Chapter Four

4 Algorithmic Trading Strategy Identification & Market Quality Impact using Unsupervised Learning & Axiomatic Feature Attribution

This chapter identifies, clusters and assesses the impact of different algorithmic trading (AT) strategies on UK equity market quality. Three separate procedures are performed to lineate the connection between algorithmic trading strategies identified in Chapter 2 and the optimised Deep Neural Network (DNN) model developed in Chapter 3 and trained on the FTSE100 dataset defined in Section 3.3, before deriving a conclusion regarding how different AT strategies impact market quality. First, internal dynamics of the optimised DNN trained for each of the 802 ISIN-Date-Trader tuples, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$, derived from the FTSE100 datasets are deconstructed by performing an axiomatic feature attribution decomposition to understand the relative saliency of each element of the n-dimensional limit order book (LOB) input feature vector $\mathbf{x}$. A total of 323 LOB features are generated and holistically explained in Appendix A. Integrated Gradients (Sundararajan, 2017) and DeepLIFT (Shrikumar, 2017) are backpropagation-based feature attribution algorithms that provide insightful information regarding the relative saliency metric, $\mathcal{FI}$, for the all feature inputs into a brokers DNN model. An assumption of this thesis is that features of the state and dynamics of the LOB serve as inputs into AT algorithms, with those DNN features that have higher feature importance representing core inputs into that firms' trading algorithm. Next, unsupervised machine learning methods are employed to cluster each Broker tuple into one of five AT strategies – Automated Market Makers (AMM), Execution, Microstructural, Momentum and Technical strategies - based on the feature importance values, $\mathcal{FI}$. Unsupervised machine learning algorithms infer functions or structural patterns in data without the aid of a response variable or class label in the observations. Identifying AT strategies requires an understanding of the financial and LOB features that drive the behaviour and actions on the LOB of firms conducting those strategies. Due to the additive properties of Integrated Gradients and DeepLIFT, raw features can be combined to build deeper levels of abstraction. This allows for AT strategies to be segmented at both the raw feature level and from a combined set of features. Various extensions of the K-Means unsupervised machine learning technique are utilised, with focus on Spherical K-Means (Buchta, 2012), to partition individual AT firms into interpretable, exclusive and distinct 'strategy' clusters based on results extracted from the feature attribution analysis and derived from the DNN for each tuple. Finally, a comprehensive analysis of the aggregated impact of each AT strategy on UK equity market quality is performed.

Accelerated adoption of algorithmic and high-frequency trading systems, technology and trading methods by market participants provides exigencies for regulators and academics to understand the dynamics of this new trader ecosystem and potential risks flowing into the real economy. Quantifying the impact emanating from interactions and interdependencies between different AT strategies first requires a model for identifying and categorising distinct market participants. AT is commonly recognised under the nom de guerre of 'high-frequency trading' (HFT), a label that has garnered scrutiny within regulator, investor and academic domains. HFT firms have been identified by the Securities and

Exchange Commission (SEC) as proprietary firms with co-located servers executing highly 'sophisticated' and 'high-speed' order submissions, routing and execution strategies over extremely low latency algorithmic trading systems.[21] This chapter provides no distinction between AT and HFT firms, rather *all* AT market participants are differentiated based on their *trading strategy* as opposed to the *characteristics* of the individual *trading firm* to allow for a holistic representation of the trader ecosystem. Untangling the complexities of the trader environment by extracting the various algorithmic trading strategies being conducted allows regulators to more accurately understand how markets function in their contemporary iteration. Implementing this new paradigm of the trader ecosystem provides the tools for policymakers to develop prescriptions using an empirical and diagnostic approach at the *strategy* level of the market, rendering suppositions based on broad generalisations of traders, such as HFT, irrelevant.

Machine learning models of the relationship between LOB features and trader actions have been deployed in this chapter to categorise traders into one of five individual algorithmic trading strategies to analyse the individual impact of these strategies on market quality. Previous academic literature has approached the task of segmenting market participants using various techniques, specifically referred to as manual, quantitative and machine learning classification methods. Several studies have focussed on understanding the behaviour of market participants within the trading ecosystem by classifying firms using a *manual* exchange or regulator identifier. Brogaard (2014) uses a flag developed by NASDAQ to categorise firms that deploy HFT strategies. Hendershott (2013) similarly uses an exchange-identified classification of orders placed by an automated system versus those placed by human traders. Benos (2012) and Aquilina (2016) utilise a similar FCA proprietary dataset to that utilised in this chapter, though they use a list of HFTs maintained by the regulator to classify firms. A second strand of identification techniques have used *quantitative* analysis of a firms' descriptive statistics to develop a feature space from which classification is made using high-level rule-based heuristics. Kirilenko (2017) and Baron (2017) implement a more complex descriptive statistic-based categorisation model using audit trail data that classifies individual trading accounts as HFT, market-makers, fundamental buyers and sellers, opportunistic traders and small traders. The authors use various intra-day metrics for categorisation including inventory positions at close, net holdings through the trading day and trading direction. Hasbrouck (2013) uses a measure of fast trading in the millisecond environment, referred to as 'strategic runs', as a proxy for HFT activity. Australian Securities and Exchange Commission (ASIC) (2013) has adopted a quantitative approach that identifies firms as engaging primarily in HFT activity based on a scoring system comprising several metrics, including the speed of messages, order-to-trade ratios, end of day positions and volume-weighted holding times. Finally, a recent strand of research has begun using *machine learning* models to segment traders. Yang (2015) model trading behaviour as a Markov Decision Process (MDP) and utilise Gaussian-based inverse reinforcement learning (IRL) and dynamic programming (DP) to solve for the optimal reward function of individual market participants that serves as a proxy for their decision-making process and actions. Their Gaussian-based IRL model infers the reward function for traders conducting a strategy based on their actions given the state and inherent market dynamics. This chapter extends the machine learning model literature by classifying algorithmic traders using a trained DNN and axiomatic feature attribution technique that extracts the

[21] Securities Exchange Act Release No. 34-61358, 75 FR 3594, 3606 (January 21, 2010).

relative importance of LOB algorithmic inputs of those traders which is utilised to classify firms through an unsupervised machine learning approach.

The structure of this chapter is as follows. Section 4.1 introduces feature attribution techniques employed to analyse the internal dynamics of the optimised DNNs trained in the previous chapter. Feature importance values are then quantified for each ISIN-Date-Trader tuple, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$. Section 4.2 involves an evaluation of backpropagation-based feature attribution methods to determine the optimal choice of what method to apply to derive importance metrics for each the data tuple. This dataset provides input into the Spherical K-Means clustering algorithm employed in Section 4.3 to segment firms by the AT strategy they deploy. Section 4.4 then assesses the aggregate impact of AT strategies on UK equity market quality.

4.1 Axiomatic Attribution for Limit Order Book Features

This section analyses the internal dynamics optimised Deep Neural Networks (DNN) trained on the 802 unique FTSE100 CLOB ISIN-Date-Broker tuples, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$, which are developed in the previous chapter of this thesis. The objective of this section is to compute feature attribution metrics, $\mathcal{FI}$, for each of the n-dimensional microstructural LOB feature input values, $\mathbf{x}$, that serve as inputs into the DNN models. Axiomatic feature attribution analysis is applied to each DNN by projecting the success or failure of predicting the correct classifier, approximated by the DNNs non-linear mapping function $\mathbf{y} = f(\mathbf{x}, \mathbf{w}, \mathbf{b})$, onto the n-dimensional microstructural LOB feature space, $\mathbf{x} \in \mathcal{F}$. Quantifying feature attribution metrics, $\mathcal{FI} = [\mathbf{r}_1, \dots, \mathbf{r}_n] \in \mathbb{R}^n$, for each of the n features in vector $\mathbf{x} = [\mathbf{x}_1, \dots, \mathbf{x}_n] \in \mathbb{R}^n$, allows for a determination of the strategy being conducted by an algorithmic trader based on the relative contribution of certain features towards making correct predictions of the traders actions executed on the LOB, $\mathbf{y} \in \mathcal{A}$.

An assumption of this thesis is that LOB feature manifestly represents algorithmic inputs into a firm's trading strategy. Intuitively, there exists an inextricable connection between the state of the LOB environment that an algorithmic trader exists within, $\mathbf{x}$, and the underlying strategy for conducting profitable actions, $\mathbf{y}$, in that environment. Certain systemic elements of the LOB environment drive traders' behaviour more than others. Identifying the components of participants' trading algorithms serves as a reference point for using an unsupervised learning algorithm employed in Section 4.3 to cluster traders into one of five trading strategies – Automated Market Making (AMM), Execution, Microstructural, Momentum and Technical trading strategies. Once feature attribution values for each Broker tuple have been quantified and traders clustered, the aggregate impact of different AT strategies on various elements of market quality can then be inferred.

