Deep Attentive Time Series Modelling for Quantitative Finance

by

Fernando Moreno Pino

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in

Multimedia and Communications

Universidad Carlos III de Madrid

Advisors:

Antonio Artés Rodríguez

Pablo Martínez Olmos

February 2023

Todo pasa y todo queda,
pero lo nuestro es pasar,
pasar haciendo caminos,
caminos sobre la mar.

— Antonio Machado

Acknowledgements

Me veo obligado a escribir unas palabras de agradecimiento a todas aquellas personas que me han acompañado en esta aventura que ha sido completar una tesis doctoral. He andado un camino que, poco tiempo antes de dar el primer paso, me era por completo desconocido, un camino que, por exigir lo mejor de ti mismo en cada zancada, se hace más fácil acompañado.

Debo agradecer a mis directores, Antonio Artés y Pablo M. Olmos, su inagotable paciencia. Sin la confianza depositada por ellos, este viaje difícilmente se habría materializado. A todos mis compañeros de laboratorio: siempre es más sencillo tener la convicción de seguir avanzando cuando estás rodeado de personas que la comparten y te afianzan en ella con su esfuerzo y dedicación. No puedo además pasar por alto el impulso del que, durante mi estancia en Oxford, tantos jóvenes y apasionados investigadores me hicieron partícipe, haciéndome ver cómo este inconmensurable esfuerzo merece la pena.

Es necesario además echar la vista atrás para poder apreciar el cariño de todas las personas con cuyo apoyo he contado en cada momento del camino. En primer lugar, a todos mis amigos, empezando por aquellos que me acompañan desde la más tierna infancia. Tener la certeza de poder contar con ellos cada vez que vuelvo al sur del sur, a casa, me hace estar eternamente agradecido. En segundo lugar, a todos aquellos con los que tuve la suerte de recorrer las calles de Madrid, compartiendo risas y forjando recuerdos que me acompañarán toda una vida. Por último, no puedo no agradecer en estas líneas a Luis y Magdalena su inconmensurable ayuda durante estos últimos años. Su generosidad y apoyo diario se han tornado de vital importancia.

Si antes mencionaba la necesidad de volver la vista atrás, ahora he de volverla hacia el origen. Gracias a mi familia, que siempre me ha acompañado. Especialmente a mis abuelos, Antonio y Daniela, vuestra alegría por vivir y vuestra forma de compartirla son una oda a la vida. A Andrés y Alfonsa, vuestro colosal esfuerzo vitalicio son un ejemplo a seguir día a día.

A Laura, a quien no puedo expresar con palabras lo agradecido que estoy de que nuestros caminos se cruzasen. Tu amor ha sido un pilar fundamental en estos años. Tu paciencia y comprensión me han sostenido y han hecho posible que esta aventura llegase a buen puerto, has sido el impulso que, a diario, me ha motivado a seguir adelante. Hemos tenido la suerte de experimentar de forma compartida

como nuestros horizontes se ampliaban durante estos años pero, sin duda, el mejor regalo es haber podido construir un hogar juntos. No puedo esperar para ver lo que el futuro nos depara juntos.

Por último, a mis padres, las personas que desde el primer instante me han llevado de la mano. Gracias por hacerme ver el valor de la educación. Gracias por vuestra eterna paciencia, por vuestro inconmensurable trabajo. Nunca podré estar lo suficientemente agradecido. Gracias por todo.

Published and presented contents

The following list of works is a short bibliography of journal articles, preprints, and working papers included as part of this thesis. In each contribution, its full or partial presence in this thesis' chapters is indicated. The material from this source included in this thesis is not singled out with typographic means and references.

1. Moreno-Pino, F., & Zohren, S. (2022). DeepVol: Volatility Forecasting from High-Frequency Data with Dilated Causal Convolutions. arXiv preprint arXiv:2210.04797. – This article is *fully included* in Chapter 4.

2. Moreno-Pino, F., Olmos, P. M., & Artés-Rodríguez, A. (2023). Deep autoregressive models with spectral attention. In Pattern Recognition, 133, 109014. – This article is *fully included* in Chapter 3.

3. Arroyo, A.*, Cartea, A., Moreno-Pino, F.*, Zohren, S. (2023). Deep Survival Analysis in the LOB: A Convolutional-Transformer Approach to Estimating Fill Probabilities. Manuscript in preparation. – This article is *fully included* in Chapter 5.

*Denotes co-first authors with equal contributions.

Other research merits

The following list of works is a short bibliography of conference papers, workshops, and preprints published during the PhD studies but *not* included as part of this thesis.

1. Ríos-Muñoz, G. R., Moreno-Pino, F., Soto, N., Olmos, P. M., Artés-Rodríguez, A., Ferández-Avilés, F., & Arenal, A. (2020, September). Hidden Markov models for activity detection in atrial fibrillation electrograms. In 2020 Computing in Cardiology (pp. 1-4). IEEE. – This article is *not included* as part of this thesis.

2. Moreno-Pino, F., Sükei, E., Olmos, P. M., & Artés-Rodríguez, A. (2022). PyHHMM: A Python Library for Heterogeneous Hidden Markov Models. arXiv preprint arXiv:2201.06968. – This article is *not included* as part of this thesis.

3. Moreno-Pino, F., Martínez-García, M., Olmos, P. M., & Artés-Rodríguez, A. (2022). Heterogeneous Hidden Markov Models for Sleep Activity Recognition from Multi-Source Passively Sensed Data. In 2022 Machine Learning for Health (ML4H). – This article is *not included* as part of this thesis.

4. Martínez-García, M.*, Moreno-Pino, F.*, Olmos, P. M., & Artés-Rodríguez, A. (2023). Sleep Activity Recognition and Characterization from Multi-Source Passively Sensed Data. arXiv preprint arXiv:2301.10156.

*Denotes co-first authors with equal contributions.

Abstract

Time series modelling and forecasting is a persistent problem with extensive implications in scientific, business, industrial, and economic areas. This thesis's contribution is twofold. Firstly, we propose a novel probabilistic time series forecasting methodology that introduces the use of Fourier domain-based attention models, merging classic signal processing spectral filtering techniques with machine learning architectures. Secondly, we take advantage of the abundance of financial intraday high-frequency data to develop deep learning-based solutions for modelling financial time series. Machine learning methods can potentially enhance the performance of traditional methodologies used by practitioners. Deep neural networks' feature extraction capabilities, which can benefit from the rising accessibility of high-frequency data, and attention mechanisms, which help to model temporal patterns, are mostly to blame for this.

Concerning our first major contribution, this thesis empirically demonstrates that spectral domain-based machine learning models can learn the properties of time series datasets and integrate this information to improve the forecasting accuracy. Simultaneously, Fourier domain-based models alleviate some of the inconveniences commonly associated with deep autoregressive models. These architectures, prone to prioritising recent past data, often ignore critical global information not contained in previous time steps. Additionally, they are susceptible to error accumulation and propagation and may not yield illustrative results. The proposed model, the Spectral Attention Autoregressive Model (SAAM), mitigates these problems by combining deep autoregressive models with a Spectral Attention (SA) module. This module uses two attention models operating over the Fourier domain representation of the time series' embedding. Through spectral filtering, SAAM differentiates between the components of the frequency domain that should be considered noise and subsequently filtered out, and the global patterns that are relevant and should be incorporated into the predictions. Empirical evaluation proves how the proposed Spectral Attention module can be integrated into various deep autoregressive models, consistently improving the results of these base architectures and achieving state-of-the-art performance.

Afterwards, this thesis shifts toward showcasing the benefits of machine learning solutions in two different quantitative finance scenarios, proving how attention-based

deep learning approaches compare favourably to classic parametric-based models and providing solutions for various algorithmic and high-frequency trading problems.

In the context of volatility forecasting, which plays a central role among equity risk measures, we show that Dilated Causal Convolutional-based neural networks offer significant performance gains compared to well-established volatility-oriented parametric models. The proposed model, called DeepVol, showcases how data-driven models can avoid the limitations of classical methods by taking advantage of the abundance of high-frequency data. DeepVol outperforms baseline methods while exhibiting robustness in the presence of volatility shocks, showing its ability to extract universal features and transfer learning to out-of-distribution data. Consequently, data-driven approaches should be carefully considered in the context of volatility forecasting, as they can be instrumental in the valuation of financial derivatives, risk management, and the formation of investment portfolios.

Finally, this thesis presents a survival analysis model for estimating the distribution of fill times for limit orders posted in the Limit Order Book (LOB). The proposed model, which does not make assumptions about the underlying stochastic processes, employs a convolutional-Transformer encoder and a monotonic neural network decoder to relate the time-varying features of the LOB to the distribution of fill times. It grants practitioners the capability of making informed decisions between market orders and limit orders, which in practice entails a trade-off between immediate execution and price premium. We offer an exhaustive comparison of the survival functions resulting from different order placement strategies, offering insight into the fill probability of orders placed within the spread. Empirical evaluation reveals the superior performance of the monotonic encoder-decoder convolutional-Transformer compared to state-of-the-art benchmarks, leading to more accurate predictions and improved economic value.

Resumen

El modelado y predicción de series temporales es un problema persistente con amplias implicaciones en áreas científicas, comerciales, industriales y económicas. Esta tesis propone una doble contribución en este ámbito. En primer lugar, formulamos una novedosa metodología para la predicción probabilística de series temporales que introduce el uso de modelos de atención basados en el dominio de la frecuencia, con la transformada de Fourier desempeñando un papel fundamental. El modelo propuesto fusiona técnicas clásicas de filtrado espectral, pertenecientes al campo del procesado de señal, con modelos de aprendizaje automático. En segundo lugar, desarrollamos varias soluciones basadas en aprendizaje profundo para el modelado de datos financieros intradía, aprovechando la cada vez mayor disponibilidad de los mismos. Los métodos de aprendizaje automático poseen el potencial para mejorar los resultados obtenidos por las metodologías clásicas que los profesionales del ámbito de las finanzas cuantitativas acostumbran a utilizar. La capacidad de extracción de características de las redes neuronales, que pueden aprovechar la creciente accesibilidad a los datos financieros de alta frecuencia, y el uso de los mecanismos de atención para el modelado temporal, son los principales responsables de ésto.

En lo relativo a la primera de las contribuciones mencionadas anteriormente, es decir, el uso de modelos de aprendizaje automático que operan sobre el dominio de la frecuencia, esta tesis demuestra de manera empírica que los modelos de aprendizaje profundo basados en el dominio espectral pueden aprender de forma más eficiente las propiedades de las series temporales a predecir. De esta manera, logran mejorar la precisión de las predicciones a la vez que solventan varios de los problemas que lastran el rendimiento de los modelos autoregresivos. Estas arquitecturas son propensas a sobreponderar los datos del pasado inmediato, ignorando a menudo valiosa información global que no está contenida en estas observaciones recientes. Además, son susceptibles a la acumulación y propagación de errores. Finalmente, los resultados que producen son difícilmente interpretables. Proponemos un nuevo modelo, llamado "Spectral Attention Autoregressive Model"(SAAM) (Modelo Autorregresivo con Atención Espectral), que mitiga estos problemas combinando modelos autorregresivos basados en aprendizaje profundo con un módulo de Atención Espectral. Dicho módulo contiene dos modelos de atención que operan sobre la representación en el dominio de Fourier del "embedding" obtenido a partir de la serie

temporal a predecir. Usando técnicas de filtrado espectral, SAAM diferencia entre los componentes del espectro que deben ser considerados ruido, y por consiguiente deben ser filtrados, y aquellos patrones globales que son relevantes y deben ser incorporados en las predicciones. Mediante una exhaustiva evaluación empírica, demostramos que nuestro modelo de Atención Espectral puede ser integrado en diversos modelos autorregresivos que forman parte del estado del arte actual, mejorando de forma consistente los resultados obtenidos.

En lo relativo a la segunda contribución principal de esta tesis doctoral, demostramos los beneficios que las metodologías de aprendizaje automático basadas en modelos de atención pueden aportar en dos problemas propios de las finanzas cuantitativas. Diversos experimentos demuestran cómo este tipo de modelos pueden mejorar los resultados obtenidos por los modelos clásicos empleados en este campo, proporcionando soluciones innovadoras para diversos problemas recurrentes dentro del trading algorítmico de alta frecuencia.

La predicción de volatilidad en mercados financieros es el primero de estos problemas en ser abordado en la presente tesis. La estimación de volatilidad desempeña un papel central entre las medidas de riesgo utilizadas en los mercados de renta variable. En esta tesis demostramos que las redes neuronales basadas en "Dilated Causal Convolutions" (Convolucionales Causales Dilatadas) ofrecen ganancias significativas en comparación con los modelos paramétricos clásicos desarrollados única y exclusivamente para predicción de volatilidad. El modelo propuesto, llamado DeepVol, evidencia que el uso de modelos de aprendizaje profundo puede evitar las numerosas limitaciones propias de los métodos clásicos, logrando aprovechar la abundancia de datos de alta frecuencia para aprender las funciones deseadas. DeepVol supera a todos los modelos de referencia usados como comparativa, a la vez que exhibe robustez en períodos que contienen shocks de volatilidad, demostrando su capacidad para extraer características universales comunes a diferentes instrumentos financieros. Los resultados obtenidos en esta parte de la tesis nos llevan a concluir que los modelos de aprendizaje automático deben considerarse cuidadosamente en el contexto de predicción de volatilidad, pudiendo ser especialmente relevantes en la valoración de derivados financieros, gestión del riesgo, y creación de carteras de inversión.

Para terminar, esta tesis presenta un modelo de análisis de supervivencia para estimar la distribución de probabilidad de ejecución subyacente a órdenes limitadas publicadas en el conocido como "Limit Order Book" (Libro de Órdenes Limitadas). El modelo propuesto, que no necesita partir de suposiciones sobre los procesos estocásticos subyacentes, emplea una arquitectura codificador/decodificador que utiliza un "Transformer" convolutional para codificar la información del libro de órdenes y una red monotónica que decodifica la función de supervivencia a estimar.

Este modelo, basado en aprendizaje profundo, relaciona por tanto las características del libro, variables en el tiempo, con la distribución de tiempos de ejecución. Nuestro modelo otorga a los profesionales del trading cuantitativo la capacidad de tomar decisiones informadas entre las órdenes de mercado y las órdenes limitadas, lo que en la práctica implica encontrar el equilibro entre una ejecución inmediata y una prima en el precio. Para evaluar el rendimiento de la metodología propuesta, ofrecemos una comparación exhaustiva de las funciones de supervivencia resultantes de diferentes estrategias de colocación de órdenes. Esta evaluación empírica revela un rendimiento superior por parte de nuestra arquitectura en comparación con el estado del arte actual. Por tanto, el empleo del modelo propuesto en esta tesis conduce a predicciones más exactas con un valor económico añadido.

Contents

List of Figures

List of Tables

List of Abbreviations

ADAM	Adaptive Moment Estimation Algorithm
AFT	Accelerated Failure Time
AGARCH	Asymmetric Generalized Auto Regressive Conditional Heteroskedasticity
APARCH	Asymmetric Power Auto regressive Conditional Heteroskedasticity
AR	Auto Regressive
ARCH	Auto regressive Conditional Heteroskedasticity
ARIMA	Auto Regressive Integrated Moving Average
ARMA	Auto Regressive Moving Average
ASGD	Averaged Stochastic Gradient Descent
CNN	Convolutional Neural Network
CDF	Cumulative Density Function
DNN	Deep Neural Network
DSSM	Deep State Space Model
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
EGARCH	Exponential Generalized Auto Regressive Conditional Heteroskedasticity
FIGARCH	Fractionally Integrated Generalized Auto Regressive Conditional Heteroskedasticity
GRU	Gated Recurrent Unit
GARCH	Generalized Auto Regressive Conditional Heteroskedasticity
HAR	Heterogeneous Autoregressive
HEAVY	High-Frequency-Based Volatility

IGARCH . . . . . Integrated Generalized Auto Regressive Conditional Heteroskedasticity

LO . . . . . Limit Order

LOB . . . . . Limit Order Book

L-BFGS . . . . . Limited Memory BFGS

LSTM . . . . . Long Short-Term Memory

ML . . . . . Machine Learning

MO . . . . . Market Order

ME . . . . . Maximum Error

MAE . . . . . Mean Absolute Error

MAPE . . . . . Mean Absolute Percentage Error

MedAE . . . . . Median Absolute Error

MN-Conv-Trans . . . . Monotonic encoder-decoder convolutional-Transformer

MN-CNN . . . . . Monotonic-Convolutional Neural Network

MN-LSTM . . . . . Monotonic-Long Short-Term Memory

MN-MLP . . . . . Monotonic-Multi Layer Perceptron

MA . . . . . Moving Average

MLP . . . . . Multi Layer Perceptron

NASDAQ . . . . . National Association of Securities Dealers Automated Quotations

NLP . . . . . Natural Language Processing

ND . . . . . Normalised Deviation

PSD . . . . . Power Spectral Density

QL . . . . . Quantile Loss

QLIKE . . . . . Quasi Log-Likelihood

RNN . . . . . Recurrent Neural Network

RCLL . . . . . Right Censored Log-Likelihood

RMSE . . . . . Root Mean Square Error

SA . . . . . Spectral Attention

SAAM . . . . . Spectral Attention Autoregressive Model

SSM . . . . . State Space Model

SMAPE . . . . . Symmetric Mean Absolute Percentage Error

TRMF . . . . . Temporal Regularized Matrix Factorization

TARCH . . . . . Threshold Auto regressive Conditional Heteroskedasticity

TGARCH . . . . . Threshold Generalized Auto Regressive Conditional Heteroskedasticity

1 Introduction

Contents

Time series modelling and forecasting, which consists in analysing historical signal patterns to predict future outcomes, is an important problem with scientific, business, industrial, and economic applications, playing an essential role in daily life. Several fields benefit from time series forecasting, such as climate prediction (H. Hu et al., 2022), forecasting of energy consumption and demand (Pang et al., 2022), product demand and supply (Merkuryeva et al., 2019), and finance applications of these models (Moreno-Pino and Zohren, 2022), toward which we focus on Chapters 4 and 5 of this thesis.

Early approaches to solving time series forecasting problems rely on statistical models, such as State Space Models (SSMs) (Durbin and Koopman, 2012), exponential smoothing (R. Hyndman et al., 2008), matrix factorisation methods (H.-F. Yu et al., 2016), or Auto Regressive Integrated Moving Average (ARIMA)

(G. E. Box and Gwilym M Jenkins, 1970), which has become one of the most popular solutions and produces its predictions as a weighted sum of past observations, thus making the model vulnerable to error accumulation, a common problem to most autoregressive architectures. We refer the readers to G. E. P. Box and G. M. Jenkins (1968), Hamilton (1994), and Lütkepohl (2005) for additional information regarding classic techniques.

Regardless, all these classic approaches share a number of weaknesses. They make linear assumptions on the data, which, together with their limited scalability, renders them unsuitable for modern large-scale forecasting tasks. Furthermore, they incorporate prior knowledge about time series composition, like trends or seasonality patterns present in the data, which requires manual feature engineering and model design by domain experts in order to achieve good results (A. C. Harvey, 1990). Moreover, they do not usually share a global set of parameters among the different time series, which implies bad generalisation and poor performance aside from single-linear time series prediction.

As the amount of data available has grown rapidly, machine learning techniques have become increasingly prevalent for the purpose of predicting time series in various scenarios. As a result, Deep Neural Networks (DNN) (Sutskever et al., 2014) have arisen as an alternative solution for time series forecasting. They are able to model non-linear temporal patterns, easily identify complex structures across time series, efficiently extract higher order features, and allow a data-driven approach that requires little to no human feature engineering (Giuliari et al., 2021). Among deep learning-based models, Recurrent Neural Networks (RNNs) (Funahashi and Nakamura, 1993) and Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) have achieved good results in temporal modelling. More recently, attention models (Bahdanau et al., 2015) have been used by these recurrent architectures to selectively focus on some segments of the data while making predictions, e.g., in machine translation, only certain words in the input sequence may be relevant for predicting the next word. To do so, these models use an inductive bias that connects each token in the input through a relevance-weighted basis of

every other token. The idea behind recurrent attention leads to Transformers models (Vaswani et al., 2017), which have become one of the most popular methods with respect to the problem of time series forecasting. Initially introduced for Natural Language Processing (NLP), Transformers proposed a completely new architecture where a self-attention mechanism is used to process data sequences. In Chapter 2, an in-depth and thorough review of both classic and machine learning-based methods within the context of time series modelling is presented.

Nevertheless, despite all these significant advancements in machine learning architectures for time series modelling and forecasting, there has been a lack of integration with the field of time series analysis, and there is potential for improved forecasting accuracy by combining neural network architectures with classic signal processing techniques. As we detail in Chapter 3, the ideas put forward in Moreno-Pino (2021) demonstrate the potential for this approach through the alignment of neural network architectures with traditional frequency domain-based filtering techniques. This integration can lead to the development of more robust and accurate models, capable of effectively handling non-stationary and highly complex time series data. Furthermore, combining these techniques can also lead to a better understanding of the underlying mechanisms that govern time series datasets, enabling the development of more interpretable models.

Regarding the application of these machine learning-based models, the rapid growth in data availability over time has made them part of the state-of-the-art in different domains, as previously referenced. In the course of this Ph.D. studies, we made research advances in the areas of human behaviour modelling (Ríos-Muñoz et al., 2020; Moreno-Pino, Sükei, et al., 2022; Moreno-Pino, Martínez-García, et al., 2022; Martínez-García et al., 2023) as well as within the quantitative finance field (Moreno-Pino and Zohren, 2022). In this thesis, we focus on the latter.

The discipline of finance has undergone a substantial transformation in the last decades, and it exhibits different challenges for the inclusion of machine learning techniques. The transition from traditional, broker-based trading to online electronic platforms greatly lowered transaction costs and enhanced market liquidity, leading

to the rise of systematic trading. Among financial trading methods, algorithm trading has been growing rapidly across all types of financial instruments in recent years, accounting for a substantial proportion of trading volume in U.S equity markets of approximately 73% (Treleaven et al., 2013).

While classic econometric methods have relied on the assumption that financial time series can be modelled by parametric stochastic processes, the behaviour of stock returns is characterised by a high degree of non-linearity and non-stationarity (Sirignano and Cont, 2019), breaking these assumptions. In this highly complex scenario, neural networks arise as a methodology able to model non-linear relationships in high-dimensional data, making them an excellent candidate to model financial time series as they can approximate any continuous function, as stated by the universal approximation theorems (Csáji et al., 2001; Y. Lu and J. Lu, 2020). In Chapters 4 and 5, we study the use of machine learning architectures for modelling high-frequency financial data in different quantitative finance scenarios.

1.1 Motivation

This thesis's motivation is twofold. Firstly, we address the integration of classic signal processing techniques into deep learning models to solve a variety of problems associated with deep autoregressive models. Despite the numerous advantages of using deep learning-based models for temporal modelling and forecasting, such as scalability, improved accuracy, and feature learning, deep autoregressive models are often hindered by various limitations that restrict their usability. Some of these constraints are:

- Heavy reliance on recent past data for making predictions about the future of the time series, potentially ignoring valuable global information that is not captured in previous forecasts.

- Potential for error accumulation and propagation (Cheng et al., 2006), similar to traditional time series forecasting methods, which can impact their accuracy.

- Lack of interpretability, making it difficult to understand how the models reach their predictions (X. Bai et al., 2021).

In Chapter 3, we show that these problems can be partially alleviated by incorporating signal processing filtering techniques into the autoregressive models that perform the time series modelling and forecasting, using Fourier domain-based deep learning models to this end.

Secondly, we aim to take advantage of the abundance of financial high-frequency data to develop deep learning-based solutions for modelling and forecasting financial time series. To this end, we propose different neural network architectures to tackle two highly relevant problems in the field of quantitative finance: volatility forecasting (Brailsford and Faff, 1996), and the estimation of fill probabilities in the Limit Order Book (LOB) (Cho and Nelling, 2000). Conventional approaches to these problems typically rely on classic parametric models, whose limitations have already been mentioned. In contrast, machine learning-based approaches can leverage the abundance of available intraday high-frequency data to construct data-driven models. In Chapters 4 and 5, we propose two novel deep learning-based solutions for the problems of volatility forecasting and limit order fill times estimation. We demonstrate that these machine learning-based solutions can yield more accurate predictions than traditional methodologies, leading to improved risk measures and more accurate estimates of limit orders' survival functions, respectively.

1.2 Contribution & Outline

In alignment with the research objectives outlined in the preceding section, the methods presented in this thesis are organised into chapters based on their primary area of contribution.

1.2.1 Spectral Domain-based Deep Learning

In Chapter 3, we propose a novel methodology for neural probabilistic time series forecasting that marries signal processing filtering techniques with deep learning-based autoregressive models. This is accomplished using an attention mechanism

that operates over the frequency domain representation of the original time series, denoted as the Spectral Attention module. Using a Fourier-based latent space allows the integration of both local spectral filtering and global patterns incorporation during the forecast process. Two attention models operate over the embedding's spectral domain representation to determine, at every time instant and for each time series, which components of the frequency domain should be considered noise and hence be filtered out, and which global patterns are relevant and should be incorporated into the predictions.

1.2.2 Volatility Forecasting from High-Frequency Data

In Chapter 4, we take advantage of the automatic feature extraction inherent to DNNs to address the problem of volatility forecasting from a purely data-driven perspective. Volatility forecasts play a central role among equity risk measures. Besides traditional statistical models, modern forecasting techniques based on machine learning can be employed when treating volatility as a univariate, daily time series. However, econometric studies have shown that increasing the number of daily observations with high-frequency intraday data helps to improve volatility predictions. To this end, we propose a deep learning model based on hierarchies of Dilated Causal Convolutions (DCC) to forecast day-ahead realised volatility from high-frequency data.

1.2.3 A Data-Driven Approach to Estimating Fill Probabilities in the Limit Order Book

In Chapter 5, a deep learning method for estimating the fill probabilities of limit orders posted in the LOB is presented. This novel model, which is tailored to the problem of survival analysis from time series literature, presents an encoder-decoder architecture that relates the time-varying features of the limit orders to the distribution of the orders' fill times. A Convolutional-Transformer encoder models the complex dependencies and interactions within the LOB data, compressing useful information into a lower-dimensional latent representation, which is then used by a

monotonic decoder to predict a monotonic decreasing survival function.

Further, in Chapter 2 we provide a comprehensive overview of the historical development of time series modelling and forecasting methods, with a particular emphasis on those models that are most pertinent to the research objectives of this thesis. Finally, Chapter 6 concludes with a succinct summary of the significant discoveries made within this thesis and discusses potential future work.

2

A Review of Time Series Modelling and Forecasting

Contents

The field of time series modelling has long been a subject of academic study. Historically, traditional methods have emphasised the use of parametric models that are guided by domain-specific expertise. However, the advent of modern machine learning techniques has enabled the development of data-driven approaches for learning temporal dynamics. In the following subsections, we provide a detailed explanation of some of the models mentioned in Chapter 1 in order to establish a shared understanding of the background for the subsequent chapters.

2.1 Classical Parametric Methods for Sequence Modelling

Before exploring the application of deep learning techniques to time series forecasting-related problems, we briefly introduce some classic methods that continue to be used in this context nowadays. These methods serve as inspiration for many state-of-the-art architectures and have a long history, dating back to the early 20th century. Early approaches to time series forecasting relied on classical statistical techniques, using parametric models informed by domain knowledge. We briefly discuss some of the most relevant ones for this thesis.

2.1.1 Autoregressive Moving Average Models

AutoRegressive Moving Average (ARMA) models were first introduced by G. E. Box and Gwilym M Jenkins (1970). These models are a combination of AutoRegressive (AR) models, which model the current value of a time series as a linear function of past values, and Moving Average (MA) models, which shape the current value as a linear function of past errors. The ARMA model is given by the following equation:

$$\hat{X}_t = c + \sum_{i=1}^{p} \phi_i \hat{X}_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t, \tag{2.1}$$

where $\hat{X}_t$ is the prediction at time step t, c is a constant, ϕ_i and θ_i are the AR and MA coefficients with orders "p" and "q", respectively, and ϵ_t is white noise, usually an independent and identically distributed (i.i.d.) normal random variable with zero mean and finite variance. This white noise is also denoted as the error term or residual, and it represents the difference between the observed value, X_t, and the predicted value, $\hat{X}_t$. ARMA models can be extended into multiple forms, and the Autoregressive Integrated Moving average (ARIMA) model (G. E. Box and Gwilym M Jenkins, 1970) is among the most well-known ones. ARIMA models include an integration component responsible for making the time series stationary through differentiation. Therefore, the ARIMA model can account for both trend and seasonality in the time series data, where trend refers to a long-term increase

or decrease in the data, while seasonality refers to patterns that repeat at regular intervals within a time horizon, such as yearly, quarterly, or monthly.

2.1.2 Exponential Smoothing

Exponential smoothing (R. Hyndman et al., 2008) is another popular technique for time series forecasting. It involves updating a forecast for the next time step based on a weighted average of past observations, with the weights decaying exponentially over time. The simplest form of exponential smoothing is given by:

$$\hat{X}_t = \alpha X_{t-1} + (1-\alpha)\hat{X}_{t-1}, \tag{2.2}$$

where $\hat{X}_t$ is the forecast for the next time step, X_{t-1} is the observed value at time $t-1$, $\hat{X}_{t-1}$ is the predicted value at time $t-1$, and α is the smoothing factor, which makes exponential smoothing a weighted average of previous observations.

2.1.3 State Space Models

State Space Models (SSM) (Durbin and Koopman, 2012) are a class of statistical models widely used for non-stationary time series forecasting. They represent a time series as a combination of two components: a latent state and the observations themselves, where the latent state is a set of variables that capture the underlying dynamics of the time series.

To forecast a time series using an SSM, we first need to specify the form of the latent state and the observation. This is typically done using a set of state equations and observation equations, which define the relationships between the latent state, the observations, and any exogenous variables, also called covariates (variables that are not part of the latent state or the observations). The state equations define the dynamics of the latent state z_t at time t in terms of the latent state at the previous time $t-1$, and any exogenous variables E_t:

$$Z_t = F_t Z_{t-1} + G_t E_t + W_t, \tag{2.3}$$

where F_t and G_t are matrices of model parameters, and w_t is a zero-mean white noise process with covariance matrix Q_t. Once the state and observation equations are

specified and their parameters are estimated, e.g., using statistical techniques such as maximum likelihood estimation, we can use the SSM to forecast future values of the time series. To forecast the latent state at time t, we can use the current estimate of the latent state at time $t-1$ and the exogenous variables at time t:

$$\hat{Z}_t = F_t \hat{Z}_{t-1} + G_t E_t. \tag{2.4}$$

Finally, the observation equations can be used to transform the prediction of the latent state into a prediction of the observations:

$$\hat{X}_t = H_t \hat{Z}_t, \tag{2.5}$$

being H_t a matrix of model parameters. One of the main advantages of SMMs with regard to other classic techniques is their ability to incorporate exogenous variables into the forecasting process, which can improve the accuracy of the forecasts. E.g., we might include weather data or time of the day as exogenous variables in an SSM to forecast electricity demand.

The foundation for numerous early time series forecasting methods was established through the adoption of these classical statistical methods, which continue to be used in various applications today. However, with the advent of deep learning, more advanced techniques that are able to model complex temporal patterns, improving the performance of time series forecasting tasks, have been developed. These techniques, which will be discussed in more detail in the following section, offer a data-driven approach that requires little to no manual feature engineering and have the potential to model large-scale forecasting tasks.

2.2 Deep Learning Models for Sequence Modelling

Deep learning models have been widely applied in the field of time series forecasting due to their ability to automatically learn features and make predictions based on data. In this section, we discuss the use of several specific types of deep learning models for time series forecasting, including Convolutional Neural Networks

(CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM) networks. Notice that there are models in the literature that we do not cover, such as the Gated Recurrent Units (GRU), as we focus on models relevant to this thesis' following chapters. We will also discuss the use of the attention mechanism, which has the potential to improve the accuracy and interpretability of deep learning models for time series forecasting and led to the development of Transformer models.

To explain the use of deep learning techniques to forecast time series, we modify the nomenclature employed in the previous section and adapt it to the standard one in the machine learning literature. We refer to the one-step-ahead forecasting at time t for the i^{th} time series as:

$$\hat{y}_t^i = f\left(y_{t-T:t-1}^i, \boldsymbol{x}_{t-T:t-1}^i\right), \tag{2.6}$$

where $\hat{y}_t^i$ is the model's forecast, $y_{t-T:t-1}^i = \left\{y_{t-T}^i, \ldots, y_t^i\right\}$ are the past T observations of the time series being forecasted (lookback window) or previous predictions in the context of autoregressive models, $\boldsymbol{x}_{t-T:t}^i = \left\{\boldsymbol{x}_{t-T}^i, \ldots, \boldsymbol{x}_t^i\right\}$ are the associated covariates, and $f(\cdot)$ is the predictive function learnt.

2.2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNN) (LeCun, Bengio, et al., 1995) are a type of deep learning model particularly well-suited for processing data with spatial structure. Originally designed for image analysis (LeCun, Boser, et al., 1989), they were adapted to time series contexts through the use of causal convolutions (Oord et al., 2016), which are convolutional filters that only use past information. While a convolutional operation of a one-dimensional input sequence x with a filter k, at time t, is defined as:

$$(x * k)_t = \sum_{\tau=-\infty}^{\infty} x_{t-\tau} k_\tau, \quad x, k \in \mathbb{R}^{\mathbb{Z}}, \tag{2.7}$$

causal convolutions modify this operation, as they only aggregate information from the past and present in order to forecast future values:

$$(x *^{\text{c}} k)_t = \sum_{\tau=0}^{S-1} x_{t-\tau} k_\tau, \quad x \in \mathbb{R}^{\mathbb{Z}}, k \in \mathbb{R}^N, \tag{2.8}$$

where k is a finite filter of length S. Finally, causal convolutions can be extended into Dilated Causal Convolutions (DCC), where the inner product with the convolutional filter is not based on consecutive entries of the time series but on entries that are a fixed number of steps apart from each other. A DCC with dilation factor d is defined as:

$$(x *_d^{\mathrm{c}} k)_t = \sum_{\tau=0}^{S-1} x_{t-d\tau} k_\tau, \quad x \in \mathbb{R}^{\mathbb{Z}}, k \in \mathbb{R}^N. \tag{2.9}$$

Figure 2.1 shows these three variations for one-dimensional convolutions. In Chapter 4, we explore in detail the use of DCCs for the quantitative-finance problem of volatility forecasting.

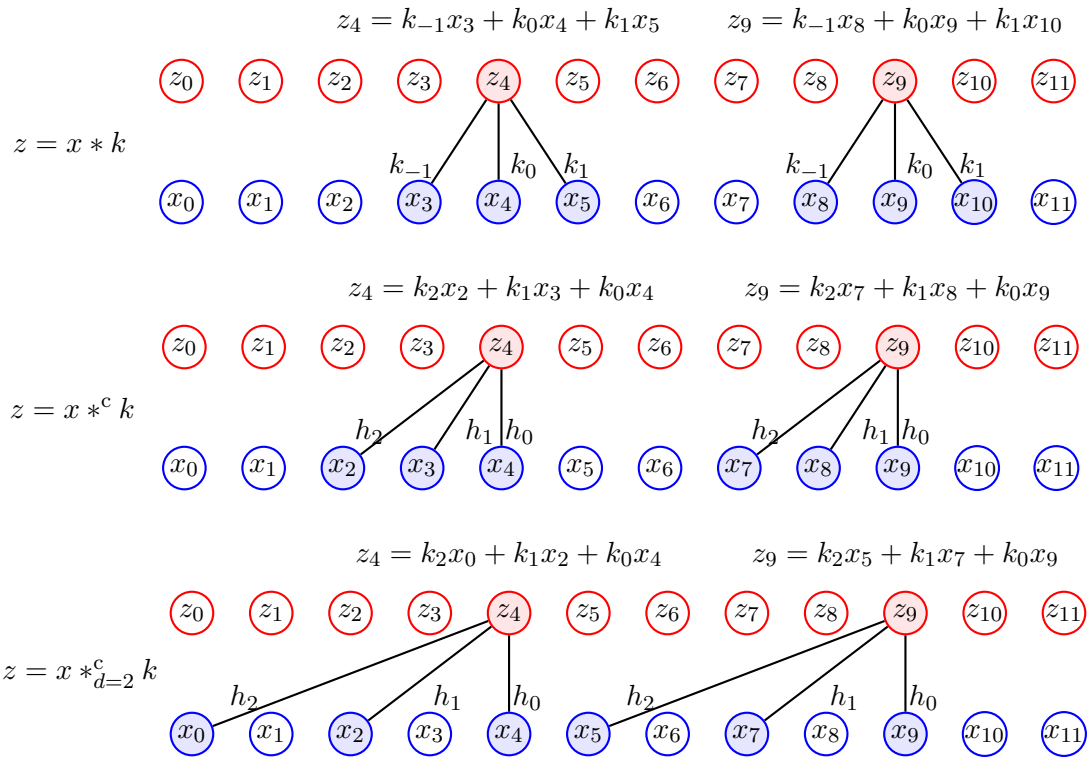

Figure 2.1: Three different types of one-dimensional convolution. (a) One-dimensional convolution of a time series x with a filter k. (b) Causal convolution of a time series x with a filter k. (c) Dilated causal convolution ($d = 2$) of a time series x with a filter k. (Adapted from Reisenhofer et al. (2022)).

