}\{T_l > t\,|\,\mathbf{x}\} = 1 - F_{T_l}(t\,|\,\mathbf{x}) \\ &= \exp\left(-\int_0^t h_{T_l}(s)\,\mathrm{d}s\right). \end{aligned}$$

The shape of survival functions is described by a vector of parameters $\boldsymbol{\theta} \in \mathbb{R}$. One may assume that the survival function is given by a tractable distribution (e.g. a Weibull distribution) and estimate the relevant parameters with standard methods. However, there is a trade-off between mathematical tractability and goodness of fit to the data. An alternative is to use neural networks to estimate the survival function, which increases the number of parameters substantially but improves the fit to the data. Here, we use the latter approach to model the survival function of limit orders because its shape is expected to have a non-linear relationship with market features. We perform *maximum likelihood estimation* to obtain the vector of parameters that best fit to the data. Specifically, we train our deep learning model to maximise the *right censored log-likelihood function*

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) = \log(L_N(\boldsymbol{\theta})) = \sum_{k=1}^{N} \delta_k \log(\hat{f}(z_k\,|\,\mathbf{x}_k,\boldsymbol{\theta})) \\ + (1-\delta_k)\log(\hat{S}(z_k\,|\,\mathbf{x}_k,\boldsymbol{\theta})), \qquad (1) \end{aligned}$$

where $\hat{S}$ and $\hat{f}$ are the neural network estimates of the survival function and the density function, respectively.† See Appendix 2 for a derivation of (1). Training on right-censored log-likelihood requires a model to output $\hat{S}(z_k\,|\,\mathbf{x}_k,\boldsymbol{\theta})$ at the exact time-instant z_k. This is challenging in deep learning models that use the softmax activation function to discretise the survival function, because these models require an interpolation scheme to train tractably on (1). Our model avoids this problem by inputting the observed time z_k only to the monotonically restricted decoder, together with the generated latent representation from the LOB time series. This allows us to generate an arbitrarily small time grid over which to evaluate the survival function at no additional parameter cost, as well as respect the monotonicity of the survival function; see Section 6 for more details.

To evaluate the quality of model fit to the survival function, we use the concept of *scoring rule*, see Rindt *et al.* (2022). A scoring rule $\mathcal{S}$ takes as input a distribution S over a set $\mathcal{Y}$, with an observed sample $y \in \mathcal{Y}$, and returns a score $\mathcal{S}(S, y)$. With positive scoring rules, higher scores indicate an improvement in model fit. In survival regression, a scoring rule $\mathcal{S}$ is *proper* if

$$\mathbb{E}_{(t,c)\sim(T,C))}[\mathcal{S}(S(t\,|\,\mathbf{x}),(z,\delta))] \geq \mathbb{E}_{(t,c)\sim(T,C))}[\mathcal{S}(\hat{S}(t\,|\,\mathbf{x}),(z,\delta))]$$

for all survival function estimates $\hat{S}(t\,|\,x)$. Here, T and C are the filltime and censoring random variables respectively, and $Z = \min(T, C)$.‡ This means that in expectation, a proper scoring rule will give higher scores to the true survival function over any other estimate. The most commonly used scoring rules in the literature are improper, see Appendix 3 for more details. On the other hand, Rindt *et al.* (2022) show that right-censored log-likelihood (RCLL) is a proper scoring rule. Throughout our analysis, we use RCLL as the scoring rule to evaluate the precision with which we fit the true survival function, and evaluate the performance of the proposed model. This guarantees that we evaluate model estimates to fit the true underlying survival function and not an incorrect proxy.

5. Empirical and statistical evidence of fill rate executions

We use the messages sent to the LOB to obtain filltime data of limit orders. In this section, we present two approaches to compute this data. The first tracks the outcome of limit orders placed in the LOB, and the second uses hypothetical orders to model limit order repegging. We compare the resulting survival functions associated to different levels of the order book (for the first approach) and the top level for the second approach. We also analyse the changes in the survival functions when orders are placed inside the spread for assets with different queue behaviour and order arrival speed.

5.1. Generation of training data

One way to obtain the dataset of triplets $\{(\mathbf{x}_i, z_i, \delta_i)\}_{i=1}^{N}$ discussed in the previous section is to track all the messages associated to a particular order after its submission. If the final message corresponds to an execution, then the order is recorded as filled, otherwise it is recorded as censored. Thus, we do not consider partial fills because they represent a negligible subset of the trading data. The time-to-fill is the time elapsed between order submission and the time the final message is observed.

Another approach is to use 'hypothetical' limit orders, which are orders of one share in volume placed last in the queue (i.e. following regular price-time priority) of a level in the order book. In particular, we consider the survival function of the order that follows the price of a particular level. We refer to this as *pegging* the limit order to that price level. Such an approach probes a set of 'fill conditions', see Appendix 1 for details, every time the order book is updated to determine if an order has been filled. We assume that hypothetical limit orders do not have market impact, which is consistent with the work of Handa and Schwartz (1996) and Maglaras *et al.* (2022). With hypothetical limit orders, we model different behaviour to what is observed in the order book data *ex post*.

Pegged orders resting outside the best level in the LOB result in censored data because very few aggressive orders walk the LOB beyond the best quotes. Therefore, we focus on estimating the fill probability of orders pegged to the best bid

† Censoring aids in the estimation because it provides information on the order not being filled before it was cancelled.

‡ Here, we drop the subscript l for clarity.

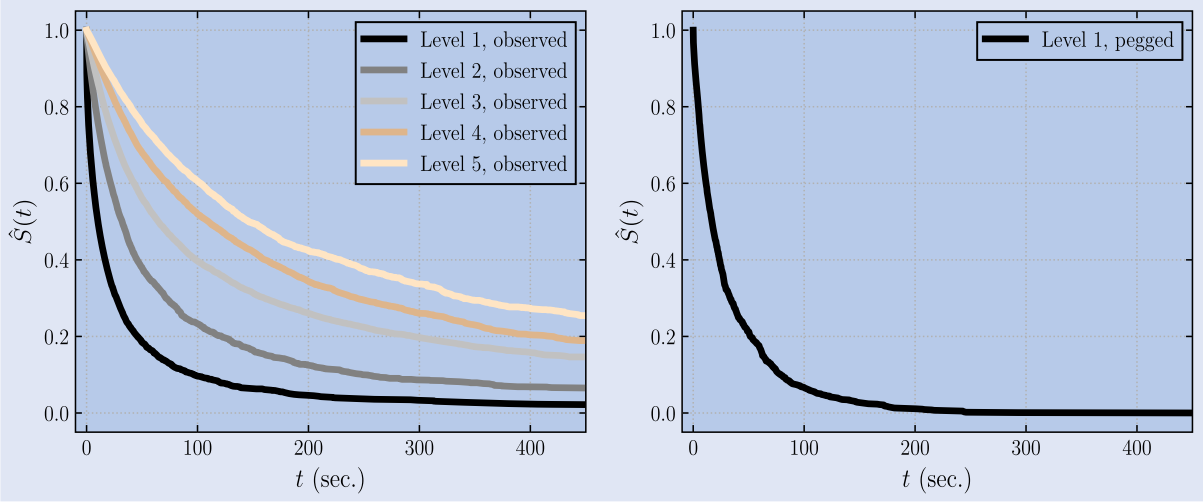


Figure 2. **Left:** Kaplan–Meier estimates of orders placed at different levels of the AAPL LOB (1 October 2022 to 27 December 2022). **Right:** Kaplan–Meier estimate of hypothetical order pegged to the first level in the LOB of AAPL, during the same period.

Table 2. Statistics of small and large tick stocks on October 2022.

	Avg. spread (ticks)	Avg. volume best ask	Avg. volume best bid	Avg. midprice	Avg. executed trades/min.
AAPL	1.44	487.19	442.03	144.34	405.02
AMZN	1.91	298.41	307.72	114.80	319.44
BIDU	8.75	138.98	97.24	101.89	20.86
COST	30.89	67.80	60.89	478.21	44.27
DELL	1.77	287.02	283.92	35.97	14.48
GOOG	1.66	299.26	232.08	99.59	120.84
MSFT	2.88	159.17	159.14	236.89	204.86
CSCO	1.15	2212.01	2193.81	41.90	74.87
INTC	1.13	5995.54	5533.79	26.55	110.88

Note: The first seven stocks are small tick, and the remaining two are large tick.

and to the best ask of the LOB.† Furthermore, we select the orders to track randomly throughout the trading day. The survival functions are visualised with Kaplan–Meier estimates in Figure 2, see Appendix 3 for more details. This shows an example of the decrease in fill probability over time for different levels, as well as the increase in fill probability for order repegging. However, this visualisation only provides an averaged view of the survival function and cannot capture the effect of microstructural features on the fill probability of the orders; this motivates our deep learning approach, which we present in the next section.

5.2. *Fill statisics of limit orders placed inside the bid-ask spread*

Here, we compute the fill probabilities of orders placed within the spread of the order book. We consider nine stocks, some of which trade with a small or large tick and also vary in their average trading activity. Table 2 shows the average spread, average volume on the best bid and best ask, and average trades per minute over the month of October 2022.‡

Intuitively, agents trading large tick stocks are incentivised to place orders inside the bid-ask spread (whenever this is possible) not to place their LOs at the end of the queue. The cost of executing this trade is almost as much as crossing the spread, and there is no guarantee of immediate execution. However, it is approximately 3 to 6 times more likely that an order is filled when it improves the best quotes and reduces the time to fill by a similar proportion, see Figure 3 and Table 3. This is expected because traders who observe orders that improve the best quotes will aim to quickly match them with a liquidity taking order before they are cancelled.§ Figure 3 shows that the fill probability of orders placed in the spread of large tick stocks, when this is possible, rises sharply for small time horizons before eventually decreasing to levels comparable to those of orders resting at the best level in the asset's limit order book. This suggests that if an order is executed within a few seconds of being placed into the book,

† For simplicity, we also focus on the best levels with tracked orders, as they also contain a large amount of censoring which results in higher computational cost to obtain data to train the model.

‡ Large tick stocks have a bid-ask spread that is on average close to one tick in size, while for small tick stocks the spread is several times larger than the tick. There is a significant difference in the dynamics between the two types, see Eisler *et al.* (2012) for more details. This is a consequence of the ratio between the price and the tick size of the exchange. When this ratio is low, traders tend to be more reluctant to cross the bid-ask spread, which results in the formation of large queues at the best bid and best ask. In our case, we consider a large tick stock to have an average spread of fewer than 1.3 ticks, as detailed in Bińkowski and Lehalle (2022).

§ This analysis is based on tracked orders only; thus, the problems arising from censoring and information bias affect this analysis.

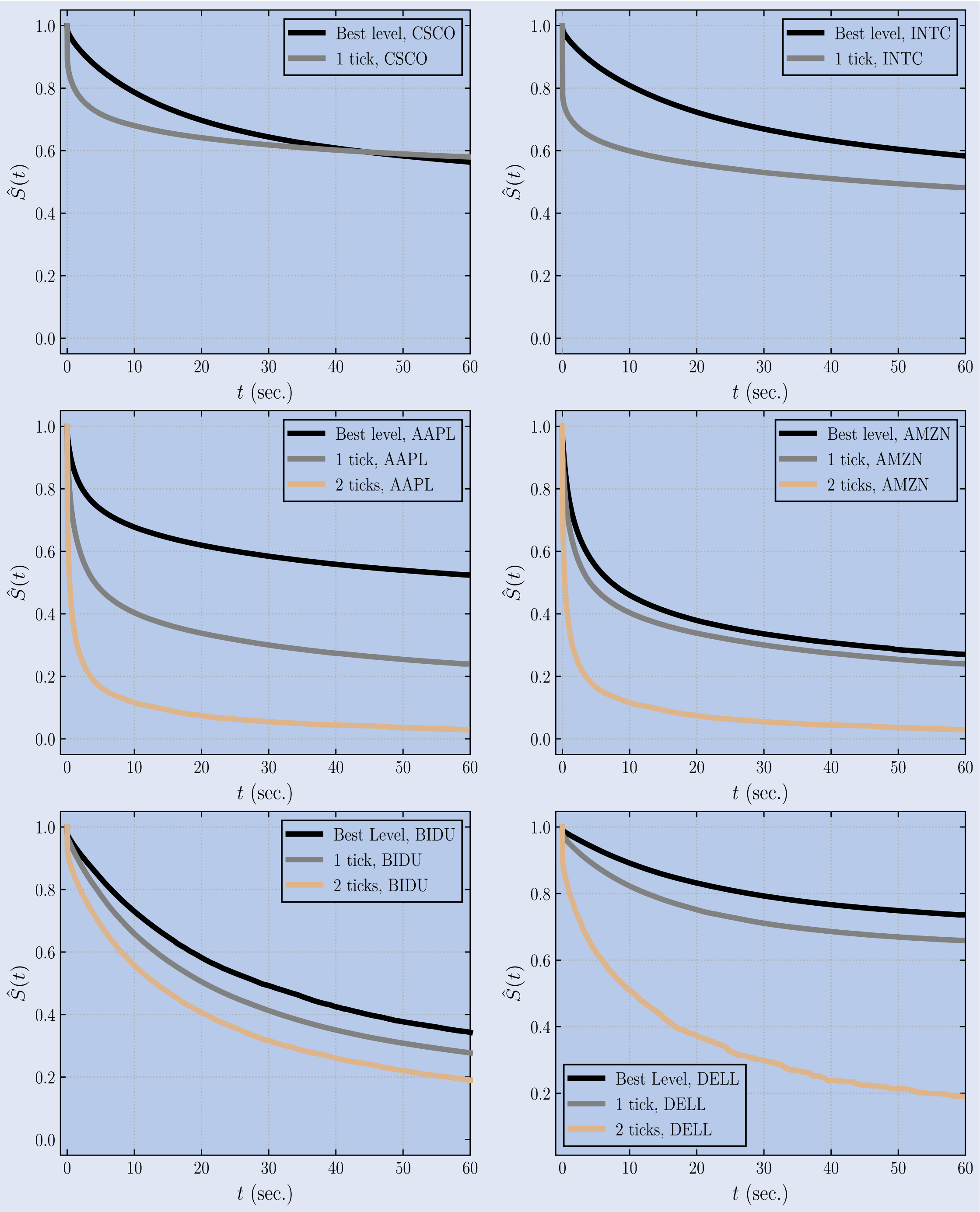


Figure 3. Kaplan-Meier estimates of survival functions when placing orders at different depths of the bid-ask spread. **Top left:** CSCO, **Top left:** INTC, **Middle left:** AAPL, **Middle right:** AMZN, **Bottom left:** BIDU, **Bottom right:** DELL.

the price moved in the favourable direction and there is a fill because the order is at the beginning of the queue. If this is not the case, it is likely that the price moved adversely, making the order rest deeper in its side of the book. Further, the survival function of stocks with larger trading activity exhibit a quicker decay than those of less active stocks.