The ubiquitous nature of neural networks in the deep learning community is founded on and driven by the strong theoretical principles (Bengio, 2012), computational efficiency and high performance (He, 2015) of these models. The successful praxis of deep learning techniques across various domains to attain these state-of-the-art results in explaining complex and abstract representations has been coupled with significant innovation in the internal components of DNNs. However, practitioners and academics in the broader domain of statistics and mathematics have retained a degree of trepidation as to the utility of these models given their reputation as a 'blackbox' machine learning model. A significant contention with DNNs is their lack of transparency and interpretability, hindering practitioners' comprehension of the synaptic connections between inputs and outputs which serves as an impediment to the mainstream adoption of machine intelligence (Mikolov, 2016). *Feature attribution methods* have the potential to bridge the disparity between interpretability and performance innate to DNNs, allowing practitioners to interact with and understand the traditionally opaque internal dynamics of the model. This potential is being driven by the increasing application of feature attribution methods in the deep learning literature used to better explain the relationship between inputs and predictions in neural networks (Shrikumar, 2017; Sundararajan 2017; Ancona, 2018).

Methods of feature attribution, also referred to as feature importance, are concerned with quantifying the relative importance of individual elements comprising the input feature vector, $\mathbf{x}$, depending on how they improve the classification system of the DNN. Application of certain attribution methods allow for the identification of the complex interdependencies that bind features within the classification system. They also help understand why an accurate prediction of the output class is made using easily interpretable importance metrics. Statistical representations of feature importance can also be deployed as a tool for feature selection by diagnosing the contribution of various input feature elements, then selecting a feature subset of the most 'important' features that improve performance (Heaton, 2017b).

The first part of this section explains the 323-dimensional LOB feature space utilised in the previous chapter to train the non-linear DNN models with these features also defined in groups based on the relevant strategy they are assumed to most commonly serve as algorithmic inputs into. The objective of this section is to develop a feature attribution space, $\mathcal{FI} \in \mathbb{R}^{323}$, for each DNN Broker tuples, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$. To do so we next define, explore and then implement several methods for attributing feature saliency for input vector elements of the optimised FTSE100 CLOB DNN tuples $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$. Attribution methods are categorised into correlation-based methods, synaptic weight methods, input perturbation approaches and backpropagation-based approaches. Two input perturbation approaches are analysed including occlusion methods (Heaton, 2017b; Zeiler, 2014), and the Locally Interpretable Model-Agnostic approach (Ribeiro, 2016). Furthermore, four backpropagation-based approaches are considered including Layer-Wise Relevance Propagation (Bach, 2015), Deep Taylor Decomposition (Montavon, 2017), DeepLIFT (Shrikumar, 2017) and Integrated Gradients (Sundararajan, 2017).

4.1.1 Feature Grouping

The 323-dimensional LOB feature space employed to train the non-linear DNN mapping functions and projected onto the corresponding feature attribution space, $\mathcal{FI} \in \mathbb{R}^{323}$, provides critical information about what motivates a trader to perform a particular action, $\mathbf{y} \in \mathcal{A}$. The high dimensionality of the dataset requires the implementation of a structural framework to relate feature importance metrics to trading strategies. Section 2.3 introduced five primary algorithmic trading strategies deployed on UK equity markets – Automated Market Makers (AMM), Execution, Microstructural, Momentum, and Technical strategies. The list of relevant LOB features that drive these strategies, derived from literature and domain knowledge, is also explained in Section 2.3, Section 2.4 and holistically defined in Appendix A. In addition, several sub-strategies are developed to partition algorithmic trading strategies into even more interpretable categories. Thus, raw features, F, have an attached strategy, S, and sub-strategy, U, with computed importance metrics, $\mathcal{FI}$, for each individual $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$ tuple. Raw features with similar attributes and properties are collected into the higher-level strategy segments to deconstruct the levels of abstraction between complex features and higher-order trading strategies. This allows for strategy identification and clustering to be performed across multiple dimensions – raw features, sub-strategies and high-level algorithmic trading strategies – whilst maintaining a parsimonious model that allows for easier interpretability of what features drive AT firms' actions and explain specific AT strategies.

The lowest-order category is the set of 323 raw LOB features, F, which act as input data into the DNN. The middle category, sub-strategies U, amalgamate raw features that have similar properties to provide a more interpretable label that when grouped together explains an important LOB sub-strategy that may be executed as part of a higher-order AT strategy, S. For example, raw features (F) feed into Informed Trading sub-strategy (U) which itself feeds into the overarching Automated Market Making strategy (S). Raw features attempt to explain the level of informed trading on the LOB. A subset of features in the Informed Trading sub-strategy category is presented in Table 4.1.1. Traders with high feature attribution values in this category can be seen as having their algorithms, which decide the correct action to take, being influenced significantly by dynamic measures of informed trading in the market. This sub-strategy combines raw features that relate to the probability of informed trading metric, PIN, and the volume-probability of informed trading metric, VPIN, which are built using various lags, techniques and hyper-parameter variables.

Strategy (S)	Sub-strategy (U)	Raw Feature (F)
AMM	Informed Trading	PINT.10
AMM	Informed Trading	PINT.100
AMM	Informed Trading	PINM.10
AMM	Informed Trading	PINM.100
AMM	Informed Trading	VPIN.BVC.10
AMM	Informed Trading	VPIN.BVC.20
AMM	Informed Trading	VPIN.TI.10
AMM	Informed Trading	VPIN.TI.20
AMM	Informed Trading	PINT.10

Table 4.1.1 – Example of Feature Grouping methodology for the sub-strategy $(U)$of Informed Trading.

By way of further explanation, the five primary AT strategies defined in Section 2.3 are represented in the first column of the feature list. These strategies encompass several sub-strategies which in turn have raw features collapse into each segment. For example, stochastic oscillators are technical indicators used by firms executing predominately Technical signal-based AT strategies. Thus, the raw features of stochastic oscillators, such as the previous fifteen event period oscillator, SO.15, are incorporated into the stochastic oscillator sub-strategy and ultimately within the Technical AT strategy category. This schematic is presented in Table 4.1.2. The objective of group categorisation is to provide interpretability of how individual raw feature attribution values, $\mathcal{FI}$, can be used ultimately to cluster trading firms into one of the five primary AT strategy categories.

Strategy (S)	Sub-strategy (U)	Raw Feature (F)
Technical	Stochastic Oscillator	SO.E15

Table 4.1.2 – Example of Feature Grouping methodology for the raw feature value (F) of SO.E15.

4.1.2 Distance Correlation & Mutual Information Methods

Understanding the univariate relationship between the FTSE100 CLOB n-dimensional feature set and actions executed on the LOB can be performed using distance correlation and mutual information metrics. These metrics provide a low-computational basis for understanding what features drive activity on modern LOBs, though they are characterised by their parochial focus on only quasi-linear relationships and their treatment of features independently, ignoring interaction effects between them.

Pearson correlation coefficients, $\rho_{\mathbf{x}_i}$, are a metaheuristic filter method commonly deployed as a feature importance metric in machine learning models. The coefficients measure the linear correlation between elements of the DNN input feature vector over the training data, $\mathbf{x}_i$, and their corresponding classifier, $\mathbf{y}$ (Biesiada, 2007). Whilst correlation between input features and classifiers serves as a relatively useful heuristic for determining the importance of a feature to predicting model outputs, it ignores both non-linear relationships with the classifier and the potential improvement in classification accuracy from abstract interactions between multiple features in the feature vector. An extended univariate filter technique capable of measuring quasi-non-linear relationships is distance correlation, $d\rho_{\mathbf{x}_i}$, which is a function of distance covariance, $dcov_{\mathbf{x}_i,\mathbf{y}}$, and the distance standard deviation for both the feature, $d\sigma_{\mathbf{x}_i}$, and classifier, $d\sigma_{\mathbf{y}}$ (Szekely, 2007):

$$\rho_{\mathbf{x}_i} = \left|\frac{cov_{\mathbf{x}_i,\mathbf{y}}}{\sigma_{\mathbf{x}_i}\sigma_{\mathbf{y}}}\right| \qquad d\rho_{\mathbf{x}_i} = \left|\frac{dcov_{\mathbf{x}_i,\mathbf{y}}}{d\sigma_{\mathbf{x}_i}d\sigma_{\mathbf{y}}}\right|$$

Absolute distance correlations for all features are evaluated across the FTSE100 CLOB dataset for each unique tuple, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$. Distance correlations are then averaged across all datasets to form importance metrics at the individual feature-level and additionally for feature 'sub-strategy groups'. Raw features are grouped by the relevant sub-strategy and strategy to which they would most likely serve as an algorithmic input into. A holistic explanation and rationale of how the feature grouping procedure is offered in the previous part of this section, which additionally explains nomenclature used in the chapter regarding feature groupings.