2.2.2 Recurrent Neural Networks

Recurrent Neural Networks (RNN) (Funahashi and Nakamura, 1993) are a type of deep learning model that are specifically designed to handle sequential data.

RNNs are able to capture dependencies between the past and present data by using hidden state vectors that are updated at each time step. The memory/hidden state of an RNN can be represented as:

$$h_t = g(h_{t-1}, \boldsymbol{x}_t) \tag{2.10}$$

where $h_t \in \mathbb{R}^{\mathcal{H}}$ is the RNN's hidden state at time t, and $g(\cdot)$ is the memory update function. Using Eq. (2.10) as basic building block, different varitions of RNN can be found, where Elman (1990) propose one straightforward approach:

$$y_{t+1} = \varphi_y \left(\boldsymbol{W}_y \boldsymbol{h}_t + \boldsymbol{b}_y \right) \tag{2.11}$$

$$\boldsymbol{h}_t = \varphi_h \left(\boldsymbol{W}_{h_1} \boldsymbol{h}_{t-1} + \boldsymbol{W}_{h_2} y_t + \boldsymbol{W}_{h_3} \boldsymbol{x}_t + \boldsymbol{W}_{h_4} \boldsymbol{s} + \boldsymbol{b}_h \right), \tag{2.12}$$

being φ the desired activation function, and $\boldsymbol{W}, \boldsymbol{b}$ the linear weights and biases, respectively.

2.2.3 Long Short-Term Memory Networks

RNNs can suffer from limitations in learning long-range dependencies in the data due to exploding and vanishing gradients. Long Short-Term Memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997) were developed to address these limitations by improving gradient flow within the network. This is achieved through the use of a cell state $\boldsymbol{c}_t$, which stores long-term information modulated through a series of gates:

$$\textit{Input gate}: \quad \boldsymbol{i}_t = \sigma(\boldsymbol{W}_{i_1} \boldsymbol{h}_{t-1} + \boldsymbol{W}_{i_2} y_t + \boldsymbol{W}_{i_3} \boldsymbol{x}_t + \boldsymbol{W}_{i_4} \boldsymbol{s} + \boldsymbol{b}_i), \tag{2.13}$$

$$\textit{Output gate}: \quad \boldsymbol{o}_t = \sigma(\boldsymbol{W}_{o_1} \boldsymbol{h}_{t-1} + \boldsymbol{W}_{o_2} y_t + \boldsymbol{W}_{o_3} \boldsymbol{x}_t + \boldsymbol{W}_{o_4} \boldsymbol{s} + \boldsymbol{b}_o), \tag{2.14}$$

$$\textit{Forget gate}: \quad \boldsymbol{f}_t = \sigma(\boldsymbol{W}_{f_1} \boldsymbol{h}_{t-1} + \boldsymbol{W}_{f_2} y_t + \boldsymbol{W}_{f_3} \boldsymbol{x}_t + \boldsymbol{W}_{f_4} \boldsymbol{s} + \boldsymbol{b}_f), \tag{2.15}$$

where σ is the activation function and $\boldsymbol{h}_{t-1}$ is the hidden state of the LSTM. The LSTM's hidden and cell states are updated as follows:

$$\textit{Hidden state}: \quad \boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t), \tag{2.16}$$

$$\textit{Cell state}: \quad \boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tanh(\boldsymbol{W}_{c_1} \boldsymbol{h}_{t-1} + \boldsymbol{W}_{c_2} y_t + \boldsymbol{W}_{c_3} \boldsymbol{x}_t + \boldsymbol{W}_{c_4} \boldsymbol{s} + \boldsymbol{b}_c), \tag{2.17}$$

where $\odot$ is the element-wise product and $\tanh(\cdot)$ is the hyperbolic tangent function. LSTMs can be easily expanded into Bidirectional-LSTMs (Bi-LSTM), whose main advantage lies in their capacity to process input sequences in two directions, both forward and backwards, being able to capture both past and future context in a sequence. To do so, while in a regular LSTM the hidden state at time t is updated based on the previous hidden state and the current input, a Bi-LSTM uses two separate LSTM networks, each processing the input sequence in forward and backward directions, respectively. The output of the Bi-LSTM at time t is a combination of the hidden states of the forward and backward LSTMs at that time, that is $h_t = \left[\overrightarrow{h_t}, \overleftarrow{h_t}\right]$, where $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ are obtained through Eq. (2.16). The ability to capture both past and future context can be particularly useful for tasks that involve understanding the meaning and relationships of words in a sentence. E.g., in NLP tasks such as language modelling and machine translation, the context in which a word is used can be important for determining its meaning.

2.2.4 Attention Mechanisms

The attention mechanism, firstly proposed by Bahdanau et al. (2015), allows recurrent architectures to focus selectively on some data segments while making predictions. This technique, originally proposed to operate on an encoder-decoder architecture, finds a context vector c_t that is a weighted sum of a sequence of T encoder's hidden states. This context vector can be specified as follows:

$$c_t = \sum_{t'=1}^{T} \alpha_{t,t'} h_{t'}, \tag{2.18}$$

where the attention weights vector $\alpha_t = [\alpha_{(t,1)}, \ldots, \alpha_{(t,T)}]$ relates current time step t to each of the T encoder's hidden states, obtained as $h_{t'} = g(h_{t'-1}, \boldsymbol{x}_{t'})$. To ponder this relation, the attention weights are computed through a softmax function:

$$\alpha_{t,t'} = \frac{\exp\left(e_{t,t'}\right)}{\sum_{k=1}^{T} \exp\left(e_{t,k}\right)}, \tag{2.19}$$

where $e_{t,t'}$ are the alignment scores, which relate the inputs around position t' with the output at position t, based on the decoder' previous output, s_{t-1}:

$$e_{t,t'} = f(s_{t-1}, h_{t'}), \tag{2.20}$$

where the function $f(\cdot)$ is usually implemented through a neural network.

2.2.5 Transformer Models

Transformer models (Vaswani et al., 2017) were recently proposed for NLP tasks. They have become a popular choice due to their ability to handle long-range dependencies and capture contextual information effectively. Furthermore, different proposals have adapted them to the time series forecasting problem idiosyncrasies (S. Li et al., 2019; Lim, Arik, et al., 2021).

At a high level, Transformer models are a type of neural network architecture solely based on the self-attention mechanism, which arises from Bahdanau et al. (2015)'s initial proposal. Therefore, Transformers process input sequences eschewing recurrence and relying entirely on the attention mechanism to find dependencies between input and output sequences, allowing for significantly more parallelization.

The self-attention mechanism in which the Transformer model bases its operation is performed simultaneously by a different number of Transformer's heads $H \in \mathbb{Z}$, comprising what is called a multi-head Transformer. Each of the multi-head self-attention sublayers simultaneously transforms the input time series $\boldsymbol{X}$ into query, key, and value matrices. For the i^{th} head, these are obtained as: $Q_i = XW_i^Q$, $K_i = XW_i^K$, $V_i = XW_i^V$, where $W_i^Q \in \mathbb{R}^{(d+1)\times d_Q}$, $W_i^K \in \mathbb{R}^{(d+1)\times d_K}$, $W_i^V \in \mathbb{R}^{(d+1)\times d_V}$ are learnable parameters, d is the dimensionality of the original time series, and d^Q, d^K, d^V are the dimensions of the learned linear projections for query, keys, and values, respectively. Finally, the Transformer output for the i^{th} head is computed through the use of the scaled dot-product attention:

$$\mathrm{h_i} = \text{Attention}\left(Q_i, K_i, V_i\right) = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i, \tag{2.21}$$

whose operative for a single Transformer head is illustrated in Figure 2.2. Finally, the different heads, whose use allows the model to jointly attend to different temporal subspaces of the original time series, are combined to obtain a joint representation:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}\left(\mathrm{h}_1, \mathrm{h}_2, \ldots, \mathrm{h}_i, \ldots \mathrm{h}_\mathrm{H}\right), \tag{2.22}$$

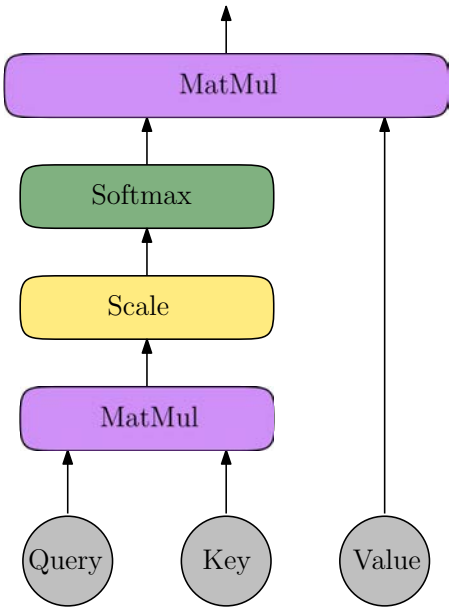

Figure 2.2: Scaled dot-product attention.

where h_i represents i^{th} head's output, $\mathrm{h_i} = \mathrm{Attention}\left(Q_i, K_i, V_i\right)$, and $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ comprise the queries, keys, and values for all heads.

This review of Transformer models marks the conclusion of our literature survey on time series modelling and forecasting. The deep learning models discussed in this chapter are currently considered state-of-the-art and are widely utilised due to their ability to extract features automatically and generate predictions based on data. Nonetheless, there are still challenges that need to be addressed in the development and implementation of these models. These issues will be addressed in the following chapter.

3

Deep Autoregressive Models with Spectral Attention

Contents

Time series forecasting is a fundamental problem across many domains, playing a crucial role in multiple real-world applications. In this chapter, we propose a

forecasting architecture that combines deep autoregressive models with a Spectral Attention (SA) module, which merges global and local frequency domain information in the model's embedded space. By characterising in the spectral domain the embedding of the time series as occurrences of a random process, our method can identify global trends and seasonality patterns.

Two spectral attention models, global and local to the time series, integrate this information within the forecast and perform spectral filtering to remove the time series's noise. The proposed architecture has several valuable properties: it can be effectively incorporated into well-known forecast architectures, requiring a low number of parameters, and producing explainable results that improve forecasting accuracy.

This chapter, which presents the contents of Moreno-Pino, Olmos, et al. (2023), is organised as follows. Section 3.1 motivates the need for our proposal and briefly introduces it, also providing an overview of methods with similar scope. Section 3.2 states the time series forecasting problem, presents a base architecture to which we append the proposed Spectral Attention module, and provides a characterization of the time series in the spectral domain. Section 3.3 describes our model, and Section 3.4 proves its effectiveness, both quantitative and qualitatively. To do so, we perform extensive experiments on both synthetic and real-world time series datasets, showing the effectiveness of the proposed Spectral Attention module, consistently outperforming the base models and reaching state-of-the-art results. In addition, ablation studies further prove the effectiveness of the designed architecture. We conclude the chapter in Section 3.5 with a short discussion.

3.1 Problems and Limitations of Autoregressive Models for Time Series Forecasting

As mentioned in Chapter 2, deep learning models have been widely applied in the field of time series forecasting due to their ability to automatically learn features and make predictions based on data, replacing the use of classic methodologies. Architectures like DeepAR (Salinas et al., 2020) and ConvTrans (S. Li et al., 2019)

set a milestone adapting state-of-the-art machine learning models to the problem of probabilistic time series forecasting. The former employs an LSTM to perform an embedding used by a probabilistic model to forecast in an encoder-decoder fashion, while the latter adapts Transformer models to the time series modelling context.

Furthermore, various attempts to merge signal processing techniques with deep neural networks can also be found in the literature. In Tamkin et al. (2020), a framework that uses spectral filtering for the problem of NLP was proposed, while Cao et al. (2020) uses the spectral domain to capture inter-series correlations and temporal dependencies jointly. Nevertheless, most of the methods in the literature that incorporate signal processing techniques into deep learning architectures solely use signal decomposition techniques to identify trend and seasonality patterns.

Likewise, many models have tried to join classical approaches with deep learning techniques, as Deep State Space Models for Time Series Forecasting (DSSM) (Rangapuram et al., 2018), or Deep Factors for Forecasting (Yuyang Wang et al., 2019). The former established an architecture that joins together State Space Models (SSM) with an RNN. The latter generalises the approach of DSSM by using global and local effects on the inference model.

Nevertheless, deep autoregressive models also present some inconveniences, as we mentioned in Chapter 1. First, deep autoregressive models tend to focus on recent past data to predict the future of the time series, frequently wasting important global information not encapsulated in previous predictions. Second, as classical time series forecasting methods, they suffer from error accumulation and propagation (Cheng et al., 2006), a problem closely related to the previous one. Third, they do not produce illustrative results, neither can we clearly explain how they reach them (X. Bai et al., 2021). In this regard, progress regarding the improvement of forecasts' representativeness has been made through the proposal of visualisation methods (Kang, R. J. Hyndman, and Smith-Miles, 2017), together with the development of evaluation tools that facilitate the benchmarking of forecasting architectures (Kang, R. J. Hyndman, and F. Li, 2020).

In the rest of this chapter, we show that the described problems can be partially alleviated by incorporating signal processing filtering techniques into the autoregressive models that perform the time series forecasting. Concerning the inability of these models to focus on the global context, we can obtain time series' most important trends via frequency domain characterization. These trends can be intelligently incorporated during the forecasting, hence making the local context aware of the time series global patterns. Regarding error accumulation and noisy local context, spectral filtering can be applied to decide at every time instant which frequencies are useful and which can be suppressed, eliminating unwanted components that do not help during the forecast. Finally, these signal processing tools that operate in the spectral domain produce more illustrative internal representations, as proved during the experiments section, making it possible to extract the explainable frequency domain features that are driving the predictions if necessary.

Thus, this chapter proposes a novel and general architecture, the Spectral Attention Autoregressive Model (SAAM), which integrates all previous solutions, representing a significant advancement in the field of temporal modelling and time series forecasting. SAAM's modularity allows it to be effectively incorporated into a variety of deep autoregressive models. This architecture uses two spectral attention models to determine, at every time instant, relevant global patterns as well as removing local context's noise while performing the forecasting. Both operations are performed in the frequency domain of the embedded space.

To the best of our knowledge, SAAM is the first deep neural autoregressive model that exploits attention mechanisms in the spectral domain. In this new frequency domain attention framework, a global-local architecture marries deep neural networks with classic signal processing techniques. This architecture, encapsulated in the Spectral Attention module, modifies the embedded representation of the time series, incorporating relevant global trends into the forecast and performing spectral filtering to mitigate error accumulation. Further, the additional complexity due to Spectral Attention is comparable to classic attention models in the temporal domain.

We perform extensive experiments on both synthetic and real-world time series datasets, showing the effectiveness of the proposed Spectral Attention module, consistently outperforming the base models and reaching state-of-the-art results. In addition, ablation studies further prove the effectiveness of the designed architecture.

3.2 Preliminaries

In this section, we formally state the problem of time series forecasting and introduce a base architecture that represents the core of most deep learning-based autoregressive models in the state-of-the-art. Also, a frequency domain characterization of the time series is proposed.

3.2.1 Problem Definition

Given a set of N univariate time series $\{z^i_{1:t_0-1}\}_{i=1}^N$, where $z^i_{1:t_0-1} = (z^i_1, z^i_2, \dots, z^i_{t_0-1})$, $t_0 \in \mathbb{N}$ is the forecast horizon, $\tau \in \mathbb{N}$ the forecast length, and $T = t_0 + \tau \in \mathbb{N}$ the sequences' total length, our goal is to model the conditional probability distribution of future trajectories of each time series given the past, namely, to predict the next τ time steps after the forecast horizon, assuming conditional independence between different time instants:

$$p\left(z^i_{t_0:t_0+\tau} \mid z^i_{1:t_0-1}, \mathbf{x}^i_{1:t_0+\tau}, \theta\right) = \prod_{t=t_0}^{t_0+\tau} p\left(z^i_t \mid z^i_{1:t-1}, \mathbf{x}^i_{1:t}, \theta\right), \tag{3.1}$$

where θ are the learnable parameters of the model and $\{\mathbf{x}^i_{1:t_0+\tau}\}_{i=1}^N \in \mathbb{R}^C$ the associated covariates. These covariates are, together with time series' past observations, the input to our predictive model.

3.2.2 Base Architecture

Several deep autoregressive models in the state-of-the-art, including DeepAR (Salinas et al., 2020) and ConvTrans (S. Li et al., 2019), can be characterised by means of a high-level architecture, represented in Figure 3.1. This general framework is composed of two parts:

1. An embedding function $\mathbf{e}_t^i = f_\phi\left(\mathbf{e}_{t-1}^i, z_{t-1}^i, \mathbf{x}_t^i\right) \in \mathbb{R}^D$, with transit function $f_\phi(\cdot)$ and parameters ϕ. This embedding receives as input, at time $t \in T$, the time series previous value z_{t-1}^i, the covariates $\mathbf{x}_t^i$, and the past value of the embedding $\mathbf{e}_{t-1}^i$. This embedding function can be implemented in different ways, such as a RNN, a LSTM, a Temporal Convolutional Network (TCN) (Lea et al., 2016), or a Transformer.

2. A probabilistic model $p\left(z_t^i \mid \mathbf{e}_t^i\right)$, with parameters ψ, which uses the embedding $\mathbf{e}_t^i$ to estimate time series' next value, $\hat{z}_t^i$. This probabilistic model is usually implemented as a function of a neural network that parameterizes the required probability distribution. E.g., a Gaussian distribution can be represented through its mean and standard deviation as: $\mu = g_\mu(\mathbf{w}_\mu^T \mathbf{e}_t^i + b_\mu)$, $\sigma = \log\left(1 + \exp\left(g_\sigma(\mathbf{w}_\sigma^T \mathbf{e}_t^i + b_\sigma)\right)\right)$, where g_μ and g_σ are neural networks.

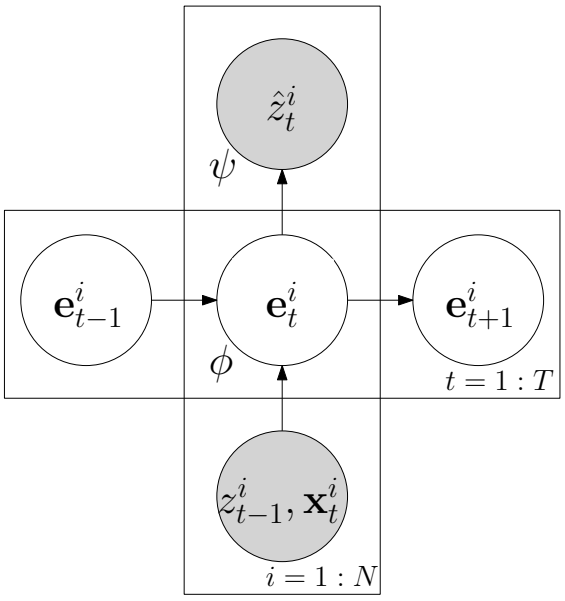

Figure 3.1: Common base architecture of deep learning-based autoregressive models. Gray represents observed variables.

The optimal model's parameters $\theta = \{\phi, \psi\}$ are obtained by maximizing the log-likelihood function $\mathcal{L}(\theta)$ for the observed data in the conditioning range, i.e.,

from $t = 1$ to $t_0 - 1$. All quantities required for computing the log-likelihood function are deterministic, which means that no inference is required:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \log p\left(z^i_{1:t_0-1} \mid \mathbf{x}^i_{1:t_0-1}, \theta\right) = \sum_{i=1}^{N} \sum_{t=1}^{t_0-1} \log p\left(z^i_t \mid \mathbf{x}^i_{1:t-1}, \theta(\phi, \psi)\right). \tag{3.2}$$

During both training and testing, the conditioning range $\{1 : t_0 - 1\}$, which is analogous to the encoder of seq2seq models (Sutskever et al., 2014), transfers information to the forecasting range $\{t_0 : t_0 + \tau\}$, analogous to the decoder. Therefore, this base framework can be interpreted as an encoder-decoder architecture, with the consideration that both encoder and decoder are the same network, as Figure 3.2 shows.

For forecasting, given the model's parameters θ, we can directly obtain joint samples from the model as $\hat{z}^i_{t_0:t_0+\tau} \sim p\left(z^i_{t_0:t_0+\tau} \mid z^i_{1:t_0-1}, \mathbf{x}^i_{1:t_0+\tau}, \theta\right)$. Therefore, during the forecasting range, the model consumes the previous time step prediction $\hat{z}^i_{t-1}$ as input, unlike during the conditioning range, where z^i_{t-1} is observed. This is illustrated in Figure 3.2.

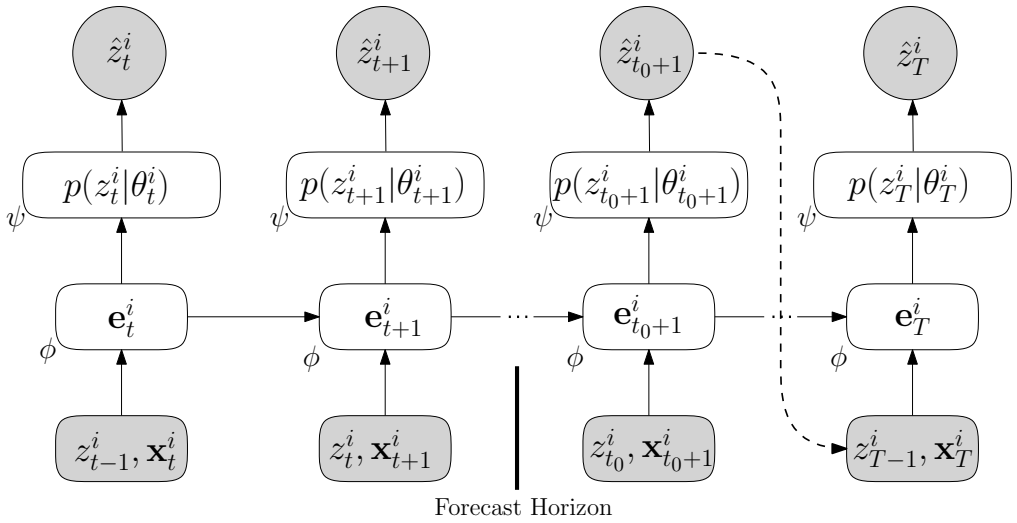

Figure 3.2: Unrolled base architecture. On the left of the forecast horizon, the conditioning range can be found $\{1 : t_0 - 1\}$. On its right, the forecasting range $\{t_0 : t_0 + \tau\}$.

Notice that Transformers, unlike RNNs or LSTMs, do not compute the embedding in a sequential manner. Accordingly, when obtaining the embedding

through a Transformer model (Wu et al., 2020) and so to use the encoder-decoder architecture previously described, we use the Transformer decoder-only mode, introduced in P. J. Liu et al. (2018).

3.2.3 Characterizing the Time Series' Embedding in the Frequency Domain

Our approach in this chapter exploits information in the spectral domain. In this regard, time series' embedding space can be statistically characterized as instances of a random process for which spectral information can be analyzed using the expected autocorrelation and the Power Spectral Density (PSD) (Buttkus, 2012).

The power spectrum per embedding's dimension can be calculated from an averaged autocorrelation, estimated from a finite number $M \in \mathbb{Z}^+$ of time series of duration $T \in \mathbb{N}$ each:

$$\hat{R}_d(\kappa) = \frac{1}{M} \sum_{j=1}^{M} \left(\frac{1}{T} \sum_{t=1}^{T} e_t^j(d)\ e_{t+\kappa}^j(d) \right) \in \mathbb{R}^T, \tag{3.3}$$

where $\hat{R}_d(\kappa) \in \mathbb{R}^T$ is the expected autocorrelation for the d^{th} dimension of the j^{th} embedded sequence, $\boldsymbol{e}^j(d)$, and κ is the lag between observations.

The Blackman-Tukey method (Blackman and Tukey, 1958), which takes advantage of the Discrete Fourier Transform (DFT) of a windowed autocorrelation, can be used to estimate the PSD once the autocorrelation has been computed:

$$\hat{S}_d(\omega) = \sum_{-T}^{T} \hat{R}_d(\kappa) e^{-j\omega\kappa} \in \mathbb{R}^{N_{\mathcal{FT}}}, \tag{3.4}$$

where $\hat{S}_d(\omega) \in \mathbb{R}^{N_{\mathcal{FT}}}$ decomposes a function into its constituent frequencies ($N_{\mathcal{FT}}$ is usually equal to the time series' length T), obtaining the estimated spectrum $\hat{S}_d(\omega)$ of the random process that generates the time series in the embedding space. Notice that $\hat{S}_d(\omega)$ is a spectral characterization of the d-dimension of the embedding space, hence not global to the process. In Section 3.3, we propose an alternative autocorrelation function that considers each embedding's dimensions as independent realizations of the same random process that generates the time series; therefore, obtaining a global spectral representation of the process.

3.3 Spectral Attention Autoregressive Model

In this section, we introduce the Spectral Attention Autoregressive Model (SAAM), a general framework that incorporates a Spectral Attention (SA) module able to exploit embedding function's spectral information using attention mechanisms to solve two main tasks:

1. Time series z_t^i are governed by global trends and seasonality structures. SAAM captures these global patterns and incorporates them into the forecast.

2. Time series exhibit noise around these primary trends that difficult the forecasting process. SAAM filters this noise using spectral filtering, improving the signal to noise ratio thus alleviating the error propagation problem autoregressive models suffer from.

These two operations, incorporating global trends into the forecast and filtering the time series' local context, are encapsulated in the SA module, responsible for all the frequency domain related operations. As such, it can be incorporated in any deep autoregressive structure.

The resulting architecture, displayed in Figure 3.3, is therefore composed of three main parts: embedding function, SA module, and probabilistic model. For further details on the embedding and probabilistic model, common to the base architecture of Figure 3.1, we refer the reader to Section 3.2.2. Concerning the SA module, we now explain in detail how it integrates both global and local information into the forecasting.

3.3.1 Global Spectral Information

To incorporate global patterns into the prediction, we perform a frequency domain characterization of the process, as explained in Section 3.2.3. We aim to exploit spectral domain information associated with the neural embeddings $\left\{\mathbf{e}_1^j, \mathbf{e}_2^j, \ldots, \mathbf{e}_{t_0-1}^j\right\}_{j=1}^M$ of a number $M \in \mathbb{N}$ of time series.

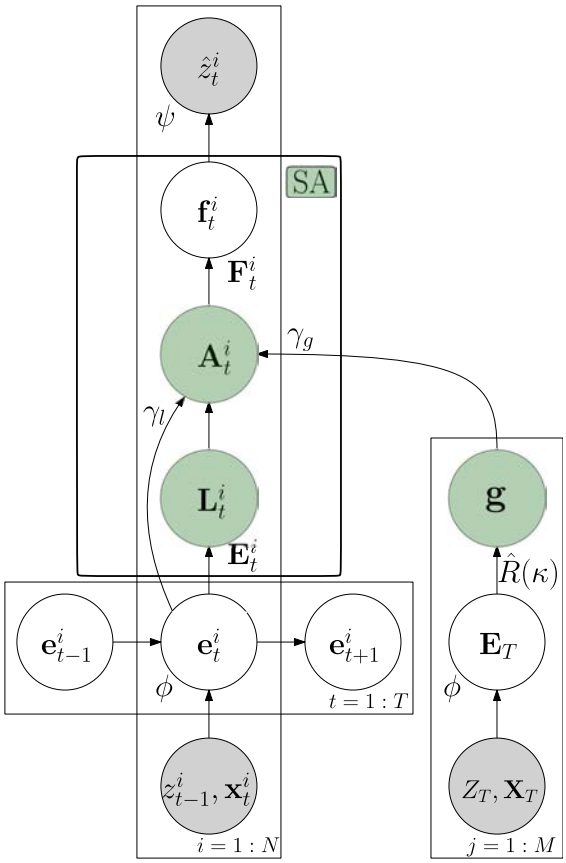

Figure 3.3: Spectral Attention Autoregressive Model general architecture. Gray represents observed variables. Green indicates frequency domain representations.

Nevertheless, we are not interested in characterizing each embedding's dimension independently, $\hat{S}_d(\omega) \in \mathbb{R}^{N_{\mathcal{FT}}}$, as using Eq. (3.3) entails. This would require a multidimensional PSD for characterizing the process over all embedding's dimensions, $\hat{S}(\omega) \in \mathbb{R}^{D \times N_{\mathcal{FT}}}$. Instead, we consider each dimension of the embedding as independent realizations of the same process we average over, resulting in the following autocorrelation function:

$$\hat{R}(\kappa) = \frac{1}{M} \sum_{j=1}^{M} \left[\frac{1}{D} \sum_{d=1}^{D} \left(\frac{1}{T} \sum_{t=1}^{T} e_t^j(d)\ e_{t+\kappa}^j(d) \right) \right] \in \mathbb{R}^T, \tag{3.5}$$

where $e_t^j(d)$ is the d^{th} dimension of the j^{th} embedded sequence and $D \in \mathbb{Z}^+$ is the embedding's number of dimensions. To apply the Blackman-Tukey method from Eq. (3.4) over the previous autocorrelation function involves obtaining the complete frequency-domain characterization of the process, incorporating all the

global patterns across the different dimensions of the embedding into a single spectral representation, $\hat{S}(\omega) \in \mathbb{R}^{N_{\mathcal{FT}}}$.

Therefore, Eqs. (3.4) and (3.5) allow us to calculate a Monte Carlo approximation of the process' PSD for a batch of $M \in \mathbb{N}$ training sequences $\{Z_T, \mathbf{X}_T\}_{j=1}^{M}$, via a Fourier Transform with $N_{\mathcal{FT}}$ points: $\mathbf{g} = \mathcal{FT}\left\{\hat{R}_T(\tau)\right\} \in \mathbb{R}^{N_{\mathcal{FT}}}$. Consequently, $\mathbf{g} \in \mathbb{R}^{N_{\mathcal{FT}}}$ is the global spectral representation of the process, shared among the $N \in \mathbb{N}$ time series to forecast.

3.3.2 Local Spectral Information

To perform the spectral local filtering, we analyze the last $T_F \in \mathbb{N}$ embedded values of each individual sample: $\mathbf{E}_t^i = \left\{\mathbf{e}_{t-T_F}^i, \mathbf{e}_{t-T_F+1}^i, \ldots, \mathbf{e}_t^i\right\} \in \mathbb{R}^{D \times T_F}$, where $\mathbf{E}_t^i$ encapsulates the previous T_F embeddings, each of dimension $\mathbb{R}^D$. This embedded buffered signal at time t, which characterizes the recent local past of the process, is transformed to the frequency domain, $\mathbf{L}_t^i = |\mathcal{FT}\{\mathbf{E}_t^i\}| \in \mathbb{R}^{D \times N_{\mathcal{FT}}}$, via a DFT with $N_{\mathcal{FT}}$ points. Note that we retain only the module of the Fourier Transform.

3.3.3 Merging Global and Local Contexts with Spectral Attention

We combine both global and local spectral information to modify the embedded representation of the time series $\mathbf{e}_t^i$. This is done through two spectral attention models contained in the SA module, with parameters $\gamma = \{\gamma_g, \gamma_l\}$:

Global Spectral Attention

This frequency-domain attention model, with parameters γ_g, is responsible for incorporating, at time $t \in T$ and for each time series, the Global Spectral Information into the forecast. To do so, it uses the time series' local context, summarized via the embedding function $\mathbf{e}_t^i$, as key to select the relevant frequency components on $\mathbf{G}$ that should be included during the prediction, where $\mathbf{G} \in \mathbb{R}^{D \times N_{\mathcal{FT}}}$ is a repetition of $\mathbf{g} \in \mathbb{R}^{N_{\mathcal{FT}}}$ along the D dimensions of the embedding: $\boldsymbol{\alpha}_{g,t}^i = f_{\gamma_g}(\mathbf{e}_t^i, \mathbf{G}) \in \mathbb{R}^{D \times N_{\mathcal{FT}}}$. The global filter's coefficients, $\boldsymbol{\alpha}_{g,t}^i$, take values $\in [0, 1]$ and $f_{\gamma_g}(\cdot)$ is a neural network.

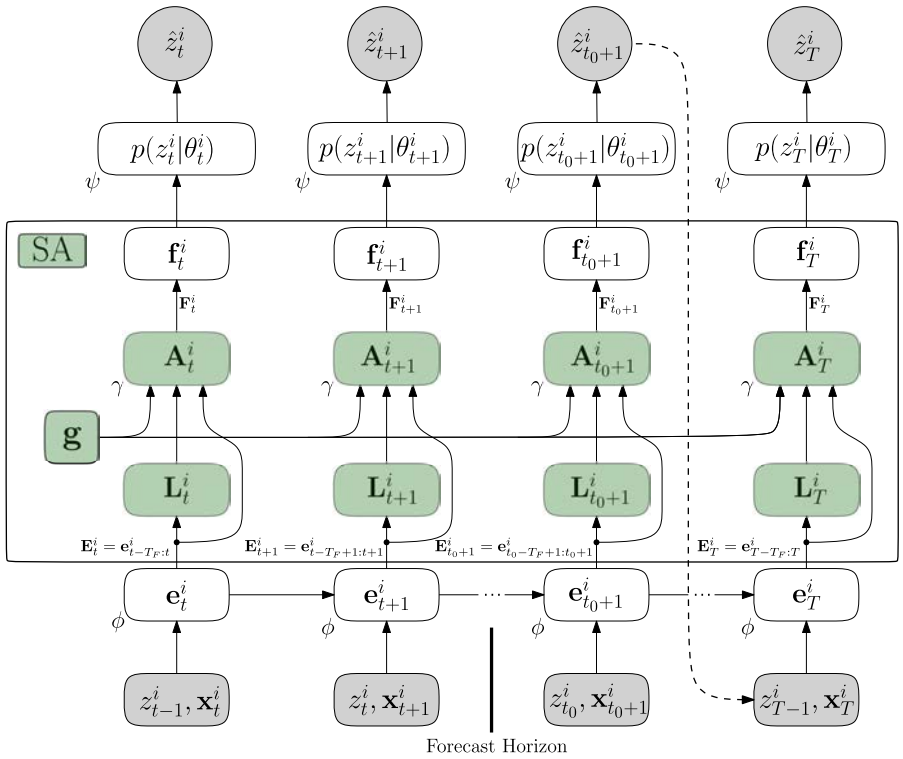

Figure 3.4: The unrolled architecture of the Spectral Attention Autoregressive Model.

Local Spectral Attention

As in the Global Spectral Attention model, the embedded representation $\mathbf{e}_t^i$ is used as a key to determine the relevant and not relevant local spectral components, $\boldsymbol{\alpha}_{l,t}^i = f_{\gamma_l}(\mathbf{e}_t^i, \mathbf{L}_t^i) \in \mathbb{R}^{D \times N_{\mathcal{F}\mathcal{T}}}$, where γ_l are the Local Spectral Attention's parameters and $\boldsymbol{\alpha}_{l,t}^i \in [0, 1]$ are the local filter's coefficients for each embedding's dimension.

Spectral Attention module: Global and Local features combination

We combine both spectral-domain attention models through multiplication and addition operations in the frequency domain: $\mathbf{A}_t^i = \mathbf{L}_t^i \odot \boldsymbol{\alpha}_{l,t}^i + \mathbf{G}_t^i \odot \boldsymbol{\alpha}_{g,t}^i \in \mathbb{R}^{D \times N_{\mathcal{T}\mathcal{F}}}$. The multiplication over the embedding's local spectrum representation $\mathbf{L}_t^i \odot \boldsymbol{\alpha}_{l,t}^i$ is performing the local spectral filtering, setting to zero not relevant frequency components through the local attention model. Furthermore, the addition of $\mathbf{G}_t^i \odot \boldsymbol{\alpha}_{g,t}^i$ includes significant global trends into the forecast via $\boldsymbol{\alpha}_{g,t}^i$, which selects relevant patterns from the process' global spectrum representation $\mathbf{G}_t^i$.