The activity inside the spread of small tick stocks is several orders of magnitude bigger than that of large tick stocks, see Table 4. Everything else being equal, this shows how improbable it is for spreads to widen in large tick stocks. Obtaining improvements in the fill probabilities for small tick assets requires placing limit orders closer to the other side of the book, where trading activity is comparable to that of LOs placed one tick closer to the other side in large tick stocks.

6. Monotonic encoder-decoder convolutional-transformer

6.1. General architecture

In this section, we present our encoder-decoder architecture which learns the mapping between states of the LOB and

Table 3. Fill statistics at the best level and at different depths in the spread between 3 October and 27 December 2022.

	Fill probability						**Avg. Filltime (s.)**					
Depth	**Best**	**1 Tick Inside**	**2 Ticks Inside**	**3 Ticks Inside**	**4 Ticks Inside**	**5 Ticks Inside**	**Best Inside**	**1 Tick Inside**	**2 Ticks Inside**	**3 Ticks Inside**	**4 Ticks Inside**	**5 Ticks Inside**
AAPL	0.0539	0.1317	0.3601	0.4165	0.4382	0.4431	1.32	0.64	0.19	0.11	0.14	0.08
AMZN	0.0923	0.1312	0.3360	0.4139	0.4343	0.4485	1.07	0.70	0.38	0.25	0.11	0.21
BIDU	0.0948	0.0528	0.1533	0.1692	0.1868	0.2140	4.40	3.82	3.93	3.59	3.26	3.07
COST	0.0458	0.0122	0.0826	0.0898	0.1061	0.1135	5.18	4.85	6.03	5.33	4.69	4.53
CSCO	0.0573	0.1753	0.3549	0.3579	0.2484	0.4558	6.26	1.71	0.47	0.51	0.25	0.36
DELL	0.0392	0.0620	0.2263	0.2762	0.2902	0.2995	6.87	4.46	3.39	2.07	2.34	1.5
GOOG	0.0618	0.1088	0.3199	0.3969	0.4450	0.5012	1.82	0.85	0.84	0.13	0.15	0.07
INTC	0.0570	0.2962	0.3215	0.2463	0.2474	0.1165	7.33	1.95	0.27	0.29	0.8	0.71
MSFT	0.0679	0.0734	0.2632	0.3690	0.4292	0.4466	0.99	0.70	0.45	0.24	0.17	0.12

Table 4. Number of LOs placed inside the spread considered between 3 October 2022 and 27 December 2022.

Depth	**1 Tick Inside**	**2 Ticks Inside**	**3 Ticks Inside**	**4 Ticks Inside**	**5 Ticks Inside**
AAPL	51 096	17 862	11 692	4059	2004
AMZN	6 215 869	169 024	24 456	6469	2470
BIDU	1 250 831	163 567	100 719	63 407	41 017
COST	2 594 091	113 709	87 820	69 689	61 457
CSCO	466 645	2 817	841	318	136
DELL	753 603	15 584	2548	937	454
GOOG	1 548 080	52 565	8344	2274	782
INTC	317 834	992	276	97	103
MSFT	9 654 578	492 218	107 990	33 885	13 710

distribution of limit order filltimes. Figure 4 illustrates the two components of our framework. The encoder, parameterised by $\mathbf{\Phi} \in \mathbb{R}^{m_\Phi}$, processes the LOB data and obtains a latent representation from it, which is used by the decoder, parameterised by $\mathbf{\Psi} \in \mathbb{R}^{m_\Psi+}$, to predict the survival function of the limit orders. The decoder comprises a monotonic neural network that guarantees a monotonically decreasing survival function. Further, we use a convolutional-Transformer encoder to model the complex dependencies and interactions within the LOB data and to compress useful information into a lower-dimensional representation, subsequently used by the monotonic-decoder.

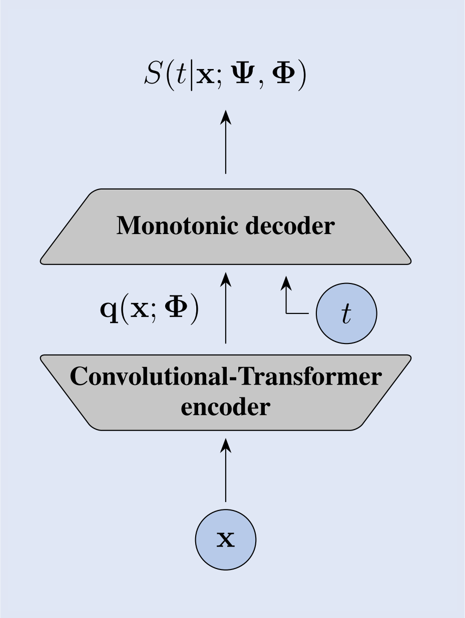

$S(t|\mathbf{x};\mathbf{\Psi},\mathbf{\Phi})$
Monotonic decoder
$\mathbf{q}(\mathbf{x};\mathbf{\Phi})$
t
Convolutional-Transformer encoder
$\mathbf{x}$

Figure 4. Encoder-decoder architecture to estimate the survival function. The first block, an encoder with parameter $\mathbf{\Phi} \in \mathbb{R}$, uses an attention-based mechanism to project the LOB observations to a latent representation that captures relevant information. The second block, a monotonic decoder takes as input both this latent representation of the time series and the response variable of the survival function, t. The weights of the decoder are positive to enforce a monotonically decreasing survival function.

6.2. *Convolutional-Transformer encoder*

We propose a convolutional-Transformer encoder to identify patterns in the LOB and to obtain accurate estimates of the fill probabilities of limit orders. The architecture of the encoder, see Figure 5, processes the LOB time series data and captures its non-Markovian dynamics through a latent representation of the time series. This representation encapsulates the most relevant information which is used by the decoder to predict the fill probabilities.

The encoder consists of two components: a locally-aware convolutional network and a Transformer model. The locally-aware convolutional network consists of three different Dilated Causal Convolutional (DCC) neural networks (Oord *et al.* 2016) that process the LOB data and generate the corresponding queries, keys, and values which serve as inputs to the Transformer model. As shown in (2), these DCCs, which are based on Convolutional Neural Networks (CNNs) (LeCun *et al.* 1989), use inner product operations based on entries that are a fixed number of steps apart from each other, contrary to CNNs and Causal-CNNs, which operate with consecutive entries. Further, causal convolutions ensure that the current position does not use future information. Previously, DCCs have been successfully applied in time series forecasting (Borovykh *et al.* 2017, Moreno-Pino and Zohren 2022).

The queries, keys, and values created by the DCCs are three different representations resulting from the convolution operation on the LOB input features and are collectively

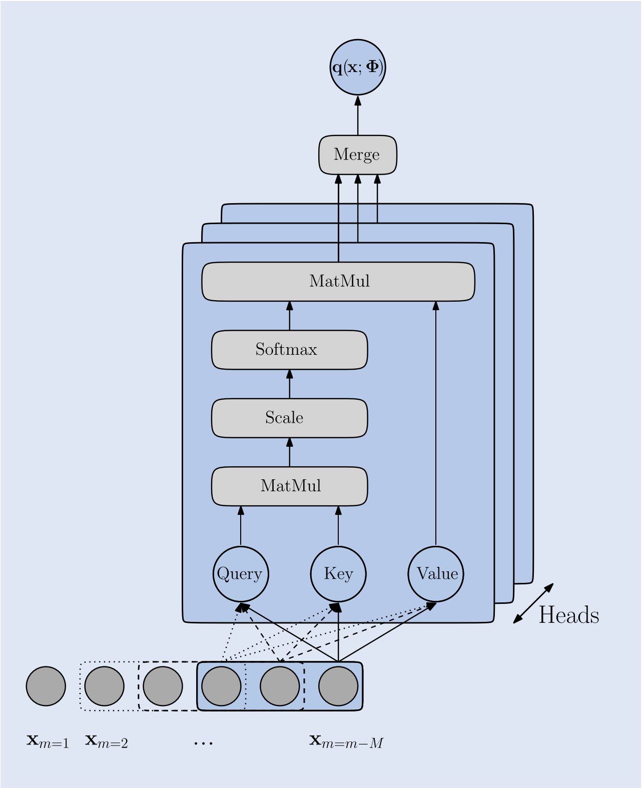


Figure 5. Structure of the convolutional-Transformer's encoder. In this diagram, the convolutional kernel is of size $s = 3$ with dilation factor $p = 1$. The locally-aware hidden representation obtained by the CNN is used to feed the Transformer, which uses self-attention over these hidden variables to obtain the latent representation $\mathbf{q}(\mathbf{x}; \Phi)$.

used by each head of the Transformer model to perform the self-attention operation and capture dependencies between different parts of the original time series. Using convolutional networks to generate the input features to the Transformer allows our encoder to be more aware of local context, granting the Transformer the ability to discern if observed values are anomalies, part of patterns, etc. This constitutes a clear advantage over using a multi-layer perceptron (MLP) to obtain the queries, keys, and values, which is the most common approach in the literature. Therefore, the operation of the DCCs can be understood as a set of data-driven local filters. The projection they perform from the original LOB time series to a hidden representation enhances the depiction of the LOB dynamics. This operation enables the Transformer's self-attention mechanism to capture complex local dependencies between datapoints, because it operates on a locally aware hidden representation, rather than point-wise values that lack local context. Additionally, the convolutional-Transformer optimises the parameters of each of the three DCCs to extract different relevant features from the LOB data. For example, some filters may be optimised to detect trends, while others may identify anomalies or changepoints. Each of the three convolutional neural networks used to obtain the corresponding query, key, and values, that serve as input to the Transformer model (Vaswani *et al.* 2017), consists of only one layer performing a causal convolutional operation between the input LOB sequence, $\mathbf{x} \in \mathbb{R}^{T\times 4L+5}$, and the corresponding convolutional kernel $\mathbf{k}$ of size $s \in \mathbb{Z}$:

$$\begin{cases} Q_i(m) = \left(\mathbf{x} *_p \mathbf{k}^Q\right)(m) = \sum_{\tau=0}^{s-1} k_\tau^Q \cdot x_{m-p\cdot\tau}, \\ K_i(m) = \left(\mathbf{x} *_p \mathbf{k}^K\right)(m) = \sum_{\tau=0}^{s-1} k_\tau^K \cdot x_{m-p\cdot\tau}, \\ V_i(m) = \left(\mathbf{x} *_p \mathbf{k}^V\right)(m) = \sum_{\tau=0}^{s-1} k_\tau^V \cdot x_{m-p\cdot\tau}, \end{cases} \tag{2}$$

where p is the dilation factor, and the dimension of $Q_i(m)$, $K_i(m)$, $V_i(m)$ is $\mathbb{R}^{T\times d_k}$, where d_k can be adjusted via the number of output channels of the convolutional operation performed by the DCC.†

We use the convolutional network solely as a feature extractor that incorporates local context into the Transformer's self-attention mechanism because, in our model, the Transformer model is only responsible for extracting the patterns within the data. Therefore, we restrict the encoder's convolutional network to a single layer, but its complexity can be easily extended to L convolutional layers, see Appendix 4.

After the CNNs extract the relevant features from the LOB data and produce the queries, keys, and values, these are fed to the Transformer model. Transformer models were initially introduced for Natural Language Processing (NLP), but they have been widely applied in time series-related problems (Moreno-Pino *et al.* 2023). These models propose a new architecture that leverages the attention mechanism (Bahdanau *et al.* 2014) to process sequences of data. They have significant advantages over more classical approaches because of their ability to maintain lookback windows with large horizons, which makes them able to detect long-term dependencies in the data. On the other hand, canonical Transformers' point-wise dot-product attention makes them prone to anomalies and optimisation issues because of their space complexity, which grows quadratically with the input length. Furthermore, canonical Transformers are locally-agnostic, because dot-product attention does not allow the model to be aware of the local context while operating with time series data. The convolutional-Transformer mitigates these problems. First, the integration of convolutional networks makes the model locally-aware. Second, with a sparse self-attention mechanism each time-step attends only to previous time-steps with an exponential step size, which mitigates its space complexity. This reduces the cost of computing the attention scores from $\mathcal{O}(M^2)$ to $\mathcal{O}(M(\log(M))^2)$, where M is the input length.

The Transformer-based encoder model grounds its operation on the well-known self-attention mechanism, performed simultaneously by a different number of Transformer's heads $H \in \mathbb{Z}$, resulting in what is known as a *multi-head Transformer*; see Figure 5. Each multi-head self-attention sublayer simultaneously applies the scaled dot-product attention over the convolutional network's output. For the i^{th} head, this scaled dot-product attention is given by

$$\mathrm{h_i}(m) = \text{Attention}\left[Q_i(m), K_i(m), V_i(m)\right]$$

† Note that $p = 1$ results in a Causal-CNN.

$$= \text{softmax}\left[\frac{Q_i(m)K_i(m)^T}{\sqrt{d_k}} W_i\right] V_i(m) \in \mathbb{R}^{T\times d_k}. \quad (3)$$

Here, the scaling constant d_k stabilises the variance of the softmax function's output, and W_i is a mask to prevent information leakage.