Results for the correlation analysis is presented in Figure 4.1.1 with a focus on the correlation between 20 interesting feature groups and two key actions – orders at the best bid, LO.B.P0, and bid market orders, MO.B. The first point to note is that four of the five most correlated feature groups relate to an individual algorithmic trading strategy. For example, momentum features are a key component of Momentum-based strategies, quote offset for AMM, quoted spread for Microstructural and trader spread for Execution type strategies or algorithms, with all four features having an average correlation higher than 0.1 for the actions analysed. This indicates that the task of clustering firms by trading strategies executed based on the importance of individual features to their trading algorithms may be possible. As expected, raw features that are components of limit order imbalance, order flow, implementation shortfall and RSI feature groups also have high correlations given their prevalence in the trading literature and the ubiquity across the corporate trading environment.

The divergence between correlation values for limit orders, represented by red dots, and market orders, represented by blue dots, is also studied. Results indicate that most of the feature group values are more strongly correlated with limit orders which aligns with the findings from the confusion matrix analysis in Section 3.6 where the market order actions were on average more difficult to predict for the DNN than limit orders. Latency arbitrage features deviate from this prescribed view as market orders have a 2.5 times higher correlation than limit orders on average, with similar results for volume traded though at a smaller magnitude. These results are in line with expectations given that latency arbitrage features relate to how traders react when price dynamics across trading venues are out of equilibrium. Note that 'arbitrage' in this context is a slight misnomer as it does not relate to explicit risk-free profit but rather an equalisation or crossing of the best bid and ask when considering the holistic UK equity market. Thus, when latency arbitrage feature values are high, there will likely be a significantly higher probability that traders will execute market orders on one or both sides of multiple exchange LOBs to benefit from discrepancies in best bid and ask prices across venues.

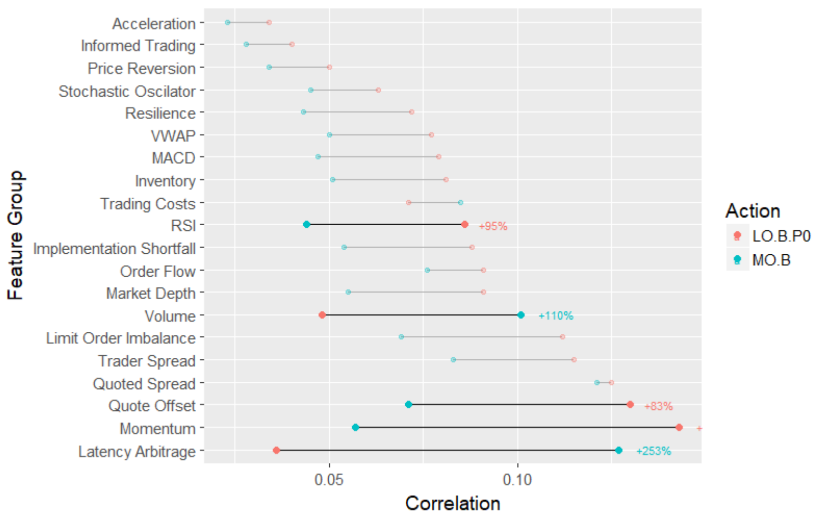

Figure 4.1.1 – Distance correlation between trader actions and feature groups. Actions are bid limit at BBO (red) and market orders (blue).

An alternative filter-based feature importance proxy takes an information theoretic approach to develop mutual information (MI) indicators, $MI_{\mathbf{x}_i,\mathbf{y}}$, for each feature-classifier pair, $\mathbf{x}_i,\mathbf{y}$. MI indicators for the FTSE CLOB dataset are measured in entropy units, $H(\mathbf{x}_i)$, which is a quantitative expression for the pervading presence of a specific input feature value within the probability distribution $P(\mathbf{x}_i)$. Therefore, if the distribution of $\mathbf{x}_i$ is biased towards a certain value $a \in \mathbf{x}_i$, then the entropy for the feature distribution is low, as there is minimal uncertainty as to its value. Alternatively, a uniform distribution of the feature values will result in a higher entropy value. For feature importance purposes, the metric $MI_{\mathbf{x}_i,\mathbf{y}}$ is developed for each feature, $\mathbf{x}_i$, which can be seen as a measure of the level of information that is shared between the input feature and output values, $\mathbf{y}$ (Estevez, 2009). This requires the calculation of the Kullback-Leibler divergence between the entropy of the input feature, $H(\mathbf{x}_i)$, or the level of uncertainty prior to knowing the output value, and the conditional entropy of the input feature value given the output value, which is the residual uncertainty once the output value is revealed, $H(\mathbf{x}_i|\mathbf{y})$. High MI values indicate features with similar distributions to, or shared information with, the classifier, acting as a proxy for higher feature importance. Given the continuous nature of features in the FTSE100 CLOB dataset, it is necessary to discretise the features into bins to approximate densities (Guyon, 2003).

$$MI_{\mathbf{x}_i,\mathbf{y}} = H(\mathbf{x}_i) - H(\mathbf{x}_i|\mathbf{y}) = -\sum P(\mathbf{x}_i,\mathbf{y}) \ln\left(\frac{P(\mathbf{x}_i,\mathbf{y})}{P(\mathbf{x}_i)P(\mathbf{y})}\right) \qquad H(A) = -\sum\nolimits_{\boldsymbol{a}\in\boldsymbol{A}} P(a)\ln\big(P(a)\big)$$

MI metrics to measure feature importance are presented in Figure 4.1.2 for 20 features averaged over each unique dataset, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$, and feature group. The same graphical framework and trader actions are employed as in the correlation analysis. Results differ slightly from the correlation analysis, with several new features included. Trader depth attains the highest feature attribution of 0.42 for limit orders, with latency arbitrage the highest for market orders with a MI value of 0.39. The trader depth feature relates to the level of current and lagged market depth for the individual trader executing the action, and evidently, the probability distribution is closely related to limit orders at the best bid. As an example, when a trader is performing an execution strategy to trade a fixed volume of shares, one would expect them to update their orders with a new limit order when their relative depth position, that is their current quantum of orders at the best bid, is low relative to the market. This could occur after another large order is placed in the queue or an opposite market order has been executed, oscillating the best bid-ask prices, though either way, the trader will seek to maintain their position in the best bid queue to complete the execution strategy. Thus, the relative trader depth becomes an important feature for execution-type algorithmic traders. Additionally, order arrival rates and aggressiveness levels attain high $MI_{\mathbf{x}_i,\mathbf{y}}$ values of 0.38 and 0.35 for limit orders, respectively. Once more, this aligns with financial logic and order flow theory (Toth, 2015), especially in high-frequency markets, that traders will follow a herding mentality and place orders at the BBO as order arrival rates increase given the competitive requirement of maintaining a high position in the order queue. Furthermore, higher aggressiveness levels may proxy for amplified levels of informed trading on the LOB, with informed traders placing orders at the BBO to profit from the demand for more immediate liquidity (Andersen, 2013; Ait-Sahalia, 2017).