Table 3.1: Summary of operations performed by SAAM to filter both local and global contexts.

Operation	Implementation
Embedding function	$\mathbf{e}_t^i = f_\phi\left(\mathbf{e}_{t-1}^i, z_{t-1}^i, \mathbf{x}_t^i\right), \in \mathbb{R}^D$
Buffered embedding	$\mathbf{E}_t^i = \mathbf{e}_{t-T_F:t}^i, \in \mathbb{R}^{D \times T_F}$
Local spectrum	$\mathbf{L}_t^i = \lvert\mathcal{FT}\left\{\mathbf{E}_t^i\right\}\rvert, \in \mathbb{R}^{D \times N_{\mathcal{FT}}}$
Global spectrum	$\mathbf{g} = \mathcal{FT}\left\{\hat{R}_T(\tau)\right\}, \in \mathbb{R}^{N_{\mathcal{FT}}}$
Local filtering	$\boldsymbol{\alpha}_{l,t}^i = f_{\gamma_l}(\mathbf{e}_t^i, \mathbf{L}_t^i), \in \mathbb{R}^{D \times N_{\mathcal{FT}}}$
Global filtering	$\boldsymbol{\alpha}_{g,t}^i = f_{\gamma_g}(\mathbf{e}_t^i, \mathbf{G}), \in \mathbb{R}^{D \times N_{\mathcal{FT}}}$
Spectral attention	$\mathbf{A}_t^i = \mathbf{L}_t^i \odot \boldsymbol{\alpha}_{l,t}^i + \mathbf{G}_t^i \odot \boldsymbol{\alpha}_{g,t}^i, \in \mathbb{R}^{D \times N_{\mathcal{FT}}}$
Filtered embedding	$\mathbf{F}_t^i = \mathcal{FT}^{-1}\left\{\mathbf{A}_t^i\right\}, \in \mathbb{R}^{D \times T_F}$
Time step filtered embedding	$\mathbf{f}_t^i = \mathbf{F}_t^i(t), \in \mathbb{R}^D$
Emission	$\hat{z}_t^i \sim p\left(z_t^i \mid \mathbf{f}_t^i, \theta\right), \in \mathbb{R}$

Finally, this spectral representation is transformed back to the time domain $\mathbf{F}_t^i = \mathcal{FT}^{-1}\left\{\mathbf{A}_t^i\right\} \in \mathbb{R}^{D \times T_F}$ and the last value of $\mathbf{F}_t^i$, $\mathbf{f}_t^i \in \mathbb{R}^D$, is feed to the probabilistic model to forecast the next time step, $\hat{z}_t^i \sim p\left(z_t^i \mid \mathbf{f}_t^i, \theta\right)$.

Figure 3.4 shows the unrolled architecture of the proposed model and Table 3.1 contains a summary of the operations performed by the SA module. If we remove this module, the base framework displayed in Figure 3.2 remains. Notice that both the computation of the expected autocorrelation and the Fourier Transform are differentiable with respect to the model parameters.

3.3.4 Training

The log-likelihood of the model, which is maximised to match the statistical properties of the data, now includes SA's parameters, γ:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \sum_{t=1}^{t_0-1} \log p\left(z_t^i \mid \mathbf{x}_{1:t-1}^i, \theta(\phi, \gamma, \psi)\right). \tag{3.6}$$

Once the parameters θ have been learned during the conditioning range, forecasts can be produced as mentioned in Section 3.2.2: $\hat{z}_{t_0:t_0+\tau}^i \sim p\left(z_{t_0:t_0+\tau}^i \mid z_{1:t_0-1}^i, \mathbf{x}_{1:t_0+\tau}^i, \theta\right)$, computing the joint distribution over the forecasting range for each time series.

Notice that, during both training and testing, global spectral information is obtained using a mini-batch of time series different to the one being forecasted.

3.4 Experiments

We conduct experiments with both synthetic and real-world datasets in order to provide evidence of the superior ability of SAAM regarding the multi-step forecasting problem. Moreover, ablation studies are conducted. The code for our proposed model is publicly available on GitHub [1] (Moreno-Pino, 2021).

3.4.1 Applying the Spectral Attention Module to State-of-the-Art Models

Many well-known deep autoregressive models that are part of the state-of-the-art can take advantage of the Spectral Attention module proposed in Section 3.3. This allows them to filter not relevant spectral components and to recognise global trends that can be incorporated into the forecast.

We integrate the SA module into two forecasting models: DeepAR (Salinas et al., 2020), a widely-known and largely used model with several industrial applications (Böse et al., 2017), which employs a LSTM to perform the embedding; and ConvTrans (S. Li et al., 2019), a more recent Transformer-based proposal, which constitutes one of the most efficient approaches for using Transformers for the problem of time series forecasting.

Notice that the proposed architecture, encapsulated in the SA module, can be applied to models that do not operate in an autoregressive manner, such as ConvTrans itself. To do so, the embedding is obtained using the decoder-only mode, as mentioned in Section 3.2.2.

We want to remark that the added complexity by the Spectral Attention module is equivalent to classic attention models in the time domain as Bahdanau et al. (2015). In these models, for a sequence of length L, computing scores between every pair causes $O(L^2)$ memory usage. The complexity of Spectral Attention leans on

[1] `https://github.com/fmorenopino/SAAM`

the embedding's dimension $D \in \mathbb{Z}^+$ and the number of points used to compute the Fourier Transform $N_{\mathcal{FT}} \in \mathbb{Z}^+$. The complexity, therefore, depends on $O(D \cdot N_{\mathcal{FT}})$, analogous to time-domain attention mechanisms.

Anyhow, it should be noticed that the SA module allows the use of the Fourier Transform's number of points, $N_{\mathcal{FT}}$, as a hyper-parameter: to decrease its value would imply a down-sampling and hence, a reduction of the algorithm's time complexity as well as the model's number of parameters.

3.4.2 Metrics

The normalised $\rho-$quantile loss, QL_ρ, with $\rho \in (0, 1)$, which quantifies the accuracy of a quantile ρ of the predictive distribution, is the main metric used to report the results of the experiments, as in many other works (Salinas et al., 2020; S. Li et al., 2019; Rangapuram et al., 2018):

$$\mathrm{QL}_\rho(\boldsymbol{z}, \hat{\boldsymbol{z}}) = 2\frac{\sum_{i,t}\left[\rho\left(z_t^{(i)} - \hat{z}_t^{(i)}\right)\mathbb{I}_{z_t^{(i)} > \hat{z}_t^{(i)}} + (1-\rho)\left(\hat{z}_t^{(i)} - z_t^{(i)}\right)\mathbb{I}_{\hat{z}_t^{(i)} \geq z_t^{(i)}}\right]}{\sum_{i,t}\left|z_t^{(i)}\right|}. \quad (3.7)$$

Rolling window predictions after the last point seen in the conditioning range are used to obtain the results. To compute this metric, we use 200 samples from the decoder to estimate $\hat{z}_t^i$ along time. Also, the normalised sum of the quantile losses is considered, as can be observed in the previous equation.

The Normalised Deviation (ND) (H.-F. Yu et al., 2016) and Root Mean Square Error (RMSE) are also used to evaluate the probabilistic forecasts, especially on the synthetic dataset experiments:

$$\begin{aligned} \mathrm{ND} &= \frac{\sum_{i,t}|z_{i,t} - \hat{z}_{i,t}|}{\sum_{i,t}|z_{i,t}|}, \\ \mathrm{RMSE} &= \frac{\sqrt{\frac{1}{N(T-t_0)}\sum_{i,t}\left(z_{i,t} - \hat{z}_{i,t}\right)^2}}{\frac{1}{N(T-t_0)}\sum_{i,t}|z_{i,t}|}. \end{aligned} \quad (3.8)$$

3.4.3 Synthetic Dataset

In order to demonstrate SAAM's capabilities, we conduct experiments on synthetic data composed of sinusoidal signals with no covariates and a duration of 200

samples. Each of these signals is divided into two halves, randomly selecting the components for each of them as:

$$f(t) = \begin{cases} \begin{cases} f \sim f_1(t), \text{ if } x_1 = 0 \\ f \sim f_2(t), \text{ if } x_1 = 1 \end{cases}, & t \in [0, 100), \\ \\ \begin{cases} f \sim f_1(t), \text{ if } x_2 = 0 \\ f \sim f_2(t), \text{ if } x_2 = 1 \end{cases}, & t \in [100, 200], \end{cases} \tag{3.9}$$

where x_1 and x_2 are independently and randomly generated from a Bernoulli distribution, $X \sim Ber(\theta)$ with probability $\theta = 1/2$. This implies that each half of the time series can take the form of one over two signals, $f_1(t)$ or $f_2(t)$, both of them composed by the addition of two sines of different frequencies plus a noise component, ν:

$$\begin{cases} f_1(t) = A_{11} \sin(2\pi f_{11} t) + A_{12} \sin(2\pi f_{12} t) + \nu \\ f_2(t) = A_{21} \sin(2\pi f_{21} t) + A_{22} \sin(2\pi f_{22} t) + \nu, \end{cases} \tag{3.10}$$

where $\nu \sim \mathcal{N}(0, \sigma_\nu^2)$ and σ_ν^2 vary during the experiments, the amplitudes have fixed values $A_{11}, A_{12}, A_{21}, A_{22} = 2$, and each of the frequencies used are chosen from a different interval as: $f_{11} \sim [1, 5], f_{12} \sim [15, 20], f_{21} \sim [5, 10], f_{22} \sim [20, 25]$. An example sampled time series is shown in Figure 3.5.

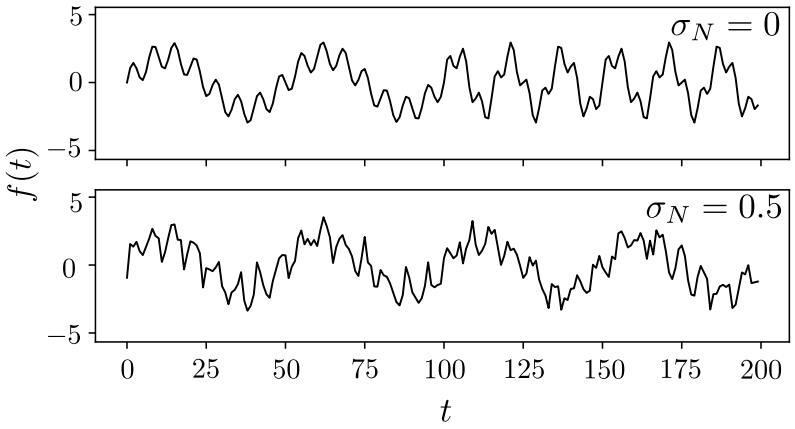

Figure 3.5: Synthetic dataset example signal, generated with $f_{11} = 2$, $f_{12} = 15$, $f_{21} = 6$, $f_{22} = 20\ Hz$. For each signal, each half randomly varies according to the Bernoulli distribution. Noise's standard deviation is increased from top to bottom.

To evaluate Spectral Attention advantages, we train two models on this dataset: 1) DeepAR as base model, and 2) SAAM (integrating the SA module into the DeepAR baseline). Both of them use the exact same architecture for the common parts: the embedding function is performed by an LSTM of 3 layers and 10 hidden units per layer and the probabilistic model is composed of a fully connected network.

Empirical Analysis

Both models, DeepAR and SAAM, are trained using 500 signals from the synthetic dataset with a noise component $N \sim \mathcal{N}(0, \sigma_\nu^2 = 0.5)$. We then evaluate the trained models in two different scenarios. First, in absence of a noise component, $N \sim \mathcal{N}(0, \sigma_\nu^2 = 0)$; second, in the same conditions they were trained, $N \sim \mathcal{N}(0, \sigma_\nu^2 = 0.5)$. Moreover, for each of the previous cases, we evaluate the models using different forecast horizons, starting with $\{t_0 = 175, \tau = 25\}$, and up to $\{t_0 = 105, \tau = 95\}$. In this last case, the final 95 time steps are forecasted after observing just the five first samples from the time series' second half. Setting $t_0 \leq 100$ would be senseless, as $t \in [0, 100)$ contains no evidence about the time series configuration during $t \in [100, 200]$.

Table 3.2 shows the results reported by DeepAR and SAAM. After training the models, 10 evaluations per scenario are conducted during testing. On average, SAAM improves the ND by a 28.4%, $RMSE$ by a 27.7%, $QL_{\rho=0.5}$ by a 28.2%, and $QL_{\rho=0.9}$ by a 28.4%, proving that Spectral Attention inclusion enhances the model's performance.

Furthermore, note that the Global Spectral Attention model contained in the SA module helps accelerate global trends detection: the earlier the forecast window starts (smaller t_0), the better the results of SAAM with respect to the DeepAR base model are. Specifically, for $N \sim \mathcal{N}(0, 0)$ and $t_0 = 175$ (the most favourable setting as there is no noise and just 25 data points to forecast), SAAM and DeepAR report very similar results while, with an increased forecast window of $t_0 = 105$, SAAM architecture thoroughly enhances base model's results: ND is improved by a 54.3%, $RMSE$ by a 40.9%, $QL_{\rho=0.5}$ by a 54%, and $QL_{\rho=0.9}$ by a 56.9%.

Table 3.2: Comparative of DeepAR vs SAAM on the synthetic dataset for different noise component's variance and forecast lengths.

DeepAR/SAAM							
σ_ν^2	**Model**	t_0	τ	**ND**	**RMSE**	$\mathbf{QL}_{\rho=0.5}$	$\mathbf{QL}_{\rho=0.9}$
$\sigma_\nu^2 = 0$	**DeepAR**	**175**	**25**	0.25494 ± 0.000	0.35004 ± 0.000	0.26250 ± 0.000	**0.21830 ± 0.000**
		150	**50**	0.27403 ± 0.000	0.37677 ± 0.000	0.27882 ± 0.000	0.26836± 0.000
		120	**80**	0.37867 ± 0.000	0.47434 ± 0.000	0.38511 ± 0.000	0.38712± 0.000
		110	**90**	0.75234 ± 0.000	1.06398 ± 0.000	0.75810 ± 0.001	0.73909 ± 0.001
		105	**95**	0.82233 ± 0.000	1.13000 ± 0.000	0.82801 ± 0.001	0.80492 ± 0.001
	SAAM	**175**	**25**	**0.24101± 0.003**	**0.28680± 0.003**	**0.24872± 0.004**	0.221701± 0.003
		150	**50**	**0.20145± 0.002**	**0.24868± 0.004**	**0.20594± 0.002**	**0.26325± 0.004**
		120	**80**	**0.17323± 0.004**	**0.21293± 0.006**	**0.17811± 0.005**	**0.13148± 0.001**
		110	**90**	**0.20326± 0.001**	**0.28207± 0.001**	**0.20828± 0.001**	**0.14980± 0.003**
		105	**95**	**0.37575± 0.003**	**0.66817± 0.001**	**0.38062± 0.004**	**0.34665± 0.002**
$\sigma_\nu^2 = 0.5$	**DeepAR**	**175**	**25**	0.58778 ± 0.000	0.75065 ± 0.000	0.59492 ± 0.001	0.59893 ± 0.001
		150	**50**	0.60786 ± 0.000	0.76756 ± 0.000	0.61322 ± 0.000	0.61094 ± 0.000
		120	**80**	0.63071 ± 0.000	0.79379 ± 0.000	0.63531 ± 0.001	0.62907 ± 0.001
		110	**90**	0.78535 ± 0.000	1.03135 ± 0.000	0.79090 ± 0.001	0.79474 ± 0.001
		105	**95**	0.86135 ± 0.000	1.12157 ± 0.000	0.86594 ± 0.001	0.85522 ± 0.001
	SAAM	**175**	**25**	**0.56469± 0.002**	**0.71553± 0.002**	**0.57267± 0.002**	**0.61914± 0.005**
		150	**50**	**0.55649± 0.001**	**0.70871± 0.002**	**0.56057± 0.001**	**0.60490± 0.003**
		120	**80**	**0.57218± 0.002**	**0.73353± 0.002**	**0.57674± 0.002**	**0.52398± 0.002**
		110	**90**	**0.67392± 0.001**	**0.89065± 0.001**	**0.67823± 0.001**	**0.61661± 0.001**
		105	**95**	**0.70326± 0.009**	**0.93480± 0.012**	**0.70809± 0.009**	**0.65422± 0.012**

SAAM's ability to detect and incorporate relevant trends into the forecast has direct implications for the reported results. Table 3.3 shows a comparison for different noise levels of the reported metrics' degradation while increasing the forecast window from $\{t_0 = 175, \tau = 25\}$ to $\{t_0 = 105, \tau = 95\}$: SAAM's deterioration is much smaller than DeepAR's.

Further, Figure 3.6 shows the evolution of the training loss and validation ND error. In this figure, it can be seen that a faster training convergence is

Table 3.3: DeepAR & SAAM accuracy deprecation comparative between $t_0 = 175/\tau = 25$ and $t_0 = 105/\tau = 95$.

		$\sigma_\nu^2 = \mathbf{0}$	$\sigma_\nu^2 = \mathbf{0.5}$
DeepAR	**ND**	-222,56 %	-46,55 %
	RMSE	-222,82 %	-49,41 %
	$\mathbf{QL}_{\rho=0.5}$	-215,43 %	-45,56 %
	$\mathbf{QL}_{\rho=0.9}$	-268,72 %	-42,79 %
SAAM	**ND**	-55,91 %	-24,54 %
	RMSE	-132,97 %	-30,64 %
	$\mathbf{QL}_{\rho=0.5}$	-53,03 %	-38,34%
	$\mathbf{QL}_{\rho=0.9}$	-56,36 %	-5,67 %

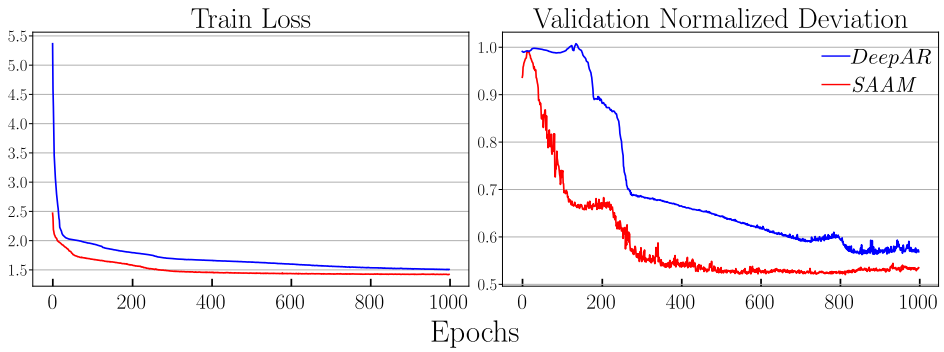

Figure 3.6: Training loss (left) and validation ND (right). The latter is for a forecast window $\tau = 50$.

achieved while using the SA module. Furthermore, SAAM reaches the minimum validation error long before DeepAR.

Explainability Analysis

We now visualise how Spectral Attention affects the model's latent space during the forecasting. To do so, we study SAAM's internal behaviour through the embedding representation while using an LSTM with 1 layer and 5 hidden units per layer as embedding. In Figure 3.7, we show the hidden representations produced by SAAM during the forecasting of a time series z_T^i. Each row in this figure represents one of those 5 hidden dimensions at time $t = 200$ (predicting the time series' final time step). SAAM's hidden variables are displayed as:

- Blue lines represent the hidden representation of the LSTM before SA. This is, the representation that DeepAR would use to forecast.

- Red lines represent SA module's output, $\mathbf{F}^i_{t=200}$ with dimension $(D \times T_F)$, being $D = 5$ and $T_F = 100$. This is the representation that DeepAR-based SAAM uses in order to perform the forecast.

- We also represent the true signal z_T^i in the first row of Figure 3.7, the noise component is removed for clarity. This specific sequence z_T^i can be described as:

$$f(t) = \begin{cases} 2\sin(2\pi t) + 2\sin(40\pi t) + \nu, \ t \in [0, 100) \\ 2\sin(10\pi t) + 2\sin(40\pi t) + \nu, \ t \in [100, 200] \end{cases} \tag{3.11}$$

Thus, Figure 3.7 shows how the SA module modifies the latent space of the model in order to incorporate global trends into the LSTM's hidden representation, making the model immediately aware of trend changes: in Dim. 0 and 2, $\mathbf{F}^i_{T_F}$ exhibits a trend associated with $f_3 = 5Hz$ as soon as time $t \approx 100$, while this component does not appear in $\mathbf{E}^i_T$ until $t \approx 160$. Furthermore, $\mathbf{F}^i_{T_F}$ incorporates in both Dim. 3 and 4 a $20Hz$ component that z^i_T exhibits for both $f_1(t)$ and $f_2(t)$.

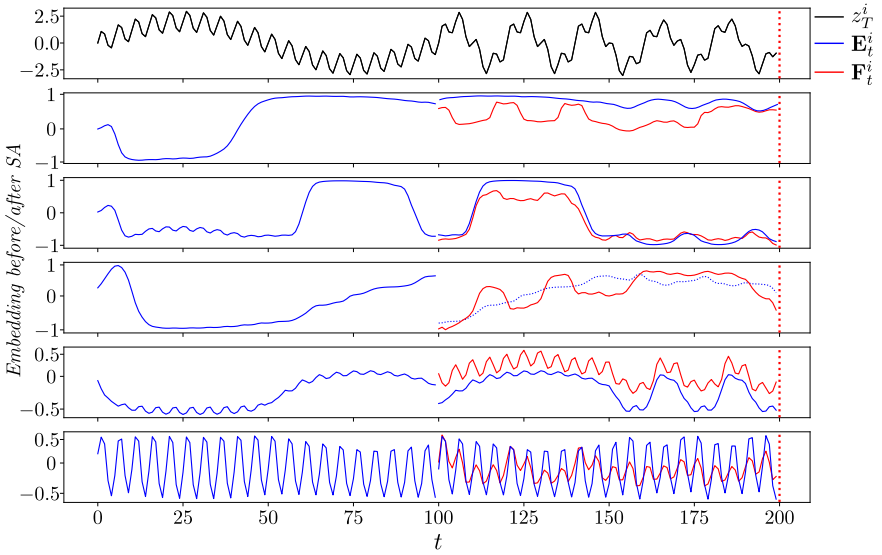

Figure 3.7: Hidden representations of SAAM model while forecasting at time $t = 200$ (marked by red dotted lines). A filtering window of size $T_F = 100$ is used.

These examples show the ability of the proposed architecture to incorporate patterns that the time series exhibit into the forecast. Therefore, in this context, explainability refers to the ability to illustrate which are the main global frequencies that are driving the forecast and how they are incorporated into the embedded representations of SAAM, as Figure 3.7 shows.

Finally, Figure 3.8 displays some forecast examples by DeepAR and SAAM. DeepAR frequently fails to detect trend changes at $t = 100$ and to incorporate certain frequency variations into the mean and standard deviation of the forecast, while SAAM correctly manages these situations, as Figure 3.8 shows.

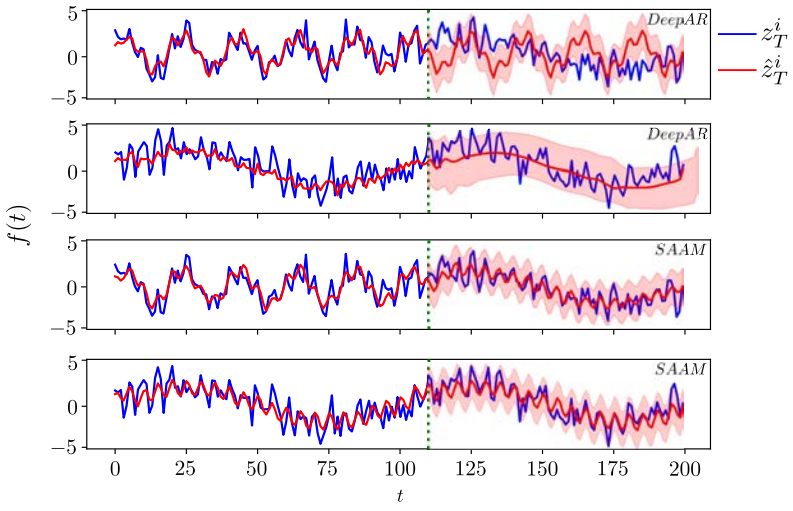

Figure 3.8: Synthetic dataset's forecasts examples by DeepAR (top rows) and SAAM (bottom rows) on the same two time series, $N \sim \mathcal{N}(0, \sigma_\nu^2 = 0.5)$, $\{t_0 = 110, \tau = 90\}$. Green dotted vertical lines mark the forecast starts. Predictions' mean and variation appear in red, and ground-truth in blue.

3.4.4 Real World Datasets

Utilising several real-world datasets, we compare the performance of SAAM with other state-of-the-art models: two classic forecasting techniques, ARIMA (G. E. Box and Gwilym M Jenkins, 1970) and ETS (R. J. Hyndman and Athanasopoulos, 2018); Facebook's open-source library for univariate time series forecasting based on additive models, Prophet (Taylor and Letham, 2018), a recent matrix factorization method, TRMF (H.-F. Yu et al., 2016); an RNN-based State Space Model, DSSM (Rangapuram et al., 2018); N-BEATS (Oreshkin et al., 2019), one of the best performing models in forecast challenges as the M4 forecast competition (Makridakis, Spiliotis, and Assimakopoulos, 2018); DeepAR (Salinas et al., 2020); and ConvTrans (S. Li et al., 2019).

Two different configurations are proposed for SAAM: the first one uses DeepAR as base model, the second combines ConvTrans with SA, and both comply with the architecture of Figure 3.3. A basic framework for the common parts is maintained during all the experiments: for the DeepAR base model, the embedding consists of 3 LSTM layers with 40 hidden units per layer while, for the ConvTrans base proposal,

a Transformer with 8 heads and 3 layers is used. Both sets of parameters appear in the original articles (Salinas et al., 2020; S. Li et al., 2019) as optimal choices.

The electricity[2] and traffic datasets [3] (Dua and Graff, 2017) are evaluated using two different forecast windows, of one and seven days. The electricity dataset contains hourly time series of energy consumption of 370 customers. Similarly, the traffic dataset contains the hourly occupancy rates, with values between zero and one, of 963 car lanes in San Francisco area freeways.

Three more datasets, with different forecast windows each, are also used. The solar dataset [4] (Measurement and Group, 2006) contains the solar power production records from January to August 2006 from 137 plants in Alabama and exhibits hourly measurements. Forecast windows of 1 day are predicted during the evaluation. The wind dataset [5] (Dane, 2015) contains daily estimates of 28 countries' wind energy potential in a period from 1986 to 2015, expressed as a percent of a power plant's maximum output. Finally, the M4-Hourly dataset [6] (Makridakis, Spiliotis, and Assimakopoulos, 2018), contains 414 hourly time series from the M4 competition (Makridakis and Spiliotis, 2020). Table 3.4 summarizes each dataset and the models' architecture used.

All datasets use covariates $\{\mathbf{x}_t^i\}_{i=1}^N \in \mathbb{R}^C$, composed by the hour of the day, day of the week, week of the year and month of the year, for daily, weekly, monthly, and yearly data, respectively. Also, covariates that measure the distance to the first observation of the time series, as well as an item index identification for each time series, are used.

Table 3.5 shows the results obtained for both electricity and traffic datasets, with forecast windows of 1 and 7 days. Table 3.6 shows the results for solar, M4, and wind datasets.

Some conclusions can be drawn from these results. The classic methods evaluated, ARIMA and ETS, perform the worst, probably due to the incapacity of detecting

[2] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
[3] https://archive.ics.uci.edu/ml/datasets/PEMS-SF
[4] https://www.nrel.gov/grid/solar-power-data.html
[5] https://www.kaggle.com/sohier/30-years-of-european-wind-generation
[6] https://www.kaggle.com/yogesh94/m4-forecasting-competition-dataset

Table 3.4: Datasets evaluated's details.

	Electricity	Traffic	Solar	Wind	M4
Length	32304	129120	5832	10957	748
# time series	370	963	137	28	414
Granularity	Hourly	Hourly	Hourly	Daily	Hourly
Domain	$\mathbb{R}$	[0,1]	$\mathbb{R}$	$\mathbb{R}$	$\mathbb{R}$
Batch Size	128	128	128	64	128
Learning Rate	1e-3	1e-3	1e-3	1e-3	1e-3
# LSTM Layers	3	3	3	3	3
# Hidden Units/Layer	40	40	40	40	40
# Heads	8	8	8	8	8
# Layers	3	3	3	3	3
Forecast Window	1 Day / 7 Days	1 Day / 7 Days	1 Day	210 Days	2 Days
Encoder Length	168 / 24	168 / 24	168	162	128
Decoder Length	24 / 168	24 / 168	24	210	48

Table 3.5: Evaluations summary, using $QL_{\rho=0.5}/QL_{\rho=0.9}$ metrics, on Electricity and Traffic datasets with forecast windows of 1 and 7 days, where $^{\triangle}$ are extracted from (S. Li et al., 2019).

Method	Dataset			
	elect-1d	**elect-7d**	**traffic-1d**	**traffic-7d**
ARIMA $^{\triangle}$	0.154/ 0.102	0.283 / 0.109	0.223 / 0.137	0.492 / 0.280
Prophet	0.108/ 0.099	0.160 / 0.154	0.256 / 0.197	0.333 / 0.296
ETS $^{\triangle}$	0.101 / 0.077	0.121 / 0.101	0.236 / 0.148	0.509 / 0.529
TRMF $^{\triangle}$	0.084 / -	0.087 / -	0.186 / -	0.202 / -
DSSM $^{\triangle}$	0.083 / 0.056	0.085 / 0.052	0.167 / 0.113	0.168 / 0.114
N-BEATS	0.069/ 0.052	0.115 / 0.097	0.125 / 0.101	**0.119** / 0.099
DeepAR	0.075 / 0.040	0.082 / 0.053	0.159 / 0.106	0.251 / 0.169
ConvTras	0.059 / 0.034	0.079 / 0.051	0.152 / 0.102	0.172 / 0.110
SAAM(DeepAR)	0.064 / 0.032	0.076 / **0.037**	0.123 / 0.099	0.246 / 0.167
SAAM(ConvTrans)	**0.056 / 0.029**	**0.073** /0.046	**0.120 / 0.083**	0.155 / **0.098**

shared patterns across the different time series. Prophet reports comparable results to classic models in the electricity and traffic datasets. Nevertheless, its performance in relation to deep-learning approaches improved with the solar, M4, and wind datasets, demonstrating its usefulness in smaller data regimes. The results reported for TRMF are slightly better than previous approaches but, for most configurations, it is not capable of beating Deep Neural Networks-based approaches, where DeepAR and ConvTrans solidly exceed DSSM. The two proposed variations of SAAM always improve the base architectures' performance and, with the exception of the

Table 3.6: Evaluations summary, using $QL_{\rho=0.5}/QL_{\rho=0.9}$ metrics, on Solar, M4, and Wind datasets with different forecast windows, where $^{\triangle}$ are extracted from (S. Li et al., 2019).

Method	Dataset		
	Solar	**M4**	**Wind**
Prophet	0.256 / 0.175	0.116 / 0.111	0.305 / 0.275
TRMF$^{\triangle}$	0.241 / -	- / -	0.311 / -
N-BEATS	0.227 / 0.168	**0.023 / 0.018**	0.302 / 0.283
DeepAR	0.222 / 0.093	0.085 / 0.044	0.286 / 0.116
ConvTras	0.210 / 0.082	0.067 / 0.049	0.287 / 0.111
SAAM(DeepAR)	**0.191 / 0.066**	0.048 / 0.029	0.282 / **0.105**
SAAM(ConvTrans)	0.197 / 0.069	0.061 / 0.044	**0.278** / 0.108

Table 3.7: Improvement percentage on $QL_{\rho=0.5}/QL_{\rho=0.9}$ metrics for each base model.

Dataset	Base Model	
	DeepAR	**ConvTrans**
elect-1d	15.33% / 20.75%	5.08% / 14.71%
elect-7d	7.32% / 30.19%	7.59% / 9.80%
traffic-1d	22.64% / 6.60%	21.1% / 18.6%
traffic-7d	1.99% / 1.18%	9.8% / 10.91 %
Solar	13.96% / 29.03%	6.19% / 15.85%
M4	43.53% / 34.09%	8.96% / 10.20%
Wind	1.05% / 9.48%	3.14% / 2.70%

traffic-7d dataset's $QL_{\rho=0.5}$ metric and the M4 dataset, it outperforms all other models. It should be noticed that the original performance difference between N-BEATS and DeepAR/ConvTrans in these two scenarios was sufficiently large to prevent SAAM from overperforming N-BEATS after including the SA module in the base architectures.

Finally, Table 3.7 shows a comparison between the base models, DeepAR and ConvTrans, and their SAAM version, which consistently improves base models' forecast accuracy. For DeepAR, $QL_{\rho=0.5}$ is improved by a 15.1%, and $QL_{\rho=0.9}$ by a 18.8% after including the SA module on SAAM. For ConvTrans, $QL_{\rho=0.5}$ is improved by a 8.8%, and $QL_{\rho=0.9}$ by a 11.8% when using our proposed SAAM architecture.

We should remark that ConvTrans, a Transformer-based architecture, operates using self-attention (Vaswani et al., 2017) in the time-domain. Therefore, measuring

the performance of the base ConvTrans model versus the enhanced version, which includes the SA module, entails a comparison between time-domain and spectral-domain attention. From this comparability, it can be observed how the incorporation of the proposed spectral-domain attention mechanism entangles an improvement in the reported results.

These experiments prove how different deep autoregressive models, with significant differences between them, can be improved by correctly incorporating frequency domain information into the forecasting, with analogous complexity to time-domain attention mechanism but featuring the possibility of using the global and local Fourier Transforms' number of points as hyper-parameters that allow reducing the model's complexity.

3.4.5 Ablation Study

Finally, to quantify the real effect of the SA module and understand the effectiveness of its components, an ablation study is conducted. SA's basic blocks: the Local Spectral Attention model and the Global Spectral Attention model, described in Section 3.3, are separately evaluated.

To assess the behaviour of both frequency-domain attention models, two ablation studies with two different datasets are conducted to secure the robustness of the conclusions. For both studies, SAAM is trained using an LSTM as embedding function.

Considering that SA's output obeys to: $\mathbf{A}_t^i = \mathbf{L}_t^i \odot \boldsymbol{\alpha}_{l,t}^i + \mathbf{G}_t^i \odot \boldsymbol{\alpha}_{g,t}^i, \in \mathbb{R}^{D \times N_{\mathcal{FT}}}$, as stated in Table 3.1, three different configurations of the model are tested: 1) SAAM; 2) SAAM without using Local Spectral Attention, $\boldsymbol{\alpha}_{l,t}^i = 1$; 3) SAAM without using Global Spectral Attention, $\boldsymbol{\alpha}_{g,t}^i = 0$.

Notice that fixing $\boldsymbol{\alpha}_{l,t}^i = 1$ makes no change in the local context, which implies that no filtering is performed. Besides, $\boldsymbol{\alpha}_{l,t}^i = 0$ disables models' abilities to incorporate global trends into the local context.

The first ablation study uses the synthetic dataset explained in Section 3.4.3 with a noise component of $\sigma_\nu^2 = 0$, no covariates, and a forecast window of $\{t_0 = 120, \tau = 80\}$. Table 3.8 shows the obtained results. The degradation produced by

Table 3.8: Ablation study on the synthetic dataset.