Each Transformer's head is therefore responsible for learning and modelling attention functions that handle the complex dependencies within the LOB data. With the different heads, the model jointly attends to different temporal subspaces of the original time series. The individual embeddings of each head are then combined to obtain a joint representation

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}\left(\text{h}_1, \text{h}_2, \ldots, \text{h}_i, \ldots \text{h}_\text{H}\right),$$

where h_i represents the output of head i^{th}, and $\text{h}_\text{i} = \text{Attention}(Q_i, K_i, V_i)$.

Merging every head's output through a linear function produces the latent representation $\mathbf{q}(\mathbf{x}; \mathbf{\Phi})$, which encodes the most relevant information from the selected features of the LOB. The vector $\mathbf{\Phi}$ includes the parameters of the convolutional layers and the attention matrices in each head.

6.3. *Monotonic decoder*

In the context of survival analysis, the survival function needs to be decreasing with respect to time. We encode this inductive bias into our architecture to avoid the well-known *crossing problem* (Tagasovska and López-Paz 2019). To this end, we use monotonically restricted neural networks (Chilinski and Silva 2020, Rindt *et al.* 2022) in our monotonic decoder, see Figure 6. This type of neural network allows us to estimate a cumulative density function (CDF) denoted by $F(t \mid \mathbf{x})$ with response variable t and conditioned on input features $\mathbf{x}$, which in our case is the latent representation obtained from the LOB time series through the encoder. The output of the decoder's network, $f_{\mathbf{\Psi}}(\,\cdot\,)$, is consistent with the properties of a CDF because it satisfies: (i) $\lim_{t\to-\infty} f_{\mathbf{\Psi}}(t, \mathbf{x}) = 0$, (ii) $\lim_{t\to\infty} f_{\mathbf{\Psi}}(t, \mathbf{x}) = 1$, (iii) $\frac{\partial f_{\mathbf{\Psi}}(t,\mathbf{x})}{\partial t} \geq 0$, where the third condition is the most difficult to guarantee because neural networks are a composition of nonlinear functions, which makes them difficult to interpret or control. See Appendix 5 for more details.

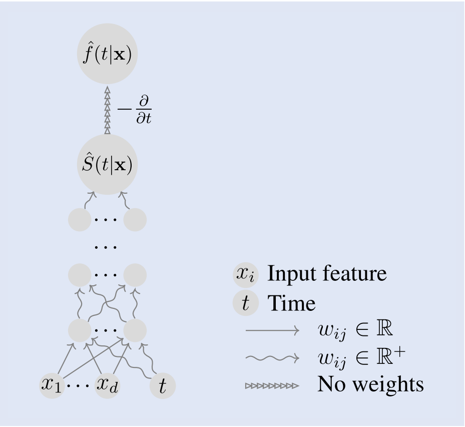


Figure 6. Monotonic decoder's structure. The last node represents the operation of differentiating the conditional survival distribution with respect to time, which results in the conditional density function of the survival time. This model guarantees a decreasing survival curve. (Adapted from Figure 1 of Chilinski and Silva (2020).)

We remark that imposing this monotonicity on the decoder does not hinder other beneficial properties of deep neural networks such as universal function approximation (Cybenko 1989, Kidger and Lyons 2020) or convexity (Littwin and Wolf 2020) in the over-parameterised case. The reason behind this is that the restriction is only enforced on the decoder, which can be made arbitrarily small (in terms of parameters) compared to the time series encoder, where the network parameters are not restricted.

7. Experiments

7.1. *Predictive features*

To estimate the survival function we distinguish between *slow-moving* and *fast-moving* features. Slow moving features provide information about intraday patterns of the trading day, which have predictive power on the probability that a limit order will be filled. One such pattern is the intraday behaviour of the volatility of returns, which is generally high at the beginning of the trading day due to uncertainty and an adjustment to overnight information; see Figure 7 for an example with AAPL over the month of October 2022.[†]

A similar effect is visible with daily traded volume. Overnight information generally causes larger trading volume at the beginning of the day, which tends to stabilise at a reduced value during the trading day, and peaks again at the end of the day when traders have more urgency to adjust their positions. This effect is shown in Figure 8, along with the evolution of the fill probability (which summarises the asymmetry between supply and demand of liquidity in the order book) during the trading day. Given the persistence of these intraday seasonalities, it is feasible to obtain good estimates of the fill probability based on this information.[‡]

Aside from seasonal patterns, we want to predict the changes in the fill probability over more granular time-scales by using *fast-moving features*. In particular, we hypothesise that some of the variables that are most important in the estimation of the survival function of limit orders include future evolution of the bid-ask spread (smaller spreads would incentivise traders to place a liquidity taking order on the other side of the book), volatility (due to its correlation with the spread), or order arrival speed (as traders who observe large queues forming will be more incentivised to cross the bid-ask spread). Volatility and the bid-ask spread exhibit significant persistence and cross-correlations, see Bińkowski and Lehalle (2022), which further motivates the use of an attention-based encoder to capture long-ranging dependencies between the different time-series.

[†] The volatility is estimated using a rolling mean of 1000 trades over the squared returns of the midprice time-series. The realised volatility is highly correlated with the evolution of the spread, which is also included as a feature.

[‡] Another possible way of homogenising these effects is to consider the evolution of the day in *transaction time*, see Appendix 8.

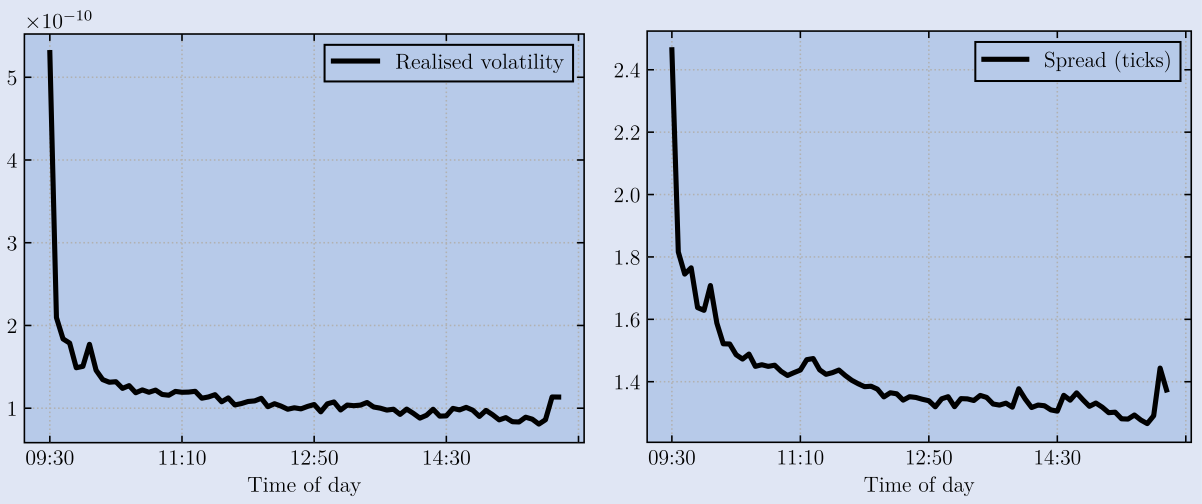


Figure 7. Realised volatility and spread of AAPL stock over October 2022. **Left:** Realised volatility. **Right:** Spread.

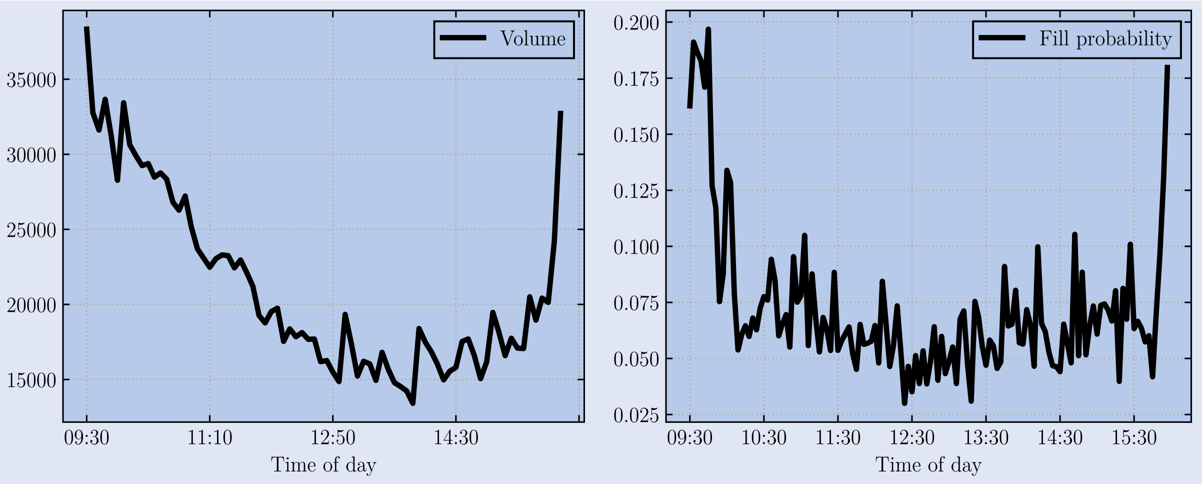


Figure 8. 5-minute bucket average statistics for AAPL stock over October 2022. **Left:** Daily traded volume. **Right:** Fill probability of limit orders posted at the best level in the book.

We build two signals with fast moving features. These are *volume imbalance*, given by

$$\Upsilon_t = \frac{v_b^1(m) - v_a^1(m)}{v_b^1(m) + v_a^1(m)} \in [-1, 1],$$

and the *microprice*, given by

$$M_t = \frac{v_b^1(m)}{v_b^1(m) + v_a^1(m)}\, p_a^1(m) + \frac{v_a^1(m)}{v_b^1(m) + v_a^1(m)}\, p_b^1(m).$$

Volume imbalance captures the difference between the buy and sell pressures in the order book at time t, and it is a predictor of the arrival of aggressive orders, limit orders, and short-term price moves, as detailed in Cartea *et al.* (2018) and Cartea *et al.* (2020). When Υ_t is close to 1, there is buy pressure, and when it is close to -1, there is sell pressure. Similarly, the microprice reflects the tendency of the price to move toward the bid or the ask side of the book. An example evolution of these indicators over a horizon of 1000 trades is shown in Figure 9.

These two signals are added as inputs to the model given their widespread use and well-known predictive power over short horizons. We also include the raw volumes and prices of the top five levels of the order book to allow our model to find more complex inter-dependencies directly from the data.

7.2. *Model fit*

In this subsection, we report the results of our model. We compare the performance of our model with classic deep learning benchmarks from the survival analysis literature. In particular, we consider the DeepSurv (Katzman *et al.* 2018) and DeepHit (Lee *et al.* 2018) models. Furthermore, we test the effectiveness of our encoder by replacing it with a Multi-Layer Perceptron (MLP) (which results in the architecture introduced in Rindt *et al.* 2022), a CNN, and a long short-term memory (LSTM) network (Hochreiter and Schmidhuber 1997). The latter two models are the ones expected to be in closest competition with the convolutional transformer, as they also model the temporal dynamics observed in the data.

We train our model with data from 3 October 2022 to 27 December 2022. We create sets to train, validate, and test with a 60, 20, 20 percent split. Further, to avoid look-ahead bias, we preserve the sequential nature of the time-series and use data from 1 October to 22 November in the training set,

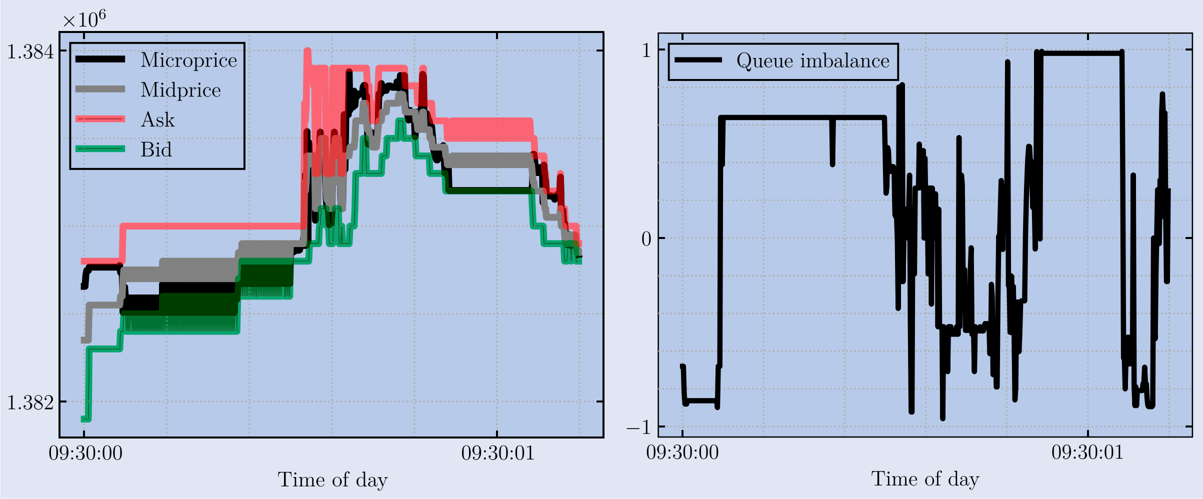


Figure 9. Evolution of indicators over the first 600 trades of 3 October 2022 for the AAPL stock. **Left:** Midprice and microprice. **Right:** Queue imbalance.