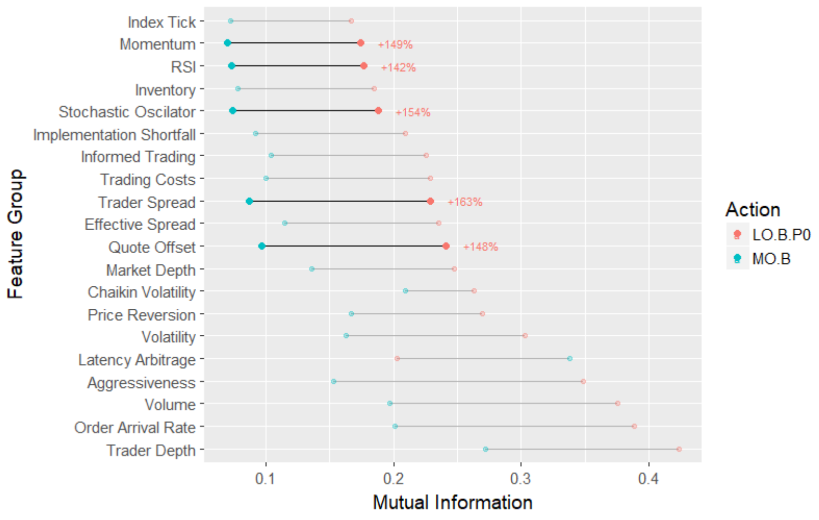

Figure 4.1.2 – Mutual Information metrics based on entropy and cross-entropy values for features and actions. Measured in entropy units. Actions are bid limit orders at the BBO (red) and market orders (blue).

Both correlation and mutual information metrics provide a genesis non-model-based heuristic method for determining what LOB features may be important to predict trader actions. The advantage of these methods is their interpretability and comparability, with values scalable allowing them to be normalised within a specific range, such as [0,1], with higher values indicating higher feature importance. Furthermore, the measures are stable given their independence from the model, equalling the same value regardless of what type of DNN is fitted. However, both these measures are a form of univariate feature importance analysis as they test the statistical relationship between only the feature and the output classifier, with feature importance considered independently of other features. This provides minimal utility for truly attributing feature importance to traders given the critical role that complex feature interdependencies and interactions play in predicting AT firms' actions on modern LOB trading systems.

4.1.3 Synaptic Weight-based Methods

Synaptic Weights are an embedded feature attribution technique that can be used to derive the saliency measures of the FTSE100 CLOB dataset features implicitly from the internal dynamics of the DNN. Specifically, the dynamics of the model investigated are the synaptic weight connections between input feature neurons and the connecting neurons in a feedforward direction towards the output layer. However, weight-based algorithms developed by Garson (1991) with extended iterations developed by Goh (1995) and Olden (2004) among others, are pertinent for an analysis of feature importance in single-layer neural networks but offer minimal application in deeper neural networks. Feature importance for input features is measured by deconstructing the synaptic connections in the neural network and applying operations to the weight values to derive importance. This section employs the Olden (2004) method that sums all absolute weights, $w_{i,j}$, connecting the input feature $\mathbf{x}_i$ to the output neuron $\mathbf{y}$, through all hidden layers, to derive a single feature importance value, $W_{\mathbf{x}_i}$. The logic of this method is that weights convey relevant information regarding the utility of a feature in explaining an output and can be seen as partially analogous to the coefficients of a regression model. The feature importance using the synaptic weight-based attribution method is computed as:

$$W_{\mathbf{x}_i} = \frac{1}{C}\sum_{j}\left|w_{i,j}\right| \qquad \textit{where } C \textit{ is the number of synaptic weights}$$

Conditional density plots of five individual raw features over the weight region [0,1] are presented in Figure 4.1.3 with results tabulated over the 802 individual FTSE100 CLOB unique datasets, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$. The horizontal axis represents the weight importance $W_{\mathbf{x}_i}$ for a specific feature whilst the conditional density is displayed on the vertical axis. From the results one can note that feature importance metrics for quoted spreads measured over the past ten events, QS.E10, tend to remain stable with a conditional probability density between 0.2 and 0.25 over the weight value range [0,1]. This indicates that the relative importance of quoted spreads compared to the other features is similar for traders that place both a low and high attribution to quoted spreads. In contrast, quote offset for the bid, QOB, has a higher relative density in the low attribution values, indicating that these features are very important for a smaller number of algorithmic traders' algorithms, ostensibly predicted to be those traders conducting automated market maker strategies.

As the synaptic weight attribution values increase, the number of firms that attain these saliency values significantly decreases, with less than 0.01% of firms attaining an average synaptic weight feature importance value greater than 2. Intuitively, this result is in line with expectations due to the deep architecture employed in the DNN with 1300 hidden neuron parameters included in the model. Evidently, a drawback of using the weight-based approach of Olden (2004) in high-dimensional, deep, and wide neural networks is the voluminous array of weights from which it becomes difficult to extract concordantial relationships between weights and feature attributions using linear techniques in highly complex models with a predisposition for non-linear interactions and transformations.

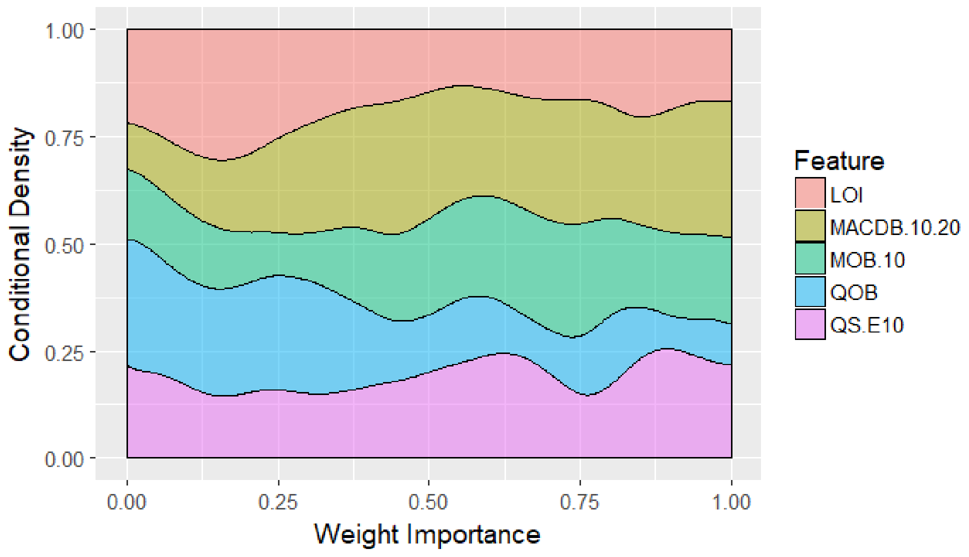

FIGURE 4.1.3 – Synaptic Weights based attribution method conditional density plot for five raw LOB features.

4.1.4 Input Perturbation Methods

Input perturbation methods attribute the importance of feature $\mathbf{x}_i$ in a trained DNN by comparing the performance of the non-corrupted network against one that perturbs, shuffles or removes the input feature $\mathbf{x}_i$. Performance is generally measured by passing the actual and perturbed dataset through the network and measuring the change in the cost function or output values. Various perturbation methods have been developed to study the attribution of pixels in Convolutional Neural Networks (CNN) (Zintgraf, 2017; Zeiler, 2014) and features in conventional feedforward DNNs (Breiman, 2001; Heaton, 2017b). Several input perturbation methods of feature attribution are explored and implemented on the FTSE CLOB dataset including the Occlusion algorithms (Heaton, 2017b; Zeiler, 2014) and Locally Interpretable Model-agnostic Explanation models (Ribeiro, 2016). Deficiencies in perturbation methods exist in their high relative computational cost, especially when compared against backpropagation methods, and instability in the presence of surprise artefacts where the DNN receives perturbed inputs that may be outside the manifold of the trained dataset leading to abnormal predictions during inference.

4.1.4.1 Occlusion Analysis

Various feature importance wrapper algorithms have been employed in the literature, primarily in the domain of image recognition, that utilise occlusion techniques to block, mask or shuffle input feature $\mathbf{x}_i$ before analysing the impact on the models' performance as a way of attributing importance to that feature (Zeiler, 2014; Heaton, 2017b). This section employs an occlusion input perturbation algorithm to analyse the importance of FTSE100 CLOB features over the set of ISIN-Date-Broker datasets, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$. The algorithm requires the fitting of the DNN for each dataset using the full input feature vector. The learnt weights, network structure, cost function and learning algorithm of the DNN are maintained throughout the application of the feature importance algorithm. The algorithm loops through each individual input feature, $\mathbf{x}_i$, of the neural network and perturbs or shuffles that feature's value, $\mathbf{x}_{P_i}$, by drawing randomly without replacement from the values of the feature vector $\mathbf{x}$. The change in loss, $\Delta\mathcal{L}$, between the non-corrupt, $\mathcal{L}(\mathbf{x},\mathbf{y})$, and perturbed dataset cost function, $\mathcal{L}_P(\mathbf{x}_P,\mathbf{y})$, is then computed. The decision to shuffle rather than fully remove or mask the input feature is based on the rationale of maintaining the distributional and statistical properties of the feature vector, whilst still inputting the feature into the DNN in a way that provides minimal information towards correctly predicting the classifiers. The deviation in loss between the initial and perturbed neural networks is a proxy for the importance, $\mathcal{FI}$, of the perturbed feature.