		ND	RMSE	$\mathbf{QL}_{\rho=0.5}$	$\mathbf{QL}_{\rho=0.9}$
$\sigma_\nu^2 = 0$	**Full SAAM**	**0.028 ± 0.000**	**0.044 ± 0.000**	**0.029 ± 0.000**	**0.025 ± 0.000**
	No global SA	0.886 ± 0.003	1.064 ± 0.004	0.893 ± 0.003	1.077 ± 0.009
	No local SA	2.084± 0.000	2.352 ± 0.000	2.100 ± 0.000	2.416 ± 0.001
$\sigma_\nu^2 = 0.5$	**Full SAAM**	**0.816 ± 0.000**	**1.084 ± 0.000**	**0.823 ± 0.001**	**0.859 ± 0.001**
	No global SA	0.949 ± 0.007	1.167± 0.009	0.956 ± 0.007	1.136 ± 0.007
	No local SA	1.936 ± 0.000	2.263 ± 0.000	1.962 ± 0.000	1.936 ± 0.001
$\sigma_\nu^2 = 1$	**Full SAAM**	**0.911 ± 0.000**	**1.145 ± 0.000**	**0.915 ± 0.000**	**0.891 ± 0.001**
	No global SA	0.990 ± 0.006	1.231 ± 0.005	0.995 ± 0.006	1.126 ± 0.008
	No local SA	1.567 ± 0.000	1.890 ± 0.000	1.591 ± 0.001	1.493 ± 0.002

Table 3.9: Ablation study on the electricity dataset with 7 days forecast windows.

	ND	RMSE	$\mathbf{QL}_{\rho=0.5}$	$\mathbf{QL}_{\rho=0.9}$
Full SAAM	**0.076 ± 0.000**	**0.496 ± 0.001**	**0.076 ± 0.000**	**0.03759 ± 0.000**
No global SA	0.236 ± 0.001	2.283 ± 0.029	0.237 ± 0.001	0.078 ± 0.001
No local SA	0.263 ± 0.000	2.665 ± 0.001	0.264 ± 0.000	0.092 ± 0.000

the ablation when $\sigma_\nu^2 = 0$ is bigger than other considered cases, which is normal considering that the model was trained on those conditions. Also, to disable the local attention $\boldsymbol{\alpha}_{l,t}^i = 1$, produces worse results than $\boldsymbol{\alpha}_{g,t}^i = 0$, when the global attention is not used. The latter causes a bigger deterioration in predictions' variance, which could be a sign of the model's inability to follow the trend after removing the global attention model.

A second ablation study is performed on the electricity dataset using a 7 days forecasting window. Again, the Local and Global Spectral Attention models are separately evaluated. For this dataset, the degradation of the results is similar for both ablations, as Table 3.9 shows. As in the synthetic dataset ablation study, not using the Global Spectral Attention model translates into a higher standard deviation on the results.

3.5 Discussion

In this chapter, we have proposed a novel methodology for neural probabilistic time series forecasting that marries signal processing techniques with deep learning-based

autoregressive models, developing an attention mechanism that operates over the frequency domain. Thanks to this combination, which is enclosed in the Spectral Attention module, local spectrum filtering and global patterns incorporation meet during the forecast. To do so, two attention models operate over the embedding's spectral domain representation to determine, at every time instant and for each time series, which components of the frequency domain should be considered noise and hence be filtered out, and which global patterns are relevant and should be incorporated into the predictions. The Spectral Attention module modifies the model's latent space in order to incorporate this information.

Experiments on synthetic and real-world datasets confirm these statements and unveil how our suggested modular architecture can be incorporated into a variety of base deep autoregressive models, consistently improving the results of these base models and achieving state-of-the-art performance. Especially, in noisy environments or short conditioning ranges, our method stands out by explicitly filtering the noise and rapidly recognizing relevant trends.

Besides the baselines used during the experiments, several forecasting architectures can be characterized by employing an embedding function and a probabilistic model, making them suitable for integrating the SA module. For instance, NBeats (Oreshkin et al., 2019) uses a sequence of stacks, each of which combines multiple blocks, to perform an embedding over the input data. NBeats' final forecast is produced through the sum of forecasts of all blocks, allowing the integration of the SA module to modify these embedded representations. Further, the recently proposed Informer model (Zhou et al., 2021) uses a encoder-decoder Transformer architecture to perform the time series embedding and probabilistic forecasting. As done with ConvTrans, the SA module could modify the original embedded representation, performing local spectral filtering and global patterns integration into the forecast.

Note that the SA module comes with some additional training parameters, as discussed in Section 3.4. Our experiments demonstrate that SA consistently enhances the baselines' performance.

Future work should explore new approaches to integrate signal processing techniques into Deep Neural Networks based approaches; such as analyzing the feasibility of alternative spectral representations, as Wavelets and the Discrete Cosine Transform (DCT), among others, and the integration of Spectral Attention in Transformer's self-attention mechanism.

4

Volatility Forecasting from High-Frequency Data with Dilated Causal Convolutions

Contents

In the previous chapters of this thesis, we conducted a review of state-of-the-art methods in the context of time series modelling, highlighting some of the challenges associated with deep learning-based techniques. Furthermore, in Chapter 3, we

proposed using models operating in the Fourier domain as a potential solution to alleviate these problems. Building on these foundations, Chapters 4 and 5 shift towards a more applied focus, showcasing the use of machine learning techniques in the context of quantitative finance.

Quantitative finance involves the use of mathematical and statistical methods to analyse and model financial markets and instruments, comprising a rich domain for the application of machine learning techniques. We propose novel and improved temporal models to address two highly relevant quantitative finance-related problems: the forecasting of realised volatility, and the estimation of fill times for limit orders posted in the Limit Order Book (LOB). Overall, Chapter 4 provides a concrete illustration of the utility and potential impact of machine learning models on the former, whereas Chapter 5 focuses on the latter. By doing so, we aim to demonstrate the versatility and effectiveness of our proposed machine learning-based techniques in real-world applications.

Regarding the specific problem of volatility forecasting, addressed in this chapter, it plays a central role among equity risk measures. Besides traditional statistical models, modern forecasting techniques based on machine learning can be employed when treating volatility as a univariate, daily time series. However, econometric studies have shown that increasing the number of daily observations with high-frequency intraday data helps to improve volatility predictions. In this chapter, we propose DeepVol, a model based on Dilated Causal Convolutions (DCC) that uses high-frequency data to forecast day-ahead volatility. We show that the dilated convolutional filters are ideally suited to extract relevant information from intraday financial data, thereby mimicking (via a data-driven approach) the econometric models which incorporate realised measures of volatility into the forecast. Simultaneously, dilated convolutional filters trained with intraday high-frequency data help us avoid the limitations of models that use daily data, such as model misspecification or manually designed handcrafted features, whose devise involves optimising the trade-off between accuracy and computational efficiency and makes models prone to lack of adaptation into changing circumstances. In

our analysis, we use two years of intraday data from NASDAQ-100 to evaluate the performance of DeepVol. Our empirical results suggest that the proposed deep learning-based approach learns global features from high-frequency data, achieving more accurate predictions than traditional methodologies and producing more appropriate risk measures.

This chapter, which presents the contents of Moreno-Pino and Zohren (2022), is organised as follows. Section 4.1 provides a brief overview of the volatility forecasting literature, motivating the need for our proposal. Section 4.2 details the dataset used, while Section 4.3 contains a brief overview of classic techniques for volatility forecasting, describing the baselines used for benchmarking purposes and the metrics that will be utilised for model comparison. Section 4.4 presents the proposed model, which is empirically evaluated in Section 4.5. Finally, Section 4.6 summarises the findings and concludes this chapter.

4.1 Introduction to Volatility Forecasting

In recent years, measures of volatility to assess the risk of portfolios have received considerable attention (Brownlees and Gallo, 2010), which has given rise to an increase in the use of volatility conditional portfolios (C. R. Harvey et al., 2018). Different studies have reported an overall gain in their Sharpe ratio (Moreira and Muir, 2017), as well as a reduction of the likelihood of observing extreme heavy-tailed returns when using them (C. R. Harvey et al., 2018). Consequently, the development of volatility forecasting models has attracted broad research efforts. In this paper, we propose a data-driven model, DeepVol, to forecast realised volatility from high-frequency data.

In this context, most of the models practitioners use for volatility forecasting are based on classic methodologies. Among them, the GARCH model (Bollerslev, 1986), which uses past volatility and daily squared returns as the driving variables for predicting day-ahead volatility, is particularly popular.

Recent articles use realised measures as predictors for realised volatility, improving the volatility prediction accuracy of classic models (Peter Reinhard Hansen

et al., 2012). These realised measures, which are non-parametric estimators of the variation of an asset's price during a time gap, are a tool that extracts and summarises information contained in high-frequency data (Torben G Andersen, Bollerslev, and Diebold, 2010).

However, methodologies that take advantage of realised measures require pre-processing steps to use them, as they cannot directly model the complex relations exhibited by intraday financial data. One concrete example of this pre-processing is the procedure followed to obtain the realised measures themselves, which summarises the vector of daily intraday high-frequency data into single scalars in order to avoid having to manage the microstructure noise associated with the former. In contrast, our work uses raw high-frequency data as input to the model, which requires no pre-processing of data and avoids its associated consequences, such as data dismissing while summarising intraday data into daily realised measures. Our proposed model is able to deal directly with the microstructure noise linked to higher intraday data sampling frequencies.

Among the methodologies employing realised measures, the HEAVY model (Shephard and Sheppard, 2010) is of special appeal among industry practitioners (Karanasos et al., 2022; Papantonis et al., 2022; Yuan et al., 2022). HEAVY is based on insights from the ARCH architecture, with superior performance over other classical benchmarks, as shown in Section 4.5.

Nevertheless, the inability of realised measures-based models, such as HEAVY and Realized GARCH (Peter Reinhard Hansen et al., 2012), to use unprocessed raw high-frequency data as input exposes them to several disadvantages. Firstly, the dependence on the realised measures for day-ahead volatility forecasting artificially limits the amount of information these architectures use, which is not the case when using raw intraday data. Furthermore, some of the most used realised measures of volatility lack robustness to microstructure noise (Baars, 2014), implying that the trained models may be based on biased data. Finally, methodologies based on realised measures often rely on manually designed handcrafted features, as the realised measures design itself, formulated to optimise the trade-off between accuracy and

increasing computational costs, which, together with common model misspecification of classical model-based approaches, undermine reported performances.

Here, we use Deep-Neural-Networks (DNN) to take advantage of the abundance of high-frequency data without prejudice, preventing the constraints of models based on realised measures in the context of day-ahead volatility forecasting. Despite the success of these DNN architectures in different areas, such as healthcare, image recognition, and text analytics, they have not been widely adopted for the problem of volatility forecasting, leading to a large gap between modern machine learning models and those applied in the volatility framework. Among DNN-based models, Recurrent Neural Networks (RNN) (Rumelhart et al., 1985) and Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) are the most popular approaches with regard to time series forecasting (Lim and Zohren, 2021). Furthermore, the addition of the attention mechanism (Bahdanau et al., 2015) into these base architectures allowed them to focus on the most relevant input data while producing predictions, making them especially prominent in fields such as Natural Language Processing (NLP). These advances also led to the appearance of Transformer models (Vaswani et al., 2017), which were initially introduced for NLP, and later used for the problem of time series forecasting (Moreno-Pino, Olmos, et al., 2023).

These models are applied in the context of financial time series through different variations (Lin et al., 2022; Su, 2021). More specifically, regarding volatility forecasting, a number of deep learning architectures are used, such as LSTMs (S. Yu and Z. Li, 2018), CNNs (Borovykh et al., 2017; Vidal and Kristjanpoller, 2020), Graph Neural Networks (GNN) (Chen and Robert, 2022), Transformer models (Ramos-Pérez et al., 2021), and NLP-based word embedding techniques (Rahimikia and Poon, 2020; Rahimikia, Zohren, et al., 2021). Furthermore, models combining traditional volatility forecasting methods with deep learning techniques can be found in the literature (Kim and Won, 2018; Mademlis and Dritsakis, 2021), as well as other approaches using DNN as calibration methods for implying volatility surfaces (Horvath et al., 2019), proving how neural network-based approaches work as complex pricing function approximators.

We capitalise on the increased availability of high-frequency data. In this work, we employ a Dilated Causal Convolutions (DCC)-based model. This architecture, initially proposed as a fully probabilistic model for audio generation (Oord et al., 2016), with equivalents for image-related problems (Van Oord et al., 2016), possesses a large receptive field that allows it to process large sequences of data without triggering an unrestrained increase in the model's complexity.

In the literature, other works use DCC in the context of realised volatility forecasting. More specifically, Reisenhofer et al. (2022) propose a model based on dilated convolutions, strongly inspired by the well-known Heterogeneous Autoregressive (HAR) model (Corsi, 2009). However, their approach does not use unprocessed raw intraday high-frequency data as input. Conversely, it still bases its predictions on the pre-computed daily realised variance, therefore requiring pre-processing steps to obtain the indispensable realised measures for forecasting the one-step-ahead volatility. This, in our judgement, does not fully explore the capabilities of DCC-based methodologies of exploiting a more dynamic representation of the intraday data. Hence, models adopting DCC-based approaches that operate from daily data still succumb to previously enumerated limitations.

Motivated by the improved performance of classical methods that employ realised measures (Peter Reinhard Hansen et al., 2012; Shephard and Sheppard, 2010), we propose using DCCs to bypass the estimation of these non-parametric estimators of assets' variance, aiming to tackle the volatility forecasting problem from a data-driven perspective. The proposed model, DeepVol, entails several advantages while performing volatility forecasting. Primarily, it does not require any pre-processing steps, as the model directly uses raw high-frequency data as input. Furthermore, DeepVol is not bounded to static realised measures whose use may be counter-productive, i.e., the optimal realised measure to use may vary depending on the traded assets' liquidity. Instead, through the attention mechanism and internal non-linearities, DeepVol intelligently performs the required transformations over the input data to maximise the accuracy of the predictions, combining relevant intraday datapoints and merging them for each day's volatility

forecast, dynamically adapting to different scenarios. Moreover, through the use of dilated convolutions, DeepVol's large receptive field easily processes long sequences of high-frequency data, enabling the model to exponentially increase its input window while performing the predictions. This approach constitutes a purely data-driven method that mimics how handcrafted realised measures condense intraday information, allowing DeepVol to hierarchically integrate the most relevant high-frequency data into the predictions. We perform extensive experiments to show the effectiveness of the proposed architecture, which consistently outperforms the base models used by practitioners.

This chapter provides three main contributions. Firstly, we empirically demonstrate the advantages offered by DCCs when employed to forecast realised volatility based on high-frequency data, providing a data-driven solution which consistently outperforms classical methodologies. The proposed model avoids the limitations of classical methods, such as model misspecification or their inability to use intraday data to perform the forecast directly. Secondly, we provide an analysis showing how the proposed deep learning model maximises the trade-off between extracting signals from high-frequency data while minimising the microstructure noise implicit to higher sampling frequencies. Reported results agree with studies validating this same trade-off for the construction of realised measures. Thirdly, the proposed volatility forecasting model generates appropriate risk measures through its predictions in an out-of-sample forecasting task, both in low and high volatility regimes. Moreover, we evaluate the proposed model's generalisation capabilities on out-of-distribution stocks, demonstrating DeepVol's capabilities to transfer learning as it performs accurate predictions into data distributions not observed during the training phase.

4.2 Data and Model Inputs

4.2.1 Data

We use intraday high-frequency data as a starting point for fitting the proposed model. DeepVol and the baseline architectures are trained and tested using two

years of NASDAQ-100 data, from September 30, 2019 to September 30, 2021. High-frequency data of different sampling frequencies (granularities), i.e., 1, 5, 15, 30, and 60 minutes, is used in our analysis. DeepVol will directly perform its prediction from raw high-frequency data, unlike the baseline models, which prior to training, require the estimation of daily statistics. The analyses conducted in this work are based on financial returns, which allow us to transform the original assets' price trend into a quasi-stationary process:

$$r_{i,t} = \log\left(\frac{p_{i,t}}{p_{i-1,t}}\right), \tag{4.1}$$

where $p_{i,t}$ is the last price of an asset in the i^{th} interval on day t, and $r_{i,t}$ is the return over this interval, at the specified sampling frequency, i.e., 1, 5, 15, 30 or 60 minutes.

4.2.2 Baselines: Data Preparation

The benchmark models used in this work are divided into two categories. Firstly, we consider methods that solely use daily returns to perform day-ahead volatility forecasts. Secondly, we examine methods that take advantage of realised measures in the forecasts.

Regarding models depending exclusively on daily returns, these are obtained through an analogous procedure to the one followed to retrieve intraday returns through Eq. (4.1), but using daily returns instead of intraday data. Figure 4.1 shows the effect of this transformation for Apple's stock price over a two year period, converting the daily price trend into daily returns, as well as the associated volatility evolution obtained through the use of a five days rolling window. Moreover, concerning methods utilising realised measures, and in consonance with other studies (C. R. Harvey et al., 2018; Peter Reinhard Hansen et al., 2012; Shephard and Sheppard, 2010), we focus on the realised variance for the scope of this work. The realised variance is a proxy measure of the volatility and is obtained as follows:

$$RV_t = \sum_{i=1}^{I} r_{i,t}^2, \tag{4.2}$$

where $r_{i,t}$ is the i^{th} intraday return for day t, see Eq. (4.1).

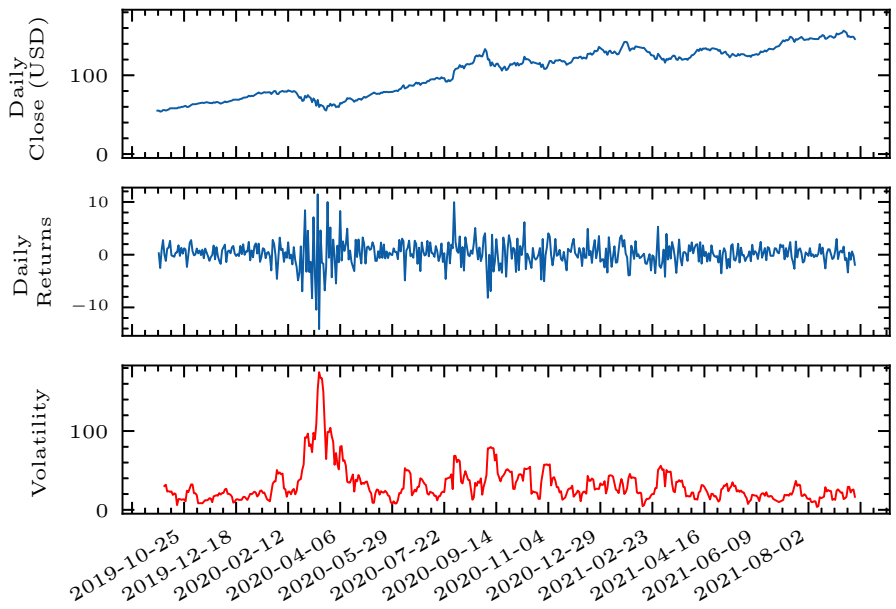

Figure 4.1: Apple's daily data. The top row shows the price trend, the second row the associated daily returns, and the bottom row shows a volatility estimation calculated from a 5-days moving window over the daily returns.

Various works (Torben G Andersen, Bollerslev, Diebold, and Ebens, 2001; Ait-Sahalia et al., 2005) study the use of different sampling frequencies to compute the realised variance through Eq. (4.2). Selecting a specific intraday data sampling frequency to compute the realised volatility (e.g., 5 or 30 minutes) involves the optimisation of a trade-off: while we aim to maximise the number of datapoints used, higher sampling frequencies entail an increase of the microstructure noise, which we want to minimise. We use 5-minutes intraday returns to compute the realised variance through Eq. (4.2), as this sampling frequency is usually accepted as the optimal value (Torben G Andersen, Bollerslev, Diebold, and Ebens, 2001; Bandi and Russell, 2006).

4.2.3 DeepVol: Data Preparation

As previously mentioned, and contrary to classical methods, DeepVol is directly fed with raw high-frequency data, with no pre-processing required. A rolling window approach is used to fit the model, meaning that DeepVol will use a window of intraday data from previous days as the model's input for predicting

the day-ahead realised volatility. Experiments conducted in Section 4.5 explore the optimal window size, hereinafter called receptive field (number of past days used for predicting the day-ahead volatility), and the best intraday data's sampling frequency. The use of this receptive field contrasts with most state-of-the-art methodologies, which operate recursively using all available time series' history. Instead, DeepVol is confined to use a specific receptive field, e.g., the previous day's high-frequency data. This non-recursive architecture reduces the input data length required by the model, which translates into faster training in comparison to purely autoregressive architectures. Finally, we should mention that DeepVol produces a forecast for the day-ahead volatility, σ_t^2, while using intraday high-frequency returns, $r_{i,t}$, as input data. This contrast with most state-of-the-art forecasting architectures, which produce predictions whose granularity (sampling frequency) is the same as the model's input data. Therefore, DeepVol is responsible for learning the necessary relations between the high-frequency data and the daily volatility, implicitly performing this time-domain transformation

4.3 Baseline Models and Metrics

4.3.1 Baseline Models

Most of the models commonly used for volatility forecasting can be traced back to Autoregressive Conditional Heteroscedastic (ARCH) models (Engle, 1982). This family of models assume volatility clustering (Cont, 2007), i.e., large shocks in prices tend to cluster together. ARCH-based models evolved into the well-known Generalized Autoregressive Conditional Heteroscedastic (GARCH) model (Bollerslev, 1986), which is still widely utilised among industry participants. A GARCH(p, q) process is given by:

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2, \tag{4.3}$$

where ω is the model's bias, q is the number of lags (order) of the observed volatility, σ_t^2; and p is the number of lags of the innovations, ε_t. In turn, the returns of

prices are related to the innovations by:

$$r_t = \mu + \varepsilon_t, \tag{4.4}$$

where μ is the expected return (usually set to zero), and the volatility is related to these innovations by means of the residuals, e_t:

$$\varepsilon_t = \sigma_t e_t, \quad e_t \sim \mathcal{N}(0,1). \tag{4.5}$$

The model's parameters $\{\mu, \omega, \alpha, \beta\}$ can be estimated by performing maximum-likelihood estimation of the joint distribution $f(\varepsilon_1, \ldots, \varepsilon_T; \{\mu, \omega, \alpha, \beta\})$. The simplest GARCH model consists on a GARCH$(1,1)$ process where $\sigma_t = 1$ and $\mu = 0$. Several variations leading to new architectures to address the volatility forecasting problem have been developed from the GARCH model. Here, we select some of them for benchmarking purposes. The integrated GARCH (IGARCH) model (Engle and Bollerslev, 1986) modifies the design of the previous model to grant a longer memory in the autocorrelation of the squared returns, allowing the model to react in a more persistent way to the impact of past squared shocks. Also, IGARCH imposes the following restriction on the model's parameters:

$$\sum_{i=1}^{p} \alpha_i + \sum_{j=1}^{q} \beta_j = 1, \tag{4.6}$$

which makes the resulting process a weakly stationary one, since the mean, variance, and autocovariance are finite and constant over time. The idea behind the IGARCH model motivated the development of the Fractionally Integrated GARCH (FIGARCH) process (Baillie et al., 1996), which is able to capture long-term volatility persistence and clustering features. To do so, it integrates a fractional difference operator (lag operator) L into the conditional variance:

$$\sigma_t^2 = \omega + \left[1 - \beta L - \phi L(1-L)^d\right] \varepsilon_t^2 + \beta \sigma_t^2, \tag{4.7}$$

where $0 < d < 1$ is known as the fractional differencing parameters. The FIGARCH model has been widely used thanks to its ability to capture the volatility's persistence and integrate it into its predictions (Cochran et al., 2012; Biage, 2019). Threshold

ARCH (TARCH) models (Rabemananjara and Zakoian, 1993) are also used for benchmarking purposes. The main difference with respect to previous methodologies is that TARCH models divide the distribution of the innovations into disjoint intervals, which are later approximated by a linear function on the conditional standard deviation (Zakoian, 1994). TARCH models are therefore capable of separately considering the influence of positive and negative innovations:

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j \varepsilon_{t-j}^2 \mathbb{I}_{\varepsilon_{t-j}<0}, \tag{4.8}$$

where $\mathbb{I}_{(\cdot)}$ is the indicator function. The main characteristic of TARCH and other threshold-based approaches, such as TGARCH (Park et al., 2009), is their ability to detect abrupt disruptions in the time series through the indicator function, which may be replaced with a continuous function if a smoother transition is desired.

Volatility usually exhibits asymmetric characteristics. This property has led to the development of different asymmetric ARCH-type models. For example, the Asymmetric Power ARCH (APARCH) model (Ding et al., 1993) assumes a parametric form for the conditional heteroskedasticity's powers. It defines the variance dynamics as follows:

$$\sigma_t^\delta = \omega + \sum_{i=1}^{q} \alpha_i \left(|\varepsilon_{t-i}| - \gamma_i \varepsilon_{t-i} \right)^\delta + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^\delta, \tag{4.9}$$

where we now also have to estimate $\delta > 0$, and γ. APARCH models nest many other volatility frameworks that can be obtained by imposing restrictions on the APARCH model's parameters. A similar idea leads to the Asymmetric GARCH model (AGARCH) (Engle and Ng, 1993), which captures the asymmetry in the volatility by using an impact curve associated with the α_i parameter:

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i \left(\varepsilon_{t-i} - \gamma_i \right)^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2. \tag{4.10}$$

Most of the mentioned models, like IGARCH or APARCH, impose restrictions on the parameters in practice, as Eq. (4.6) states. These restrictions are lifted in the Exponential GARCH (EGARCH) model (Nelson, 1991), which is defined as:

$$\ln \sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i \left(|\epsilon_{t-i}| + \gamma_i \epsilon_{t-i} \right) + \sum_{j=1}^{q} \beta_j \ln \sigma_{t-j}^2. \tag{4.11}$$

As evidenced by the definition above, the EGARCH model integrates one powerful volatility clustering assumption into its architecture: negative shocks at time $t-1$ produce a stronger impact on the value of the volatility at time t than positive shocks do, allowing for asymmetric effects between positive and negative asset returns. This asymmetry is known in the volatility forecasting literature as leverage effect (Bouchaud et al., 2001).

All the methods described previously operate through the use of daily returns. However, more recent proposals have included the use of realised measures obtained from high-frequency data as additional input features for daily volatility forecasting. Among these methodologies, the High-Frequency-Based Volatility (HEAVY) model (Shephard and Sheppard, 2010), has shown superior forecasting capabilities (Shephard and Sheppard, 2010; Noureldin et al., 2012; Sheppard and W. Xu, 2019). Formally, the model is defined as follows:

$$\begin{aligned} \operatorname{var}\left(r_t \mid \mathcal{F}_{t-1}^{\mathrm{HF}}\right) &= \sigma_t^2 = \omega + \alpha \mathrm{RM}_{t-1} + \beta \sigma_{t-1}^2, \\ \mathbb{E}\left(\mathrm{RM}_t \mid \mathcal{F}_{t-1}^{\mathrm{HF}}\right) &= \mu_t = \omega_R + \alpha_R \mathrm{RM}_{t-1} + \beta_R \mu_{t-1}, \end{aligned} \tag{4.12}$$

where r_t denotes daily returns, RM_t denotes daily realised measures, and $\mathcal{F}_{t-1}^{\mathrm{HF}}$ denotes the high-frequency data utilised to obtain these realised measures. In the previous equation, the restrictions $\{\omega, \alpha \geq 0, \beta \in [0,1)\}$ are imposed on the variation of the returns, and $\{\omega_R, \alpha_R, \beta_R \geq 0, \alpha_R + \beta_R \in [0,1)\}$ on the realised measures' evolution, while observing the following variables:

$$\begin{aligned} r_t &= \sqrt{\sigma_t^2 z_t}, \\ x_t &= \mu_t z_{RV,t}^2, \end{aligned} \tag{4.13}$$

with:

$$\begin{pmatrix} z_t \\ z_{RV,t} \end{pmatrix} \sim \mathcal{N}(0, I). \tag{4.14}$$

Eq. (4.12) shows that HEAVY consists of two parts. While σ_t^2 explains the development of the unobserved conditional variance, μ_t is responsible for explaining the development of the realised measures. The HEAVY model is clearly motivated by GARCH methodologies, which makes it simple to understand while reporting additional gains in the performance. For further details about it, like parameters inference, we refer the readers to (Shephard and Sheppard, 2010).

4.3.2 Evaluation Metrics

In this section, we define a series of metrics that will be used to assess the day-ahead volatility forecast of our proposed architecture against previously defined baseline models. The Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE) constitute two of the most common error functions to evaluate the performance of volatility forecasting architectures. While a number of articles focus entirely on those two metrics to report performance (Shen et al., 2021; Izzeldin et al., 2019), we complement them with the use of the Symmetric Mean Absolute Percentage Error (SMAPE). This relative error measure has both a lower and upper bound, contrary to the Mean Absolute Percentage Error (MAPE), and it is scale independent. Also, the Maximum Error (ME) is used to illustrate which models produce the more significant inaccuracies: poor performance adapting to new regimes, as volatility shocks, leads certain models to substantial momentary discrepancies between the forecast and the actual volatility, which leads to an increase in the ME. We complement the ME with the Median Absolute Error (MedAE), an outliers-robust metric. Lastly, we include the Quasi Log-Likelihood (QLIKE), which has proven to be a noise-robust loss function in the volatility proxy (Patton, 2011). Both the QLIKE and the RMSE will be used as loss functions to optimise the model's parameters during Section 4.5, while the rest of the metrics will be used to assess the models' performance. Eq. (4.15) summarises the definition of all metrics:

$$
\begin{aligned}
\ell_{MAE}(\sigma_t^2, \hat{\sigma}_t^2) &= \frac{1}{T}\sum_{t=1}^{T}\left|\sigma_t^2 - \hat{\sigma}_t^2\right|, \\
\ell_{rmse}(\sigma_t^2, \hat{\sigma}_t^2) &= \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\sigma_t^2 - \hat{\sigma}_t^2)^2}, \\
\ell_{SMAPE}(\sigma_t^2, \hat{\sigma}_t^2) &= \frac{1}{T}\sum_{t=1}^{T}\frac{|\sigma_t^2 - \hat{\sigma}_t^2|}{(\sigma_t^2 - \hat{\sigma}_t^2)/2}, \\
\ell_{ME}(\sigma_t^2, \hat{\sigma}_t^2) &= max\left(\left|\sigma_t^2 - \hat{\sigma}_t^2\right|\right), \\
\ell_{MedAE}(\sigma_t^2, \hat{\sigma}_t^2) &= median(\left|\sigma_1^2 - \hat{\sigma}_1^2\right|, \cdots, \left|\sigma_T^2 - \hat{\sigma}_T^2\right|), \\
\ell_{QLIKE}(\sigma_t^2, \hat{\sigma}_t^2) &= \frac{1}{T}\sum_{t=1}^{T}\log\left(\hat{\sigma}_t^2\right) + \frac{\sigma_t^2}{\hat{\sigma}_t^2},
\end{aligned}
\tag{4.15}
$$

where $\hat{\sigma}_t^2$ and σ_t^2 represent the volatility forecast and the volatility proxy measure, respectively, with T the total amount of rolling forecasts.

4.4 Model

4.4.1 Problem Definition

Considering a set of assets, $\Delta \in \mathbb{R}^d$, where $d \in \mathbb{N}$ denotes the dimension of the input vector, with $T \in \mathbb{N}$ days' intraday high-frequency data associated to them, $\{\mathbf{r}_t^{1:J}\}_{t=1}^T$, where $\mathbf{r}_t^{1:J} = (r_t^1, r_t^2, \ldots, r_t^J)$ are the intraday returns of the t^{th} day, with T being referred to as receptive field, and with $J \in \mathbb{N}$ the length of each day's intraday data, our goal is to forecast the day-ahead realised volatility:

$$\hat{\sigma}_{T+1}^2 = f_\theta \left(r_{t=1}^1, r_{t=1}^2, \ldots, r_{t=1}^J, r_{t=2}^1, \ldots, r_{t=T}^1, \ldots, r_{t=T}^J \right), \tag{4.16}$$

where $f_\theta : \mathbb{R}^d \to \mathbb{R}^m, m \in \mathbb{N}$, is a function implemented through a DCC-based neural network, with $\theta \in \Theta$ being the learnable parameters of the model from a set $\Theta \in \mathbb{R}^n$, for some $n \in \mathbb{N}$.

These parameters fully specify the corresponding volatility forecast. Therefore, we aim to obtain the set of optimal parameters $\hat{\theta} \in \Theta$ that minimises the difference between the forecasted volatility, $\hat{\sigma}_t^2$, and the volatility's proxy measure σ_t^2 for the considered assets:

$$\hat{\theta} = \underset{\theta \in \Theta}{\operatorname{argmin}} \, \mathcal{L} \left(f_\theta(\Delta), \sigma_t^2(\Delta) \right), \tag{4.17}$$

where $\mathcal{L}(\cdot)$ is the selected metric for evaluating the forecast accuracy.

4.4.2 Dilated Causal Convolutions

Our volatility forecasting proposal, DeepVol, uses DCCs as a technique to integrate high-frequency information into the realised volatility prediction. The deployment of such an architecture allows the use of a large receptive field, permitting an increase in the size of the input sequences while preserving the number of parameters of the network, yielding improved computational efficiency. The proposed architecture

consists of L convolutional layers. The convolution operation performed by the first layer, between the input sequences x and the kernel k, can be defined as follows:

$$F^{(l=1)}(t) = \left(x *_d k^{(l=1)}\right)(t) = \sum_{\tau=0}^{s-1} k_\tau^{(l=1)} \cdot x_{t-d\tau}, \tag{4.18}$$

being d the dilation factor and k the filter, with size $s \in \mathbb{Z}$. For each of the rest l^{th} layers, we can define the convolution operation as:

$$F^{(l)}(t) = \left(F^{(l-1)} *_d k^{(l)}\right)(t) = \sum_{\tau=0}^{s-1} k_\tau^{(l)} \cdot F_{t-d\tau}^{l-1}(t). \tag{4.19}$$

As previous equations state, the inner product performed by the dilated causal convolutions is based on entries that are a fixed number of steps apart from each other, contrary to CNN and Causal-CNN, which operate with consecutive entries. Furthermore, each of the layers in this hierarchical structure defines the kernel operation as an affine function acting between layers:

$$k^{(l)} : \mathbb{R}^{N_l} \longrightarrow \mathbb{R}^{N_{l+1}}, 1 \leq l \leq L. \tag{4.20}$$

As previous equations show, through the use of residual connections, firstly proposed in He et al., 2016, the model connects l^{th} layer's output to $(l+1)^{\text{th}}$ layer's input, enabling the use of deeper models with larger receptive fields. The complete operation of the proposed model can be defined as follows:

$$\sigma_{T+1}^2(\{\mathbf{r}_t^{1:J}\}_{t=1}^T) = \alpha_0 + \sum_{l=1}^{L} \alpha_l \, \sigma_{ReLu}(F^{(l)}(\{\mathbf{r}_t^{1:J}\}_{t=1}^T)), \tag{4.21}$$

where $\sigma_{ReLu} : \mathbb{R} \mapsto \mathbb{R}$ is the selected non-linearity and $\{\alpha_0, \cdots, \alpha_l, \cdots, \alpha_L\}$ is a set of weights applied to the convolutional operations. Figure 4.2 presents an overview of the utilised architecture, where a receptive field with previous T days' intraday data is processed through a hierarchy of dilated convolutions to forecast the day-ahead realised volatility.

DeepVol, like any deep-feed-forward neural network, approximates the volatility's unknown function through sample pairs of input and output data (x, y). Formally speaking, DeepVol is approximating some function $f_\theta(\cdot)$ which is not available in closed form by finding the optimal model's parameters $\hat{\theta}$ derived from the best function approximation $f_\theta^*(\cdot)$.

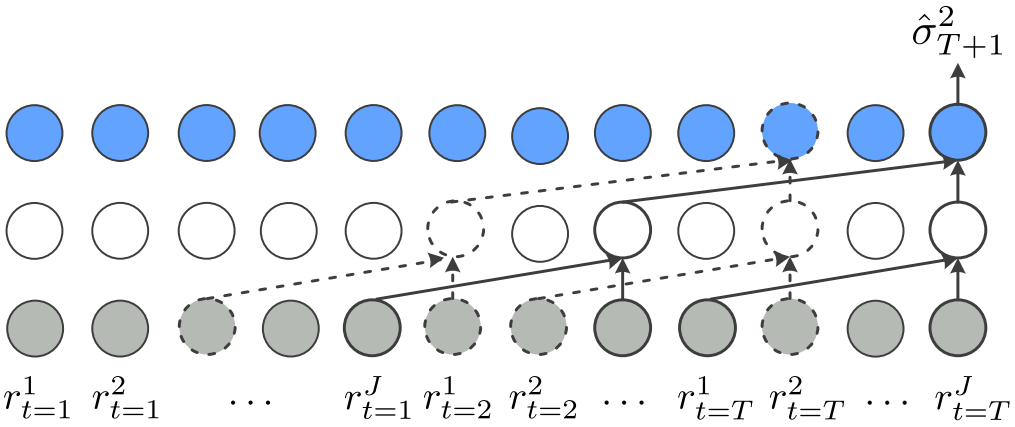

Figure 4.2: DeepVol intrinsic architecture. The dilation factor grows exponentially, allowing an increase in the receptive field without increasing the model's complexity.