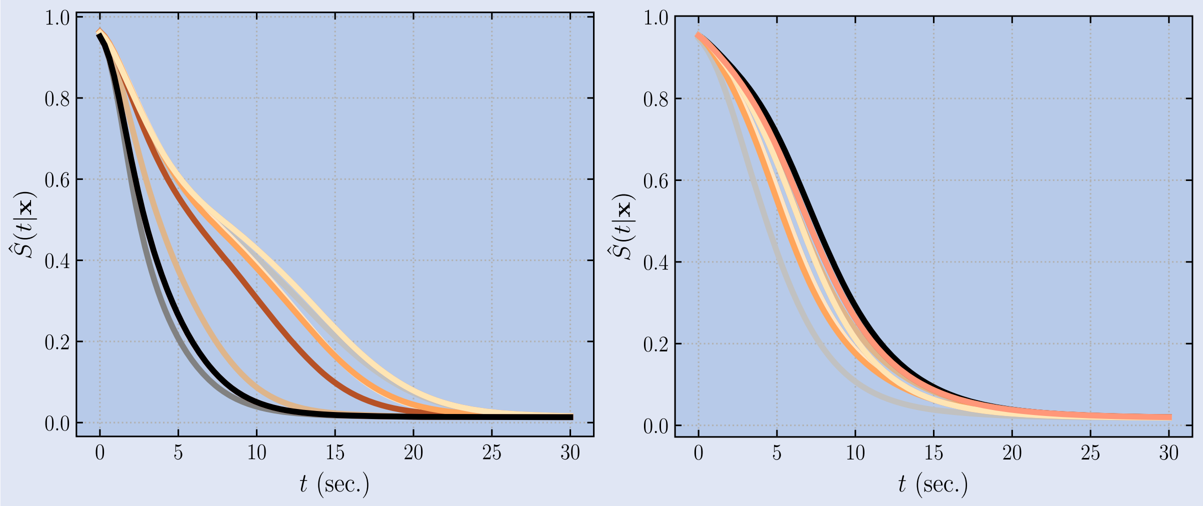


Figure 10. Survival functions predicted by the encoder-decoder monotonic convolutional-Transformer model for a batch of different limit orders. **Left:** AAPL, **Right:** AMZN.

23 November to 10 December for hyperparameter tuning, and 11 December to 27 December for out-of-sample testing. We use the tickers shown in Table 2, and the features described in the previous subsection (time of day, volatility, volume imbalance, microprice, and prices and volumes of the best five levels). For each trading day, we choose either 100 orders that were placed by market participants into the order book (we only consider orders placed at-the-touch because most orders in the book are placed at that level), or we place and track the same number of hypothetical limit orders pegged to the best level of the book.† We store the features over different lookback horizons of $M = 50$, 500, and 1000 trades to explore whether information in the distant past is informative to predict the fill probability (Figure 10).

The out-of-sample performance of the model in terms of negative RCLL is provided in Tables 5 and 6 for the tracked and hypothetical limit orders, respectively. Tables 7 and 8 show the performance improvement of each model over the one using an MLP as an encoder for both order types. The best result for each ticker is in bold, with our proposed model outperforming all benchmarks. All models which account for the time-varying dynamics of the LOB achieve significant performance gains, which suggests that high-frequency microstructural information plays a major role in the estimation. Furthermore, models with an LSTM or CNN encoder do not exhibit as significant gains (or even incur in some performance degradation) when considering longer lookback windows, due to the inability to summarise the information of the entire horizon. In contrast, the convolutional-Transformer encoder can weigh the information over the entire horizon by its relevance to the final estimate, allowing it to achieve the best performance for longer time windows. As mentioned, our model achieves this performance with fewer parameters compared to other models such as those using the CNN as an encoder.

For completeness, we use an order flow representation of the order book to carry out the same analysis and summarise the results in Appendix 9. The benefits of such an approach are summarised in Kolm *et al.* (2021) and Lucchese *et al.* (2022), where authors suggest that considering order flow or volume representations of the order book increase predictive

† We choose random times during the trading day to track both hypothetical and tracked limit orders.

Table 5. Model performance for pegged orders dataset, in terms of RCLL.

	Mean ± STD Negative RCLL								
	AAPL	**AMZN**	**BIDU**	**COST**	**CSCO**	**DELL**	**GOOG**	**INTC**	**MSFT**
No recurrence									
DeepSurv	1.572 ± 0.032	1.155 ± 0.369	1.913 ± 0.192	1.243 ± 0.326	1.214 ± 0.286	0.998 ± 0.035	1.432 ± 0.261	1.288 ± 0.214	1.282 ± 0.049
DeepHit	1.230 ± 0.054	1.418 ± 0.098	1.926 ± 0.093	1.304 ± 0.243	1.122 ± 0.089	0.922 ± 0.096	1.314 ± 0.116	1.188 ± 0.399	1.362 ± 0.075
MN-MLP	1.465 ± 0.364	1.754 ± 0.295	1.943 ± 0.972	1.987 ± 0.261	1.273 ± 0.410	1.312 ± 0.349	1.553 ± 1.405	1.511 ± 0.352	1.912 ± 1.058
T=50									
MN-CNN	0.571 ± 0.095	0.594 ± 0.071	0.954 ± 0.042	0.647 ± 0.037	1.200 ± 0.165	0.835 ± 0.057	0.657 ± 0.006	0.757 ± 0.174	0.676 ± 0.109
MN-LSTM	0.820 ± 0.611	0.390 ± 0.198	1.265 ± 0.808	0.883 ± 0.409	0.884 ± 0.377	0.559 ± 0.468	1.234 ± 0.731	1.066 ± 0.650	0.963 ± 0.268
MN-Conv-Trans	**0.142 ± 0.185**	0.132 ± 0.195	0.406 ± 0.364	0.178 ± 0.299	0.242 ± 0.303	0.168 ± 0.255	0.233 ± 0.210	0.341 ± 0.454	0.180 ± 0.201
T=500									
MN-CNN	1.043 ± 0.177	0.629 ± 0.059	0.714 ± 0.104	0.362 ± 0.132	0.805 ± 0.121	0.754 ± 0.108	0.854 ± 0.117	1.188 ± 0.166	1.456 ± 0.122
MN-LSTM	0.977 ± 0.500	0.414 ± 0.097	0.966 ± 0.374	0.078 ± 0.129	1.145 ± 0.517	0.449 ± 0.311	0.620 ± 0.208	0.959 ± 0.423	1.322 ± 0.546
MN-Conv-Trans	0.180 ± 0.153	0.177 ± 0.166	**0.296 ± 0.300**	**0.027 ± 0.040**	0.365 ± 0.323	**0.167 ± 0.250**	0.308 ± 0.236	**0.281 ± 0.293**	0.188 ± 0.119
T=1000									
MN-CNN	0.757 ± 0.28	0.581 ± 0.092	0.919 ± 0.075	1.345 ± 0.068	1.185 ± 0.277	0.904 ± 0.138	0.657 ± 0.074	1.303 ± 0.247	0.677 ± 0.092
MN-LSTM	1.013 ± 0.495	0.402 ± 0.173	1.633 ± 0.475	1.053 ± 0.663	0.889 ± 0.406	0.871 ± 0.386	1.240 ± 0.647	1.405 ± 0.501	0.858 ± 0.374
MN-Conv-Trans	0.192 ± 0.168	**0.129 ± 0.171**	0.405 ± 0.320	0.241 ± 0.236	**0.236 ± 0.291**	0.179 ± 0.206	**0.208 ± 0.197**	0.313 ± 0.345	**0.179 ± 0.252**

performance for step-ahead forecasts, while making the use of more complex models unnecessary. In our case, however, we find that the convolutional transformer still outperforms all other models in this setting. This suggests that some of the representations used to perform directional price forecasts are not as relevant when estimating the survival functions of limit orders. A possible reason for this is that price prediction is a harder task given that it is a directional forecast. Moreover, the prediction of the fill probability is closely linked to the behaviour of the spread, whose prediction is non-directional, and is closely linked to the volatility (which exhibits higher persistence than a returns time series). As such, the model might be focussing on properties of the features that aid in the prediction of future volatility, making representations that aid in directional price forecasts redundant.

7.3. Model interpretability

In this section, we focus on the interpretability of our model. To do so, we analyse both the time and feature domains. In particular, we use attention heatmaps to visualise which parts of the past values of the signals are more important to the model to estimate the survival function. Finally, we use Shapley values (Hart 1989) to quantify the relative importance of each input feature to the output of the model.

7.3.1. Attention heatmaps. The convolutional-Transformer employs (2) to obtain, through the convolutional network, the self-attention input features described in Section 6.2: the query, key, and value matrices. The model then performs the dot-product computation between the attention's queries and keys, see (3). This operation results on a matrix of dimensions

Table 6. Model performance for observed orders dataset, in terms of RCLL.

Mean±STD Negative RCLL									
	AAPL	**AMZN**	**BIDU**	**COST**	**CSCO**	**DELL**	**GOOG**	**INTC**	**MSFT**
No recurrence									
DeepSurv	8.109 ± 0.065	8.557 ± 0.034	7.915 ± 0.103	7.982 ± 0.066	9.978 ± 0.067	8.885 ± 0.064	9.131 ± 0.024	8.462 ± 0.096	8.564 ± 0.008
DeepHit	10.098 ± 0.065	10.119 ± 0.028	9.716 ± 0.146	9.731 ± 0.165	9.978 ± 0.021	9.816 ± 0.081	10.046 ± 0.065	9.874 ± 0.022	10.074 ± 0.055
MN-MLP	10.554 ± 0.364	10.709 ± 0.351	13.300 ± 0.171	13.603 ± 0.470	12.364 ± 0.422	13.151 ± 0.445	11.330 ± 0.398	12.110 ± 0.415	10.784 ± 0.354
***T* = 50**									
MN-CNN	5.075 ± 0.366	4.795 ± 0.288	13.743 ± 0.158	9.474 ± 0.139	6.569 ± 0.126	8.778 ± 0.553	5.535 ± 0.246	6.588 ± 0.206	5.351 ± 0.045
MN-LSTM	3.896 ± 0.497	3.302 ± 0.067	6.452 ± 0.181	9.539 ± 1.958	6.065 ± 0.221	7.056 ± 0.114	4.385 ± 0.215	6.954 ± 0.169	4.077 ± 0.255
MN-Conv-Trans	3.201 ± 0.827	2.997 ± 0.675	5.930 ± 0.863	5.957 ± 0.808	5.019 ± 1.149	**5.522 ± 0.860**	3.730 ± 0.901	5.002 ± 1.167	3.533 ± 1.090
***T* = 500**									
MN-CNN	5.056 ± 0.165	5.401 ± 0.077	10.801 ± 0.221	8.910 ± 0.253	6.699 ± 0.130	7.422 ± 0.135	5.536 ± 0.157	6.768 ± 0.307	5.284 ± 0.165
MN-LSTM	3.701 ± 0.202	4.135 ± 0.270	7.658 ± 0.303	7.681 ± 0.272	5.636 ± 0.224	7.479 ± 0.385	4.338 ± 0.169	6.545 ± 0.232	4.369 ± 0.189
MN-Conv-Trans	**3.171 ± 0.157**	3.111 ± 0.249	6.428 ± 1.231	**5.822 ± 0.290**	4.997 ± 0.255	5.814 ± 0.424	3.729 ± 0.925	5.077 ± 0.352	**3.326 ± 0.309**
***T* = 1000**									
MN-CNN	5.925 ± 0.010	4.796 ± 0.142	7.485 ± 0.271	8.052 ± 0.022	6.581 ± 0.290	7.171 ± 0.174	5.536 ± 0.157	7.676 ± 0.540	5.347 ± 0.093
MN-LSTM	5.404 ± 0.023	3.932 ± 0.154	6.945 ± 0.06	7.199 ± 0.030	6.043 ± 0.219	7.202 ± 0.583	4.338 ± 0.169	5.52 ± .241	3.904 ± 0.141
MN-Conv-Trans	3.724 ± 1.226	**2.980 ± 0.713**	**5.887 ± 0.918**	6.089 ± 1.073	**4.974 ± 0.834**	5.625 ± 0.919	**3.453 ± 0.498**	**4.951 ± 1.053**	3.560 ± 1.122

$\mathbb{R}^{T\times T}$, where $T \in \mathbb{Z}$ is the lookback window's length. Finally, after the softmax function is applied to this matrix, see (3), the output is used to multiply the previously obtained self-attention's values. Therefore, with the matrix resulting from the dot-product operation, one visualises which regions of the lookback window (more precisely, of its non-linear projection), are given the highest weighting by the model when estimating the survival function. These are commonly known as *attention heatmaps*.

Figure 11 shows the four attention heatmaps of our model, one per head, for a single estimate. Further, Appendix 6 shows the corresponding evolution in time of the features. The attention heatmaps display the self-attention weights and provide information on the weight given for each time-step by the model. As shown in the plots, head 0 focuses on samples of 400 trades ago when there was a significant reduction in volatility and the size of the spread. The remaining of the heads have a sparse attention pattern, showing that the models accounts for both short and long-term information to make the forecast.

7.3.2. Shapley values. Shapley values, originally developed in the cooperative game theory literature (Faigle and Kern 1992, Hsiao and Raghavan 1993), offer a mechanism to allocate importance among the participants of cooperative games. In machine learning, these techniques are used to quantify the impact of features on the predictions of deep learning models, see Lundberg and Lee (2017), Ghorbani and Zou (2019), which helps in model interpretation, feature importance assessment and selection.

In our setting, Shapley values help us to understand which of the market features are most important and

Table 7. Percentage improvement over the MN-MLP model for pegged limit orders dataset.