The occlusion input perturbation method is applied to the data with boxplots for 11 sub-strategy feature groups depicted in Figure 4.1.4. Each boxplot is a representation of the change in loss metric, $\Delta\mathcal{L}$, which serves as a proxy for feature importance given that higher loss values are associated with features critical to the performance of the network, thus, explain trader actions in the FTSE100 CLOB dataset and proxy as inputs into trader algorithms. From the figure it is apparent that when quoted spread feature values are perturbed and shuffled randomly, the average $\Delta\mathcal{L}$ is 16% higher for the trained DNNs, represented by the white asterisk. Furthermore, the results for quoted spread feature groups indicate a high degree of skewness toward higher losses when inputs are perturbed, illustrated by the upper quartile value of 20.8%, with quartiles represented by the box hinges, and an upper whisker value of 51.1%, represented by the thin line which extends 1.5 times the interquartile range. Beyond the whisker line there are further outlier data points. Several other features analysed follow a similar pattern though with smaller magnitudes, including the limit order imbalance, quote offset and latency arbitrage feature groups. As expected all features attain mean change in loss values that are greater than zero, indicating that shuffling the feature values and propagating them through the network induces a higher loss than if the correct values were used. Several of the features have average loss changes close to zero, such as informed trading and resilience feature groups, indicating that on average the DNNs loss function does not increase significantly when their respective input features are perturbed.

Occlusion analysis provides an embedded model framework for assessing feature importance using the actual trained DNN with the results presented showing the change in loss metric to be a valid measure of feature importance. As discussed, the primary drawback from applying input perturbation methods is the risk that surprise artefacts exist in the data leading to an unstable change in loss measure $\Delta\mathcal{L}$ that captures deficiencies in the model's ability to process such artefacts, as opposed to the underlying feature saliency properties. The FTSE100 CLOB dataset is susceptible to this concern given the nature of input vector $\mathbf{x}$ as a mathematical representation of LOB phenomena, in contrast to image or text inputs that allow for the user identification of artefact issues through visual inspection. Normalisation and standardisation pre-processing techniques do combat this issue to a degree, but it is difficult to test for robustness in the measure.

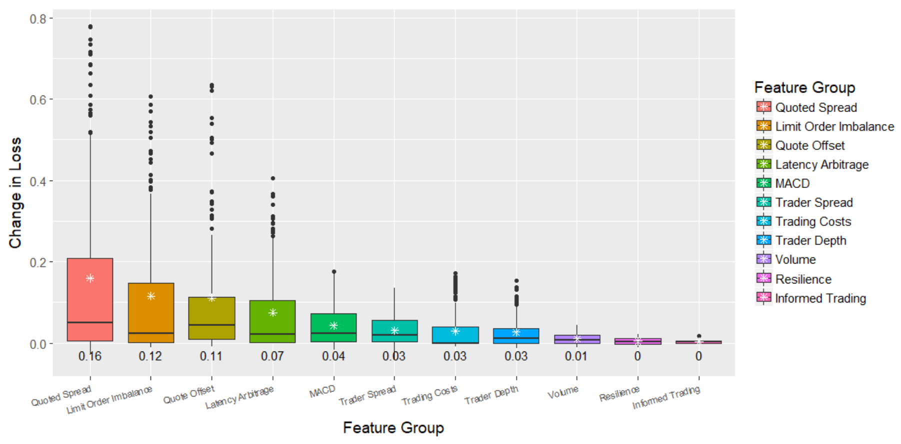

Figure 4.1.4 – Occlusion analysis measuring change in loss, $\Delta\mathcal{L}$, from perturbations of input features, $\mathbf{x}$, for 11 sub-strategy feature groups.

4.1.4.2 Locally Interpretable Model-Agnostic Explanations

The second perturbation-based method implemented is the Locally Interpretable Model-agnostic Explanation (LIME) model (Ribeiro, 2016). The primary objective of the LIME approach for feature importance is to define an *interpretable* explanation model, $g: \mathbb{R}^n$, within a class of linear models G where $g \in G$, that is both locally faithful and capable of approximating the global non-linear predictor function, $f: \mathbb{R}$, locally. The assumption of LIME is that the non-linear FTSE100 CLOB DNN predictor function, f, optimised during training, must exist within an 'interpretable' representation (Ribeiro, 2016). Mathematically, the interpretable explanation model g is chosen as that which minimises the function $\delta(x)$. This function measures the local fidelity of model g in approximating the locality of the global model f. That is, how well the linear model explains the non-linear model in a 'locality region' between constants z and x, ϑ_z. Additionally, the complexity of the linear model, $\varphi(g)$, is included in the objective function to arrive at:

$$\delta(x) = \arg\min_{g \in G} C(f, g, \vartheta_z) + \varphi(g)$$

LIME takes a model-agnostic approach to explaining complex machine learning models by separating the original model from ex-post explanation and interpretability, given that many model types, particularly very deep neural networks, still have 'blackbox' properties that minimise their interpretability. This approach of model agnosticism allows for significant global non-linear model flexibility given simpler local models are capable of approximately explaining any model irrespective of their complexity, negotiating the flexibility-human interpretability trade-off (Freitas, 2014) inherent in complex models. Feature importance metrics, $\mathcal{FI}$, are developed by quantifying the distance between the complex non-linear DNN model predictions and the predicted values of the local linear approximate model g on a local scale when minor perturbations are made in the input feature $\mathbf{x}_i$. The initial step of LIME is to calculate these importance metrics for a specific data sample by first generating n permutations of a sample instance being explained by perturbing the input feature. Predictions are made by the complex DNN on the permuted dataset before computing a 'weight' metric for each input feature that measures the proximity of the sample prediction of the complex DNN model f, for all permuted observations, to the actual values of the non-permuted data. As a result, higher weighted features better explain the complex non-linear DNN model's predictor outcomes. Once the 'weight' metrics for the features are quantified, the most important m features are extracted from the complex DNN using a feature selection criterion, before fitting the local linear model g to the permuted data using those m features. Accordingly, the local linear model can predict its own 'weights' for each of the m features using basic linear regression. These new weights serve as a proxy to explain the non-linear complex model's behaviour in the local region and evidently can be used as feature attribution measures for each of the m features utilised for the analysis of the individual data sample.

LIME models of feature importance are fitted to the individual FTSE100 CLOB DNN trained to model the relationship between LOB features and trader actions. Figure 4.1.5 provides a micro-level overview of the results for a single DNN trained for one ISIN-Date-Broker tuple $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$, for two near consecutive limit orders placed at the best ask by the trader. The figure provides an analysis of the LIME feature importance scores for 50 features that are chosen using a feature selection algorithm based on the absolute feature importance weights of a linear fitted ridge regression that serves as a local approximate for the DNNs non-linear function. Two separate samples are analysed for the action of a limit order at the best ask, LO.A.P0, with the DNN softmax outputting probabilities of 0.99 and 0.96 for each sample, with both samples correctly predicted by the DNN as the action taken by the trader.

An initial point to note is that both samples have slightly different features that explain each individual action, even though the actions were executed within several seconds of each other. However, given that both actions are performed within a close event time-frame, by the same trader and on the same security, there is also a lot of commonality between the feature importance scores for both samples. For both cases, LIME selects quote offset on the ask side, QOA, as the most important feature that contradicts the DNN with a value of -0.16 and -0.14 for the first and second case, respectively. For example, the trader may be in a situation where they are expected to widen the quotes on the ask side by placing a limit order one or two ticks from the BBO, however, other features in the dataset may additionally serve as an important input into the firm's algorithm that motivates them to place the order at the BBO. The next series of important features may fit this criterion. For example, the latency arbitrage feature, LAOB, in the first case indicates that there is a disequilibrium across the UK trading venues for the security being analysed at the best bid-ask, which may motivate the trader to place the order at the best ask to gain from order queue priority. Additionally, three momentum features in the first case – MOB.10, MOB.5, and MOA.5 – all have high importance feature values greater than 0.05 and support the DNN model's predictions. Once more a potential explanation overriding some of the contradictory features such as QOA or the traders market depth features, MDTR.0, could be that excitatory momentum signals are prevalent on the LOB, driving the motivation of the trader to place the order at the ask BBO.