4.5 Experiments

In this section, DeepVol's volatility forecasts will be evaluated and compared with the baseline models described in Section 4.3.1. For benchmarking purposes, we will utilise the metrics described in Section 4.3.2. Besides classic out-of-sample forecast comparisons, we perform different experiments to present some additional insights into the inner workings of DeepVol. We analyse the model's behaviour while varying the intraday data sampling frequency, studying the discrepancies in model behaviour when trained on different granularity regimes. In close relation to this, we also explore the use of different receptive field sizes and how this affects the model performance. Finally, we analyse the inclusion of realised measures as an extra input to the model, studying if its addition can improve the forecasting accuracy. Besides the models presented in Section 4.3.1, we also include a martingale process for comparison purposes.

4.5.1 Experiments Setup

We apply the same architecture to all the experiments in this section, using the Quasi Log-Likelihood as a loss function to train the model parameters. We choose Adaptive Moment Estimation Algorithm (ADAM) (Kingma and Ba, 2014) as optimiser, even though different experiments were conducted exploring the use

Table 4.1: Out-of-sample forecast: experiments results for the NASDAQ-100 dataset.

Method	MAE	RMSE	SMAPE	QLIKE	ME	MedAE
martingale	5.180	11.410	0.324	747.480	96.654	**1.614**
TARCH	4.849	10.320	0.301	351.310	71.659	2.804
IGARCH	5.008	10.534	0.302	351.702	72.048	2.797
FIGARCH	4.631	10.356	0.294	349.245	71.050	2.460
APARCH	4.730	10.096	0.299	349.974	**70.088**	2.859
AGARCH	4.833	10.304	0.324	351.217	71.215	2.819
EGARCH	4.793	10.180	0.300	348.614	70.615	2.917
HEAVY	4.565	10.239	0.292	343.490	72.404	2.545
DeepVol	**3.903**	**8.457**	**0.279**	**340.779**	71.779	2.008

of Averaged Stochastic Gradient Descent (ASGD) (Kingma and Ba, 2014) and Limited Memory BFGS (L-BFGS) (D. Liu and Nocedal, 1989). While the use of these optimisers usually entails smoother predictions, the reported performance declined with respect to ADAM. Hence, they were not further considered. Early stopping is used during the training process. DeepVol is implemented in Pytorch-Lightning (Falcon and The PyTorch Lightning team, 2019), and the experiments are conducted using an NVIDIA Titan Xp GPU.

4.5.2 Out-of-sample Forecast

This section aims to provide an out-of-sample performance comparison between the proposed model and some classical methodologies widely used in the finance industry. For this purpose, we use the NASDAQ-100 dataset described in Section 4.2.1, which is split into two folds. The first of them contains the intraday data from September 30, 2019, to December 31, 2020. The second fold includes data from January 1, 2021, to September 30, 2021. The 15 months of data of the first fold are used for training purposes, while the remaining nine months are used to evaluate the out-of-sample performance of the models. For this specific study, a sampling frequency of 5 minutes for the intraday data and a receptive field of one day (using t-day intraday's data to predict the realised volatility for day $t+1$) are utilised.

Table 4.1 summarises the out-of-sample forecast performance of the different models we evaluate. It can be seen that the proposed architecture, DeepVol,

improves the baseline results for most metrics, with the exception of ME and MedAE. Peculiarly, for the latter, the martingale process proves to provide the best results. Considering that the MedAE is an outlier-robust error function, this behaviour is not surprising, as the martingale process is the most conservative among the evaluated strategies. For this particular error metric, DeepVol is the second-best in performance terms.

As mentioned before, the evaluated baseline methodologies operate in a recurrent manner, utilising all available past data, while DeepVol uses just the previous day's intraday information for the day-ahead prediction. Considering these facts, DeepVol's good performance with respect to the MedAE is especially surprising, as noisier behaviour could be expected due to the lack of recurrence. Furthermore, some of the baseline models evaluated, such as the HEAVY model, integrate momentum indicators into their architecture, something that we do not explicitly model in DeepVol. Consequently, DeepVol's accurate predictions in terms of MAE and RMSE are particularly interesting, considering how the model maintains a low MedAE. In conclusion, the proposed architecture shows robustness in the presence of volatility shocks and avoids an escalation on the ME and MedAE as unstable methods would report.

Table 4.2 extends the results of Table 4.1, displaying the improvement/degradation for each evaluated method relative to a basic martingale process and the HEAVY model. For the former, we aim to report how much improvement each model provides over the most basic modelling of the problem. For the latter, considering that the HEAVY model is the best performer among the baseline architectures, a direct comparison with it is especially useful for analysing DeepVol's performance.

DeepVol thoroughly overperforms HEAVY concerning the MAE, RMSE, SMAPE, and MedAE errors, while the differences in QLIKE and ME are tighter. As previously mentioned, we consider particularly interesting DeepVol's ability to overperform the rest of the models while proving a robust noise behaviour, avoiding an escalation in the ME and MedAE while performing more accurate predictions.

Table 4.2: Out-of-sample forecast: percentage of improvement/degradation over the martingale process and the HEAVY model, for each of the evaluated models.

Method	MAE	RMSE	SMAPE	QLIKE	ME	MedAE
	Improvement over martingale (%)					
martingale	-	-	-	-	-	-
TARCH	6.398	9.553	7.099	53.001	25.860	-73.730
IGARCH	3.320	7.677	6.790	52.948	25.458	-73.296
FIGARCH	10.598	9.238	9.259	53.277	26.490	-52.410
APARCH	8.687	11.516	7.716	53.179	**27.486**	-77.138
AGARCH	6.699	9.693	0.000	53.013	26.319	-74.659
EGARCH	7.471	10.780	7.407	53.361	26.940	-80.731
HEAVY	11.873	10.263	9.877	54.047	25.089	-57.677
DeepVol	**24.653**	**25.881**	**13.889**	**54.410**	25.736	**-24.411**
	Improvement over HEAVY (%)					
martingale	-13.472	-11.437	-10.959	-117.613	-33.492	**36.579**
TARCH	-6.212	-0.791	-3.082	-2.277	1.029	-10.181
IGARCH	-9.704	-2.881	-3.425	-2.391	0.492	-9.906
FIGARCH	-1.446	-1.143	-0.685	-1.675	1.870	3.340
APARCH	-3.614	1.397	-2.397	-1.888	**3.120**	-12.342
AGARCH	-5.871	-0.635	-10.959	-2.250	1.642	-10.771
EGARCH	-4.995	0.576	-2.740	-1.492	2.471	-14.621
HEAVY	-	-	-	-	-	-
DeepVol	**14.502**	**17.404**	**4.452**	**0.789**	0.863	21.097

4.5.3 Receptive Field and Sampling Frequency Analysis

The receptive field size and the intraday sampling frequency are two model parameters that shed light on the inner workings of DeepVol when analysed further. Therefore, its analysis is particularly interesting in order to understand the model's behaviour. As mentioned in Section 4.2.2, a number of studies have validated the optimal intraday data sampling frequency for the computation of the realised measures from high-frequency data (T. Andersen et al., 2006; Peter R Hansen and Lunde, 2006; Corradi and Distaso, 2006), commonly concluding that using a granularity of 5 or 10 minutes minimises the microstructure noise effect while maximising the use of high-frequency information. In this section, we study this same trade-off in the proposed deep learning architecture, analysing the effect that using different sampling frequencies has on model performance.

Furthermore, increasing the receptive field size is a practical way of extending the network's capabilities without modifying its architecture or increasing its complexity.

Table 4.3: Receptive field and sampling frequency study.

Sampling Frequency	Receptive Field	MAE	RMSE	SMAPE	QLIKE	ME	MedAE
1 min	**1**	4.096	8.462	0.287	342.313	71.749	2.396
5 min	**1**	**3.903**	8.457	**0.279**	**340.779**	71.779	2.008
	2	4.429	9.495	0.308	367.209	**70.036**	1.756
	3	4.054	**8.379**	0.285	343.359	70.457	2.334
15 min	**1**	3.993	8.436	0.283	343.412	70.915	2.185
	2	4.651	10.437	0.312	365.893	70.926	1.836
	3	5.817	9.520	0.323	362.235	72.338	4.577
	5	6.736	10.217	0.336	366.192	72.240	5.594
30 min	**1**	4.259	9.699	0.318	689.633	75.793	1.632
	2	4.503	10.140	0.327	789.326	79.345	1.724
	3	4.473	9.931	0.324	784.843	75.059	1.705
	5	4.705	10.802	0.326	833.966	83.416	1.676
	10	4.732	11.084	0.327	853.981	85.656	**1.591**
60 min	**1**	4.988	10.516	0.297	366.509	82.207	2.402
	2	5.616	12.596	0.324	709.082	99.828	2.178
	3	5.441	12.615	0.319	688.996	99.612	2.017
	5	5.520	12.456	0.326	714.871	95.763	2.071
	10	4.997	11.186	0.322	706.917	87.096	1.927

For example, while DeepVol could be easily modified to integrate a momentum indicator, increasing its receptive field should entail a similar effect, providing DeepVol with the possibility of incorporating past data if it is informative enough for the realised volatility forecasting.

Table 4.3 collects the results of an analysis whose main objective is to evaluate if DeepVol's performance is robust to increasing the receptive field or modifying the sampling frequency. It is interesting to note that using intraday data from one day with a sampling frequency of 5-minutes proves to be optimal. This scenario reports the best results with regard to all the considered metrics but the MedAE. Secondly, the increment of the receptive field leads to a degradation of performance. This indicates that, for the proposed architecture, all the relevant information for forecasting the day-ahead volatility can be obtained from previous day's high-frequency data. Otherwise, the model yields more conservative predictions that degrade its performance. Thirdly, the best performance in terms of MedAE is

obtained when using a 30 minutes sampling frequency, together with a receptive field of ten days. This result can be directly related to the hypothesis previously mentioned: a longer receptive field leads to a more conservative forecast, resulting in a lower Median Absolute Error. In this scenario, the model is less prone to forecast volatility jumps, a behaviour commonly associated with integrating momentum indicators. However, this specific setup leads to a deterioration in all the other metrics. The growth in the receptive field size prevents the model from forecasting more drastic changes in the presence of volatility shocks, leading to more conservative predictions than the ones reported when using just the previous day's intraday data.

4.5.4 Linearity Study

The analyses conducted in the previous sections have shown that the use of a receptive field of one day and a sampling frequency of 5-minutes reports the most accurate results for forecasting the day-ahead realised volatility. In addition, we want to study the possible gains of including realised measures in our methodology. The intuition behind this idea is that integrating previous days' realised measures as an extra input would allow DeepVol to observe a bigger window of past data, allowing the model to complement high-frequency data with extra historical information about the time series.

To integrate the realised measures into DeepVol, we slightly modify its architecture, adding a linear output as a final layer. This last layer merged the results of the dilated convolutions performed over the high-frequency data with the realised measures, each weighted by its corresponding terms. Different receptive fields were validated while integrating the realised measures. The reported results are shown in Table 4.4, where it can be noted that our DNNs-based proposal does not benefit from the inclusion of realised measures as an extra input feature. Adding the past realised measures results in an analogous behaviour to increasing the receptive field, highlighting again that DeepVol is especially efficient in utilising recent high-frequency data for volatility forecasting, not requiring a more extended lookback window to do so.

Table 4.4: Linearity Study. DeepVol + RV merges DeepVol's predictions with the realised variance through a linear layer and additional non-linearities.

	Receptive Field	MAE	RMSE	SMAPE	QLIKE	ME	MedAE
DeepVol	**1**	4.130	8.720	0.286	345.150	72.292	2.193
+	**2**	6.804	10.036	0.335	369.529	69.309	5.728
RV	**3**	6.899	10.008	0.340	371.762	69.903	6.577
	1	**3.903**	8.457	**0.279**	**340.779**	71.779	2.008
DeepVol	**2**	4.429	9.495	0.308	592.600	78.690	**1.756**
	3	4.054	**8.379**	0.285	343.359	**70.457**	2.334

4.5.5 Generalisation and Transfer Learning Analysis

Previous experiments used all NASDAQ-100 tickers during training and testing, preserving a portion of the dataset's dates for the out-of-sample forecast. In this section, we split the dataset into two folds. During training, just half of the tickers are used, while the other half is utilised for testing. For training purposes, we use the first four months of data corresponding to the first half of tickers, that is, from September 30, 2019, through January 30, 2020. The model is later tested on the remainder tickers, using data from February 01, 2020, through September 30, 2021. This setup allows us to evaluate the quality of the models' forecasts during the volatility shocks provoked by the COVID-19 crisis, which started in February 2020. Further, it provides an evaluation of our model's generalisation capabilities, predicting the day-ahead volatility for tickers that were not previously available. As done in Section 4.5.5, a 5-minutes sampling frequency and a receptive field of one day are used. The results of this out-of-sample-stocks forecast study are collected in Table 4.5. In these conditions, DeepVol still reports the best MAE, RMSE, and SMAPE results, while the HEAVY model reports a better QLIKE than the rest of the evaluated methods. Concerning the MeadAE, DeepVol reports the best results, immediately followed by the martingale process, which still outperforms the rest of the baseline models. These results, which are similar to the obtained in the out-of-sample forecast study of Section 4.5.5, confirm that DeepVol still shows a conservative behaviour in this new forecast scenario, proving its generalisation capabilities to transfer learning

Table 4.5: Out-of-sample-stocks forecast. Generalisation Study: experiments results for the NASDAQ-100 dataset.

Method	MAE	RMSE	SMAPE	QLIKE	ME	MedAE
martingale	9.673	35.235	0.324	2142.795	341.457	2.169
TARCH	8.525	28.178	0.295	893.795	282.096	3.236
IGARCH	9.208	27.753	0.312	947.409	279.080	3.982
FIGARCH	7.805	26.752	0.299	899.955	267.533	3.581
APARCH	8.179	26.749	0.297	896.063	**265.910**	3.557
AGARCH	7.928	26.486	0.294	893.682	269.577	3.191
EGARCH	8.180	26.767	0.297	897.432	277.022	3.530
HEAVY	8.315	26.322	0.294	**874.409**	277.780	2.158
DeepVol	**7.288**	**23.396**	**0.292**	894.283	275.255	**1.927**

from training to test, learning global features of the data that allow the model to perform well on out-of-distribution data. As in Section 4.5.5, Table 4.6 reports the improvement/degradation for each evaluated method with respect to a basic martingale process and the HEAVY model on the test set tickers.

Table 4.6: Out-of-sample-stocks forecast: percentage of improvement/degradation over the martingale process and the HEAVY model. for each of the evaluated models.

Method	MAE	RMSE	SMAPE	QLIKE	ME	MedAE
	Improvement over martingale (%)					
martingale	-	-	-	-	-	-
TARCH	11.863	20.030	9.068	58.288	17.385	-49.161
IGARCH	4.804	21.235	3.671	55.786	18.268	-83.565
FIGARCH	19.315	24.075	7.711	58.001	21.650	-65.089
APARCH	15.440	24.083	8.267	58.183	**22.125**	-63.973
AGARCH	18.043	24.829	9.251	58.294	21.051	-47.105
EGARCH	15.443	24.032	8.452	58.119	18.871	-62.737
HEAVY	14.037	25.295	9.200	**59.193**	18.649	0.534
DeepVol	**24.660**	**33.599**	**9.932**	58.266	19.388	**11.165**
	Improvement over HEAVY (%)					
martingale	-16.330	-33.859	-10.132	-145.056	-22.924	-0.537
TARCH	-2.529	-7.048	-0.145	-2.217	-1.554	-49.962
IGARCH	-10.741	-5.434	-6.090	-8.349	-0.468	-84.551
FIGARCH	6.140	-1.632	-1.640	-2.921	3.689	-65.975
APARCH	1.632	-1.622	-1.028	-2.476	**4.273**	-64.854
AGARCH	4.660	-0.623	0.056	**-2.204**	2.953	-47.895
EGARCH	1.624	-1.691	-0.824	-2.633	0.273	-63.611
HEAVY	-	-	-	-	-	-
DeepVol	**12.357**	**11.116**	**0.806**	-2.273	0.909	**10.688**

Finally, Figures 4.3 to 4.6 show different examples of how the evaluated models

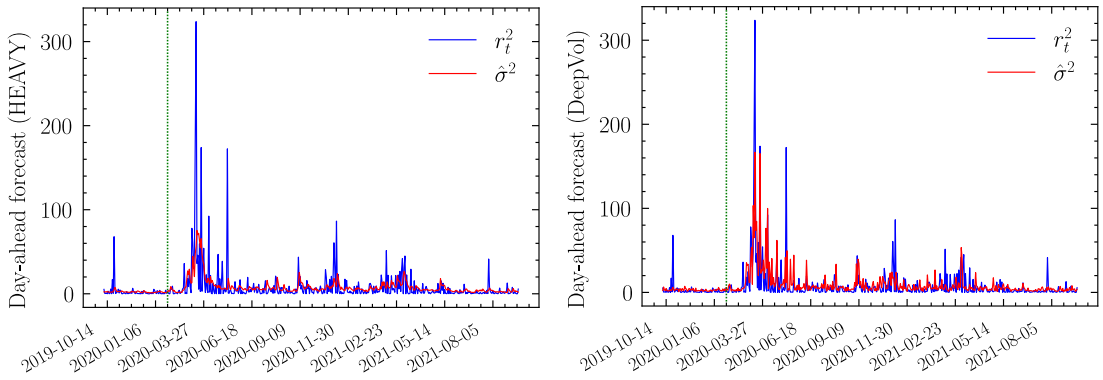

Figure 4.3: Out-of-sample-stocks: HEAVY's and DeepVol's forecast on PYPL. Green dotted vertical lines mark the forecast's start.

generalise and transfer learning from the tickers used during training to the test distribution. Model forecasts are shown together with the daily squared returns, allowing a direct comparison between forecasts from DeepVol and the baselines. Note that classical methodologies return smoother predictions, a phenomenon especially visible in the HEAVY model as it integrates a momentum indicator. This behaviour, associated with more conservative predictions, clearly poses a disadvantage in terms of slower adaptation to volatility shocks. Several of these volatility shocks, provoked by the COVID-19 crisis in 2020 and 2021, are easily recognisable in the associated figures. All the evaluated models reacted to the bigger of these shocks, the 2020 stock market crash, starting in February 2020, in one way or another. Otherwise, during the minor shocks that followed that year, baseline predictions are almost negligible with the exception of IGARCH in Figure 4.6. HEAVY and EGARCH exhibit an invariable behaviour in this turbulent environment, showing a lack of adaptability to changing conditions. We should remark that DeepVol requires just one day of intraday data to perform the out-of-sample volatility forecasting, unlike classical methodologies, which operate recursively, forcing them to use a sufficiently long window of past data. This places DeepVol in an advantaged position in situations of low data availability, such as the inclusion of new tickers in the stock market, as it does not require a long horizon of historical data to perform the forecasting.

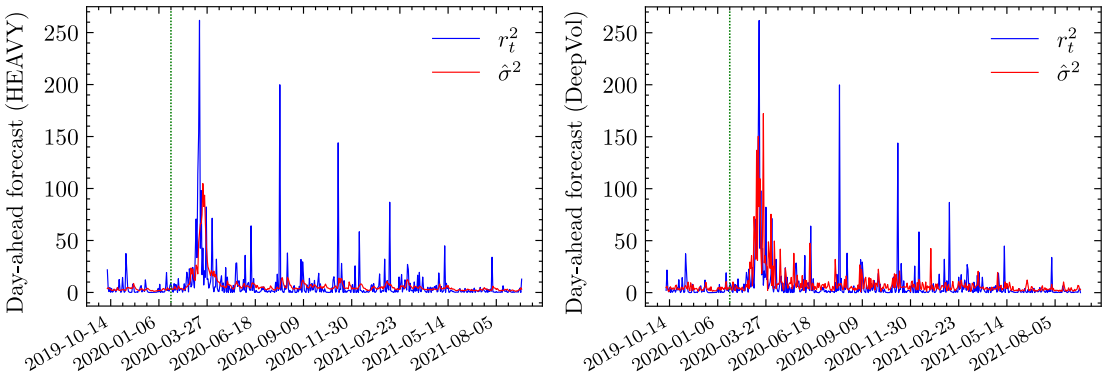

Figure 4.4: Out-of-sample-stocks: HEAVY's and DeepVol's forecast on QCOM. Green dotted vertical lines mark the forecast's start.

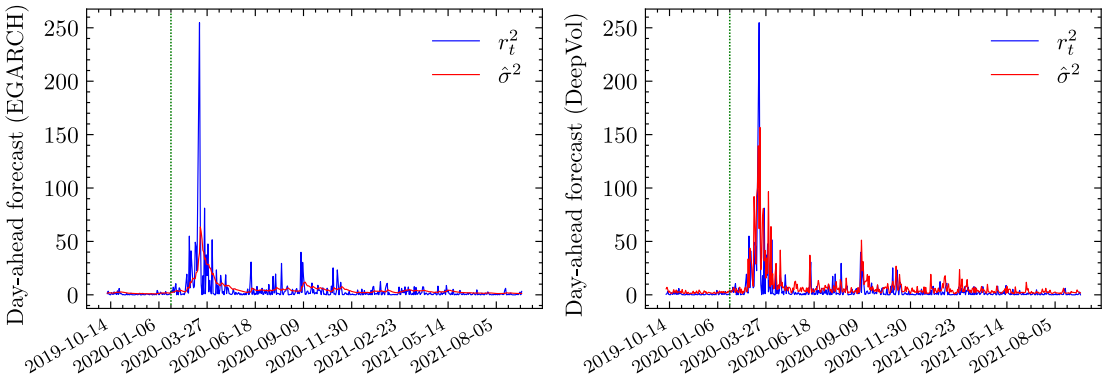

Figure 4.5: Out-of-sample-stocks: EGARCH's and DeepVol's forecast on MSFT. Green dotted vertical lines mark the forecast's start.

4.5.6 Discussion of Results

Several findings from the experiments are worth highlighting with regard to the use of Dilated Causal Convolutions for the day-ahead realised volatility forecasting. Firstly, DeepVol generally outperforms traditional autoregressive architectures, showing a quicker adaptation to volatility shocks while maintaining some conservatism in its predictions, as the reported MedAE and ME in previous experiments show. Specifically, DeepVol's reported accuracy seems particularly interesting, specially considering that the experiments were conducted in high-volatility regimes. The reported results resemble extensive literature indicating that deep learning-based volatility forecasting architectures (Ramos-Pérez et al., 2021) and hybrid models (Baek and Kim, 2018; Kim and Won, 2018) consistently outperform classical

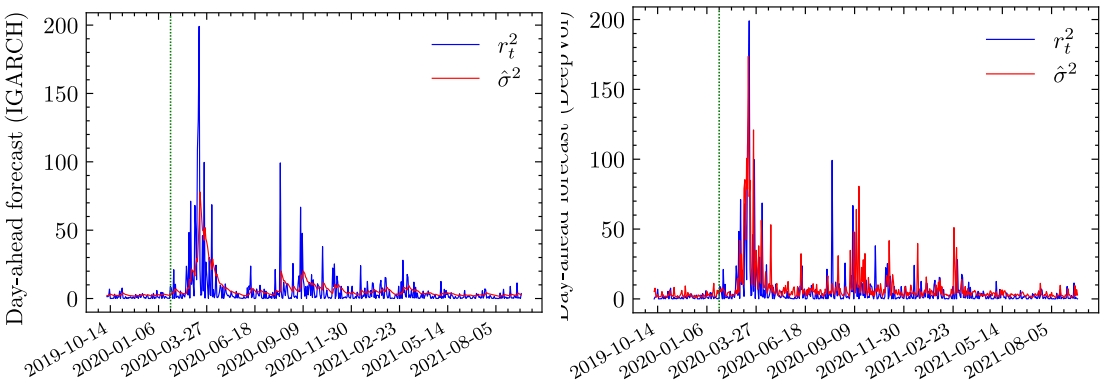

Figure 4.6: Out-of-sample-stocks: IGARCH's and DeepVol's forecast on AAPL. Green dotted vertical lines mark the forecast's start.

methodologies.

Experiments in Section 4.5.3 have highlighted that, for the proposed method, data from the previous trading day contains enough information for predicting the day-ahead realised volatility with high accuracy. Furthermore, a sampling frequency of 5-minutes has been shown to maximise the trade-off between noise and intraday information, echoing studies analysing this same trade-off in the context of estimating realised measures from high-frequency data. Finally, DeepVol consistently outperforms baseline methods while reporting good results in outlier-robust metrics such as MedAE, proving that the model quickly adapts to volatility shocks while exhibiting noise robustness.

4.6 Discussion

In this chapter, we have proposed a deep learning model based on hierarchies of Dilated Causal Convolutions – termed DeepVol – to forecast day-ahead realised volatility from high-frequency data. Our model takes advantage of the automatic feature extraction inherent to Deep Neural Networks to bypass the estimation of the realised measures, tackling the problem of volatility forecasting from a pure data-driven perspective. At the same time, the use of dilated convolutions enables DeepVol to exponentially increase its input window, performing a similar operation to how handcrafted realised measures condense high-frequency information. Reported

results show how DeepVol's predictions significantly improve the baseline models performance, proving that the proposed data-driven approach avoids the limitations of classical methods, such as model misspecification or the use of hand-crafted noisy realised measures, by taking advantage of the abundance of high-frequency data.

The proposed architecture outperforms baseline methods while exhibiting robustness in the presence of volatility shocks, avoiding an increase in Maximum and Median Absolute Errors, as reported by other unstable methods. Those results are especially relevant considering that experiments were conducted in high volatility regimes, such as the 2020 stock crisis caused by the COVID-19 pandemic. In the context of the generalisation study, where out-of-sample-stocks forecasts are conducted, DeepVol shows its ability to extract universal features and transfer learning to out-of-distribution data. Additionally, we observe that for DeepVol, the previous day's intraday data makes the most significant contribution to predicting the day-ahead volatility. Therefore, increasing the receptive field of DeepVol does not generally lead to better performance. Moreover, we show that using a 5-minutes sampling frequency optimises the trade-off between maximising the use of high-frequency data information while minimising the microstructure noise implicit to higher sampling frequencies. This result is particularly interesting as it is reminiscent of earlier studies validating this same trade-off for the construction of realised measures. The empirical results collected in this chapter suggest that models based on Dilated Causal Convolutions should be carefully considered in the context of volatility forecasting and, as a result, can play a key role in the valuation of financial derivatives, risk management, and portfolio construction.

5

Deep Survival Analysis in the LOB: A Convolutional-Transformer Approach to Estimating Fill Probabilities

Contents

This chapter shifts the focus towards another relevant problem in the quantitative finance literature, specifically, the estimation of fill times for limit orders posted in

the Limit Order Book (LOB). Choosing between a limit order and a market order to execute a trade is an important problem in optimal execution strategies. Key to this choice is the fill probability of a limit order placed at different levels of the book. In this chapter, we propose a deep learning-based method to estimate the fill times of limit orders posted in the LOB. In particular, we present a novel survival analysis model, based on a convolutional-Transformer encoder and a monotonic decoder, which relates the time-varying features of the LOB to the distribution of fill times. We provide an exhaustive comparison between survival functions resulting from different order placement strategies and offer insight into the fill probability of orders placed inside the spread. Finally, we compare our method with competing benchmarks from the survival analysis literature in terms of proper scoring rules and perform an interpretability analysis to shed light on the informativeness of features when estimating fill probabilities.

The remainder of this chapter is organised as follows. Section 5.1 provides a brief introduction to the problem of limit orders' fill times estimation, as well as an overview of the main contributions of our work. Section 5.2 presents a succinct summary of the survival analysis literature, while Section 5.3 provides an overview of LOBs. In Section 5.4, we formalise the survival analysis framework used in the rest of the chapter, as well as the concept of proper scoring rules. Further, Section 5.5 performs a statistical analysis on the fill probabilities observed in LOBs for assets with different characteristics. Section 5.6 presents the proposed model, and Section 5.7 provides an empirical evaluation of its performance and an interpretability analysis. Finally, Section 5.8 summarises the findings and concludes this chapter.

5.1 Introduction

Electronic financial exchanges typically use LOBs to organise and clear the demand and supply of liquidity in various asset classes. LOBs offer several types of orders, where limit orders (LOs) and market orders (MOs) are the most common types. There is a trade-off between the selection of each order type: market orders cross the bid-ask spread and obtain immediate execution, while limit orders can be placed

at various levels of the order book, which, if executed, obtain a better price than that of a market order. However, there is no guarantee that a limit order will be executed, as the limit order rests in the LOB until it is filled by a market order that crosses the spread, or it is cancelled. The amount of time a limit order takes to get filled is known as the time-to-fill and it can be estimated using different methods.

In this chapter, we use survival analysis to estimate the fill probabilities of limit orders. We present a novel deep learning-based method to estimate the survival functions of orders placed at different levels of the order book, where the matching of orders is determined by price-time priority. To this end, we introduce an encoder-decoder neural network architecture based on the Transformer model (Vaswani et al., 2017) and partially monotonic neural networks (Chilinski and Silva, 2020). The proposed model allows us to estimate the fill probability of different assets with minimal assumptions, conditioning on the most recent events in the LOB. Our methodology is general and suitable for other application areas that perform survival analysis from longitudinal data, representing a novel improvement to the existing literature in this area.

Along with this novel predictive methodology, we provide additional financial insights into the value of orders positioned at different levels of the book and inside the spread. To evaluate the performance of the estimated fill probabilities, we use proper scoring rules (Gneiting and Ranjan, 2011; Avati et al., 2020; Rindt et al., 2022), which ensures that we maximise the fit with respect to the true survival function instead of an incorrect proxy, hence producing more robust results. In this regard, we provide a novel contribution to the literature by generalising the proof of Rindt et al. (2022) to the multi-event case.

Finally, we show through empirical evaluation that the proposed method is able to summarise LOB information prior to the order submission. This entails significant performance improvements compared to baseline off-the-shelf architectures and standard benchmarks from the survival analysis literature. Further, the Shapley value analysis conducted in Section 5.7 shows that the model heavily relies on

high-frequency information to perform the prediction, paying less attention to slow-moving features such as time of day.

5.2 Literature Review

Time-to-event analysis, also known as survival analysis, is widely used in different fields, including the estimation of patients' time-to-recovery (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 to the features (covariates) describing the system. Simple parametric models, such as the Cox Proportional Hazards Model (Cox, 1972) or the Accelerated Failure Time (AFT) model (Wei, 1992) are commonly used to make these connections. Nevertheless, the use of deep learning-based models for survival analysis has become increasingly prevalent in recent times, particularly in the medical field. Early examples of this are Faraggi and Simon (1995), which uses a feed-forward neural network to extend the Cox Proportional Hazards Model, and Katzman et al. (2018) and Kvamme et al. (2019), which incorporate deep learning techniques, such as dropout, into the survival analysis literature. Subsequent work has aimed to improve upon these models (Z. Wang and Sun, 2022; Zhong et al., 2021; S. Hu et al., 2021), with Rindt et al. (2022) highlighting the importance of using proper scoring rules in survival analysis and illustrating the shortcomings of previous models in this regard. Other notable approaches include the use of Gaussian processes (Fernández et al., 2016; Alaa and van der Schaar, 2017), random forests (Ishwaran et al., 2008), and adversarial approaches (Chapfuwa et al., 2018).

In the quantitative finance field, Cho and Nelling (2000) employs a survival analysis approach to calculate limit orders' fill probabilities, assuming that the shape of the survival function follows a Weibull distribution. To determine the fill probability, the authors track limit orders throughout the trading day, considering cancellations as right-censoring. If a limit order is matched against multiple market orders, the total time-to-fill is the weighted average of the individual time-to-fills,

with the size of each execution serving as its weight. Further, Lo et al. (2002) utilises the generalised gamma distribution to model the baseline distribution and incorporate market features through the use of the AFT model. The authors establish separate models for time-to-completion, time-to-first fill, and time-to-cancellation, also introducing an in-depth discussion on hypothetical limit orders. Finally, Cartea, Jaimungal, et al. (2015) derives optimal trading strategies under the assumption of a fill probability that is static in time and decreases exponentially between levels of the order book.

More recently, several studies have employed deep learning for modelling market microstructure (Z. Zhang, Zohren, and Roberts, 2019; Z. Zhang and Zohren, 2021; Kolm et al., 2021; Z. Zhang, Lim, et al., 2021), mainly focusing on short-term price prediction. Further, Maglaras et al. (2021) uses deep learning to predict limit orders' fill probability, employing Recurrent Neural Networks (RNNs) (Funahashi and Nakamura, 1993) to do so. In contrast, our work improves this setup by introducing a novel Transformer-based architecture, proposing and analysing new ways to generate fill time data. We compare our method to multiple benchmark models from the survival analysis literature, evaluating the reported performances using proper scoring rules and showing how the proposed model outperforms the state-of-the-art.

5.3 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 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. Other types of orders also exist, such as hidden orders, which do not reveal their quantity, fill-or-kill orders, which are to be executed immediately (entirely or partially) at a predetermined price range, and immediate-or-cancel orders, which are to be executed immediately in their entirety (within a predetermined price range) or the entire order is cancelled. The list of pending limit orders is stored in the LOB until they are matched or cancelled.

Table 5.1: Example message data from the LOB. First five trades on 01/10/2012 for AAPL ticker.

Time	Event Type	Order ID	Size	Price	Direction
34200.395473	1	4141926	45	6713500	1
34200.395473	1	1972838	30	6715100	-1
34200.395473	1	4485151	1	6713400	1
34200.395473	1	6037010	10	6715500	-1
34200.395473	1	4719061	20	6712500	1

The matching engine of most exchanges follows a price-time priority rule where orders are first ranked according to their price and then according to the time they arrived at 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 orders, and the bid side containing all the buy orders. A snapshot of the LOB at time t is described by the vector:

$$s_t = \left\{ p_a^l(t), v_a^l(t), p_b^l(t), v_b^l(t) \right\}_{l=1}^{L}, \tag{5.1}$$

where $p_a^l(t), v_a^l(t), p_b^l(t), v_b^l(t)$ denote the ask price, ask volume, bid price, and bid volume for price level $l \in \{1, ..., L\}$ at time t, which is typically measured in microseconds after midnight. Multiple snapshots assemble a matrix $\mathbf{x}_t \in \mathbb{R}^{T \times 4L}$, which contains the discrete-time dynamics of the LOB from time t to $t - T$. Concerning the empirical evaluation of our model, we consider Lobster message data (R. Huang and Polak, 2011) containing information on events that update the state of the order book in the NASDAQ stock exchange. These include messages to post or cancel orders, the direction (buy or sell) and volume of the orders, etc. Table 5.1 shows the first five messages observed for AAPL's LOB on a trading day randomly chosen.

5.4 Survival Analysis

We consider an event time, $T_l \in \mathbb{R}_{\geq 0}$, which represents a positive valued random variable describing the time-to-fill of a limit order placed at level l of the order book. We are interested in predicting the set of event times by conditioning on a

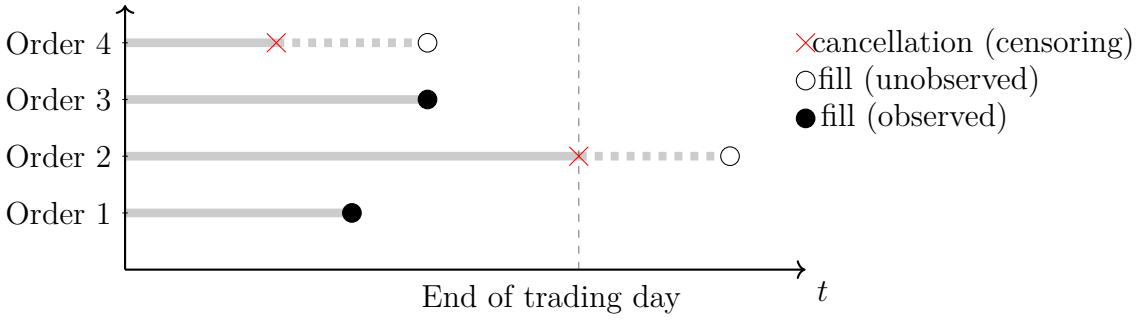

Figure 5.1: Events which can occur after the submission of a limit order.

set of market features $\mathbf{x} \in \mathbb{R}^p$. All events are subject to right-censoring, meaning the final event is not observed. When a limit order is cancelled or reaches the end of the trading day without being filled, it is considered a censored event [1], as Figure 5.1 shows. We consider a set of N observations of the triplet $(\mathbf{x}_i, z_i, d_i)$, where $d_i = \mathbb{I}\{z_i = t_i\}$ is an indicator function equal to zero if the event is censored, z_i is the observed event time, and $\mathbf{x}_i$ are the observed market features up to the instant of the order submission. We use this data to estimate the survival function, $S_{T_l}(t|\mathbf{x}) = \mathbb{P}\{T_l > t|\mathbf{x}\}$[2].