Improvement over MN-MLP (%)										
	AAPL	**AMZN**	**BIDU**	**COST**	**CSCO**	**DELL**	**GOOG**	**INTC**	**MSFT**	**Average**
No recurrence										
DeepSurv	– 7.30	34.15	1.54	37.44	4.63	23.93	7.79	14.76	32.95	16.66
DeepHit	16.04	19.16	0.87	34.37	11.86	29.73	15.39	21.38	28.77	19.73
T=50										
MN-CNN	61.02	66.13	50.90	67.44	5.73	36.36	57.69	49.90	64.64	51.09
MN-LSTM	44.03	77.77	34.89	55.56	30.56	57.39	20.54	29.45	49.63	44.42
MN-Conv-Trans	**90.31**	92.47	79.10	91.04	80.99	87.20	85.00	77.43	90.59	**86.01**
T=500										
MN-CNN	28.81	64.14	63.25	81.78	36.76	42.53	45.01	21.38	23.85	45.28
MN-LSTM	33.31	76.40	50.28	96.07	10.05	65.78	60.08	36.53	30.86	51.04
MN-Conv-Trans	87.71	89.91	**84.77**	**98.64**	71.33	**87.27**	80.17	**81.40**	90.17	85.71
T=1000										
MN-CNN	60.34	66.88	52.70	32.31	6.91	31.10	57.69	13.77	64.59	35.77
MN-LSTM	30.85	77.08	15.95	47.01	30.16	33.61	20.15	7.02	55.12	30.75
MN-Conv-Trans	86.89	**93.51**	79.15	87.87	**81.46**	86.36	**86.61**	79.29	**90.64**	85.22

Table 8. Percentage improvement over the MN-MLP model for observed limit orders dataset.

Improvement over MN-MLP (%)										
	AAPL	**AMZN**	**BIDU**	**COST**	**CSCO**	**DELL**	**GOOG**	**INTC**	**MSFT**	**Average**
No Recurrence										
DeepSurv	23.17	20.10	40.49	41.32	19.30	32.44	19.41	30.12	20.59	27.44
DeepHit	4.32	5.51	26.95	28.46	19.30	25.36	11.33	18.46	6.58	16.25
$\boldsymbol{T = 50}$										
MN-CNN	51.91	55.22	-3.33	30.35	46.87	33.25	51.15	45.60	50.38	40.16
MN-LSTM	63.09	69.17	51.49	29.88	50.95	46.35	61.30	42.58	62.19	53.00
MN-Conv-Trans	69.67	72.01	55.41	56.21	59.41	**58.01**	67.08	58.70	67.24	62.64
$\boldsymbol{T = 500}$										
MN-CNN	52.09	49.57	18.79	34.50	45.82	43.56	51.14	44.11	51.00	43.40
MN-LSTM	64.93	61.39	42.42	43.53	54.42	43.13	61.71	45.95	59.49	53.00
MN-Conv-Trans	**69.95**	70.95	51.67	**57.20**	59.58	55.79	67.09	58.08	**69.16**	62.16
$\boldsymbol{T = 1000}$										
MN-CNN	54.56	55.22	43.72	40.81	46.77	45.47	51.15	36.61	50.42	47.19
MN-LSTM	48.80	63.28	47.78	47.08	51.12	45.24	61.30	54.42	63.80	53.65
MN-Conv-Trans	64.71	**78.09**	**55.74**	55.24	**59.77**	57.23	**69.45**	**59.12**	66.99	**62.66**

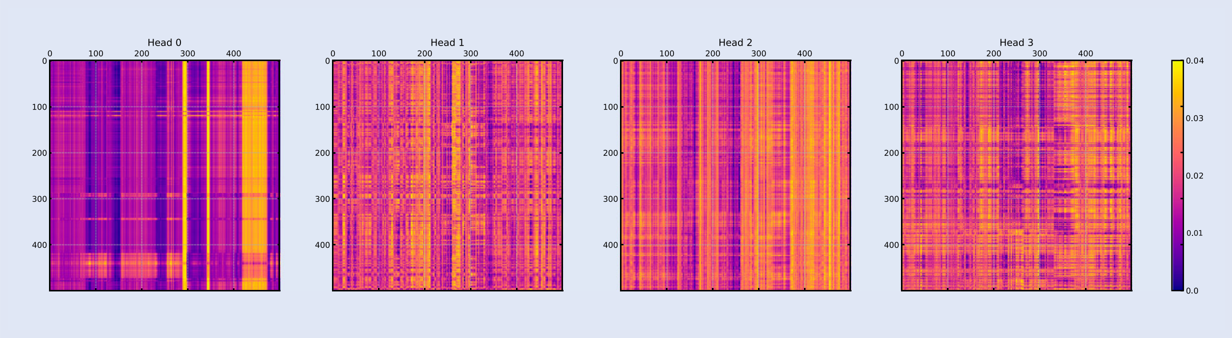


Figure 11. Attention heatmaps obtained for an example order.

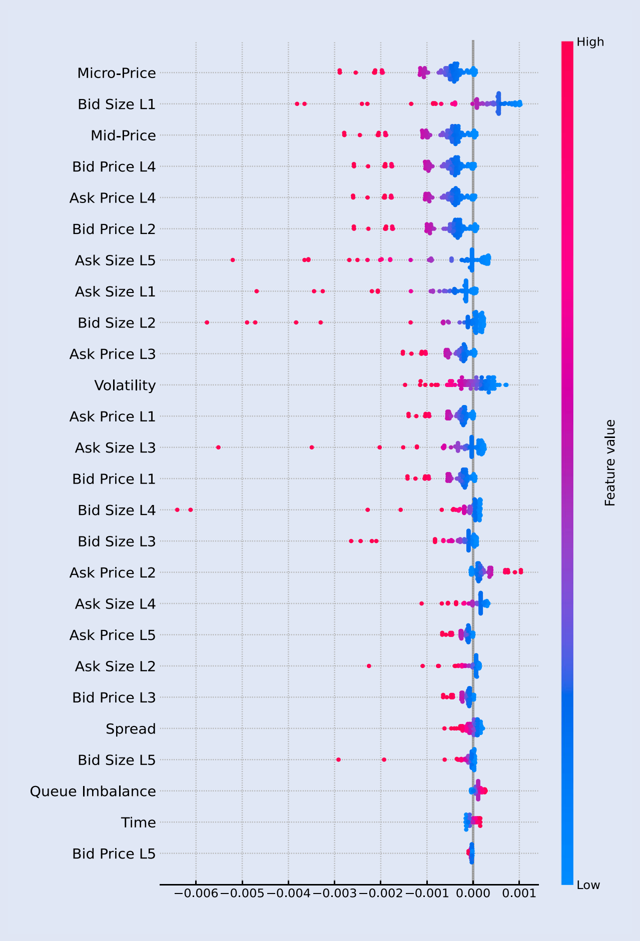


Figure 12. Shapley values for 100 predictions of the MN-MLP model.

how they contribute to the model's overall prediction. We follow the DeepSHAP approach in Lundberg and Lee (2017) to calculate the Shapley values (see Appendix 11 for more details). Figure 12 depicts a Beeswarm plot associated with the MN-MLP model, which shows in the vertical axis the relative importance of each feature and its relationship with the predicted outcome. Each of the datapoints (a full batch) is represented with a single dot, where colours ranging from blue to red indicate lower to higher values per input feature. The vertical axis is ordered in accordance with the importance on average of each feature, while the horizontal axis displays the associated Shapley value.

The figure shows that the model gives the most importance to fast-moving features. For example, the microprice is the most important feature because it provides a relatively good estimate of the perceived fundamental value of the asset, which has an effect on where liquidity taking orders are placed. The volatility of the asset also plays a major role in the prediction, which could be attributed to the fact that it is a good proxy for the volume being traded. If more volume is being traded in the market, there is a higher chance that a liquidity taking order will cross the spread and match an outstanding order. Time of day, as well as other features, shows little to no importance, which is likely due to the fact that time of day shows a very persistent pattern it is an intraday pattern, meaning that small perturbations in its value should not affect the output of the model significantly.

8. Conclusion

This paper presented a novel approach to estimating the fill probabilities of limit orders posted in the LOB. The proposed data-driven approach integrates a novel convolutional-Transformer model to raise local-awareness from LOB data, and a monotonic neural network to guarantee the monotonicity of the survival function of limit orders. To train and evaluate the model, we use right-censored log-likelihood, which is a proper scoring rule, unlike other scoring rules commonly used in the literature. To demonstrate the effectiveness of the proposed method, we conducted a set of experiments on real LOB data. These experiments showed that the monotonic encoder-decoder convolutional-Transformer significantly outperforms state-of-the-art benchmarks, and provides a new general framework with which to perform survival analysis from time-series observations. Finally, we provided an interpretability analysis based on Shapley values and attention heatmaps, which provides insight on which predictive features are the most influential.

In this paper we have focussed on the financial applications of survival analysis. However, future work could make use of the same architecture for alternative application domains like healthcare, where the use of survival analysis is prevalent. From a financial standpoint, it would be interesting to further explore if the fill probability results in improved LOB modelling when used in a realistic market simulator. Furthermore, we considered a setup in which we summarised the time-series representation up to a point in time and did not use any information after order submission. To this end, an interesting follow-up could leverage or extend recent ideas of temporally-consistent survival analysis, see Maystre and Russo (2022), to obtain improved estimates of the survival function of limit orders. Finally, it would be interesting to explore a multi-asset framework, see Bergault *et al.* (2022), Drissi (2022, 2023), to understand how the fill probability is affected by correlated instruments, which opens up the possibility of using graphs, see Arroyo *et al.* (2022) and de Ocáriz Borde *et al.* (2023), as a modelling technique.

Acknowledgments

We are grateful to Fayçal Drissi and Leandro Sánchez-Betancourt for insightful comments and suggestions. We also thank the participants of the Victoria seminar for interesting discussions. We thank the participants of the 67th EWGCFM Meeting for comments.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

Álvaro Arroyo acknowledges support from the Rafael del Pino Foundation. Fernando Moreno-Pino acknowledges support from Spanish Ministerio de Ciencia, Innovación y Universidades (FPU18/00470), Ministerio de Ciencia e Innovación jointly with the European Commission

(ERDF) (PID2021-123182OB-I00,PID2021-125159NB-I00), by Comunidad de Madrid (and IND2020/TIC-17372), and by the European Union (FEDER) and the European Research Council (ERC) through the European Union's Horizon 2020 research and innovation program underGrant 714161.

References

Alaa, A.M. and van der Schaar, M. Deep multi-task Gaussian processes for survival analysis with competing risks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 2326–2334, 2017.

Antolini, L., Boracchi, P. and Biganzoli, E., A time-dependent discrimination index for survival data. *Stat. Med.*, 2005, **24**(24), 3927–3944.

Arroyo, A., Scalzo, B., Stanković, L. and Mandic, D.P., Dynamic portfolio cuts: A spectral approach to graph-theoretic diversification. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5468–5472, 2022 (IEEE).

Avati, A., Duan, T., Zhou, S., Jung, K., Shah, N.H. and Ng, A.Y., Countdown regression: Sharp and calibrated survival predictions. In *Uncertainty in Artificial Intelligence*, pp. 145–155, 2020 (PMLR).

Bahdanau, D., Cho, K. and Bengio, Y., Neural machine translation by jointly learning to align and translate. Preprint, 2014. arXiv:1409.0473.

Bergault, P., Drissi, F. and Guéant, O., Multi-asset optimal execution and statistical arbitrage strategies under ornstein–uhlenbeck dynamics. *SIAM J. Financ. Math.*, 2022, **13**(1), 353–390.

Bińkowski, M. and Lehalle, C.A., Endogenous dynamics of intraday liquidity. *J. Portf. Manag.*, 2022, **48**(6), 145–169.

Borovykh, A., Bohte, S. and Oosterlee, C.W., Conditional time series forecasting with convolutional neural networks. Preprint, 2017. arXiv:1703.04691.

Cartea, Á., Donnelly, R. and Jaimungal, S., Enhancing trading strategies with order book signals. *Appl. Math. Finance*, 2018, **25**(1), 1–35.

Cartea, Á., Jaimungal, S. and Penalva, J., *Algorithmic and High-Frequency Trading*, 2015 (Cambridge University Press).

Cartea, Á., Jaimungal, S. and Wang, Y., Spoofing and price manipulation in order-driven markets. *Appl. Math. Finance*, 2020, **27**(1–2), 67–98.

Chapfuwa, P., Tao, C., Li, C., Page, C., Goldstein, B., Duke, L.C. and Henao, R., Adversarial time-to-event modeling. In *International Conference on Machine Learning*, pp. 735–744, 2018 (PMLR).

Chilinski, P. and Silva, R., Neural likelihoods via cumulative distribution functions. In *Conference on Uncertainty in Artificial Intelligence*, pp. 420–429, 2020 (PMLR).

Cho, J.W. and Nelling, E., The probability of limit-order execution. *Financ. Anal.*, 2000, **56**(5), 28–33.

Cox, D.R., Regression models and life-tables. *J. R. Stat. Soc.: Ser. B (Methodol.)*, 1972, **34**, 187–202.

Cybenko, G., Approximation by superpositions of a sigmoidal function. *Math. Control. Signals, Syst.*, 1989, **2**(4), 303–314.

Ding, Q., Wu, S., Sun, H., Guo, J. and Guo, J., Hierarchical multiscale gaussian transformer for stock movement prediction. In *IJCAI*, pp. 4640–4646, 2020.

Dirick, L., Claeskens, G. and Baesens, B., Time to default in credit scoring using survival analysis: A benchmark study. *J. Oper. Res. Soc.*, 2017, **68**(6), 652–665.

Drissi, F., Solvability of differential Riccati equations and applications to algorithmic trading with signals. 2022. Available at SSRN 4308008.

Drissi, F., Models of market liquidity: Applications to traditional markets and automated market makers. 2023. Available at SSRN 4424010.

Eisler, Z., Bouchaud, J.P. and Kockelkoren, J., The price impact of order book events: Market orders, limit orders and cancellations. *Quant. Finance*, 2012, **12**(9), 1395–1419.