Arguments against implementing LIME as an attribution method for FTSE100 CLOB features are based on its failure to meet the axiom described in the literature as 'sensitivity' (Sundararajan, 2017; Shrikumar, 2017; Ancona, 2018). Meeting this axiom requires that samples which differ in a certain feature, with a corresponding divergence in predictor classes, must place some non-zero attribution towards that feature. When the DNNs non-linear function f flattens in a certain region, common when ReLU activations are employed, it may be approximated by a local linear function. When predictions are made by LIME they may not meet the 'sensitivity' axiom as oscillations in the non-zero feature input value may remain on the flat region of the local linear model, thus, changes in the feature value do not result in a non-zero feature attribution value, breaking the sensitivity axiom.

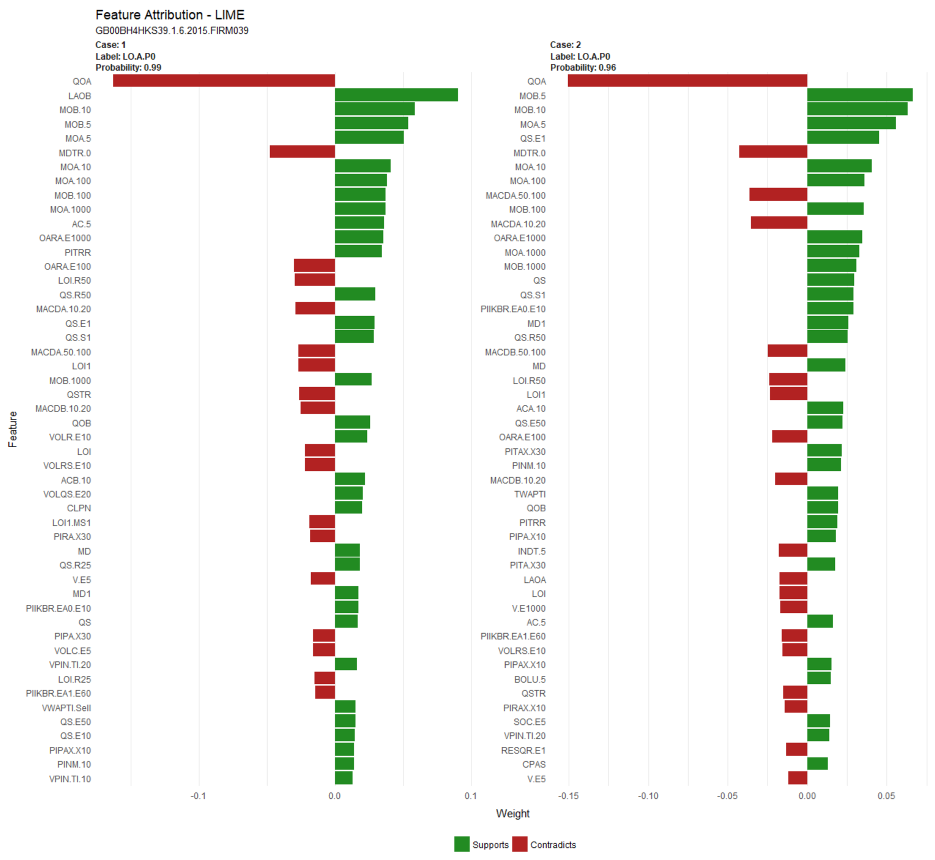

Figure 4.1.5 – LIME model for two limit orders at the ask side of the BBO, LO.A.P0, for ISIN GB00BH4HKS39, Date 1/6/2015, Undisclosed Broker, 10:45 am. Both orders are placed within a one second timeframe. Label refers to the correct order classification and probability refers to the models prediction of the correct class.

4.1.5 Backpropagation Approaches

The common binding concept behind backpropagation-based approaches to feature attribution is the reliance on evaluating the gradients of the DNN output, $\mathbf{y}$, with respect to individual input neurons, $\mathbf{x}_i$, to derive saliency metrics for neuron i. These techniques differ from previous approaches given their model-dependent nature with attribution based on the level and magnitude of signals propagating through the DNN. Backpropagation-based approaches address the computational inefficiency of input perturbation methods which require continuous permutations of the features to extract attribution whilst backpropagation can be completed with a single pass forwards and backwards of an input feature through the network. Several issues common to perturbation attribution methods, including the presence of artefacts in the data, and basic gradient backpropagation methods, including the 'saturation' of gradient and 'threshold' issues (Shrikumar, 2017), have been overcome by recent innovations in backpropagation methods. In particular, Integrated Gradient (Sundararajan, 2017) and DeepLIFT (Shrikumar, 2017) algorithms for feature attribution present reliable contemporary methods applicable to measuring the saliency of FTSE100 CLOB DNN input features.

This section introduces five gradient-based approaches for feature attribution – Gradient*Input (Simonyan, 2014), Layer-wise Relevance Propagation (Bach, 2015), Deep Taylor Decompositions (Montavon, 2017), DeepLIFT (Shrikumar, 2017) and Integrated Gradients (Sundararajan, 2017) - with the approaches then evaluated to determine the relevant method to implement for the FTSE100 CLOB dataset. The backpropagation-based approaches for DNNs are mathematically rooted in the partial derivative of output w.r.t input, $\partial \mathbf{y}_c / \partial \mathbf{x}_i$. Recent work by Ancona (2018) posits conditions of equivalence and approximation between several of the methods and attempts to provide a unifying framework. Henceforth, the term 'gradient' is used interchangeably with the analogous partial derivative of the output w.r.t input term, unless specified otherwise.

*4.1.5.1 Gradient*Input*

Backpropagation of signals through the DNN is a common thread that connects the various attribution methods explored in this section, with the Gradient*Input method (GI) serving as a theoretically tractable and simple estimation of feature saliency (Simonyan, 2014). GI methods were initially implemented by Simonyan (2014) as an instance-specific class saliency mapping technique to visualise feature importance, or spatial support, by highlighting features, or pixels, on attribution maps, based on the gradient of the classifier output w.r.t input pixel features in a deep CNN. Mathematically, these methods measure feature saliency by computing the absolute partial derivative, $\partial \mathbf{y}_c / \partial \mathbf{x}_i$, for the majority class prediction, $\mathbf{y}_c$, w.r.t the input feature vector, $\mathbf{x}_i$. Springenberg (2014) developed a 'guided backpropagation' method similar to Simonyan (2014) in that feature importance is conditioned on backpropagating gradients through the network, though the method suppresses negative gradients to align with properties of the ReLU activation function employed in the author's network. Intuitively, larger absolute gradients require only a minimal perturbation to the input feature to drastically affect the output classifier, thus, reinforcing the importance of that input feature value to the network. Shrikumar (2017) extended these techniques to incorporate the magnitude of the input activation signal, $\mathbf{x}_i$, culminating in the GI attribution for input feature i, $\mathbf{r}_i$, defined by:

$$\mathbf{r}_i = \left| \frac{\partial \mathbf{y}_c}{\partial \mathbf{x}_i} \right| \mathbf{x}_i$$

The logic behind these scores is that gradients reflect the impact of an input feature on the classifier value, indicating importance for explaining the neural networks relationship between features and classifiers. Higher scores imply that features are more useful for classifying predictors. Several observations regarding GI follow. First, the inclusion of the multiplicative input feature term, $\mathbf{x}_i$, allows for the computation of a 'local' attribution measured by the expected change in output from an infinitesimally small change in input *around* the input feature. Ancona (2018) differentiates the local attribution method from global attribution methods that are explored by the following four backpropagation algorithms in this section which quantify the marginal impact of a change in the input feature on the output value relative to a baseline defined by the user. Second, GI employs the absolute value of the gradient to attribute importance which prohibits the detection of information regarding whether the importance is positive or negative resulting in less practical interpretability. Third, GI methods commonly suffer from 'saturation' and 'threshold' issues common when backpropagating signals through the network without applying any normalisation measures (Shrikumar, 2017).