The survival function returns the probability that a limit order posted at level l will not be filled before time t. The link between the survival function, $S_{T_l}(t|\mathbf{x})$, and the Cumulative Density Function (CDF), $F_{T_l}(t|\mathbf{x})$, is given by $S_{T_l}(t|\mathbf{x}) = 1 - F_{T_l}(t|\mathbf{x})$. Therefore, the density function can be defined as follows: $f_{T_l}(t|\mathbf{x}) = -\frac{d}{dt} S_{T_l}(t|\mathbf{x})$. Further, the hazard rate, which indicates the propensity that an order will be filled after time t, is given by:

$$h_{T_l}(t|\mathbf{x}) = \frac{f_{T_l}(t|\mathbf{x})}{1 - F_{T_l}(t|\mathbf{x})}. \tag{5.2}$$

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

$$S_{T_l}(t|\mathbf{x}) = \mathbb{P}\{T_l > t|\mathbf{x}\} = 1 - F_{T_l}(t|\mathbf{x}) = \exp\left(- \int_0^t h_{T_l}(s) ds \right). \tag{5.3}$$

[1] Orders that reach the end of the trading day without being filled are treated as censored in this work, even though all orders are removed from the book at the end of the trading day. However, this is consistent with the literature, where this event is treated as "end-of-study" censoring.

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

Survival functions are parameterized by a vector of parameters $\boldsymbol{\theta} \in \mathbb{R}$ that describes its shape. One may assume that the survival function is determined 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 the goodness of the model's fit. Using neural networks to estimate the survival function is a possible alternative; it increases the number of parameters substantially, but also improves the fit to the data. Here, we use this approach to model the survival function of limit orders, as its shape is expected to have a non-linear relationship with the market features. To find the parameters that best explain the observed data, we perform maximum likelihood estimation. Specifically, we train our deep learning model to maximize the Right Censored Log-Likelihood (RCLL) function, defined as:

$$\mathcal{L}(\boldsymbol{\theta}) = \log(L_N(\boldsymbol{\theta})) = \sum_{k=1}^{N} d_k \log(\hat{f}(z_k|\mathbf{x}_k, \boldsymbol{\theta})) + (1 - d_k) \log(\hat{S}(z_k|\mathbf{x}_k, \boldsymbol{\theta})), \quad (5.4)$$

where $\hat{S}$ and $\hat{f}$ are the neural network estimates for the survival and density functions, respectively. See Appendix B for a derivation of Eq. (5.4).

Training on the RCLL requires the model to output $\hat{S}(z_k|\mathbf{x}_k, \boldsymbol{\theta})$ at the exact time-instant z_k. This is challenging for deep learning models that use the softmax activation function to discretise the survival function, as they require an interpolation scheme to tractably train using Eq. (5.4). Our model avoids this problem by introducing the observed time z_k mid-way through the network, together with the generated latent representation from the LOB time series, as Section 5.6 shows.

To evaluate the quality of our model fit, we introduce 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

$$\mathbb{E}_{t,c,\mathbf{x}}[\mathcal{S}(S(t|\mathbf{x}), (z, d))] \geq \mathbb{E}_{t,c,\mathbf{x}}[\mathcal{S}(\hat{S}(t|\mathbf{x}), (z, d))] \quad (5.5)$$

for all survival function estimates $\hat{S}(t|x)$ (Rindt et al., 2022). The most commonly used scoring rules in the literature are improper, see Appendix D. On the other hand,

the RCLL is shown to be a proper scoring rule in Rindt et al. (2022). Therefore, throughout our analysis, we employ the RCLL as a proper scoring rule to evaluate the precision with which the evaluated models fit the true survival function.

5.5 Empirical and Statistical Evidence of Fill Rate Executions

To generate the data for the network to train on, we must infer whether an order was filled or not purely from observational LOB data. In this scenario, providing a good estimate of the fill probability does not only involve the design of a good survival analysis model, but it also requires a realistic data collection setup. In this section, we provide an empirical analysis of the effect of different data generation strategies on the resulting survival function. Additionally, we provide a statistical analysis of the survival functions and fill probabilities of orders placed at different levels of the LOB, as well as those placed within the spread.

5.5.1 Generation of Synthetically Tracked Orders

We explore two different ways to generate our dataset of triples. The first method consists of directly analysing messages data from the LOB. Specifically, we track all the messages associated to a particular order after its submission. If the final message associated to that order is a cancellation, we treat the order as censored, and filled otherwise. The time-to-fill is calculated as the time between order submission and the time the final message is observed.

This first approach does not allow for gauging the difference in the fill probability when performing a change to the original characteristics of the limit order, such as cancellation and repositioning when there is an adverse price move, commonly employed strategies. For these reasons, we consider "hypothetical" limit orders, which are orders of one share in volume, placed at the top of the queue of a price level l in the LOB. Such an approach relies on "fill conditions" to determine if an order has been filled. Furthermore, hypothetical limit orders are assumed not to cause any market impact. These assumptions are consistent with previous works

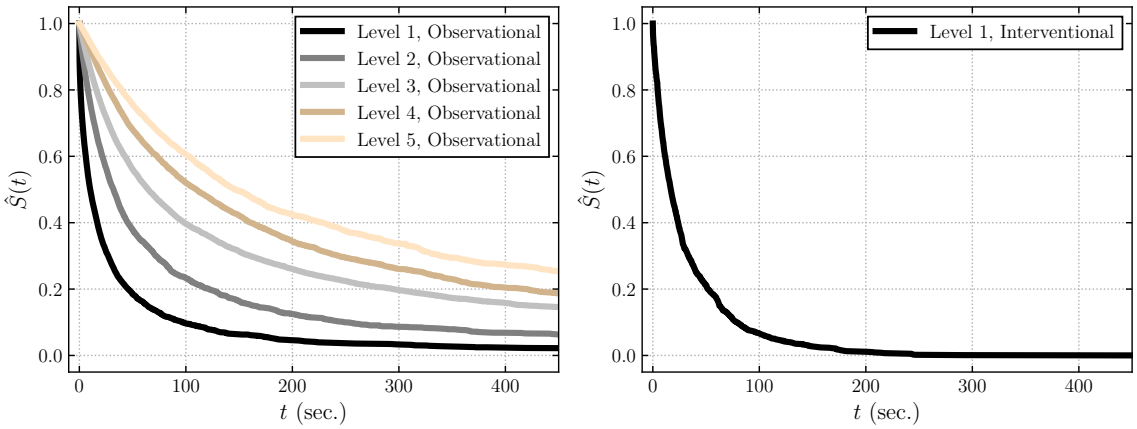

Figure 5.2: Left: Kaplan-Meier estimates of orders placed at different levels of the AAPL LOB. **Right:** Repositioning on the first level of AAPL LOB.

(Handa et al., 2003; Maglaras et al., 2021). All these considerations are explained in more detail in Appendices A and D.

We generate this hypothetical limit order data by randomly selecting times throughout the trading day, and placing the hypothetical one-share limit order at the best ask or best bid. The limit order is automatically re-positioned to the new best price whenever there is a price change at that level. This order repositioning results in a significant amount of censoring, as a small proportion of orders go beyond the first level. For this reason, we focus on estimating the fill probability of orders repositioned on the best bid and best ask (level one) of the LOB. Figure 5.2 shows the Kaplan-Meier estimates of the survival functions associated to LOB data tracking.

5.5.2 Fill Statistics of LOs placed inside the Spread

Here, we compute the fill probabilities of orders placed within the spread of the order book. To do so, we isolate orders posted n ticks into the spread, measured from the best bid or best ask. We consider a small set of assets comprised of small and large tick stocks[3], which also vary in the average arrival rate of LOs. The assets considered, along with their average spread, average volume on the best bid and best ask, and average events per minute, are shown in Table 5.2.

[3]We consider assets with a tick size of 0.01 US dollars, but differentiate between large tick and small tick assets. Large tick stocks are such that the bid-ask spread is approximately one tick in size, while for small tick stocks, the spread is several times larger than that. There is a significant difference in the dynamics between the two types, see Eisler et al. (2012) for more details.

Table 5.2: Statistics of small and large tick stocks on 1st of October 2022.

	Av. Spread (ticks)	Av. Vol. Best Ask	Av. Vol. Best Bid	Av. event/min
AAPL	1.30	494.65	488.86	412.23
AMZN	1.76	289.45	278.53	218.67
BIDU	9.37	88.54	97.24	20.86
COST	26.22	61.36	46.46	44.27
DELL	1.58	287.12	283.92	20.93
MSFT	2.74	146.15	159.14	204.86
GOOG	1.64	295.76	232.08	120.84
CSCO	1.10	2401.47	2314.83	68.58
INTC	1.12	5676.63	5533.79	110.88

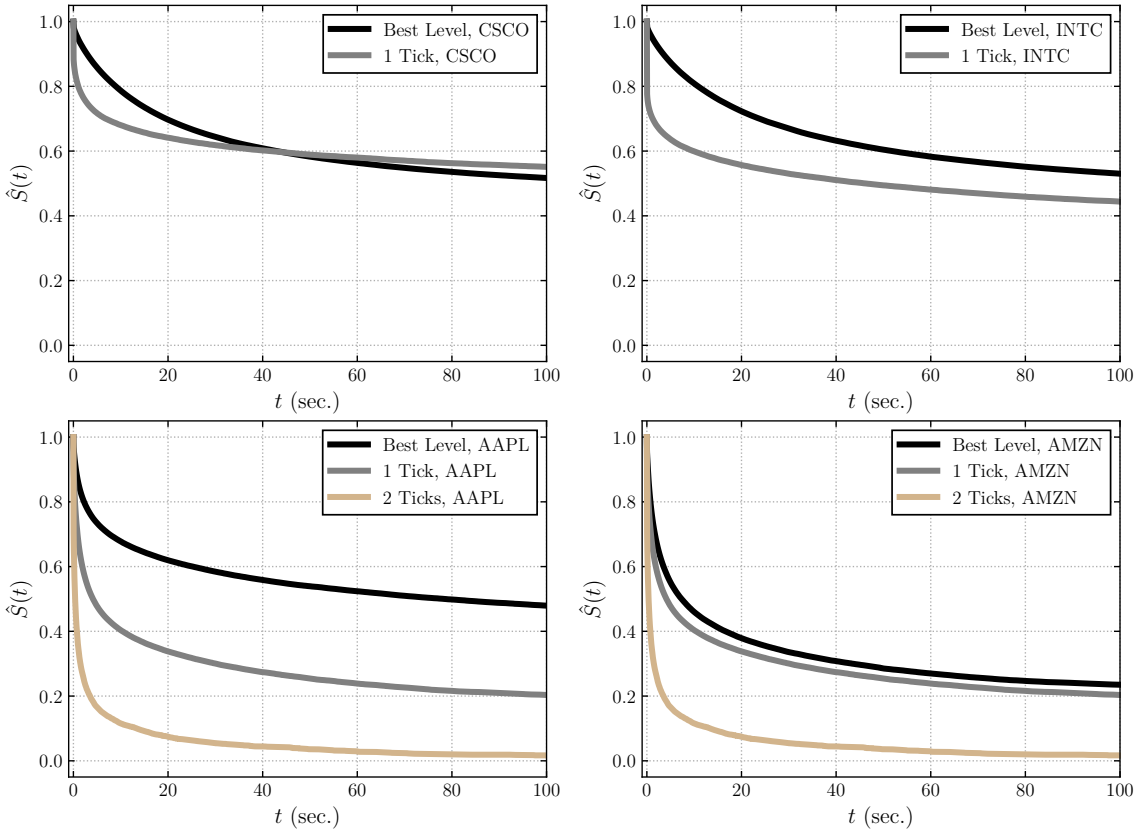

Figure 5.3: Survival functions when placing orders at different depths of the bid-ask spread. **Top left:** CSCO, **Top left:** INTC, **Bottom left:** AAPL, **Bottom left:** AMZN.

Table 5.3: Fill statistics for all assets considered, between 1 October 2022 and 27 December 2022.

		AAPL	AMZN	BIDU	COST	CSCO	DELL	GOOG	INTC	MSFT
Best Level	**Number of LOs**	17491885	18891270	2406523	5193897	14242140	4461087	2261258	17894682	18921692
	$\mathbb{P}\{T < 60, d = 1\}$	5.39%	9.23%	9.48%	4.58%	5.73%	3.92%	6.18%	5.70%	6.79%
	$\mathbb{E}\{T \mid T < 60, d = 1\}$	1.32 *s.*	1.07 *s.*	4.40 *s.*	5.18 *s.*	6.26 *s.*	6.87 *s.*	1.82 *s.*	7.33 *s.*	0.99 *s.*
1 Tick	**Number of LOs**	51096	6215869	1250831	2594091	466645	753603	1548080	317834	9654578
	$\mathbb{P}\{T < 60, d = 1\}$	13.17%	13.12%	5.28%	1.22%	17.53%	6.2%	10.88%	29.62%	7.34%
	$\mathbb{E}\{T \mid T < 60, d = 1\}$	0.64 *s.*	0.70 *s.*	3.82 *s.*	4.85 *s.*	1.71 *s.*	4.46 *s.*	0.85 *s.*	1.95 *s.*	0.70 *s.*
2 Ticks	**Number of LOs**	17862	169024	163567	113709	2817	15584	52565	992	492218
	$\mathbb{P}\{T < 60, d = 1\}$	36.01%	33.60%	15.33%	8.26%	35.49%	22.63%	31.99%	32.15%	26.32%
	$\mathbb{E}\{T \mid T < 60, d = 1\}$	0.19 *s.*	0.38 *s.*	3.93 *s.*	6.03 *s.*	0.47 *s.*	3.39 *s.*	0.84 *s.*	0.27 *s.*	0.45 *s.*
3 Ticks	**Number of LOs**	11692	24456	100719	87820	841	2548	8344	276	107990
	$\mathbb{P}\{T < 60, d = 1\}$	41.65%	41.39%	16.92%	8.98%	35.79%	27.62%	39.69%	24.63%	36.90%
	$\mathbb{E}\{T \mid T < 60, d = 1\}$	0.11 *s.*	0.25 *s.*	3.59 *s.*	5.33 *s.*	0.51 *s.*	2.07 *s.*	0.13 *s.*	0.29 *s.*	0.24 *s.*
4 Ticks	**Number of LOs**	4059	6469	63407	69689	318	937	2274	97	33885
	$\mathbb{P}\{T < 60, d = 1\}$	43.82%	43.43%	18.68%	10.61%	24.84%	29.02%	44.50%	24.74%	42.92%
	$\mathbb{E}\{T \mid T < 60, d = 1\}$	0.14 *s.*	0.11 *s.*	3.26 *s.*	4.69 *s.*	0.25 *s.*	2.34 *s.*	0.15 *s.*	0.8 *s.*	0.17 *s.*
5 Ticks	**Number of LOs**	2004	2470	41017	61457	136	454	782	103	13710
	$\mathbb{P}\{T < 60, d = 1\}$	44.31%	44.85%	21.40%	11.35%	45.58%	29.95%	50.12%	11.65%	44.66%
	$\mathbb{E}\{T \mid T < 60, d = 1\}$	0.08 *s.*	0.21 *s.*	3.07 *s.*	4.53 *s.*	0.36 *s.*	1.5 *s.*	0.07 *s.*	0.71 *s.*	0.12 *s.*

Intuitively, agents trading large tick stocks are incentivised to place their orders inside the bid-ask spread not to have 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 makes it approximately 3-6 times more likely for an order to get filled, while reducing the time to fill by a

similar proportion, as shown in Figure 5.3 and Table 5.3. Small tick stocks have a larger trading activity occuring inside the spread, even beyond one level in depth. Obtaining similar improvements in the fill probabilities for these assets requires placing LOs deeper in the book. It should be noticed that stocks with reduced trading activity, such as BIDU, COST or DELL, have a significantly lower fill probability at the best level and all depths inside the spread. Finally, notice that the markedly different dynamics of all assets considered further motivates using a model-free approach for modelling the survival functions in the LOB.

5.6 Monotonic Encoder-Decoder Convolutional-Transformer

5.6.1 General Architecture

In this work, we present an encoder-decoder architecture to learn the distribution of limit order survival times directly from the LOB. In contrast to previous approaches in the literature, our model does not rely on strong parametric assumptions to understand the relationship between the fill rates distribution and the predictors used. Figure 5.4 illustrates the two components of our framework. The encoder, parameterized by ϕ, processes the LOB data and obtains a latent representation from it, which is used by the decoder, parameterized by ψ, to predict the survival function of the limit orders. This decoder comprises a fixed monotonic neural network that guarantees a monotonic decreasing survival function. Further, a convolutional-Transformer encoder is used 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. We denote $\theta = \{\phi, \psi\}$ to the full set of parameters.

5.6.2 Convolutional-Transformer Encoder

We propose a convolutional-Transformer encoder to identify patterns in the LOB, which allows for a better estimate of the fill probabilities of limit orders. The architecture of the encoder, illustrated in Figure 5.5, processes the LOB time series

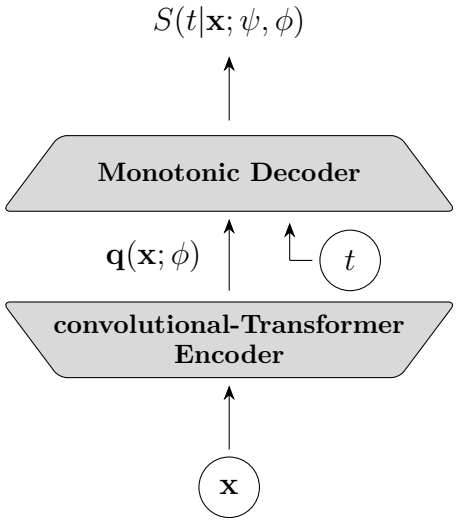

Figure 5.4: Encoder-decoder architecture for predictive tasks. The first block, an encoder with parameter $\phi \in \mathbb{R}$, uses an attention-based mechanism to project the irregularly sampled LOB observations to a latent representation that comprises critical information. The second block, a monotonic decoder, takes as input both this latent representation of the time series and the time input t. This decoder has positive parameters $\psi \in \mathbb{R}^+$, enforcing monotonicity on the survival function.

data and captures its non-Markovian dynamics through a latent representation of the time series. This representation encapsulates the most relevant information and is later used by the decoder to predict the fill probabilities.

Our encoder uses data to learn an optimal pattern extraction function, rather than relying on traditional model-based approaches, which are prone to model misspecification. This encoder consists of two main components: a locally-aware convolutional network and a Transformer model. The locally-aware convolutional network comprises 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 (Niu et al., 2021) that serve as input to the Transformer model. These DCCs, which are based on Convolutional Neural Networks (CNNs) (LeCun, Boser, et al., 1989), operate through an inner product 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, as Eq. (5.6) manifests. Further, using causal convolutions ensure that the current position does

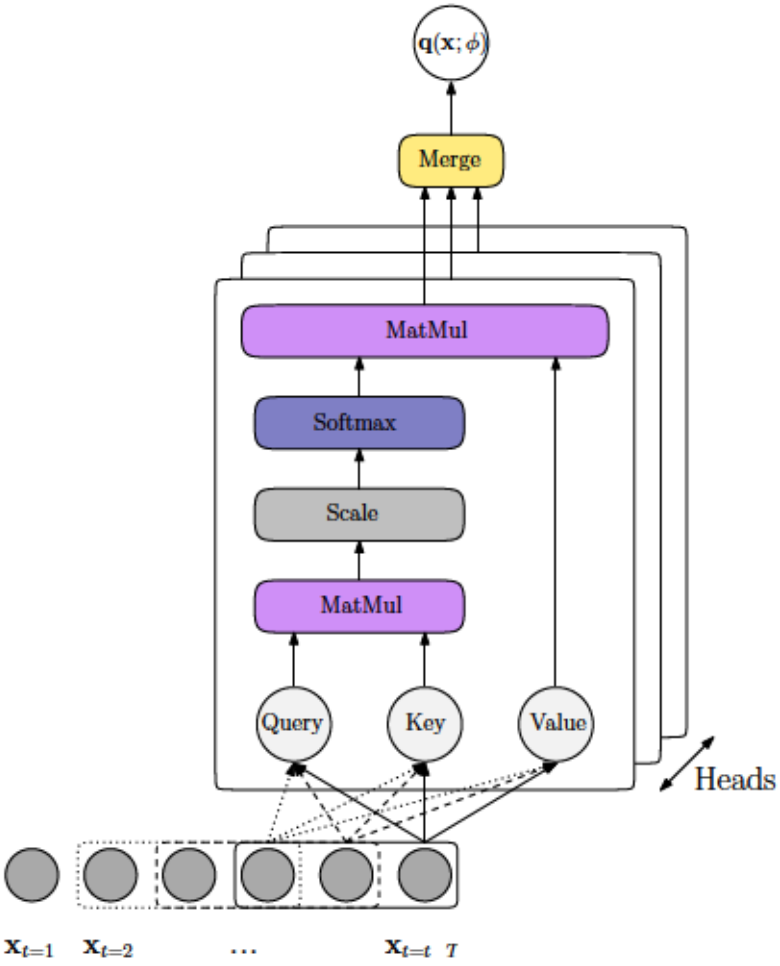

Figure 5.5: Convolutional-Transformer's encoder architecture. A convolutional kernel of size $s = 3$ with dilation factor $d = 1$ is used in this diagram. The locally-aware hidden representation obtained by the CNN is used to feed the Transformer, which uses self-attention over these hidden variables to obtain the latent representation $\mathbf{q}(\mathbf{x};\phi)$.

not use future information. DCCs have been successfully applied in time series forecasting (Borovykh et al., 2017) and quantitative finance (Moreno-Pino and Zohren, 2022), motivating our architectural choice.

The queries, keys, and values created by the DCCs are three different representations of the input data and are collectively used by the Transformer model to perform self-attention and capture dependencies between different parts of the original time series. Using convolutional networks to generate these Transformer's input features 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 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, as it now operates on a locally aware hidden representation, rather than point-wise values that lack local context. Additionally, the convolutional-Transformer optimizes the parameters of each of the three DCCs to extract different relevant features from the LOB data. For example, some filters may be optimized 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 consist of only one layer performing a causal convolutional operation between the input sequence, $x \in \mathbb{R}$, and the corresponding convolutional kernel k of size $s \in \mathbb{Z}$:

$$\begin{cases} Q(t) &= \left(x *_d k^Q\right)(t) = \sum_{\tau=0}^{s-1} k_\tau^Q \cdot x_{t-d\tau}, \\ K(t) &= \left(x *_d k^K\right)(t) = \sum_{\tau=0}^{s-1} k_\tau^K \cdot x_{t-d\tau}, \\ V(t) &= \left(x *_d k^V\right)(t) = \sum_{\tau=0}^{s-1} k_\tau^V \cdot x_{t-d\tau}, \end{cases} \tag{5.6}$$

where d is the dilation factor and $d = 1$ results in a Causal-CNN. Notice that 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 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, as detailed in Appendix E. In Section 5.7, we test the model's performance for different convolutional kernel' sizes; the reported results are shown in Table 5.8. It is worth noting that a convolutional operation with kernel size $s = 1$ is equivalent to canonical self-attention.

After the CNNs extract the relevant features from the LOB data, producing the queries, keys, and values, they 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, Olmos, et al., 2023). These models propose a completely new architecture that leverages the attention mechanism (Bahdanau et al., 2015) to process sequences of data. Transformers offer significant advantages over more classical approaches, as their ability to maintain lookback windows with large horizons, which makes them able to detect long-term dependencies in the data. Nevertheless, they also suffer from different weaknesses. Among those, canonical Transformers' point-wise dot-product attention makes them prone to anomalies and optimization issues due to the fact that these models' space complexity grows quadratically with the input length. Furthermore, canonical Transformers are locally-agnostic, as dot-product attention does not allow the model to be aware of the local context while operating with time series data. The convolutional-Transformer alleviates these problems: the integration of convolutional networks makes the model locally-aware and a sparse self-attention mechanism mitigate its space complexity, reducing the cost of computing the attention scores from $\mathcal{O}(L^2)$ to $\mathcal{O}(L(\log(L))^2)$, where L 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 $H \in \mathbb{Z}$, comprising what is called a multi-head Transformer, as shown in Figure 5.5. 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 computed as follows:

$$\mathrm{h_i} = \text{Attention}\left(Q_i, K_i, V_i\right) = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i. \tag{5.7}$$

Each Transformer's head is therefore responsible for learning and modelling attention functions able to handle the complex dependencies within the LOB data. The use of different heads allows the model to jointly attend to different temporal subspaces of

the original time series, they are finally combined to obtain a joint representation:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}\left(\text{h}_1, \text{h}_2, \ldots, \text{h}_i, \ldots \text{h}_\text{H}\right), \tag{5.8}$$

where h_i represents head i^{th}'s output, $\text{h}_\text{i} = \text{Attention}\left(Q_i, K_i, V_i\right)$. Merging every head's output through a linear function produces the latent representation $\mathbf{q}(\mathbf{x}; \phi)$, which encodes the most relevant information from the selected features of the LOB. This latent representation is used by the decoder to predict the orders' conditional density of survival times in a monotonically-decreasing manner.

It is worth mentioning that, despite the proposed use of the convolutional-Transformer, this paper presents a model-agnostic encoder because different architectures can perform the encoder's tasks. Thus, Section 5.7 shows the performance's variation for a number of stocks when different machine learning models act as encoders: a Multi-Layer Perceptron (MLP), a Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997), and a CNN are employed to this end, where modelling recurrence in the data is expected to be critical to obtain satisfactory results. In this regard, it should be noted that very liquid markets may not need complex models performing the encoder's functions, as relevant data for predicting the survival function is already encoded on the very recent time windows.

5.6.3 Monotonic Decoder

In the context of survival analysis, the survival function needs to be decreasing with respect to time. This inductive bias is something we would like to encode into our architecture to avoid the well-known crossing problem (Tagasovska and Lopez-Paz, 2019). To this end, we implement a monotonic decoder employing monotonically restricted neural networks (Chilinski and Silva, 2020; Rindt et al., 2022), see Figure 5.6. This type of neural network allows us to estimate a Cumulative Density Function (CDF), $F(t|\mathbf{x})$, 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 $f_\psi(\cdot)$ is consistent with the properties of a CDF because it satisfies:

(i) $\lim_{t\to-\infty} f_\psi(t,\mathbf{x}) = 0,$

(ii) $\lim_{t\to\infty} f_\psi(t,\mathbf{x}) = 1,$

(iii) $\dfrac{\partial f_\psi(t,\mathbf{x})}{\partial t} \geq 0,$

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. To clarify how this condition is satisfied, we consider a set-up where all the intermediate layers h of the network, with $1 < h < H$ (not to be confused with the use of h_i in Section 5.6.2 to accredit the Transformer model's head $h_i \in H$), have the following input-output relationship for each node j and input $\mathbf{x}^h \in \mathbb{R}^{M^h}$:

$$\mathbf{h}_j^l = tanh\left(\sum_{i=1}^{M^h} w_{ij}^h x_i + b_{ij}^h\right), \tag{5.9}$$

where the final output is given by:

$$f_\psi(t,\mathbf{x}) = P(T \leq t|\mathbf{X}=\mathbf{x}) = \sigma\left(\sum_{i=1}^{M^H} w_{i1}^H x_i + b_{i1}^H\right), \tag{5.10}$$

being w_{ij} and b_{ij} the individual weight and bias terms associated to node j, and $tanh(\cdot)$ and $\sigma(\cdot)$ the hyperbolic tangent and sigmoid functions, respectively. To enforce monotonicity of the output with respect to the response variable t, we impose $w_{ij}^h \geq 0 \quad \forall h \in \{1,\ldots,H\}$, as the derivative of the output of the node associated to the response variable in the first layer of the decoder is given by:

$$\frac{\partial \mathbf{h}_j^1}{\partial t} = \tanh'\left(w_{1M^1}^1 t + b_{1M^1}^1\right) w_{1M^1}^1, \tag{5.11}$$

which requires positivity of the weight to guarantee the monotonic condition. A similar argument holds for all nodes in subsequent layers of the network following from the chain rule. Finally, the remaining two conditions are satisfied empirically given the chosen likelihood-based training method. We stress that placing this monotonic restriction 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-parameterized case. This stem from the fact that the applied restriction is

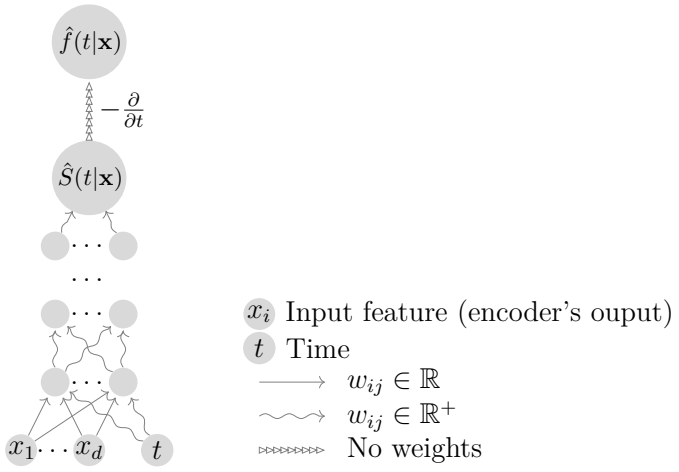

Figure 5.6: Monotonic decoder's architecture. The last node represents the operation of differentiating the conditional survival distribution with respect to time, which results in the conditional density function of the survival time. This model guarantees a decreasing survival curve. [Adapted from Figure 1 of Chilinski and Silva (2020)].

only enforced to the decoder, which can be made arbitrarily small (in terms of parameters) compared to the time series encoder, where no restriction is placed on the network parameters.

5.7 Experiments

5.7.1 Predictive Features

We now describe the set of features we use as input signals to the model, summarised in Table 5.4. While many machine learning models in the context of LOB modelling aim to extract information directly from the observed order book (Z. Zhang, Zohren, and Roberts, 2019), we also introduce additional indicators. In particular, we split

Table 5.4: Features used to estimate the fill probability.

	Feature
Slow Moving Features	Volatility
	Time of Day
Fast Moving Features	Prices and Volumes
	Midprice
	Spread
	Imbalance
	Microprice

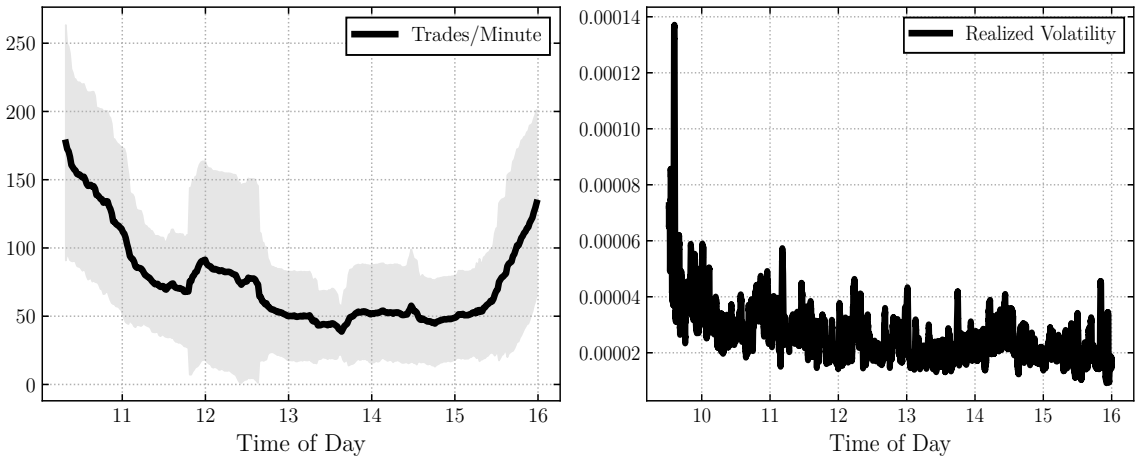

Figure 5.7: Left: Number of traded volume per minute. **Right:** Realized volatility for a given trading day.

our dataset into slow moving and fast moving signals, which extract information at different timescales.

Slow Moving Features

Slow moving features, such as time of day or volatility, provide a global view of the trading day. Regarding the time of day, it is informative because there are a larger number of trades happening at the beginning and at the end of the trading day, which increases the chance of a limit order being filled, see Figure 5.7. Regarding the volatility, different studies have validated its predictive capabilities in different settings (Heston, 1993). We employ the realised volatility estimator (Gould et al., 2013) as the volatility proxy measure, which is defined as:

$$V_t = std(\{r^m_{i,i+1}, i = t, \dots, t-k\}), \tag{5.12}$$

where the log-returns of the midprice time series $r^m_{t,t+1}$ for a lookback window of k trades is considered, being $std(\cdot)$ the standard deviation operator.

Fast Moving Features

Fast moving features are expected to yield more granular information about the order book, providing information based on the high-frequency signals observed in the market. Other works have performed short-term price looking at raw limit

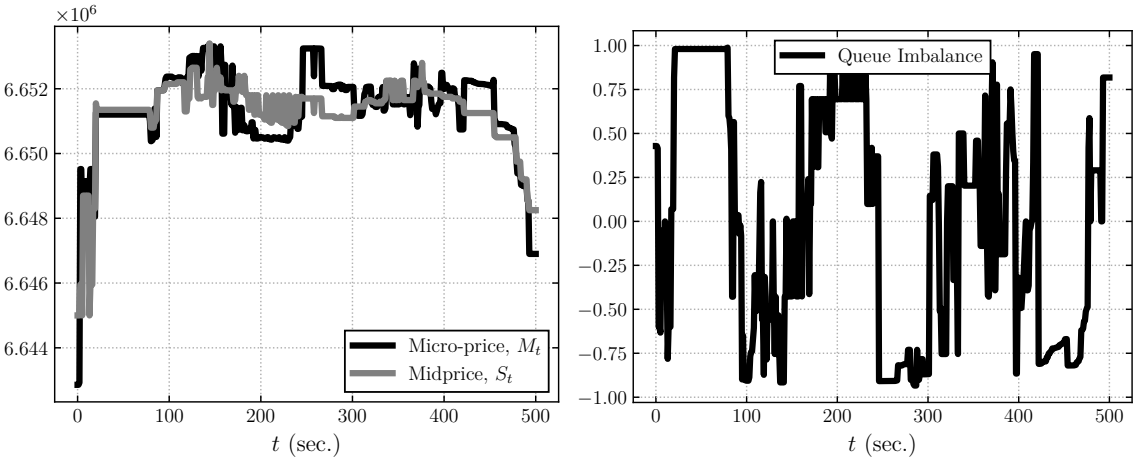

Figure 5.8: Left: Evolution of microprice (where S_t denotes the midprice) **Right:** Queue imbalance over the first 500 seconds of a trading day.

order book data (Z. Zhang, Zohren, and Roberts, 2019) and, despite the proven success of these approaches, we also consider some extra indicators such as the volume imbalance signal, given by:

$$\Upsilon_t = \frac{v_b^1(t) - v_a^1(t)}{v_b^1(t) + v_a^1(t)} \in [-1, 1]. \tag{5.13}$$

Volume imbalance captures the difference between the buy and sell pressures in the order book at time t, and it is a predictor of short-term price moves, see Cartea, Donnelly, et al. (2018). When Υ_t is close to $1/-1$, there is buy/sell pressure. If volume imbalance is a predictor of future price moves, this should be a strong indicator of whether an order (at different depths in the book) is going to fill or not. An alternative indicator of future price moves is the microprice, which follows the definition:

$$M_t = \frac{v_b^1(t)}{v_b^1(t) + v_a^1(t)} p_a^1(t) + \frac{v_a^1(t)}{v_b^1(t) + v_a^1(t)} p_b^1(t). \tag{5.14}$$

As detailed in Cartea, Jaimungal, et al. (2015), the microprice can be used as a proxy for an asset's transaction cost-free price, which measures the proneness of a price of moving towards the bid/ask side depending on the sell/buy pressure, see Figure 5.8.