Faigle, U. and Kern, W., The shapley value for cooperative games under precedence constraints. *Int. J. Game Theory*, 1992, **21**(3), 249–266.

Faraggi, D. and Simon, R., A neural network model for survival data. *Stat. Med.*, 1995, **14**(1), 73–82.

Fernández, T., Rivera, N. and Teh, Y.W., Gaussian processes for survival analysis. *Adv. Neural Inf. Process. Syst.*, 2016, **29**.

Ghorbani, A. and Zou, J., Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pp. 2242–2251, 2019 (PMLR).

Gneiting, T. and Ranjan, R., Comparing density forecasts using threshold-and quantile-weighted scoring rules. *J. Bus. Econ. Stat.*, 2011, **29**(3), 411–422.

Graf, E., Schmoor, C., Sauerbrei, W. and Schumacher, M., Assessment and comparison of prognostic classification schemes for survival data. *Stat. Med.*, 1999, **18**, 2529–2545.

Guéant, O., *The Financial Mathematics of Market Liquidity: From Optimal Execution to Market Making*, 2016 (CRC Press).

Handa, P. and Schwartz, R.A., Limit order trading. *J. Finance*, 1996, **51**(5), 1835–1861.

Harrell, F.E., Califf, R.M., Pryor, D.B., Lee, K.L. and Rosati, R.A., Evaluating the yield of medical tests. *Jama*, 1982, **247**(18), 2543–2546.

Hart, S., Shapley value. In *Game Theory*, pp. 210–216, 1989 (Springer).

He, K., Zhang, X., Ren, S. and Sun, J., Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Hochreiter, S. and Schmidhuber, J., Long short-term memory. *Neural Comput.*, 1997, **9**(8), 1735–1780.

Hsiao, C.R. and Raghavan, T., Shapley value for multichoice cooperative games, i. *Games Econ. Behav.*, 1993, **5**(2), 240–256.

Hu, S., Fridgeirsson, E., van Wingen, G. and Welling, M., Transformer-based deep survival analysis. In *Survival Prediction-Algorithms, Challenges and Applications*, pp. 132–148, 2021 (PMLR).

Ishwaran, H., Kogalur, U.B., Blackstone, E.H. and Lauer, M.S., Random survival forests. *Ann. Appl. Stat.*, 2008, **2**(3), 841–860.

Kaplan, E.L. and Meier, P., Nonparametric estimation from incomplete observations. *J. Am. Stat. Assoc.*, 1958, **53**(282), 457–481.

Katzman, J.L., Shaham, U., Cloninger, A., Bates, J., Jiang, T. and Kluger, Y., DeepSurv: Personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC Med. Res. Methodol.*, 2018, **18**(1), 1–12.

Kidger, P. and Lyons, T., Universal approximation with deep narrow networks. In *Conference on Learning Theory*, pp. 2306–2327, 2020 (PMLR).

Kolm, P.N., Turiel, J. and Westray, N., Deep order flow imbalance: Extracting alpha at multiple horizons from the limit order book. 2021. Available at SSRN 3900141.

Kvamme, H., Borgan, Ø. and Scheel, I., Time-to-event prediction with neural networks and Cox regression. Preprint, 2019. arXiv preprint arXiv:1907.00825.

Larivière, B. and Van den Poel, D., Investigating the role of product features in preventing customer churn, by using survival analysis and choice modeling: The case of financial services. *Expert Syst. Appl.*, 2004, **27**(2), 277–285.

Laurie, J.A., Moertel, C.G., Fleming, T.R., Wieand, H.S., Leigh, J.E., Rubin, J., McCormack, G.W., Gerstner, J.B., Krook, J.E. and Malliard, J., Surgical adjuvant therapy of large-bowel carcinoma: An evaluation of levamisole and the combination of levamisole and fluorouracil. *J. Clin. Oncol.*, 1989, **7**(10), 1447–1456.

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W. and Jackel, L.D., Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1989, **1**(4), 541–551.

Lee, C., Yoon, J. and Van Der Schaar, M., Dynamic-deephit: A deep learning approach for dynamic survival analysis with competing risks based on longitudinal data. *IEEE. Trans. Biomed. Eng.*, 2019, **67**(1), 122–133.

Lee, C., Zame, W., Yoon, J. and Van Der Schaar, M., Deephit: A deep learning approach to survival analysis with competing risks.

In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Lezmi, E. and Xu, J., Time series forecasting with transformer models and application to asset management. 2023. Available at SSRN 4375798.

Littwin, E. and Wolf, L., On the convex behavior of deep neural networks in relation to the layers' width. Preprint, 2020. arXiv:2001.04878.

Lo, A.W., MacKinlay, A.C. and Zhang, J., Econometric models of limit-order executions. *J. Financ. Econ.*, 2002, **65**(1), 31–71.

Lucchese, L., Pakkanen, M. and Veraart, A., The short-term predictability of returns in order book markets: A deep learning perspective. Preprint, 2022. arXiv:2211.13777.

Lundberg, S.M. and Lee, S.I., A unified approach to interpreting model predictions. Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 4768–4777, December 2017.

Maglaras, C., Moallemi, C. and Wang, M., A deep learning approach to estimating fill probabilities in a limit order book. *Quant. Finance*, 2022, **22**, 1989–2003.

Maystre, L. and Russo, D., Temporally-consistent survival analysis. *Adv. Neural Inf. Process. Syst.*, 2022, **35**, 10671–10683.

Mishev, K., Gjorgjevikj, A., Vodenska, I., Chitkushev, L.T. and Trajanov, D., Evaluation of sentiment analysis in finance: From lexicons to transformers. *IEEE Access*, 2020, **8**, 131662–131682.

Moreno-Pino, F. and Zohren, S., Deepvol: Volatility forecasting from high-frequency data with dilated causal convolutions. Preprint, 2022. arXiv:2210.04797.

Moreno-Pino, F., Olmos, P.M. and Artés-Rodríguez, A., Deep autoregressive models with spectral attention. *Pattern Recognit.*, 2023, **133**, 109014.

de Ocáriz Borde, H.S., Arroyo, A. and Posner, I., Projections of model spaces for latent graph inference. In *ICLR 2023 Workshop on Physics for Machine Learning*, 2023.

Oord, A.v.d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. and Kavukcuoglu, K., Wavenet: A generative model for raw audio. Preprint, 2016. arXiv:1609.03499.

Rindt, D., Hu, R., Steinsaltz, D. and Sejdinovic, D., Survival regression with proper scoring rules and monotonic neural networks. In *25th International Conference on Artificial Intelligence and Statistics (AISTATS 2022)*, 2022.

Singh, R. and Mukhopadhyay, K., Survival analysis in clinical trials: Basics and must know areas. *Perspect. Clin. Res.*, 2011, **2**(4), 145.

Susto, G.A., Schirru, A., Pampuri, S., McLoone, S. and Beghi, A., Machine learning for predictive maintenance: A multiple classifier approach. *IEEE Trans. Industr. Inform.*, 2014, **11**(3), 812–820.

Tagasovska, N. and López-Paz, D., Single-model uncertainties for deep learning. *Adv. Neural Inf. Process. Syst.*, 2019, **32**.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., Attention is all you need. *Adv. Neural Inf. Process. Syst.*, 2017, **30**.

Wallbridge, J., Transformers for limit order books. Preprint, 2020. arXiv:2003.00130.

Wang, Z. and Sun, J., SurvTRACE: Transformers for survival analysis with competing events. In *Proceedings of the 13th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, pp. 1–9, 2022.

Wang, C., Chen, Y., Zhang, S. and Zhang, Q., Stock market index prediction using deep transformer model. *Expert. Syst. Appl.*, 2022, **208**, 118128.

Wei, L.J., The accelerated failure time model: A useful alternative to the Cox regression model in survival analysis. *Stat. Med.*, 1992, **11**(14-15), 1871–1879.

Zhong, Q., Mueller, J.W. and Wang, J.L., Deep extended hazard models for survival analysis. *Adv. Neural Inf. Process. Syst.*, 2021, **34**, 15111–15124.

Ziehm, M. and Thornton, J.M., Unlocking the potential of survival data for model organisms through a new database and online analysis platform. *Aging Cell*, 2013, **12**(5), 910–916.

Appendices

Appendix 1. Conditions for a hypothetical order fill

A hypothetical limit order placed in the LOB follows the standard rules of price-time queue priority. As in Maglaras *et al.* (2022), the fill conditions for hypothetical limit orders are:

(1) A new hypothetical limit order is filled if:
- A new buy/sell limit order or liquidity taking order (i.e. a market order or a marketable limit order) order arrives at a higher/lower price than that of the hypothetical sell/buy limit order;
- An incoming liquidity taking order fills a limit order resting in the order book with lower execution priority than that of the hypothetical order. In this event, we assume that the hypothetical order of unit size is also filled.

Appendix 2. Derivation of right-censored log-likelihood

Assume we have a dataset of observations $\mathcal{D} = \{(\mathbf{x}_k, z_k, \delta_k)\}_{k=1}^N$, where $z_k = \min\{T_l, C_l\}$, where the random variables T_l and C_l denote the random fill and cancellation (censoring) times, respectively. For clarity, in the remaining derivations of the appendix, we drop the subscript l. The likelihood function is

$$\begin{aligned} L &= f(z_1, \delta_1, \ldots, z_N, \delta_N) \\ &= \prod_{k=1}^N f(z_k, \delta_k), \end{aligned} \tag{A1}$$

where we sample at random times during the trading day so that the pairs are independent. To re-write this equation depending on the values of the indicator variable δ_k, we first consider the case in which the order is filled, and the event is therefore observed ($\delta_k = 1$)

$$\begin{aligned} f(z_k, \delta_k) &= \mathbb{P}\{Z = z_k, \delta_k = 1\} \\ &= \mathbb{P}\{T = z_k, T \le C\} \\ &= \mathbb{P}\{T = z_k, z_k \le C\} \\ &= \mathbb{P}\{T = z_k\}\mathbb{P}\{z_k \le C\} \quad \text{(assuming } T \text{ is independent of } C) \\ &= f_T(z_k) S_C(z_k). \end{aligned}$$

Moreover, if the observation is censored, we have that $\delta_k = 0$ and

$$\begin{aligned} f(z_k, \delta_k) &= \mathbb{P}\{Z = z_k, \delta_k = 0\} \\ &= \mathbb{P}\{C = z_k, T > C\} \\ &= \mathbb{P}\{C = z_k, z_k < T\} \\ &= \mathbb{P}\{C = z_k\}\mathbb{P}\{z_k < T\} \quad \text{(assuming } T \text{ is independent of } C) \\ &= f_C(z_k) S_T(z_k), \end{aligned}$$

and re-write (A1) as

$$\begin{aligned} L &= \prod_{k=1}^N [f_T(z_k) S_C(z_k)]^{\delta_k} [f_C(z_k) S_T(z_k)]^{(1-\delta_k)} \\ &= \prod_{k=1}^N \left[f_T(z_k)^{\delta_k} S_T(z_k)^{(1-\delta_k)} \right] \left[f_C(z_k)^{(1-\delta_k)} S_C(z_k)^{\delta_k} \right]. \end{aligned}$$

We are concerned with the estimation of $S_T(t)$ and not $S_C(t)$, because $S_C(t)$ contains information related to censoring mechanism. Only $S_T(t)$ contains information about the filltimes, which is the variable of interest. Thus, the terms that do not involve T are considered constants, and the log-likelihood is

$$\mathcal{L} = \log(L) = \sum_{k=1}^N \delta_k \log(\hat{f}(z_k)) + (1 - \delta_k) \log(\hat{S}(z_k)).$$

Appendix 3. Typical survival models and scoring rules

A. Survival models

An initial approach is to use a Kaplan–Meier estimate (Kaplan and Meier 1958) to estimate the survival function i.e.

$$\hat{S}(t) = \prod_{i:t_i \le t} \left(1 - \frac{k_i}{n_i}\right),$$

where t_i is the time when at least one event occurred, k_i is the number of events that occurred at time t_i, and n_i denotes the limit orders known to have survived up to time t_i. A follow-up model to the Kaplan–Meier estimate which conditions on a feature vector is the Cox proportional hazards model (Cox 1972), where the hazard rate follows the definition

$$h(t \mid \mathbf{x}) = h_0(t)\ \exp(\boldsymbol{\beta}^T \mathbf{x}),$$

where $\boldsymbol{\beta}$ are coefficients for the feature vector $\mathbf{x}$, and $h_0(t)$ is a baseline hazard directly estimated from the data. Given N observations, the regression coefficients which maximise

$$\mathcal{L}(\boldsymbol{\beta}) = \prod_{\delta_i = 1} \frac{\exp(\boldsymbol{\beta}^T \mathbf{x})}{\sum_{j:t_j \ge t_i} \exp(\boldsymbol{\beta}^T \mathbf{x})}.$$

Another popular model used in survival analysis is the accelerated failure time model (Wei 1992), in which the hazard rates are determined by

$$h(t \mid \mathbf{x}) = \phi(\boldsymbol{x})\, h_0(\phi(\boldsymbol{x})t),$$

where $\phi(\boldsymbol{x})$ models the effect of the covariates, usually through the relationship $\phi(\boldsymbol{x}) = \exp(\boldsymbol{\beta}^T \mathbf{x})$. In practice, the assumption of linear interaction between features and the proportional hazards assumption are often violated. This motivated the extension of the Cox model with deep learning to capture non-linearities between features of interest (Katzman *et al.* 2018, Kvamme *et al.* 2019)

$$h(t \mid \mathbf{x}) = h_0(t)\ \exp(f_{\boldsymbol{\theta}}(\mathbf{x}, t)).$$

More recent work (Lee *et al.* 2018, 2019, Rindt *et al.* 2022) focuses on directly learning the survival function conditioning on input features

$$S(t \mid \mathbf{x}) = f_{\boldsymbol{\theta}}(\mathbf{x}, t).$$

B. Scoring rules

Commonly used scores for survival functions include time-dependant concordance (Antolini *et al.* 2005), given by

$$\begin{aligned} C_{td} &= \mathbb{P}[\hat{S}(z_i \mid \mathbf{x}_i) < \hat{S}(z_i \mid \mathbf{x}_j) \mid z_i < z_j, \delta_i = 1] \\ &\approx \frac{\sum_{i=1}^{N} \sum_{j=1;i\neq j}^{N} \mathbb{1}[\hat{S}(z_i \mid \mathbf{x}_i) < \hat{S}(z_j \mid \mathbf{x}_j)]\pi_{ij}}{\sum_{i=1}^{N} \sum_{j=1;i\neq j}^{N} \pi_{ij}}, \end{aligned}$$

where π_{ij} is an indicator of the pair (i,j) being amenable for comparison, i.e. if the pair is 'concordant'. The intuition behind time-varying concordance (Harrell *et al.* 1982) is based on the idea that the predicted survival probability for an order i evaluated at time z_i and conditioned on market features $\mathbf{x}_i$ should be lower than that of order j evaluated at the same time and conditioned on market features $\mathbf{x}_j$ if order i was filled faster than order j. In addition to time-varying concordance, another frequently used scoring is the Brier score for right-censored data (Graf *et al.* 1999), defined as

$$\beta = \frac{\hat{S}(t \mid x)^2 \mathbb{1}\{z \le t, \delta = 1\}}{\hat{G}(z)} + \frac{(1 - \hat{S}(t \mid x))^2 \mathbb{1}\{z > t\}}{\hat{G}(z)},$$

where $\hat{G}$ is the Kaplan–Meier estimate of the censoring distribution.