Saturation occurs when a multitude of mathematical operations are applied through the non-linearities of the backward pass, as gradients are multiplied across very deep networks using the chain rule to derive the gradient of the output w.r.t input. Activation functions that have gradients with significant saturation regions are most often affected as the gradient reduces to a near indecipherable and ultimately uninterpretable value (Glorot, 2010). Sundararajan (2017) also analysed vanilla gradient attribution using the GoogleNet 22-layer deep CNN Inception architecture (Szegedy, 2014) trained on the ImageNet dataset, positing that input gradients in and of themselves fail to provide accurate quantitative attributions for the importance of individual pixel features. As identified, this is due to saturation of backpropagated gradients which essentially are local linear approximations. Thresholding issues occur due to discontinuities in the gradients for certain activation functions, particularly ReLUs. This results in undesirable artefacts becoming prevalent in the feature attribution values, especially around the threshold value where the discontinuity occurs as gradients shift significantly when input values lie on either side of the threshold, providing a misleading interpretation of importance.

The introduction of exponential linear unit activation functions (Clevert, 2015) and batch normalisation (Ioffe, 2015) for deep networks can go some way to alleviating the saturation and threshold issues. However, there remain practical issues with employing the GI method which requires consideration of other techniques that placate many of the concerns inherent in GI to provide more analytically tractable attribution methods. Applying difference from reference or baseline global attribution methodologies used in Integrated Gradients and DeepLIFT methods can overcome these issues (Ancona, 2018).

4.1.5.2 Layer-Wise Relevance Propagation

Layer-wise relevance propagation (LRP) is a rule-based method that performs feature attribution by decomposing the probabilistic value outputted by the classifier predictor function of the DNN at layer L, $\hat{\mathbf{y}}$, into a 'relevance' score measure, $\mathbf{r}$, over the input vector $\mathbf{x}$ (Bach, 2015). Similar to other attribution methods, higher relevance scores can be qualitatively interpreted as LOB features that provide stronger evidence for the correct prediction of the action a trader executes on the LOB. For the DNN applied in this chapter, LRP first requires the computation of relevance scores for each softmax output neuron, $\mathbf{r}_c^L = \hat{\mathbf{y}}_c$, where, $c \in C$, for all classes C. Relevance scores for output values are decomposed in a recursive manner as signals propagate backwards through the network using a conservative relevance redistribution procedure (Bach, 2015), attributing importance or 'relevance' scores for each neuron layer-by-layer. Signals backpropagated through the network in the form of relevance scores maintain a conservation property where the sum of scores in each layer should be approximately equal to the sum over output values in $\hat{\mathbf{y}}$. The relevance score for neuron $i \in I$ in layer l, $\mathbf{r}_i^l$, is computed as a function of scores in $j \in J$ neurons comprising the adjacent layer, $\mathbf{r}_j^{l+1}$, and the series of input activation values connecting the two hidden layers, $\mathbf{z}_{i,j}^{l+1} = \mathbf{w}_{i,j}^l \mathbf{x}_i^l + \mathbf{b}_i^l$, using the recursive ε rule (Bach, 2015), where ε is a minor permuted value. Feature importance using LRP can be defined as:

$$\mathbf{r}_i^l = \sum\nolimits_j \frac{\mathbf{z}_{i,j}^{l+1}\mathbf{r}_j^{l+1}}{\sum_{i\in I}\mathbf{z}_{i,j}^{l+1} + \varepsilon\sum_{i\in I}\mathbf{z}_{i,j}^{l+1}} \qquad where \sum\nolimits_{i\in I}\mathbf{r}_i^l = \sum\nolimits_{j\in J}\mathbf{r}_j^{l+1} \quad (Conservation\ property)$$

The propagation rule is applied by iteratively distributing relevance scores layer-by-layer backwards through the network before culminating at the input layer $\mathbf{x}$ where relevance scores of the vector, $\mathbf{r}^0$, serve as a proxy for the attribution of each input feature to the output of the DNN. The LRP technique overcomes many of the issues inherent in saturation or threshold concerns evident in the GI method applied in the previous section. LRP also serves as an approximate alternative to a full Taylor-decomposition of the DNN, explored in the next section.

4.1.5.3 Deep Taylor Decompositions

Deep Taylor Decomposition (DTD) methods extend the relevance score method employed in LRP (Bach, 2015) and saliency map techniques (Simonyan, 2014) by performing a Taylor decomposition of the DNNs output value, $\mathbf{y}$, to obtain output layer relevance values, $\mathbf{r}$ (Montavon, 2017). These relevance signals are then parsed backwards through the network to ultimately derive an attribution value for the feature input vector $\mathbf{x}$. The key component of the method is its application of a Taylor decomposition to *individual* neurons that are independently projected onto an adjacent hidden layer in the direction of the input layer, rather than as a collective which is the case in LRP methods. DTD relevance scores are theoretically derived to meet the axiom of 'consistency', aligning attribution values for the input features with total relevance in the DNN output (Montavon, 2017).

In a similar vein to LRP, the DTD method initialises relevance scores for neurons in the output layer based on the softmax values outputted by the DNN. The algorithm iterates through recursive decompositions that propagate relevance signals backwards through the deep network, first by attributing importance to neurons in deeper layers that have higher levels of feature abstractions, before terminating the process at the input layer. Determining the relevance score, $\mathbf{r}_i^l$, for hidden neuron i, $\mathbf{x}_i^l$, first requires the assumption that relevance scores for neurons in the adjacent layer, $\mathbf{x}_j^{l+1}$, can be expressed in terms of a positive constant $c_j \geq 0$, that represents the attribution of neuron j to the output of the DNN in a local region, such that $\mathbf{r}_j^{l+1} = c_j \mathbf{x}_j^{l+1}$. Using the DNN architecture employed in this chapter, the relevance score for neuron j can be distributed into neuron i in the lower layer using a decomposition based on a first-order Taylor expansion of neuron j's newly defined relevance score, $c_j \mathbf{x}_j^{l+1}$. The resulting expression, $\mathbf{r}_{i(j,l+1)}^l$, defines the *individual* attribution from neuron i in layer l towards the output of neuron j in layer $l+1$, based on the partial derivative of the higher layer neurons activation value w.r.t the lower layer's neuron. The expansion also requires the setting of a unit root point, $\tilde{\mathbf{x}}_{i(j,l+1)}$, in a region where score functions are infinitesimally small (Montavon, 2017), resulting in the relevance computed as:

$$\mathbf{r}_{i(j,l+1)}^l = c_j \left.\frac{\partial \mathbf{r}_j^{l+1}}{\partial \mathbf{x}_i^l}\right|_{\tilde{\mathbf{x}}_{i(j,l+1)}} \left(\mathbf{x}_i^l - \tilde{\mathbf{x}}_{i(j,l+1)}\right)$$

Evidently, performing DTD feature attribution for a single neuron requires the computation of Taylor decompositions and the setting of unit roots for all neurons in the adjacent contiguous layer of the DNN. Unit roots can be set to the closest point in Euclidean space to the input neuron i's activation value *when the feature space is unconstrained*, as is the case for the exponential linear unit activation function for the DNN employed in this chapter. According to Montavon (2017), the point for the unit root sits at the intersection between two subspaces – a plane equation that constrains the forward propagated input activation value for neuron j to zero, $\sum_i \mathbf{x}_i^l \mathbf{w}_{ij}^l + \mathbf{b}_i^l = 0$, and a line of maximum descent between neuron i and j. Solving for the point where the subspaces connect results in the setting of the unit root and quantification of the final backwards propagated relevance score from neuron j to neuron i, $\mathbf{r}_{i(j,l+1)}^l$, where $i \in I$ and I is the number of neurons in layer l, leading to:

$$\mathbf{r}_{i(j,l+1)}^l = \sum_j \frac{\left(\mathbf{w}_{ij}^l\right)^2}{\sum_{i \in I}\left(\mathbf{w}_{ij}^l\right)^2} \mathbf{r}_j^{l+1}$$

Through the structural decomposition of the DNN into a multitude of relevance score sub-functions for each neuron in the network, the DTD method can ultimately arrive at feature attribution scores for the input vector, $\mathbf{r}^0$. Interestingly, after applying consistent Taylor decompositions (Montavon, 2017), the DTD feature attribution rule transforms into a functional form similar to the synaptic weight-based approaches of Olden (2004) and Goh (1995), where weights of the network, $\mathbf{w}$, explain feature importance.