5.7.2 Model Fit

We train our model using Lobster data (R. Huang and Polak, 2011) from 1st September 2022 to 26th December 2022. One hundred time points are chosen

randomly at each available trading day to position the "synthetic" orders, see Section 5.5.1. The side of the book the order is placed in is chosen randomly as well. Furthermore, at the time of submission of the synthetic order, we store the top five levels of the book as well as our handcrafted indicators with lookback windows of $T = 50/500/1000$. These elements compose the features used to perform our predictions. Figures 5.7 and 5.8 show examples of some of these features' evolution for a window of $T = 500$.

In the rest of this section, we provide an overview of the usefulness of the proposed survival model. To do so, we use the monotonic encoder-decoder convolutional-Transformer (MN-Conv-Trans) to estimate the fill times of limit orders posted in the LOB for nine different tickers. We compare the results of our model with DeepSurv (Katzman et al., 2018) and DeepHit (C. Lee, Zame, et al., 2018a), two highly popular deep learning-based models for survival analysis. Additionally, in order to assess the benefits of utilising the convolutional-Transformer model as encoder, see Section 5.6.2, we define a collection of monotonic baseline models based on the encoder-decoder architecture outlined in Section 5.6 and summarised in Figure 5.4. Each of these encoder-decoder architectures substitutes the convolutional-Transformer with a well-known machine learning model that performs the encoder's tasks. Firstly, the monotonic-MLP (MN-MLP) follows the approach presented in Rindt et al. (2022), where a MLP is used to predict the fill times probabilities, therefore not using any kind of recurrence. Secondly, the monotonic-CNN (MN-CNN) employs a CNN to extract the most important features from the dataset. Finally, the monotonic-LSTM (MN-LSTM) employs an LSTM network as encoder model. Notice that both the MN-CNN and the MN-LSTM are able to model recurrence in the dataset, as the convolutional-Transformer does. These recurrent models use a lookback window of past observations that allows them to integrate the temporal dynamics of the dataset into the prediction. To assess the effect of varying this lookback window's length, different experiments using $T = \{50, 500, 1000\}$ are conducted.

Table 5.5 provides a summary of the predictive performance of these models using the negative RCLL, a proper scoring rule as detailed in Section 5.4, for different

Table 5.5: Models evaluated using the negative right-censored log-likelihood, at different levels of the order book, for lookback windows of $T = \{50, 500, 1000\}$.

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

lookback window's sizes. Further, the best result for each ticker is remarked. Clearly, the proposed monotonic encoder-decoder convolutional-Transformer, whose predictions for AAPL and AMZN are shown in Figure 5.9, outperforms all other models in all the considered set-ups. We observe that using recursion significantly enhances the predictive capabilities of the models. Specifically, as seen in Table 5.6, which shows the improvement percentages of DeepSurv/DeepHit/MN-CNN/MN-LSTM/MN-Conv-Trans over the more simplistic MN-MLP, employing the MN-CNN with a loopaback window of $T = 50$ improves MN-MLP's predictions by more than 40%, on average. This aligns with financial intuition, as considering information from the recent past is expected to improve short-term forecasts. Additionally, we find that the MN-LSTM generally improves the performance of the MN-CNN, while the Transformer-based model exhibits the best performance overall, improving MN-MLP's prediction by 62%. Notice that, as mentioned in Section 5.6.2, the convolutional network employed by the MN-CNN is far more complex than the one employed by the convolutional-Transformer. The former employs several layers of CNNs to boost its capabilities, while the later employs a single layer performing a dilated causal convolution. Finally, we should remark that the performance of both the MN-LSTM and the MN-Conv-Trans models does not strongly change for different lookback window's sizes, while the MN-CNN perform better with larger

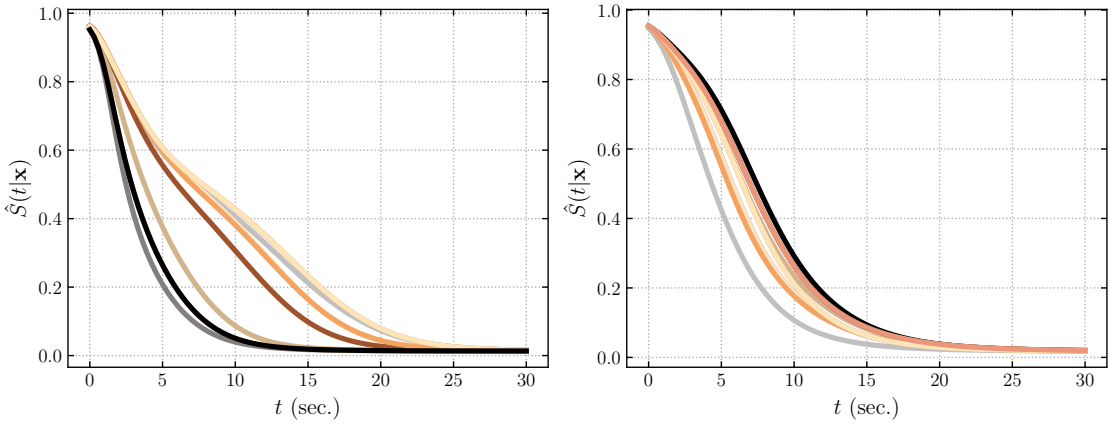

Figure 5.9: Example of survival functions predicted by the monotonic encoder-decoder monotonic convolutional-Transformer model for different limit orders. **Left:** AAPL, **Right:** AMZN.

Table 5.6: Percentage improvement, over the MN-MLP, for each of the evaluated recurrent models for lookback windows of $T = \{50, 500, 1000\}$.

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
$T = 50$										
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
$T = 500$										
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
$T = 1000$										
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	67.08	**59.12**	66.99	**62.66**

horizons, suggesting that most relevant information for estimating the fill times of limit orders dwell within the most recent observations.

In Table 5.7, we repeat this same analysis but considering an order flow representation of the order book, as recommended by Kolm et al. (2021) and Lucchese et al. (2022). In these works, the authors suggest that employing order flow or volume representations of the order book increases predictive performance for step-ahead forecasts, making unnecessary the use of more complex models. Anyhow, contrary to the case of midprice forecasting, we find no consistent improvement when using this type of data representation.

Finally, Table 5.8 sheds light on the convolutional operation performed by the monotonic convolutional-Transformer. This table explores different kernel sizes, $s \in \{1, 2, 3, 5, 10, 25, 50\}$, for the MN-Conv-Tran's convolutional operation while predicting AAPL's survival function using a lookback window of $T = 500$. Recall that a convolutional network with kernel size $s = 1$ results on canonical self-attention. In that case, there is an evident decline in the negative RCLL compared to larger kernel sizes. No substantial variations in the performance are observed among the

other kernel sizes examined. Therefore, a value of $s = 3$ seems reasonable to avoid an unnecessary increase in parametric complexity.

Table 5.7: 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, 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	8.441 ± 0.022	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	9.843 ± 0.083	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 ± 0.317	5.646 ± 0.163
MN-LSTM	3.746 ± 0.136	4.838 ± 0.248	7.320 ± .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	30.58	23.69	51.99	16.84
DeepHit	4.94	6.47	28.78	19.72	25.19	19.05	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 5.8: Performance variation while using different kernel's sizes for the MN-Conv-Trans's DCC network, for AAPL and a lookback window of $T = 500$.

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

5.7.3 Model Interpretability

In this section, we interpret the convolutional-Transformer's predictions to shed light on how it produces them. To do so, we analyse both the time and feature domain of the model's operation. Firstly, through attention heatmaps, we visualise which parts of the input signals' past values are more important to the model while predicting the survival function. Secondly, we use Shapley values (Hart, 1989) to quantify the importance of each input feature, which allows us to determine the most relevant of these input signals and how their value affects the model's predictions.

Attention Heatmaps

The convolutional-Transformer employs Eq. (5.6) to obtain, through the convolutional network, the self-attention input features: query, key, and value. Once these are obtained, the Transformer model, through Eq. (5.7), performs the dot-product computation between the attention's queries and keys, applying a scaling factor $\sqrt{d_k}$ that guarantees a variance equal to one: $Q_i K_i^T \,/\, \sqrt{d_k}$. This operation results on a matrix of dimensions $\mathbb{R}^{T\times T}$, where $T \in \mathbb{Z}$ is the lookback window's length. Finally, after a softmax function is applied to this matrix, see Eq. (5.7), the output is used to multiply the previously obtained self-attention's values. Through this last operation, each Transformer's head selects the most relevant time instants. Therefore, the matrix resulting from the dot-product operation, $Q_i K_i^T \,/\, \sqrt{d_k}$, allows us to visualise which regions of the lookback window (more specifically, of its non-linear projection), the model pays more attention to while predicting the survival function. This process is illustrated through Figures 5.10 and 5.11. More precisely, Figure 5.10 represents the evolution of the input features when a lookback window of $T = 500$ is used. These are the signals the model uses to perform its prediction. On the other hand, Figure 5.11 shows four different attention heatmaps, one per each head of the Transformer model. These attention heatmaps, which shows the self-attention weights, comprise information regarding the importance of each time-step, encoded in the intensity of each colour. Thus, Figure 5.11 shows which parts of the input sequence each Transformer's head is paying more attention to.

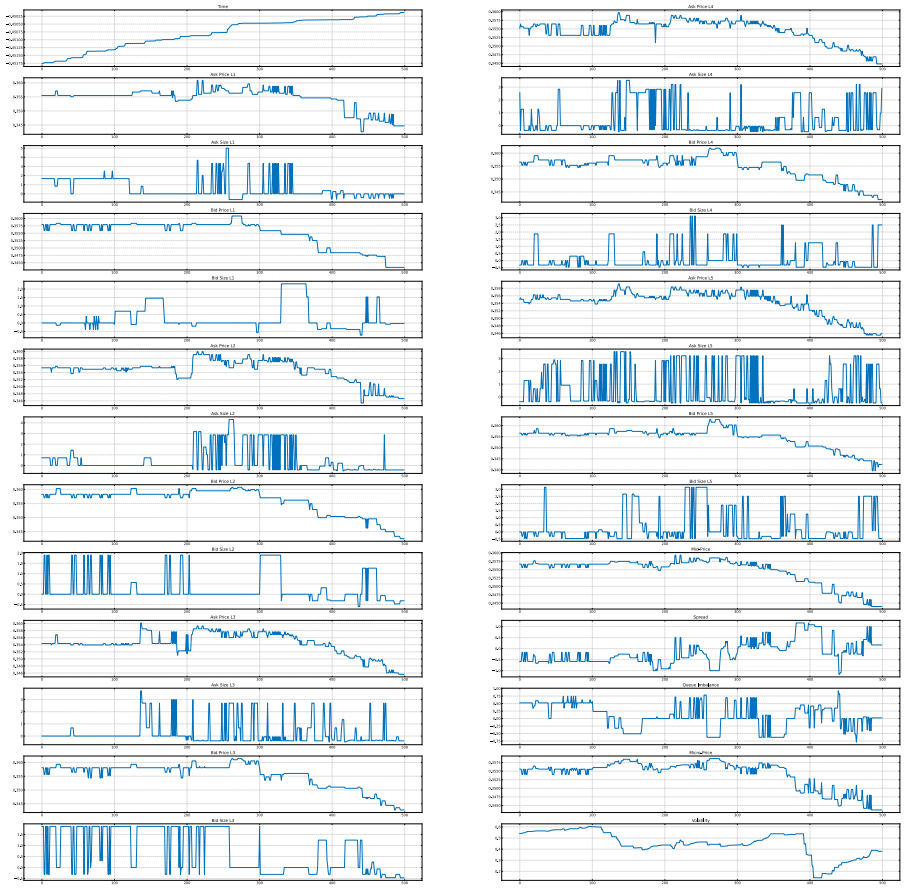

Figure 5.10: Features used to predict the survival function for a specific limit order, using $T = 500$.

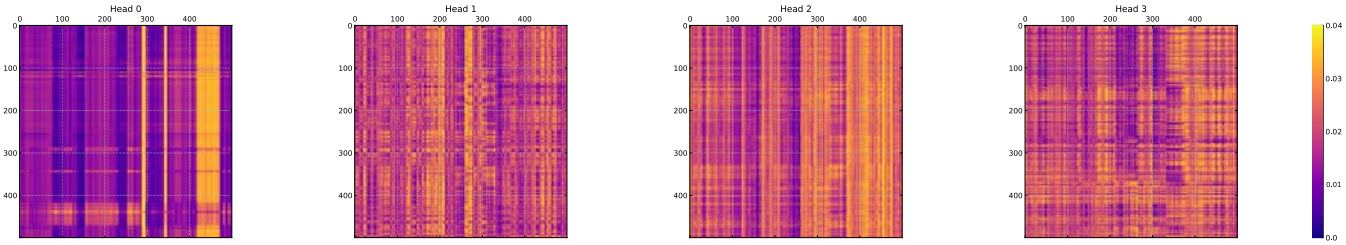

Figure 5.11: Attention heatmaps obtained for the order represented in Figure 5.10.

While head-1 shows quite a sparse attention pattern, it is clearly visible that head-0 focuses on samples after time instant $T = 400$, where a volatility shock is clearly visible, see Figure 5.10. Heads 2 and 3 also focus on the last samples when, as clearly noticeable in Figure 5.10, a large deprecation in the ask and bid prices occurred, certainly comprising relevant information for the prediction of the fill probability.

Shapley Values

Shapley values, on the other hand, provide a method to attribute model's predictions to the individual features of the input. This can help us understand which of the order's features are most important and how they contribute to the model's overall prediction. To calculate these Shapley values, we follow the DeepSHAP approach in Lundberg and S.-I. Lee (2017), where it is shown that the per node attribution rules used by DeepLIFT (Shrikumar et al., 2017) can be chosen to approximate them. To measure the importance per feature, we define $\mathbf{S} \subseteq \mathbf{F}$ as all possible feature subsets, where $\mathbf{F}$ is the set of all features. We firstly integrate over many 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 Shapley values such that they sum up to the difference between this expected model's output and the predicted values: $\hat{f}_{\mathbf{S}}(t|\mathbf{x}) - \mathbb{E}[\hat{f}(t|\mathbf{x})]$. The contribution of feature i^{th}, that is, its importance, can be measured as:

$$C_i = \hat{f}_{\mathbf{S}}(t|\mathbf{x}) - \hat{f}_{\mathbf{S}\setminus\{i\}}(t|\mathbf{x}), \tag{5.15}$$

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

$$\phi_i = \frac{1}{n!}\sum_{p=0}^{n} p!(n-p)!C_i, \tag{5.16}$$

where n is the total number of features and p the number of features present in the input sample, which can be any subset over the total number of features depending on the specific input being evaluated.

Overall, the study of Shapley values provides a powerful tool to interpret the behaviour of our deep learning model. Figure 5.12 shows a Beeswarm plot, a complex and information-rich display of Shapley values, which reveals the relative importance of each feature and their relationships with the predicted outcome. Each of the datapoints is represented with a single dot in Figure 5.12, where colours ranging from blue to red indicate lower to higher values per feature. The vertical axis' order represents the importance of each feature for the convolutional-Transformer

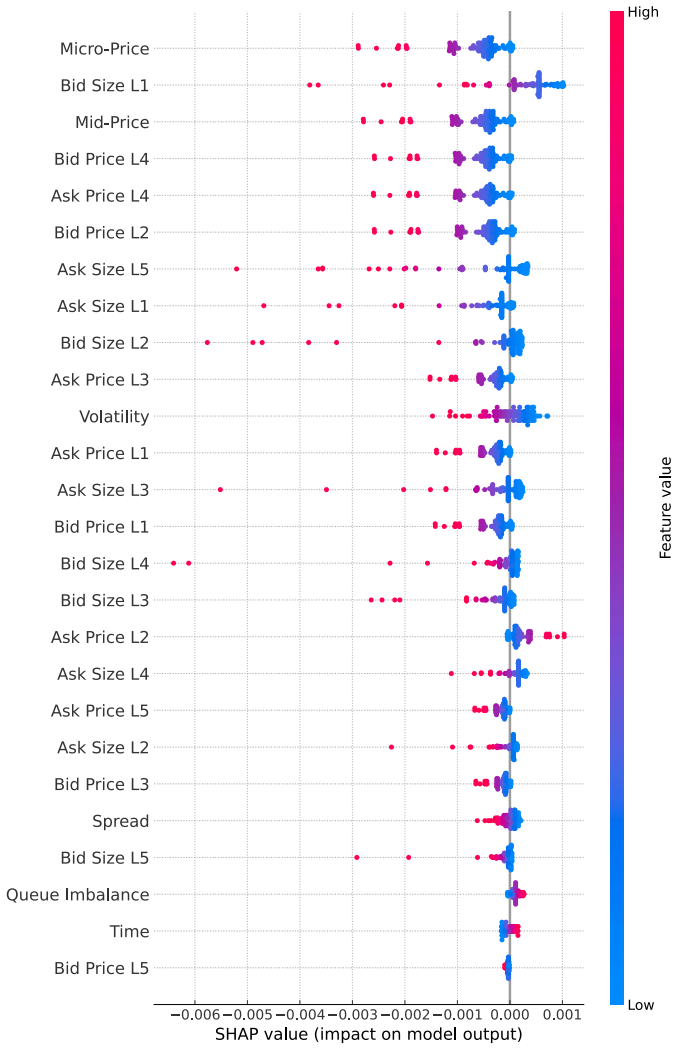

Figure 5.12: Shapley values for 100 limit orders.

on average, while the horizontal axis comprises the associated Shapley value. One can appreciate how certain input features, e.g., the bid price in Level 5, or the time of day itself (which probably is already integrated into the other features), are not critical. Further, features such as the microprice and the bid size on Level 1 are driving the model's predictions. Finally, it should be mentioned that the distribution of the points itself is also informative, e.g., Figure 5.12 clearly reflects how certain extreme observations (sparse red dots) have a significant effect on the model's output: extreme values in Level 4's bid price may entail a large order which already consumed the previous levels. On the other hand, dense clusters of averaged

values are usually associated with small Shapley values.

To conclude this section, we should remark that by visualising attention heatmaps along with Shapley values, we provide a more interpretable view of the monotonic encoder-decoder convolutional-Transformer's performance, gaining valuable insights into how it interacts with the model's inputs. The attention heatmaps visualisation allows us to observe the regions of the input sequence that the model focuses on at every time step of the prediction process. The Shapley values, on the other hand, provide a clear view of which features are most important to the model and how they contribute to its overall prediction. This allows us to gain a better understanding of the model's decision-making process and to identify potential areas for improvement, such as ensuring that the model considers the most relevant features when making its predictions.

5.8 Discussion

This paper presents a novel approach to estimating the fill probabilities of limit orders posted in the LOB. Our data-driven approach integrates a novel convolutional-Transformer model to raise local-awareness from LOB data, and a monotonic neural network which guarantees the theoretical correctness of the survival function estimation. To train and evaluate the proposed model, we use the right censored log-likelihood, which is a proper scoring rule, unlike other scoring rules that are commonly used in the literature. To demonstrate the effectiveness of our method, we conduct a set of experiments on real LOB data. These experiments show that the monotonic encoder-decoder convolutional-Transformer significantly outperforms state-of-the-art benchmarks, providing a novel general framework to perform survival analysis from time-series observations. Finally, we provide an interpretability analysis based on Shapley values, providing insight into which features are the most influential.

6
Conclusions and Future Work

Contents

The preceding chapters have conducted a detailed analysis of three main areas of study. Chapter 3 presented key technical contributions to improve the performance of deep learning models for time series modelling and forecasting, while Chapters 4 and 5 introduced new approaches for solving common quantitative-finance problems with machine learning-based techniques. In the final chapter of this dissertation, we briefly recapitulate our primary contributions, including a succinct summary of the key findings and a thorough examination of potential avenues for future research development.

6.1 Methods and Contributions

Chapter 3 proposed a novel probabilistic time series forecasting methodology that introduces the use of spectral domain-based deep learning models, merging classic signal processing filtering techniques with machine learning architectures. The proposed solution can alleviate some of the inconveniences commonly associated

with deep autoregressive models. These architectures are often prone to prioritising recent past data, hence ignoring critical global information not contained in previous time steps. Additionally, they are susceptible to error accumulation and propagation and may not yield illustrative results. The proposed model, the Spectral Attention Autoregressive Model (SAAM), mitigates these problems by combining deep autoregressive models with a Spectral Attention (SA) module. This module uses two attention models operating over the Fourier domain representation of the time series' embedding. Through spectral filtering, Spectral Attention differentiates between the components of the frequency domain that should be considered noise and subsequently filtered out, and the global patterns that are relevant and should be incorporated into the predictions. The Spectral Attention module modifies the model's latent representation of the time series and incorporates this information within the forecast. Further, experiments on synthetic and real-world datasets empirically proved these statements and unveiled how our suggested modular architecture can be incorporated into a variety of base deep autoregressive models, consistently improving the results of these base models and achieving state-of-the-art performance.

Chapters 4 and 5 shifted toward showcasing the benefits of machine learning-based solutions for different problems in the field of quantitative finance, proving how data-driven approaches compare favourably to classic parametric-based models and providing solutions for various algorithmic and high-frequency trading scenarios.

Specifically, Chapter 4 addressed the problem of volatility forecasting, which plays a central role among equity risk measures. In this chapter, an empirical analysis was conducted to evaluate the performance of Dilated Causal Convolutional-based neural networks in the context of volatility forecasting. A set of robust loss functions was used during this performance assessment, conducted in a large dataset comprising two years of high-frequency intraday financial data. The results showed that the proposed model, DeepVol, obtained significant performance gains compared to well-established volatility-oriented parametric models such as GARCH and its variants, e.g., EGARCH and the HEAVY model. Further, this empirical evaluation

is especially relevant considering that the experiments were conducted in high volatility regimes, such as the 2020 stock crisis caused by the COVID-19 pandemic. Out-of-sample forecasts revealed how, in this tumultuous scenario, our machine learning-based solution outperformed baseline methods while exhibiting robustness in the presence of volatility shocks, showing its ability to extract universal features and transfer learning to out-of-distribution data. Thus, Chapter 4 demonstrated how models based on Dilated Causal Convolutions can avoid the limitations of classical methods, such as model misspecification or the use of hand-crafted noisy realised measures, by taking advantage of the abundance of high-frequency data. Consequently, we concluded that these data-driven approaches should be carefully considered in the context of volatility forecasting, as they can be instrumental in the valuation of financial derivatives, risk management, and the formation of investment portfolios.

Finally, Chapter 5 presented a novel methodology for estimating the distribution of fill times for limit orders posted in the Limit Order Book (LOB). The proposed data-driven approach does not make assumptions about the underlying stochastic processes. It employs state-of-the-art deep learning methodologies to grant practitioners the capability of making informed decisions between market orders and limit orders, a tradeoff between immediate execution and price premium. The survival analysis model introduced in this chapter examines the relationship between the time-varying features of the LOB and the distribution of fill times. The proposed model is based on a convolutional-Transformer encoder and a monotonic decoder. We offer an exhaustive comparison of the survival functions resulting from different order placement strategies, offering insight into the fill probability of orders placed within the spread. Further, the effectiveness of the presented methodology was assessed through an empirical evaluation based on the right-censored log-likelihood, a proper scoring rule. The experiments conducted revealed the superior performance of the monotonic encoder-decoder convolutional-Transformer compared to state-of-the-art benchmarks, leading to more accurate predictions and improved economic value.

6.2 Future Work

We conclude this thesis with a brief discussion of future research directions in light of the problems addressed in this doctoral project. In this regard, we separately consider the two primary themes covered: the use of spectral domain-based machine learning models and the development of data-driven solutions in the context of quantitative finance.

Concerning Fourier domain-based deep learning models, future work should explore new approaches to take advantage of signal processing techniques in the machine learning context, as we do when combining attention models with spectral filtering. Different alternatives for the statistical characterisation of the process should be studied. E.g., the Discrete Cosine Transform (DCT), a real-to-real function, would simplify the process of modifying the phase of the time series embedded representation in comparison with the Discrete Fourier transform (DFT), a real-to-complex transformation, employed in Chapter 3. Further, using Spectral Attention as a replacement for Transformer models' self-attention mechanism would allow Transformer-based architectures to function with frequency-domain representations of the data, bringing opportunities to enhance self-attention explainability in conjunction with models' complexity reduction.

When it comes to the use of data-driven models in the field of algorithmic high-frequency trading, state-of-the-art machine learning methods have the potential to enhance the performance of traditional methodologies used by practitioners. Deep neural networks' feature extraction capabilities, which can benefit from the rising accessibility of high-frequency data, are mostly to blame for this. Additionally, since the availability of large financial datasets would endorse research efforts in the field of quantitative finance, it is important to prioritise the development of a shared set of standardised high-frequency financial time series datasets to facilitate the evaluation and comparison of various models. Finally, the field of algorithmic high-frequency trading requires the adoption of robust loss functions for accurate benchmarking. Otherwise, model comparisons lack systematic rigour. Patton

(2011) analytically proves that the presence of noise in the volatility proxy can lead to the selection of an imperfect volatility forecast over the true conditional variance for certain choices of the loss function. Additionally, Rindt et al. (2022) reaches similar conclusions in the context of survival analysis, a field in which the literature frequently makes use of improper scoring rules. Consequently, it is crucial to establish necessary and sufficient conditions for loss functions to yield rankings of both volatility forecasts and fill times of limit orders that are robust to noise, thereby leading to the development of proper benchmarking loss functions.

References

Ait-Sahalia, Yacine, Per A Mykland, and Lan Zhang (2005). "How often to sample a continuous-time process in the presence of market microstructure noise". In: *The review of financial studies* 18.2, pp. 351–416.

Alaa, Ahmed M and Mihaela van der Schaar (2017). "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.

Andersen, TG, Tim Bollerslev, and Nour Meddahi (2006). "Market microstructure noise and realized volatility forecasting". In: *Unpublished paper: Department of Economics, Duke University*.

Andersen, Torben G, Tim Bollerslev, and Francis X Diebold (2010). "Parametric and nonparametric volatility measurement". In: *Handbook of financial econometrics: Tools and techniques*. Elsevier, pp. 67–137.

Andersen, Torben G, Tim Bollerslev, Francis X Diebold, and Heiko Ebens (2001). "The distribution of realized stock return volatility". In: *Journal of financial economics* 61.1, pp. 43–76.

Antolini, Laura, Patrizia Boracchi, and Elia Biganzoli (2005). "A time-dependent discrimination index for survival data". In: *Statistics in medicine* 24.24, pp. 3927–3944.

Avati, Anand et al. (2020). "Countdown regression: sharp and calibrated survival predictions". In: *Uncertainty in Artificial Intelligence*. PMLR, pp. 145–155.

Baars, Bjorn (2014). *HEAVY and Realized (E) GARCH Models*. GlobeEdit.

Baek, Yujin and Ha Young Kim (2018). "ModAugNet: A new forecasting framework for stock market index value with an overfitting prevention LSTM module and a prediction LSTM module". In: *Expert Systems with Applications* 113, pp. 457–480.

Bahdanau, Dzmitry, Kyung Hyun Cho, and Yoshua Bengio (2015). "Neural machine translation by jointly learning to align and translate". In: *3rd International Conference on Learning Representations, ICLR 2015*.

Bai, Xiao et al. (2021). "Explainable deep learning for efficient and robust pattern recognition: A survey of recent developments". In: *Pattern Recognition* 120, p. 108102.

Baillie, Richard T, Tim Bollerslev, and Hans Ole Mikkelsen (1996). "Fractionally integrated generalized autoregressive conditional heteroskedasticity". In: *Journal of econometrics* 74.1, pp. 3–30.

Bandi, Federico M and Jeffrey R Russell (2006). "Separating microstructure noise from volatility". In: *Journal of Financial Economics* 79.3, pp. 655–692.

Biage, Milton (2019). "Analysis of shares frequency components on daily value-at-risk in emerging and developed markets". In: *Physica A: Statistical Mechanics and its Applications* 532, p. 121798.

Blackman, Ralph Beebe and John Wilder Tukey (1958). "The measurement of power spectra from the point of view of communications engineering—Part I". In: *Bell System Technical Journal* 37.1, pp. 185–282.

Bollerslev, Tim (1986). "Generalized autoregressive conditional heteroskedasticity". In: *Journal of econometrics* 31.3, pp. 307–327.

Borovykh, Anastasia, Sander Bohte, and Cornelis W Oosterlee (2017). "Conditional time series forecasting with convolutional neural networks". In: *arXiv preprint arXiv:1703.04691*.

Böse, Joos-Hendrik et al. (2017). "Probabilistic demand forecasting at scale". In: *Proceedings of the VLDB Endowment* 10.12, pp. 1694–1705.

Bouchaud, Jean-Philippe, Andrew Matacz, and Marc Potters (2001). "Leverage effect in financial markets: The retarded volatility model". In: *Physical review letters* 87.22, p. 228701.

Box, G. E. P. and G. M. Jenkins (1968). "Some Recent Advances in Forecasting and Control". In: *Applied Statistics* 17.2, p. 91.

Box, George EP and Gwilym M Jenkins (1970). *Time Series Analysis Forecasting and Control*. Tech. rep. WISCONSIN UNIV MADISON DEPT OF STATISTICS.

Brailsford, Timothy J and Robert W Faff (1996). "An evaluation of volatility forecasting techniques". In: *Journal of Banking & Finance* 20.3, pp. 419–438.

Brownlees, Christian T and Giampiero M Gallo (2010). "Comparison of volatility measures: a risk management perspective". In: *Journal of Financial Econometrics* 8.1, pp. 29–56.

Buttkus, Burkhard (2012). *Spectral analysis and filter theory in applied geophysics*. Springer Science & Business Media.

Cao, Defu et al. (2020). "Spectral Temporal Graph Neural Network for Multivariate Time-series Forecasting". In: *Advances in Neural Information Processing Systems*.

Cartea, Álvaro, Ryan Donnelly, and Sebastian Jaimungal (2018). "Enhancing trading strategies with order book signals". In: *Applied Mathematical Finance* 25.1, pp. 1–35.

Cartea, Álvaro, Sebastian Jaimungal, and José Penalva (2015). *Algorithmic and high-frequency trading*. Cambridge University Press.

Chapfuwa, Paidamoyo et al. (2018). "Adversarial time-to-event modeling". In: *International Conference on Machine Learning*. PMLR, pp. 735–744.

Chen, Qinkai and Christian-Yann Robert (2022). "Multivariate Realized Volatility Forecasting with Graph Neural Network". In: *Proceedings of the Third ACM International Conference on AI in Finance*, pp. 156–164.

Cheng, Haibin et al. (2006). "Multistep-ahead time series prediction". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, pp. 765–774.

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

Cho, Jin-Wan and Edward Nelling (2000). "The probability of limit-order execution". In: *Financial Analysts Journal* 56.5, pp. 28–33.

Cochran, Steven J, Iqbal Mansur, and Babatunde Odusami (2012). "Volatility persistence in metal returns: A FIGARCH approach". In: *Journal of Economics and Business* 64.4, pp. 287–305.

Cont, Rama (2007). "Volatility clustering in financial markets: empirical facts and agent-based models". In: *Long memory in economics*. Springer, pp. 289–309.

Corradi, Valentina and Walter Distaso (2006). "Semi-parametric comparison of stochastic volatility models using realized measures". In: *The Review of Economic Studies* 73.3, pp. 635–667.

Corsi, Fulvio (2009). "A simple approximate long-memory model of realized volatility". In: *Journal of Financial Econometrics* 7.2, pp. 174–196.

Cox, David R (1972). "Regression models and life-tables". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 34.2, pp. 187–202.

Csáji, Balázs Csanád et al. (2001). "Approximation with artificial neural networks". In: *Faculty of Sciences, Etvs Lornd University, Hungary* 24.48, p. 7.

Cybenko, George (1989). "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4, pp. 303–314.

Dane, Sohier (2015). *30 Years of European Wind Generation*. URL: `https://www.kaggle.com/sohier/30-years-of-european-wind-generation`.

Ding, Zhuanxin, Clive WJ Granger, and Robert F Engle (1993). "A long memory property of stock market returns and a new model". In: *Journal of empirical finance* 1.1, pp. 83–106.

Dirick, Lore, Gerda Claeskens, and Bart Baesens (2017). "Time to default in credit scoring using survival analysis: a benchmark study". In: *Journal of the Operational Research Society* 68.6, pp. 652–665.

Dua, Dheeru and Casey Graff (2017). *UCI Machine Learning Repository*. URL: `http://archive.ics.uci.edu/ml`.

Durbin, James and Siem Jan Koopman (2012). *Time series analysis by state space methods*. Oxford university press.

Eisler, Zoltan, Jean-Philippe Bouchaud, and Julien Kockelkoren (2012). "The price impact of order book events: market orders, limit orders and cancellations". In: *Quantitative Finance* 12.9, pp. 1395–1419.

Elman, Jeffrey L (1990). "Finding structure in time". In: *Cognitive science* 14.2, pp. 179–211.

Engle, Robert F (1982). "Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation". In: *Econometrica: Journal of the econometric society*, pp. 987–1007.

Engle, Robert F and Tim Bollerslev (1986). "Modelling the persistence of conditional variances". In: *Econometric reviews* 5.1, pp. 1–50.

Engle, Robert F and Victor K Ng (1993). "Measuring and testing the impact of news on volatility". In: *The journal of finance* 48.5, pp. 1749–1778.

Falcon, William and The PyTorch Lightning team (Mar. 2019). *PyTorch Lightning*. Version 1.4. URL: `https://github.com/Lightning-AI/lightning`.

Faraggi, David and Richard Simon (1995). "A neural network model for survival data". In: *Statistics in medicine* 14.1, pp. 73–82.

Fernández, Tamara, Nicolás Rivera, and Yee Whye Teh (2016). "Gaussian processes for survival analysis". In: *Advances in Neural Information Processing Systems* 29.

Funahashi, Ken-ichi and Yuichi Nakamura (1993). "Approximation of dynamical systems by continuous time recurrent neural networks". In: *Neural networks* 6.6, pp. 801–806.

Giuliari, Francesco et al. (2021). "Transformer networks for trajectory forecasting". In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, pp. 10335–10342.

Gneiting, Tilmann and Roopesh Ranjan (2011). "Comparing density forecasts using threshold-and quantile-weighted scoring rules". In: *Journal of Business & Economic Statistics* 29.3, pp. 411–422.

Gould, Martin D et al. (2013). "Limit order books". In: *Quantitative Finance* 13.11, pp. 1709–1742.

Graf, Erika et al. (1999). "Assessment and comparison of prognostic classification schemes for survival data". In: *Statistics in medicine* 18.17-18, pp. 2529–2545.

Groha, Stefan, Sebastian M Schmon, and Alexander Gusev (2020). "A general framework for survival analysis and multi-state modelling". In: *arXiv preprint arXiv:2006.04893*.

Hamilton, James Douglas (1994). *Time series analysis*. Vol. 2. Cambridge Univ Press.

Handa, Puneet, Robert Schwartz, and Ashish Tiwari (2003). "Quote setting and price formation in an order driven market". In: *Journal of financial markets* 6.4, pp. 461–489.

Hansen, Peter R and Asger Lunde (2006). "Realized variance and market microstructure noise". In: *Journal of Business & Economic Statistics* 24.2, pp. 127–161.

Hansen, Peter Reinhard, Zhuo Huang, and Howard Howan Shek (2012). "Realized GARCH: a joint model for returns and realized measures of volatility". In: *Journal of Applied Econometrics* 27.6, pp. 877–906.

Harrell, Frank E et al. (1982). "Evaluating the yield of medical tests". In: *Jama* 247.18, pp. 2543–2546.

Hart, Sergiu (1989). "Shapley value". In: *Game theory*. Springer, pp. 210–216.

Harvey, Andrew C (1990). "Forecasting, structural time series models and the Kalman filter". In.

Harvey, Campbell R et al. (2018). "The impact of volatility targeting". In: *The Journal of Portfolio Management* 45.1, pp. 14–33.