Time-varying concordance and Brier score are the two most common scores to evaluate models in the survival analysis literature (Lee *et al.* 2018, 2019, Zhong *et al.* 2021). However, recent work (Rindt *et al.* 2022) shows that both scoring rules (as well as a number of others) are improper, meaning that they can potentially give higher scores to wrongly fitted distributions.† Right-censored log-likelihood is a proper scoring rule, which is the key result in (Rindt *et al.* 2022).

Appendix 4. Extending the Encoder's Dilated Causal Convolutional Neural Network to *L* layers

To extend the original the DCC in the encoder, which is responsible for obtaining the corresponding queries, keys, and values in the proposed convolutional-Transformer, the causal convolutional network should be modified as follows. The first layer would perform the same operation, convolving the input LOB sequences x and the kernel k:

$$F^{(l=1)}(m) = \left(x *_p k^{(l=1)}\right)(m) = \sum_{\tau=0}^{s-1} k_\tau^{(l=1)} \cdot x_{m-p\cdot\tau},$$

where p is the dilation factor and k the convolutional filter with size $s \in \mathbb{Z}$. For each of the remaining $L-1$ layers, we define the convolution operation as

$$F^{(l)}(m) = \left(F^{(l-1)} *_d k^{(l)}\right)(m) = \sum_{\tau=0}^{s-1} k_\tau^{(l)} \cdot F_{m-p\cdot\tau}^{l-1}(m).$$

Each of the layers in this hierarchical structure defines the kernel operation as an affine function acting between layers

$$k^{(l)} : \mathbb{R}^{N_l} \longrightarrow \mathbb{R}^{N_{l+1}}, \quad 1 \le l \le L.$$

The previous equation shows how, through the use of residual connections, firstly proposed in He *et al.* (2016), the encoder's convolutional network could connect lth layer's output to $(l+1)$th layer's input, enabling the usage of deeper models with larger receptive fields to generate the hidden representation that is used by the Transformer self-attention's inputs.

Appendix 5. Monotonicity of the Decoder

The setup of the decoder network ensures that the monotonicity condition is satisfied. In particular, we consider a feed-forward network with intermediate layers denoted by h, with $1 < h < H$ (not to be confused with h_i in Section 6.2 that accredits the Transformer model's head $h_i \in H$). Each element k of the output vector of each of the nodes j has output

$$y_{j,k}^h = \tanh\left(\sum_{i=1}^{M_h} w_{i,k}^{h,j}\, x_i^h\right),$$

for an input $\mathbf{x}^h \in \mathbb{R}^{M_h}$ (ignoring the bias terms, and assuming that the input includes the response variable t) in all layers except the final one. In turn, the final layer (which contains a single node and outputs a scalar) is set up as

$$P(T \le t \mid \mathbf{X} = \mathbf{x}) = \sigma\left(\sum_{i=1}^{M_H} w_i^H\, x_i^H\right).$$

† Brier score is a proper scoring rule under the assumption of independence between censoring and covariates, as well as a perfect estimate of the censoring distribution. These assumptions do not hold in the context of limit order executions, given that orders of larger size are prone to cancellations (which are interpreted as censoring in this work) and there is not a tractable way of obtaining a perfect estimate of the distribution of cancelled orders.

Here, $w_{i,k}^{h,j}$ are the individual weight terms associated to node j, layer h and column k of the weight matrix, respectively, and $\tanh(\cdot)$ and $\sigma(\cdot)$ are the hyperbolic tangent and sigmoid functions, respectively. To enforce monotonicity of the output with respect to the response variable t, we impose $w_{i,k}^{h,j} \geq 0 \, \forall \, h \in \{1, \ldots, H\}$, because the derivative of the output of each node in the first layer is (assuming that the first element of the input is the response variable)

$$\frac{\partial y_{j,k}^{1}}{\partial t} = \tanh'\left(w_{1,k}^{1,j} t + \sum_{i=2}^{M_1} w_{i,k}^{1,j} x_i^1 \right) w_{1,k}^{1,j},$$

which requires positivity of the $w_{1,k}^{1,j}$ weights to guarantee the monotonicity condition. From the chain rule, a similar argument holds for all nodes in subsequent layers of the network. Finally, the remaining two conditions are satisfied empirically given the likelihood-based training method. We highlight that we only impose this restriction in the decoder network, which takes the response variable as input, along with the latent representation of the LOB generated by the encoder.

Appendix 6. Order Features

The evolution over 500 trades of the features considered to produce the attention heatmaps are shown in Figure A1.

Appendix 7. Kernel Sizes

Table A1 explores different kernel sizes, $s \in \{1, 2, 3, 5, 10, 25, 50\}$, for the convolutional operation of the MN-Conv-Trans model for the estimation of AAPL's survival function with a lookback window of $T = 500$ trades. Recall that a convolutional network with kernel size $s = 1$ results on canonical self-attention, in which case, there is an

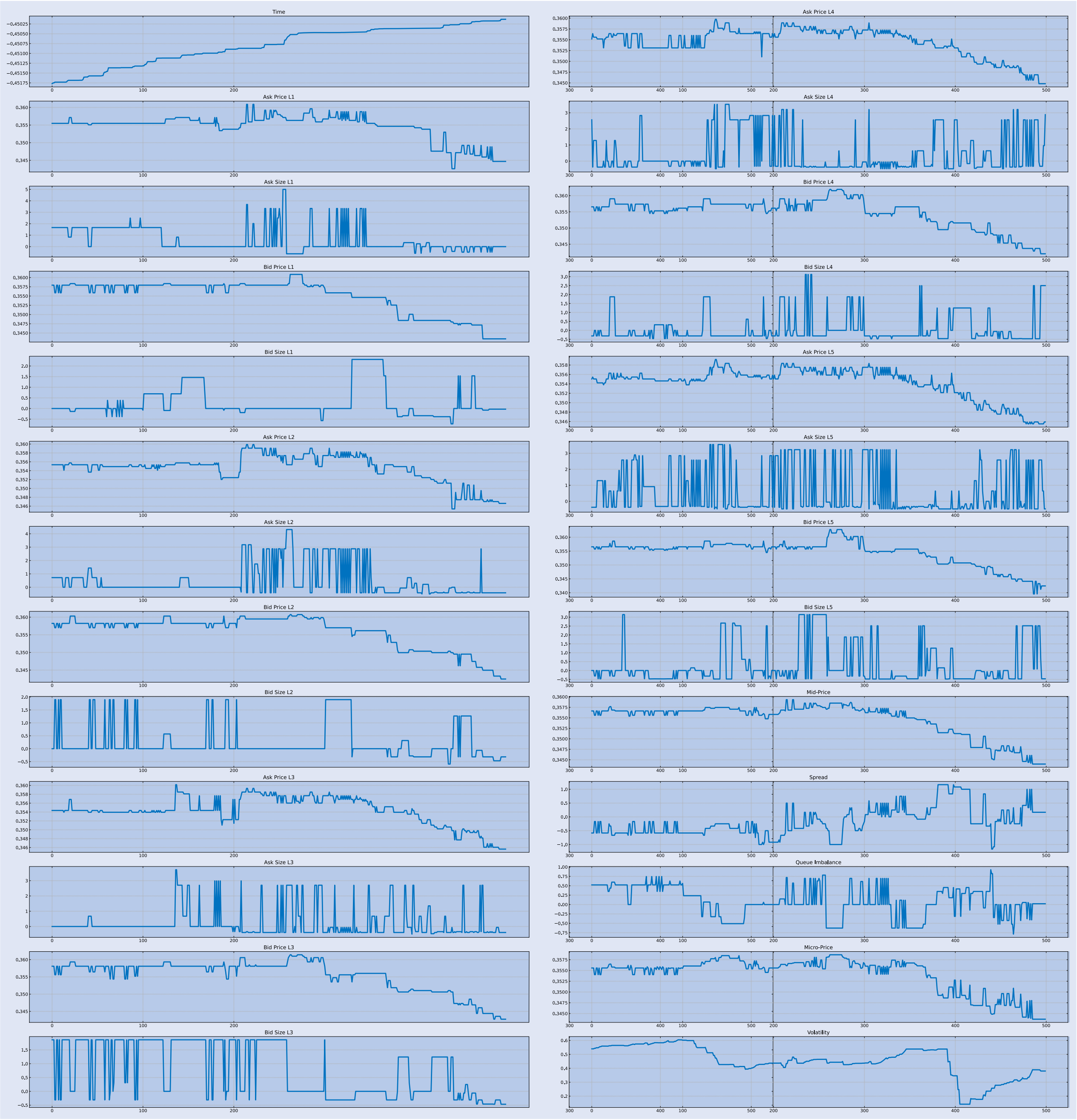


Figure A1. Evolution of order features over 500 trades.

evident decline in the negative RCLL in comparison to larger kernel sizes. There are no substantial variations in the performance among the other kernel sizes. Therefore, a value of $s = 3$ seems reasonable to avoid an unnecessary increase in parametric complexity.

Table A1. Performance variation while when different kernel's sizes for the MN-Conv-Trans DCC network.

Kernel Size	**Mean±STD Negative RCLL**
$s = 1$	3.245 ± 0.207
$s = 2$	3.184 ± 0.343
$s = 3$	3.171 ± 0.157
$s = 5$	3.189 ± 0.433
$s = 10$	3.179 ± 0.367
$s = 25$	3.196 ± 0.383
$s = 50$	3.170 ± 0.370

Note: AAPL stock with a lookback window of 500 trades.

Appendix 8. Survival Functions in Transaction Time

In this Appendix, we plot the survival functions with a change in clock from regular time to transaction time. This removes the effect of interarrival times, which makes the shapes of the survival functions more similar.

Appendix 9. Performance of Order Flow Representations

Appendix 10. Results for Cox Proportional Hazards model

We carry out an additional experiment to test the effectiveness of linear models in estimating the survival function. Specifically, we test the performance of the Cox Proportional Hazards model Cox (1972), which is one of the most widely used models in the survival analysis literature, see Table A3.

Appendix 11. Calculation of the Shapley Values

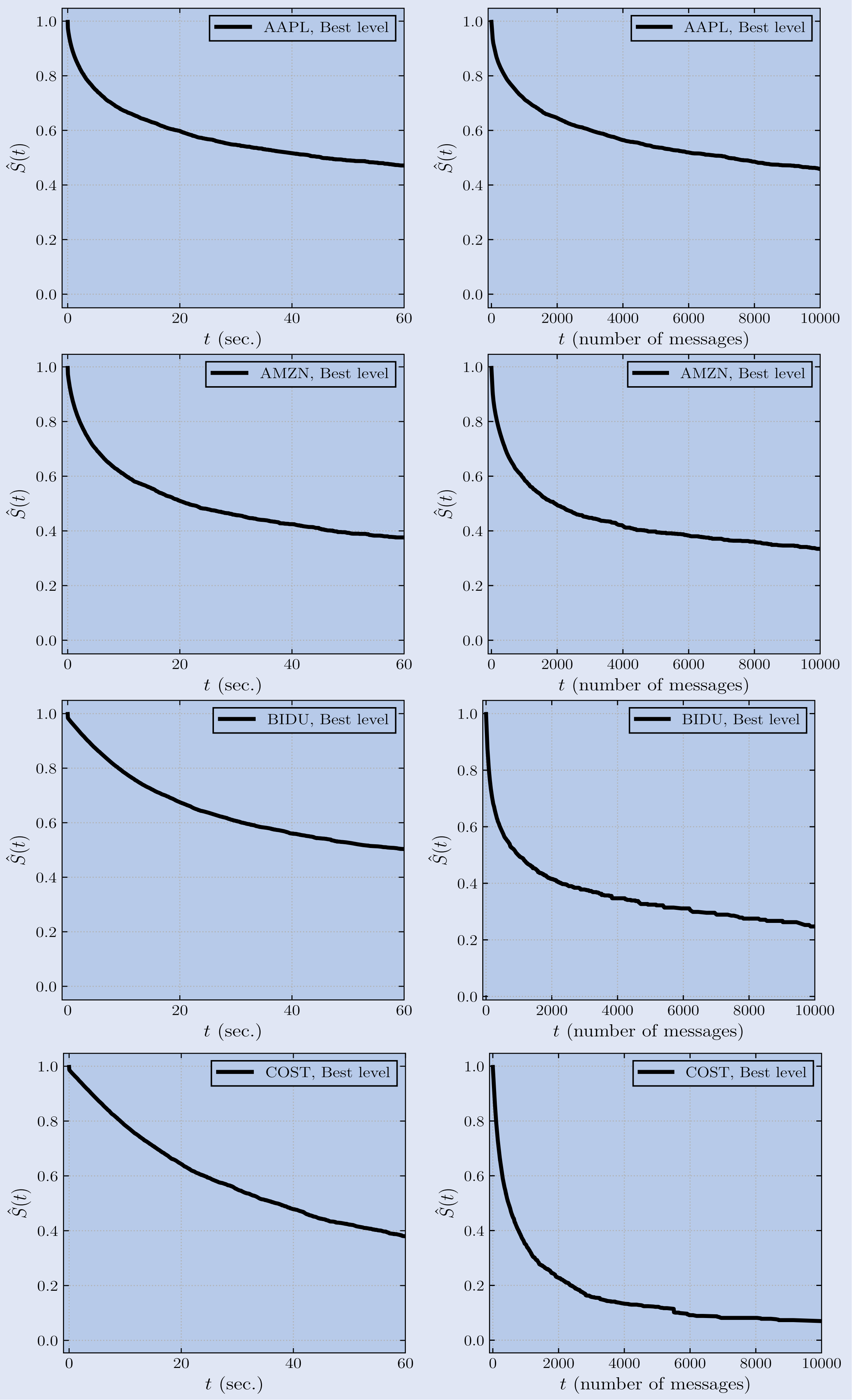
To calculate the Shapley values, we define $\mathbf{S} \subseteq \mathbf{F}$ as all possible feature subsets, where $\mathbf{F}$ is the set of all features. Then, to measure

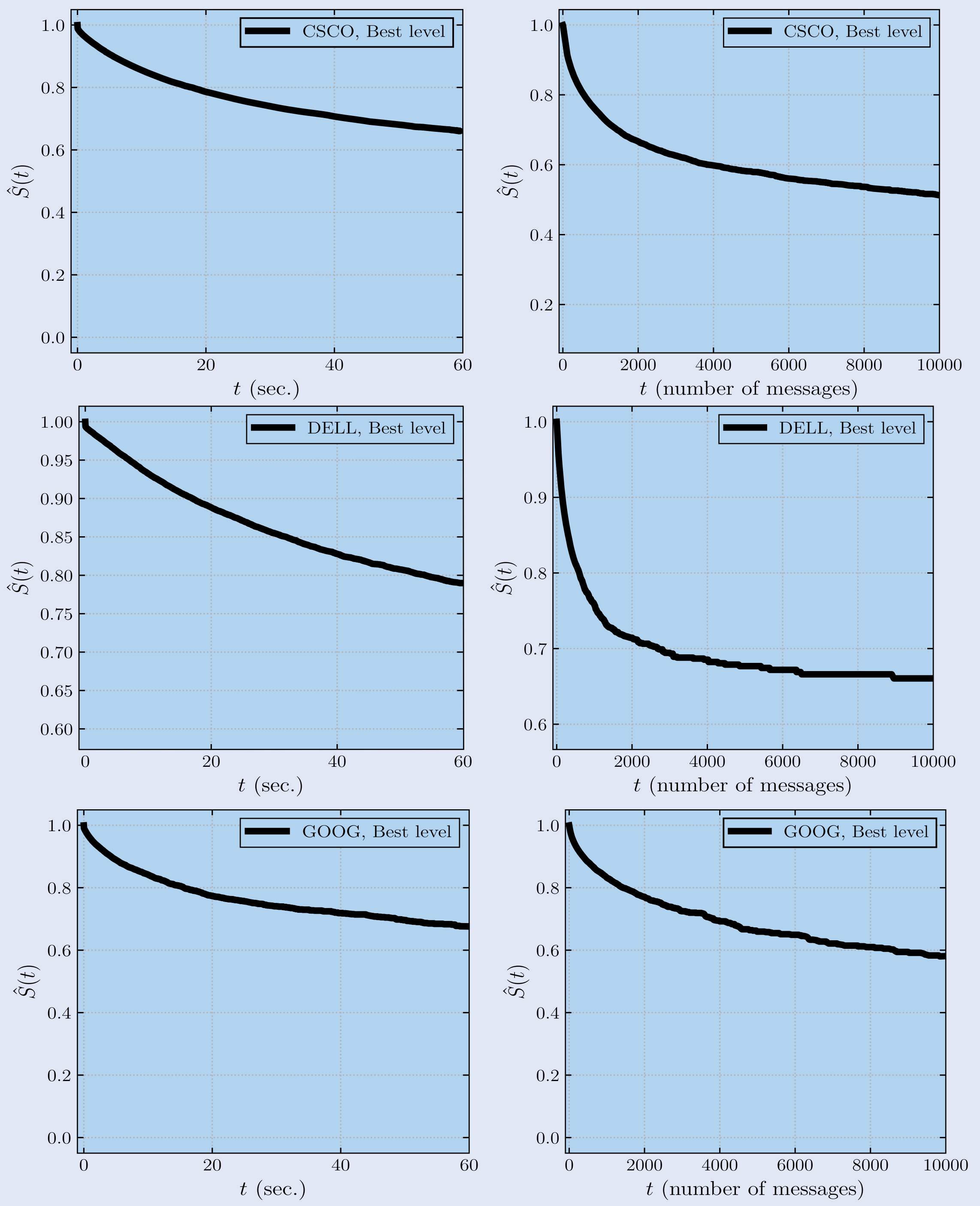
Table A2. Models evaluated using the negative right-censored log-likelihood for a lookback window of $T = 500$ on the order flow data, and percentage improvement over the MN-MLP model, for each of the evaluated models.

Mean±STD Negative RCLL									
	AAPL	**AMZN**	**BIDU**	**COST**	**CSCO**	**DELL**	**GOOG**	**INTC**	**MSFT**
DeepSurv	8.053 ± 0.019	8.346 ± 0.011	7.236 ± 0.051	9.712 ± 0.047	8.441 ± 0.022	8.138 ± 0.013	8.242 ± 0.004	6.211 ± 0.057	9.117 ± 0.043
DeepHit	10.015 ± 0.091	10.102 ± 0.024	9.627 ± 0.064	10.036 ± 0.043	9.843 ± 0.083	9.247 ± 0.051	10.142 ± 0.042	9.795 ± 0.157	10.112 ± 0.062
MN-MLP	10.535 ± 0.339	10.801 ± 0.347	13.518 ± 0.497	12.502 ± 0.431	13.158 ± 0.495	12.160 ± 0.451	10.800 ± 0.311	12.938 ± 0.422	10.963 ± 0.326
MN-CNN	5.176 ± 0.343	5.427 ± 0.173	7.921 ± 0.162	6.741 ± 0.224	7.434 ± 0.212	6.754 ± 0.291	5.469 ± 0.621	7.331 ± 0.285	5.646 ± 0.163
MN-LSTM	3.746 ± 0.136	4.838 ± 0.248	7.320 ± 0.613	5.941 ± 0.195	7.193 ± 0.129	6.487 ± 0.198	4.890 ± 0.538	8.583 ± 0.387	4.352 ± 0.231
MN-Conv-Trans	**3.158 ± 1.138**	**3.546 ± 1.002**	**6.107 ± 1.008**	**5.220 ± 1.013**	**6.211 ± 1.103**	**5.245 ± 1.162**	**3.595 ± 1.001**	**5.997 ± 1.173**	**3.772 ± 0.918**
Improvement over MN-MLP (%)									
	AAPL	**AMZN**	**BIDU**	**COST**	**CSCO**	**DELL**	**GOOG**	**INTC**	**MSFT**
DeepSurv	23.56	22.73	46.47	22.32	35.85	33.08	23.69	51.99	16.84
DeepHit	4.94	6.47	28.78	19.72	25.19	23.96	6.09	24.29	7.76
MN-MLP	-	-	-	-	-	-	-	-	-
MN-CNN	50.87	49.75	41.40	46.08	43.50	44.46	49.36	43.34	48.50
MN-LSTM	64.44	55.21	45.85	52.48	45.33	46.65	54.72	33.66	60.30
MN-Conv-Trans	**70.02**	**67.17**	**54.82**	**58.25**	**52.80**	**56.87**	**66.71**	**53.65**	**65.59**

Table A3. Percentage improvement of the MN-MLP model over Cox PH for all assets.

Improvement over MN-MLP (%)									
	AAPL	**AMZN**	**BIDU**	**COST**	**CSCO**	**DELL**	**GOOG**	**INTC**	**MSFT**
CoxPH	– 28.88	– 19.91	– 8.35	– 4.47	– 42.91	– 41.68	– 11.19	– 23.87	– 19.79


AAPL, Best level
AMZN, Best level
BIDU, Best level
COST, Best level
$\hat{S}(t)$
t (sec.)
t (number of messages)
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
0
2000
4000
6000
8000
10000



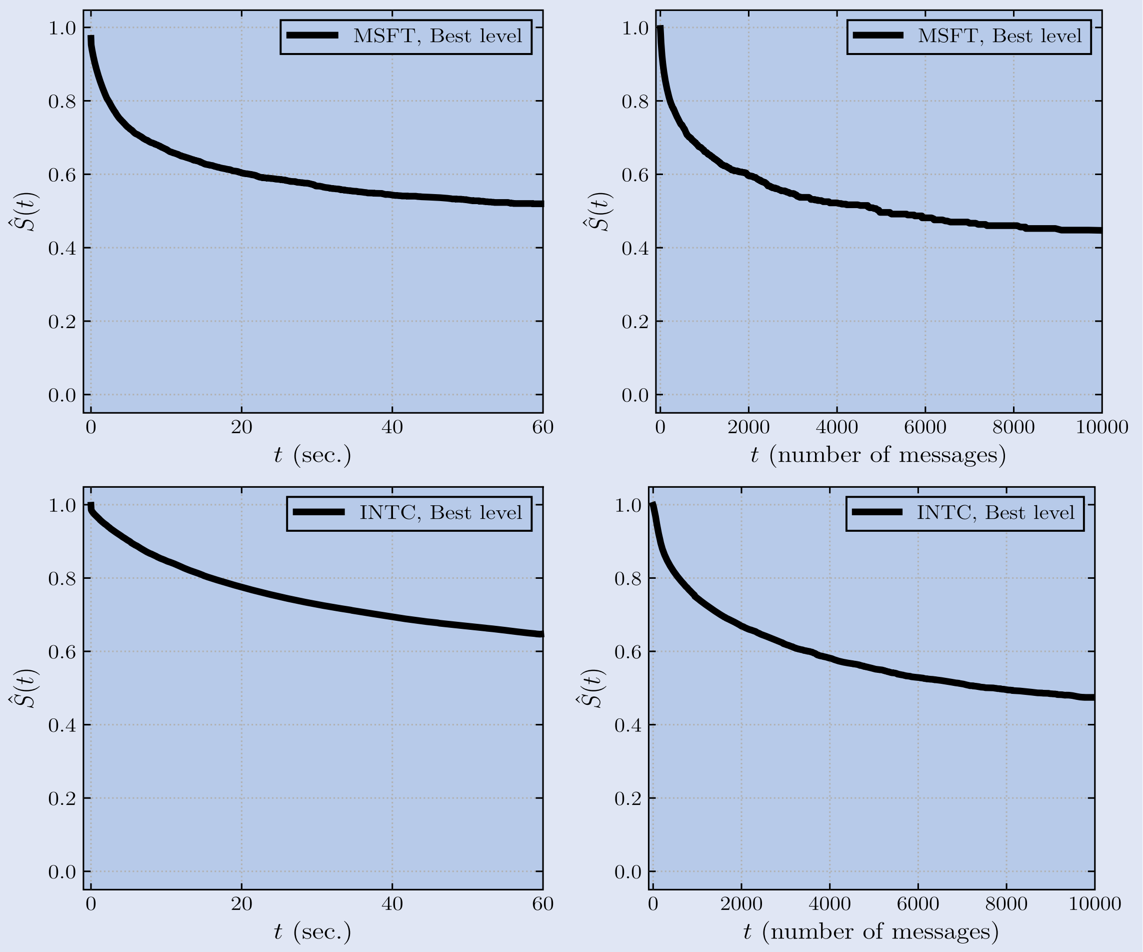
the importance per feature, we first integrate over multiple background samples to obtain the expected model output $\mathbb{E}[\hat{f}_{\mathbf{S}}(t\,|\,\mathbf{x})]$, where $\hat{f}_{\mathbf{S}}(t\,|\,\mathbf{x})$ denotes the model's prediction while using all the available features. Next, we approximate the Shapley values as the difference between the expected output of the model and the predicted values, i.e. $\hat{f}_{\mathbf{S}}(t\,|\,\mathbf{x}) - \mathbb{E}[\hat{f}(t\,|\,\mathbf{x})]$. Therefore, the contribution of the i^{th} feature can be stated as:

$$C_i = \hat{f}_{\mathbf{S}}(t\,|\,\mathbf{x}) - \hat{f}_{\mathbf{S}\setminus\{i\}}(t\,|\,\mathbf{x}),$$

where $\hat{f}_{\mathbf{S}\setminus\{i\}}(x)$ is the model's prediction without using the i^{th} feature, whose Shapley value is given by

$$\partial_i = \frac{1}{n!}\sum_{p=0}^{n} p!\,(n-p)!\,C_i,$$

where n is the total number of features and p is the number of features present in the input sample.


MSFT, Best level
$\hat{S}(t)$
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
t (sec.)
MSFT, Best level
0
2000
4000
6000
8000
10000
t (number of messages)
INTC, Best level
t (sec.)
INTC, Best level
t (number of messages)