He, Kaiming et al. (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Heston, Steven L (1993). "A closed-form solution for options with stochastic volatility with applications to bond and currency options". In: *The review of financial studies* 6.2, pp. 327–343.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Horvath, Blanka, Aitor Muguruza, and Mehdi Tomas (2019). "Deep learning volatility". In: *arXiv preprint arXiv:1901.09647*.

Hu, Haize et al. (2022). "A novel hybrid model for short-term prediction of wind speed". In: *Pattern Recognition* 127, p. 108623.

Hu, Shi et al. (2021). "Transformer-based deep survival analysis". In: *Survival Prediction-Algorithms, Challenges and Applications*. PMLR, pp. 132–148.

Huang, Ruihong and Tomas Polak (2011). "Lobster: Limit order book reconstruction system". In: *Available at SSRN 1977207*.

Hyndman, Rob et al. (2008). *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media.

Hyndman, Rob J and George Athanasopoulos (2018). *Forecasting: principles and practice*. OTexts.

Ishwaran, Hemant et al. (2008). "Random survival forests". In: *The annals of applied statistics* 2.3, pp. 841–860.

Izzeldin, Marwan et al. (2019). "Forecasting realised volatility using ARFIMA and HAR models". In: *Quantitative Finance* 19.10, pp. 1627–1638.

Kang, Yanfei, Rob J Hyndman, and Feng Li (2020). "GRATIS: GeneRAting TIme Series with diverse and controllable characteristics". In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 13.4, pp. 354–376.

Kang, Yanfei, Rob J Hyndman, and Kate Smith-Miles (2017). "Visualising forecasting algorithm performance using time series instance spaces". In: *International Journal of Forecasting* 33.2, pp. 345–358.

Kaplan, Edward L and Paul Meier (1958). "Nonparametric estimation from incomplete observations". In: *Journal of the American statistical association* 53.282, pp. 457–481.

Karanasos, Menelaos, Stavroula Yfanti, and John Hunter (2022). "Emerging stock market volatility and economic fundamentals: the importance of US uncertainty spillovers, financial and health crises". In: *Annals of operations research* 313.2, pp. 1077–1116.

Katzman, Jared L et al. (2018). "DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network". In: *BMC medical research methodology* 18.1, pp. 1–12.

Kidger, Patrick and Terry Lyons (2020). "Universal approximation with deep narrow networks". In: *Conference on learning theory*. PMLR, pp. 2306–2327.

Kim, Ha Young and Chang Hyun Won (2018). "Forecasting the volatility of stock price index: A hybrid model integrating LSTM with multiple GARCH-type models". In: *Expert Systems with Applications* 103, pp. 25–37.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Kolm, Petter N, Jeremy Turiel, and Nicholas Westray (2021). "Deep Order Flow Imbalance: Extracting Alpha at Multiple Horizons from the Limit Order Book". In: *Available at SSRN 3900141*.

Kvamme, Håvard, Ørnulf Borgan, and Ida Scheel (2019). "Time-to-Event Prediction with Neural Networks and Cox Regression". In: *Journal of machine learning research* 20.129, pp. 1–30.

Larivière, Bart and Dirk Van den Poel (2004). "Investigating the role of product features in preventing customer churn, by using survival analysis and choice modeling: The case of financial services". In: *Expert Systems with Applications* 27.2, pp. 277–285.

Laurie, John A et al. (1989). "Surgical adjuvant therapy of large-bowel carcinoma: an evaluation of levamisole and the combination of levamisole and fluorouracil. The North Central Cancer Treatment Group and the Mayo Clinic." In: *Journal of Clinical Oncology* 7.10, pp. 1447–1456.

Lawless, Jerald F (2011). *Statistical models and methods for lifetime data*. John Wiley & Sons.

Lea, Colin et al. (2016). "Temporal convolutional networks: A unified approach to action segmentation". In: *European Conference on Computer Vision*. Springer, pp. 47–54.

LeCun, Yann, Yoshua Bengio, et al. (1995). "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks* 3361.10, p. 1995.

LeCun, Yann, Bernhard Boser, et al. (1989). "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4, pp. 541–551.

Lee, Changhee, Jinsung Yoon, and Mihaela Van Der Schaar (2019). "Dynamic-deephit: A deep learning approach for dynamic survival analysis with competing risks based on longitudinal data". In: *IEEE Transactions on Biomedical Engineering* 67.1, pp. 122–133.

Lee, Changhee, William Zame, et al. (2018a). "Deephit: A deep learning approach to survival analysis with competing risks". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1.

Lee, Changhee, William Zame, et al. (2018b). "Deephit: A deep learning approach to survival analysis with competing risks". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1.

Leung, Kwan-Moon, Robert M Elashoff, and Abdelmonem A Afifi (1997). "Censoring issues in survival analysis". In: *Annual review of public health* 18.1, pp. 83–104.

Li, Shiyang et al. (2019). "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting". In: *Advances in Neural Information Processing Systems*, pp. 5243–5253.

Lim, Bryan, Sercan Ö Arik, et al. (2021). "Temporal fusion transformers for interpretable multi-horizon time series forecasting". In: *International Journal of Forecasting*.

Lim, Bryan and Stefan Zohren (2021). "Time-series forecasting with deep learning: a survey". In: *Philosophical Transactions of the Royal Society A* 379.2194, p. 20200209.

Lin, Yu et al. (2022). "Forecasting the realized volatility of stock price index: A hybrid model integrating CEEMDAN and LSTM". In: *Expert Systems with Applications*, p. 117736.

Littwin, Etai and Lior Wolf (2020). "On the convex behavior of deep neural networks in relation to the layers' width". In: *arXiv preprint arXiv:2001.04878*.

Liu, DC and J Nocedal (1989). "On the limited memory method for large scale optimization: Mathematical Programming B". In.

Liu, Peter J et al. (2018). "Generating Wikipedia by Summarizing Long Sequences". In: *International Conference on Learning Representations*.

Lo, Andrew W, A Craig MacKinlay, and June Zhang (2002). "Econometric models of limit-order executions". In: *Journal of Financial Economics* 65.1, pp. 31–71.

Lu, Yulong and Jianfeng Lu (2020). "A universal approximation theorem of deep neural networks for expressing probability distributions". In: *Advances in neural information processing systems* 33, pp. 3094–3105.

Lucchese, Lorenzo, Mikko Pakkanen, and Almut Veraart (2022). "The Short-Term Predictability of Returns in Order Book Markets: a Deep Learning Perspective". In: *arXiv preprint arXiv:2211.13777*.

Lundberg, Scott M and Su-In Lee (2017). "A unified approach to interpreting model predictions". In: *Advances in neural information processing systems* 30.

Lütkepohl, Helmut (2005). *New introduction to multiple time series analysis*. Cambridge Univ Press.

Mademlis, Dimitrios Kartsonakis and Nikolaos Dritsakis (2021). "Volatility Forecasting using Hybrid GARCH Neural Network Models: The Case of the Italian Stock Market". In: *International Journal of Economics and Financial Issues* 11.1, p. 49.

Maglaras, Costis, Ciamac Moallemi, and Muye Wang (2021). *A Deep Learning Approach to Estimating Fill Probabilities in a Limit Order Book*.

Makridakis, Spyros and Evangelos Spiliotis (2020). "The M4 Competition: 100,000 time series and 61 forecasting methods". In: *International Journal of Forecasting* 36.1, pp. 54–74.

Makridakis, Spyros, Evangelos Spiliotis, and Vassilios Assimakopoulos (2018). "The M4 Competition: Results, findings, conclusion and way forward". In: *International Journal of Forecasting* 34.4, pp. 802–808.

Martínez-García, María et al. (2023). "Sleep Activity Recognition and Characterization from Multi-Source Passively Sensed Data". In: *arXiv preprint arXiv:2301.10156*.

Measurement and Forecasting Group (2006). *Solar Power Data for Integration Studies*. URL: `https://www.nrel.gov/grid/solar-power-data.html`.

Merkuryeva, Galina, Aija Valberga, and Alexander Smirnov (2019). "Demand forecasting in pharmaceutical supply chains: A case study". In: *Procedia Computer Science* 149, pp. 3–10.

Moreira, Alan and Tyler Muir (2017). "Volatility-managed portfolios". In: *The Journal of Finance* 72.4, pp. 1611–1644.

Moreno-Pino, Fernando (July 2021). *fmorenopino/SAAM: First code release of 'Deep Autoregressive Models with Spectral Attention'*. Version v1.0. URL: `https://doi.org/10.5281/zenodo.5086179`.

Moreno-Pino, Fernando, María Martínez-García, et al. (2022). "Heterogeneous Hidden Markov Models for Sleep Activity Recognition from Multi-Source Passively Sensed Data". In: *arXiv preprint arXiv:2211.10371*.

Moreno-Pino, Fernando, Pablo M Olmos, and Antonio Artés-Rodríguez (2023). "Deep autoregressive models with spectral attention". In: *Pattern Recognition* 133, p. 109014.

Moreno-Pino, Fernando, Emese Sükei, et al. (2022). "PyHHMM: A Python Library for Heterogeneous Hidden Markov Models". In: *arXiv preprint arXiv:2201.06968*.

Moreno-Pino, Fernando and Stefan Zohren (2022). "DeepVol: Volatility Forecasting from High-Frequency Data with Dilated Causal Convolutions". In: *arXiv preprint arXiv:2210.04797*.

Nelson, Daniel B (1991). "Conditional heteroskedasticity in asset returns: A new approach". In: *Econometrica: Journal of the econometric society*, pp. 347–370.

Niu, Zhaoyang, Guoqiang Zhong, and Hui Yu (2021). "A review on the attention mechanism of deep learning". In: *Neurocomputing* 452, pp. 48–62.

Noureldin, Diaa, Neil Shephard, and Kevin Sheppard (2012). "Multivariate high-frequency-based volatility (HEAVY) models". In: *Journal of Applied Econometrics* 27.6, pp. 907–933.

Oord, Aaron van den et al. (2016). "Wavenet: A generative model for raw audio". In: *arXiv preprint arXiv:1609.03499*.

Oreshkin, Boris N et al. (2019). "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting". In: *International Conference on Learning Representations*.

Pang, Yue et al. (2022). "Hierarchical Electricity Time Series Prediction with Cluster Analysis and Sparse Penalty". In: *Pattern Recognition*, p. 108555.

Papantonis, Ioannis, Leonidas Rompolis, and Elias Tzavalis (2022). "Improving variance forecasts: The role of Realized Variance features". In: *International Journal of Forecasting*.

Park, JA, JS Baek, and SY Hwang (2009). "Persistent-threshold-GARCH processes: Model and application". In: *Statistics & Probability Letters* 79.7, pp. 907–914.

Patton, Andrew J (2011). "Volatility forecast comparison using imperfect volatility proxies". In: *Journal of Econometrics* 160.1, pp. 246–256.

Rabemananjara, Roger and Jean-Michel Zakoian (1993). "Threshold ARCH models and asymmetries in volatility". In: *Journal of applied econometrics* 8.1, pp. 31–49.

Rahimikia, Eghbal and Ser-Huang Poon (2020). "Big data approach to realised volatility forecasting using HAR model augmented with limit order book and news". In: *Available at SSRN* 3684040.

Rahimikia, Eghbal, Stefan Zohren, and Ser-Huang Poon (2021). "Realised Volatility Forecasting: Machine Learning via Financial Word Embedding". In: *arXiv preprint arXiv:2108.00480*.

Ramos-Pérez, Eduardo, Pablo J Alonso-González, and José Javier Núñez-Velázquez (2021). "Multi-transformer: A new neural network-based architecture for forecasting S&P volatility". In: *Mathematics* 9.15, p. 1794.

Ranganathan, Priya, CS Pramesh, et al. (2012). "Censoring in survival analysis: potential for bias". In: *Perspectives in clinical research* 3.1, p. 40.

Rangapuram, Syama Sundar et al. (2018). "Deep state space models for time series forecasting". In: *Advances in neural information processing systems* 31, pp. 7785–7794.

Reisenhofer, Rafael, Xandro Bayer, and Nikolaus Hautsch (2022). "HARNet: A convolutional neural network for realized volatility forecasting". In: *arXiv preprint arXiv:2205.07719*.

Rindt, D et al. (2022). "Survival regression with proper scoring rules and monotonic neural networks". In: *25th International Conference on Artificial Intelligence and Statistics (AISTATS 2022)*.

Ríos-Muñoz, Gonzalo R et al. (2020). "Hidden Markov models for activity detection in atrial fibrillation electrograms". In: *2020 Computing in Cardiology*. IEEE, pp. 1–4.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1985). *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science.

Salinas, David et al. (2020). "DeepAR: Probabilistic forecasting with autoregressive recurrent networks". In: *International Journal of Forecasting* 36.3, pp. 1181–1191.

Shen, Ze, Qing Wan, and David J Leatham (2021). "Bitcoin Return Volatility Forecasting: A Comparative Study between GARCH and RNN". In: *Journal of Risk and Financial Management* 14.7, p. 337.

Shephard, Neil and Kevin Sheppard (2010). "Realising the future: forecasting with high-frequency-based volatility (HEAVY) models". In: *Journal of Applied Econometrics* 25.2, pp. 197–231.

Sheppard, Kevin and Wen Xu (2019). "Factor high-frequency-based volatility (HEAVY) models". In: *Journal of Financial Econometrics* 17.1, pp. 33–65.

Shrikumar, Avanti, Peyton Greenside, and Anshul Kundaje (2017). "Learning important features through propagating activation differences". In: *International conference on machine learning*. PMLR, pp. 3145–3153.

Singh, Ritesh and Keshab Mukhopadhyay (2011). "Survival analysis in clinical trials: Basics and must know areas". In: *Perspectives in clinical research* 2.4, p. 145.

Sirignano, Justin and Rama Cont (2019). "Universal features of price formation in financial markets: perspectives from deep learning". In: *Quantitative Finance* 19.9, pp. 1449–1459.

Su, Jung-Bin (2021). "How to promote the performance of parametric volatility forecasts in the stock market? a neural networks approach". In: *Entropy* 23.9, p. 1151.

Susto, Gian Antonio et al. (2014). "Machine learning for predictive maintenance: A multiple classifier approach". In: *IEEE transactions on industrial informatics* 11.3, pp. 812–820.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014). "Sequence to Sequence Learning with Neural Networks". In: *Advances in Neural Information Processing Systems* 27, pp. 3104–3112.

Tagasovska, Natasa and David Lopez-Paz (2019). "Single-model uncertainties for deep learning". In: *Advances in Neural Information Processing Systems* 32.

Tamkin, Alex, Dan Jurafsky, and Noah Goodman (2020). "Language Through a Prism: A Spectral Approach for Multiscale Language Representations". In: *Advances in Neural Information Processing Systems* 33.

Taylor, Sean J and Benjamin Letham (2018). "Forecasting at scale". In: *The American Statistician* 72.1, pp. 37–45.

Treleaven, Philip, Michal Galas, and Vidhi Lalchand (2013). "Algorithmic trading review". In: *Communications of the ACM* 56.11, pp. 76–85.

Van Oord, Aaron, Nal Kalchbrenner, and Koray Kavukcuoglu (2016). "Pixel recurrent neural networks". In: *International conference on machine learning*. PMLR, pp. 1747–1756.

Vaswani, Ashish et al. (2017). "Attention is All you Need". In: *NIPS*.

Vidal, Andrés and Werner Kristjanpoller (2020). "Gold volatility prediction using a CNN-LSTM approach". In: *Expert Systems with Applications* 157, p. 113481.

Wang, Yuyang et al. (2019). "Deep factors for forecasting". In: *International Conference on Machine Learning*. PMLR, pp. 6607–6617.

Wang, Zifeng and Jimeng Sun (2022). "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.

Wei, Lee-Jen (1992). "The accelerated failure time model: a useful alternative to the Cox regression model in survival analysis". In: *Statistics in medicine* 11.14-15, pp. 1871–1879.

Wu, Neo et al. (2020). "Deep transformer models for time series forecasting: The influenza prevalence case". In: *arXiv preprint arXiv:2001.08317*.

Yu, Hsiang-Fu, Nikhil Rao, and Inderjit S Dhillon (2016). "Temporal Regularized Matrix Factorization for High-dimensional Time Series Prediction." In: *NIPS*, pp. 847–855.

Yu, ShuiLing and Zhe Li (2018). "Forecasting stock price index volatility with LSTM deep neural network". In: *Recent developments in data science and business analytics*. Springer, pp. 265–272.

Yuan, Huiling, Guodong Li, and Junhui Wang (2022). "High-Frequency-Based Volatility Model with Network Structure". In: *arXiv preprint arXiv:2204.12933*.

Zakoian, Jean-Michel (1994). "Threshold heteroskedastic models". In: *Journal of Economic Dynamics and control* 18.5, pp. 931–955.

Zhang, Zihao, Bryan Lim, and Stefan Zohren (2021). "Deep learning for market by order data". In: *Applied Mathematical Finance* 28.1, pp. 79–95.

Zhang, Zihao and Stefan Zohren (2021). "Multi-horizon forecasting for limit order books: Novel deep learning approaches and hardware acceleration using intelligent processing units". In: *arXiv preprint arXiv:2105.10430*.

Zhang, Zihao, Stefan Zohren, and Stephen Roberts (2019). "Deeplob: Deep convolutional neural networks for limit order books". In: *IEEE Transactions on Signal Processing* 67.11, pp. 3001–3012.

Zhong, Qixian, Jonas W Mueller, and Jane-Ling Wang (2021). "Deep extended hazard models for survival analysis". In: *Advances in Neural Information Processing Systems* 34, pp. 15111–15124.

Zhou, Haoyi et al. (2021). "Informer: Beyond efficient transformer for long sequence time-series forecasting". In: *Proceedings of AAAI*.

Ziehm, Matthias and Janet M Thornton (2013). "Unlocking the potential of survival data for model organisms through a new database and online analysis platform: S urv C urv". In: *Aging cell* 12.5, pp. 910–916.

Appendices

A

Conditions for an Order Fill

Figure A.1 shows a depiction of a Limit Order Book (LOB). To check if an order is filled or not, a hypothetical order is placed in the book at a certain level or tick distance from the midprice. This is illustrated in Figure A.2.

A limit order is considered filled if certain fill conditions are satisfied. However, if the fill conditions are not met by the time the market closes, the limit order is cancelled and is considered unfilled and right censored by end-of-study censoring. Fill conditions are checked by monitoring the price of new incoming limit orders and market orders after submitting the hypothetical limit order. We proceed to detail these analogously to Maglaras et al. (2021):

1. A new limit/market order:

    - If a new buy/sell order comes at a higher/lower price than our synthetic sell/buy order, then the order is filled.
    - If an order in front of our synthetic order is filled/partially filled, we consider the synthetic order to be filled as well.

2. A new market order comes in at the same price than our synthetic order:

    - If the market order crossed against any of the orders being tracked, then our synthetic order has been filled.

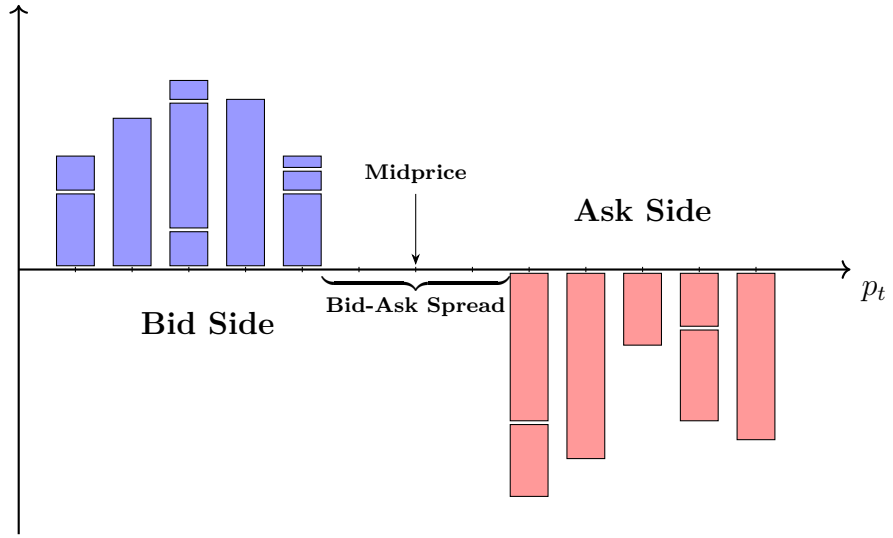

Figure A.1: Example diagram of the top five levels on the ask and on the bid side of a limit order book.

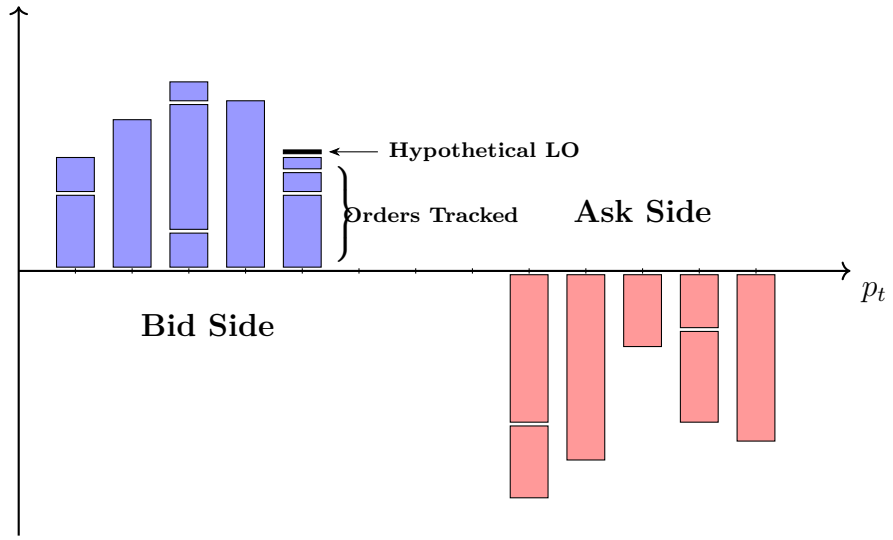

Figure A.2: Example diagram of a hypothetical limit order placed in the first level of the bid side of the order book. To check for an order fill, we track execution messages order for order in front of the in the queue, as well as levels deeper in the book.

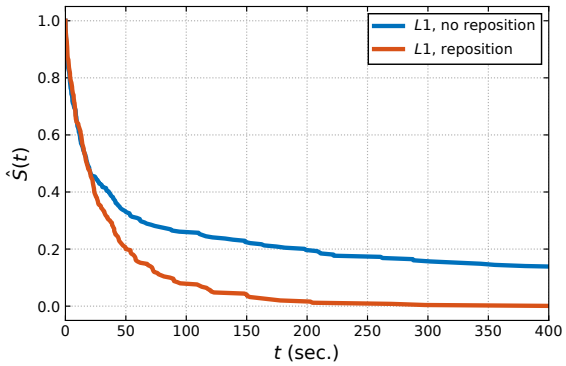

Figure A.3: Kaplan-Meier estimates of orders repositioned at the best level and orders with no repositionwith .

From an implementation perspective, we do not require maintaining a full log of all the orders in the queue using a linked list. It suffices to track the messages associated with orders in front of the hypothetical limit order in the queue and market orders arriving above/below the hypothetical order, depending on the side it had been placed. Figure A.3 shows the difference between the estimates resulting from using or not order re-positioning. Both survival functions show a similar form of decay, and their shapes are consistent with the well-known decrease in fill probability at deeper levels of the order book.

B

Derivation of the Right Censored Log-Likelihood

In consistency with the notation used in previous sections, we assume we have a dataset of observations, $\mathcal{D} = \{(\mathbf{x}_k, z_k, d_k)\}_{k=1}^{N}$, where $z_k = \min\{T_l^k, C_l^k\}$,with random variables T_l and C_l denoting the random fill and cancellation (censoring) times, respectively. The likelihood function is given by:

$$L = f(z_1, d_1, \ldots, z_N, d_N) = \prod_{k=1}^{N} f(z_k, d_k). \tag{B.1}$$

To re-write this equation with respect to the values of the indicator variable d, we first consider the case in which the order is filled, and the event is therefore

observed ($d = 1$). In that case, we have that:

$$\begin{aligned} f(z_k, d_k) &= \mathbb{P}\{Z = z_k, d_k = 1\} = \\ &= \mathbb{P}\{T = z_k, T \leq C\} = \\ &= \mathbb{P}\{T = z_k, z_i \leq C\} = \\ &= \mathbb{P}\{T = z_k\}\mathbb{P}\{z_i \leq C\} = \quad \text{(assuming } T \perp\!\!\!\perp C\text{)} \\ &= f_T(z_k)S_C(z_k). \end{aligned} \tag{B.2}$$

Moreover, if the order is cancelled or is not filled before the end of the trading day (and is therefore censored), we have that $d_i = 0$, which implies:

$$\begin{aligned} f(z_k, d_k) &= \mathbb{P}\{Z = z_k, d_k = 0\} = \\ &= \mathbb{P}\{C = z_k, T > C\} = \\ &= \mathbb{P}\{C = z_k, z_k > C\} = \\ &= \mathbb{P}\{C = z_k\}\mathbb{P}\{z_k \leq C\} = \quad \text{(assuming } T \perp\!\!\!\perp C\text{)} \\ &= f_C(z_k)S_T(z_k), \end{aligned} \tag{B.3}$$

which means that Eq. (B.1) can be rewritten as:

$$\begin{aligned} L &= \prod_{k=1}^{N} \Big[f_T(z_k)S_C(z_k)\Big]^{\delta_k}\Big[f_C(z_k)S_T(z_k)\Big]^{(1-\delta_k)} = \\ &= \prod_{k=1}^{N} \Big[f_T(z_k)^{\delta_k}S_T(z_k)^{(1-\delta_k)}\Big]\Big[f_C(z_k)^{(1-\delta_k)}S_C(z_k)^{\delta_k}\Big]. \end{aligned} \tag{B.4}$$

In most cases, we are concerned with the estimation of $S_T(t)$ and not $S_C(t)$, since $S_C(t)$ contains information related to the censoring mechanism. Only $S_T(t)$ contains the information about the filltime, which is the variable of interest. So, the terms that do not involve T are considered constant, and the likelihood is proportional to:

$$L = \prod_{k=1}^{N} f_T(z_k)^{\delta_k}S_T(z_k)^{(1-\delta_k)}. \tag{B.5}$$

Finally, taking the log we obtain:

$$\mathcal{L} = \log(L) = \sum_{k=1}^{N} d_k \log(\hat{f}(z_k|\mathbf{x}_k)) + (1 - d_k)\log(\hat{S}(z_k|\mathbf{x}_k)). \tag{B.6}$$

This proof empirically validates the independence assumption between the censoring mechanism and the random filltime, used to derive the expression for the right-censored log-likelihood in Eq. (B.5). This was done for all the tested levels of

the order book, calculating the Pearson correlation coefficient between T and C. The results are reported in Table B.1.

Table B.1: Pearson correlation coefficient between filltime, T, and censoring time, C, for all levels of the order book.

	L1	**L2**	**L3**	**L4**	**L5**
$\rho_{T,C}$	-0.003	-0.012	-0.015	-0.016	-0.017

C

Validating Assumptions of Survival Analysis in the LOB

In this section, we scrutinise the reliability of some of the implicit assumptions of this work. The first of those assumptions concerns censored events. Censored data helps in the estimation of a survival function because the fact that an order has not been filled before being censored also provides information (Lawless, 2011). However, censoring in survival analysis should be "non-informative", which means that censoring distribution is independent of the time-to-event distribution (Leung et al., 1997), as "informative" censoring has been shown to yield biased results (Ranganathan, Pramesh, et al., 2012).

It is important to validate the independence assumption mentioned above due to the high proportion of cancellations present in LOBs. Figure C.1 reports the number of fills and cancellations (as well as their proportion) per level for data collected from AAPL LOB between September 2012 and September 2014. As expected, due to the rules of matching (price-time priority), the largest proportion

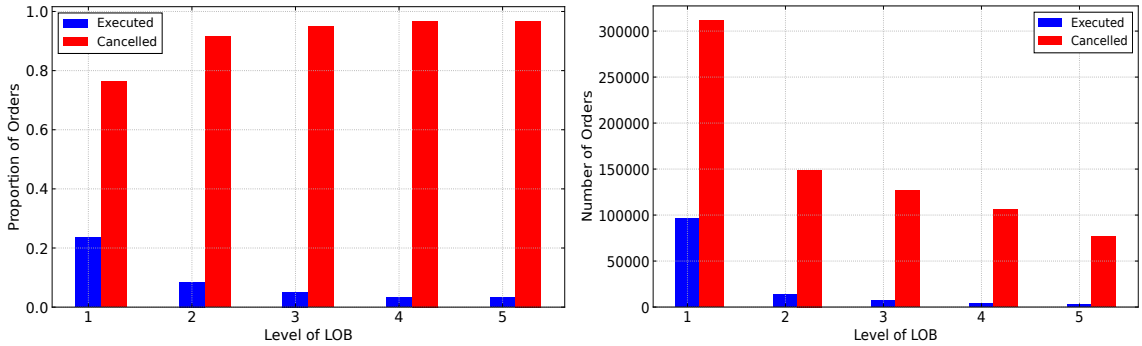

Figure C.1: Number (left) and proportion (right) of orders executed and cancelled at different levels of the AAPL LOB between 2012 and 2014.

of filled orders occurs in the first level. As we go deeper into the book, most orders are cancelled. This is well known in the literature, see Cartea, Jaimungal, et al. (2015). It entails that survival functions estimated at deeper levels of the LOB will be subject to a large censoring proportion.

As previously mentioned, this study is restricted to analysing "terminal" events. However, larger-size LOs in the book might be "partially filled" or "partially cancelled" before reaching the final terminal state. Modelling these intermediate events in survival analysis is not trivial. Recent work (Groha et al., 2020) suggests using intermediate state representations modelled with a Markov process to account for non-terminal events, such as partial fills. However, the difficulty of describing the transition probabilities of such a model while allowing a flexible use of the covariates is challenging. This, together with the computational expense associated

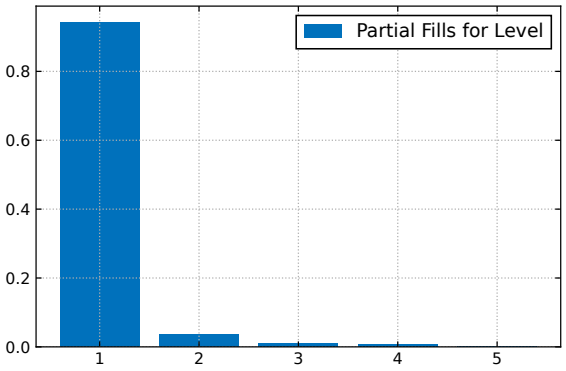

Figure C.2: Number of partial fills required for full order execution when considering fills at different levels.

with the method proposed in Groha et al. (2020), discourages its use in this initial study of survival analysis in the order book and it is left for future work. However, Figure C.2 shows the proportion of partial fills needed before the complete execution of an order. Since more than 98% of orders are filled against a single market order, we consider this assumption to be satisfactory.

D

Typical Survival Models and Scoring Rules

An initial approach to estimate the survival function is to use a Kaplan-Meier estimate (Kaplan and Meier, 1958), which is given by:

$$\hat{S}(t) = \prod_{i:t_i \leq t} \left(1 - \frac{d_i}{n_i}\right), \tag{D.1}$$

where t_i is the time when at least one event occurred, d_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. We present a series of methods that approximate survival functions. A follow-up model which conditions on a feature vector is the Cox Proportional-Hazards model (Cox, 1972), where the hazard rate follows:

$$h(t|\mathbf{x}) = h_0(t)\exp(\boldsymbol{\beta}^T\mathbf{x}), \tag{D.2}$$

where $\boldsymbol{\beta}$ are the 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 are chosen to maximise

$$\mathcal{L}(\boldsymbol{\beta}) = \prod_{d_i=1} \frac{\exp(\boldsymbol{\beta}^T\mathbf{x})}{\sum_{j:t_j \geq t_i} \exp(\boldsymbol{\beta}^T\mathbf{x})}. \tag{D.3}$$

Another popular model used in survival analysis is the Accelerated Failure Time model (Wei, 1992), in which the hazard rates are determined by:

$$h(t|\mathbf{x}) = \phi(\boldsymbol{x})h_0(\phi(\boldsymbol{x})t), \tag{D.4}$$

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 is often violated. This has recently motivated the extension of the Cox model using deep learning in order to capture non-linearities between features of interest (Katzman et al., 2018; Kvamme et al., 2019) while maintaining the structure of the Cox model:

$$h(t|\mathbf{x}) = h_0(t)\exp(f_{\boldsymbol{\theta}}(\mathbf{x}, t)). \tag{D.5}$$

More recent work (C. Lee, Zame, et al., 2018b; C. Lee, Yoon, et al., 2019; Rindt et al., 2022) focuses on directly learning the survival function conditioning on input features:

$$S(t|\mathbf{x}) = f_{\boldsymbol{\theta}}(\mathbf{x}, t), \tag{D.6}$$

which is also the approach we take.

Commonly used scores for survival functions include time-dependent concordance (Antolini et al., 2005), which is defined as:

$$\begin{aligned} C_{td} &= \mathbb{P}[\hat{S}(z_i|\mathbf{x}_i) < \hat{S}(z_i|\mathbf{x}_j)|z_i < z_j, d_i = 1] \approx \\ &\approx \frac{\sum_{i=1}^{N}\sum_{j=1;i\neq j}^{N}\mathbb{I}[\hat{S}(z_i|\mathbf{x}_i) < \hat{S}(z_j|\mathbf{x}_j)]\pi_{ij}}{\sum_{i=1}^{N}\sum_{j=1;i\neq j}^{N}\pi_{ij}}, \end{aligned} \tag{D.7}$$

where π_{ij} is an indicator of the pair (i, j) being amenable for comparison, i.e., if the pair is "concordan". 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 an 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:

$$\beta = \frac{\hat{S}(t|x)^2 \mathbb{I}\{z \leq t, d = 1\}}{\hat{G}(z)} + \frac{(1 - \hat{S}(t|x))^2 \mathbb{I}\{z > t\}}{\hat{G}(z)}, \tag{D.8}$$

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 (C. Lee, Zame, et al., 2018b; C. Lee, Yoon, et al., 2019; Zhong et al., 2021). However, recent work (Rindt et al., 2022) shows that the previously presented scoring rules (as well as a number of others) are actually improper, meaning that they can potentially give higher scores to wrongly fitted distributions [1]. Right-censored log-likelihood is a proper scoring rule, with the key result in (Rindt et al., 2022) showing that:

$$\begin{aligned} &\mathbb{E}[\mathcal{S}(S, (z, d)) - \mathcal{S}(\hat{S}, (z, d))) \mid C] = \\ &F(C)\mathrm{KL}(f(t)/F(C) \| \hat{f}(t)/\hat{F}(C)) + \mathrm{KL}(\mathrm{Ber}(S(C)) \| \mathrm{Ber}(\hat{S}(C))) \geq 0, \end{aligned} \tag{D.9}$$

where $\mathrm{KL}(\cdot)$ denotes the Kullback-Leibler divergence, $\mathrm{Ber}(\cdot)$ is the Bernoulli distribution, C is the random censoring time, and F denotes the probability mass function.

[1] 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.

E

Extending the Encoder's Dilated Causal Convolutional Neural Network to L layers

If we desire to extend the original encoder's DCC in charge of obtaining the corresponding queries, keys, and keys in the proposed convolutional-Transformer, the causal convolutional network would be modified as follows. The first layer would perform the same operation, convoluting the input sequences x and the kernel k, which can be defined as follows:

$$F^{(l=1)}(t) = \left(x *_d k^{(l=1)}\right)(t) = \sum_{\tau=0}^{s-1} k_\tau^{(l=1)} \cdot x_{t-d\tau}, \tag{E.1}$$

being d the dilation factor and k the filter, with size $s \in \mathbb{Z}$. For each of the rest l-th layers, we could define the convolution operation as:

$$F^{(l)}(t) = \left(F^{(l-1)} *_d k^{(l)}\right)(t) = \sum_{\tau=0}^{s-1} k_\tau^{(l)} \cdot F_{t-d\tau}^{l-1}(t). \tag{E.2}$$

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

$$k^{(l)} : \mathbb{R}^{N_l} \longrightarrow \mathbb{R}^{N_{l+1}}, 1 \leq l \leq L. \tag{E.3}$$

The previous equation shows how, through the usage of residual connections, firstly proposed in He et al. (2016), the encoder's convolutional network could connect l-th 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 model.