4.1.5.4 DeepLIFT

DeepLIFT applies a novel computationally efficient backpropagation-based algorithm deployed with the objective of attributing feature importance values, $\mathbf{r}$, for elements of a non-linear DNN input feature vector (Shrikumar, 2017). This technique overcomes the saturation and threshold issue identified in the previous sections by framing feature importance as a deterministic function of the divergence between input and output values from a reference point, determined by the network designer using domain knowledge. Utilising the 'difference from reference' approach allows the algorithm to overcome issues of saturation when propagating a signal back from the output to input by shifting the input values away from saturated regions into those with higher interpretable properties. Similarly, thresholding issues arising due to discontinuities in the functions derivative are mitigated by avoiding artefacts or misleading importance attribution values by using difference from reference adjustments that produce continuous and logical functions. DeepLIFT is a particularly relevant attribution method for the FTSE100 CLOB DNN datasets due to DeepLIFTs ability to solve for saturation and threshold issues, caused by the deep architecture of the optimised model and the use of exponential linear unit activations in the network.

Following from the optimised DNN architecture employed to solve the mapping function, $\mathbf{y} = f(\mathbf{x}, \mathbf{w}, \mathbf{b})$, for the Broker tuple datasets, DeepLIFT can be applied to both hidden, $\mathbf{x}^l$, and input, $\mathbf{x}$, neurons in the network. For the hidden layer l input, $\mathbf{x}^l$, composed of n feature elements in space $\mathbb{R}^n$, that feeds into a target neuron in the adjacent layer, $\mathbf{x}_j^{l+1}$, DeepLIFT first requires the definition of a corresponding 'reference' activation value, $\mathbf{x}_j^{0,l+1}$, from which the difference from reference activation value for neuron j, $\Delta\mathbf{x}_j^{l+1} = \mathbf{x}_j^{l+1} - \mathbf{x}_j^{0,l+1}$, can be computed. The reference value is the activation value of the target neuron when the DNN propagates the reference input forward, hence, once the input layer reference values are defined, the hidden and output layer neurons reference values are determined naturally using forward propagation of the new reference input values through the DNN. Thus, initial input reference values are critical and require domain knowledge of the machine learning problem, though Shrikumar (2017) provides guidance by recommending that references be set at values that provide a 'neutral' base for comparison for which feature importance can be attributed. This chapter utilises activation values of zero due to the normalised and zero mean centred nature of the FTSE100 DNN input feature data.

The DeepLIFT algorithm assigns feature importance or contribution scores of, $\mathbf{r}_{\Delta\mathbf{x}_i^l \Delta\mathbf{x}_j^{l+1}}$, for hidden layer l neuron i, $\mathbf{x}_i^l \in \mathbb{R}^n$, by backpropagating signals from neuron j in layer $l+1$ backwards through the network towards the target neuron i in layer l. Target neuron contribution scores can be heuristically explained as the difference in the output neuron from its reference, $\Delta\mathbf{x}_j^{l+1}$, that is attributable to the difference in the input or target neuron in the previous layer from its reference, $\Delta\mathbf{x}_i^l$. The algorithm requires that the summation-to-delta property (Shrikumar, 2017) holds such that the sum of all input neuron contribution scores for the n neurons in layer l, that are attributable to adjacent layer output neuron j, should sum to the divergence in the output neuron from its reference:

$$\sum_{i=1}^{n} \mathbf{r}_{\Delta \mathbf{x}_i^l \Delta \mathbf{x}_j^{l+1}} = \Delta \mathbf{x}_j^{l+1}$$

Multiplier values, $\mathbf{m}_{\Delta \mathbf{x}_i^l \Delta \mathbf{x}_j^{l+1}}$, are calculated as the contribution score of neuron j to i, $\mathbf{r}_{\Delta \mathbf{x}_i^l \Delta \mathbf{x}_j^{l+1}}$, divided by $\Delta \mathbf{x}_i^l$, and is analogous to the partial derivative of the output with respect to the input value, over finite differences. Shrikumar (2017) highlights the reliance on the '*chain rule for multipliers'* to calculate the multipliers between an input neuron a and output neuron c, where a is fully connected to the adjacent hidden layer comprising b neurons, by using dynamic programming and backpropagation to derive $m_{\Delta x_a \Delta x_c}$ through the summation of all b' neurons in adjacent layers, leading to:

$$\mathbf{m}_{\Delta \mathbf{x}_i^l \Delta \mathbf{x}_j^{l+1}} = \frac{\mathbf{r}_{\Delta \mathbf{x}_i^l \Delta \mathbf{x}_j^{l+1}}}{\Delta \mathbf{x}_i^l} \qquad m_{\Delta x_a \Delta x_c} = \sum_{b'} m_{\Delta x_a \Delta x_b} m_{\Delta x_b \Delta x_c}$$

Due to the non-linear and dense feedforward nature of the DNN architecture applied in this chapter, it is necessary to deploy the DeepLIFT 'RevealCancel' rule to assign multipliers and corresponding contribution scores to the networks neurons (Shrikumar, 2017). This requires an independent treatment for the positive, $\Delta \mathbf{x}_j^{+,l+1}$, and negative, $\Delta \mathbf{x}_j^{-,l+1}$, difference from reference activations for the output neuron and corresponding multiplier computations, that together sum to the total output difference from reference, $\Delta \mathbf{x}_j^{l+1}$ (Shrikumar, 2017). Furthermore, the non-linear hidden layer activation functions of the network require the algorithm to split how contributions flow through both the linear, $\mathbf{z}_j^{l+1} = \sum_i \left[\left(\mathbf{w}_{ij}^{l+1} \mathbf{x}_i^l \right) + \mathbf{b}_j^{l+1} \right]$, and then non-linear, $\mathbf{x}_j^{l+1} = \phi\left(\mathbf{z}_j^{l+1}\right)$, components of the DNN.

The 'chain rule for multipliers' provides a basis for decomposing the multiplier between neurons in adjacent layers, $\mathbf{m}_{\Delta \mathbf{x}_i^l \Delta \mathbf{x}_j^{l+1}}$, into component multiplier elements, expressed using the backwards pass of $\mathbf{x}_j^{l+1} \to \mathbf{z}_j^{l+1}$ and $\mathbf{z}_j^{l+1} \to \mathbf{x}_i^l$. The affine transformed linear multiplier, $\mathbf{m}_{\Delta \mathbf{x}_i^{+/-,l} \Delta \mathbf{z}_j^{+/-,l+1}}$, and the non-linear multiplier, $\mathbf{m}_{\Delta \mathbf{z}_j^{+/-,l+1} \Delta \mathbf{x}_j^{+/-,l+1}}$, are attributed differently for the positive and negative components $(+/-)$, leading to the computation of multipliers as:

$$\mathbf{m}_{\Delta \mathbf{x}_i^{+,l} \Delta \mathbf{z}_j^{+,l+1}} = \mathbf{m}_{\Delta \mathbf{x}_i^{-,l} \Delta \mathbf{z}_j^{+,l+1}} = \mathbf{1}\left\{ \mathbf{w}_{ij}^{l+1} \Delta \mathbf{x}_i^l > 0 \right\} \mathbf{w}_{ij}^{l+1} \qquad (linear\ connections)$$

$$\mathbf{m}_{\Delta \mathbf{z}_j^{+,l+1} \Delta \mathbf{x}_j^{+,l+1}} = \frac{\Delta \mathbf{x}_j^{+,l+1}}{\Delta \mathbf{z}_j^{+,l+1}} \qquad \mathbf{m}_{\Delta \mathbf{z}_j^{-,l+1} \Delta \mathbf{x}_j^{-,l+1}} = \frac{\Delta \mathbf{x}_j^{-,l+1}}{\Delta \mathbf{z}_j^{-,l+1}} \qquad (non-linear\ connections)$$

$$\mathbf{m}_{\Delta \mathbf{x}_i^l \Delta \mathbf{x}_j^{l+1}} = \mathbf{m}_{\Delta \mathbf{x}_i^l \Delta \mathbf{z}_j^{l+1}} \mathbf{m}_{\Delta \mathbf{z}_i^{l+1} \Delta \mathbf{x}_j^{l+1}} \qquad (chaining\ linear\ and\ non-linear\ multipliers)$$

Evidently, the multiplier for the linear activation has an equivalent form for both the negative and positive difference from reference activations. From the equations, the difference from reference for the output neuron for non-linear functions, $\Delta \mathbf{x}_j^{+/-}$, is taken as the average of the impact from the input neuron with the same side, $\Delta \mathbf{z}_j^{+/-}$, with no terms added, and the impact when, $\Delta \mathbf{z}_j^{-/+}$, has been added. Shrikumar (2017) implements this method to minimise the risk of positive and negative differences from reference values annulling each other. Application of the RevealCancel rule and computation of the multipliers, $\mathbf{m}_{\Delta\mathbf{x}_i^l \Delta\mathbf{x}_j^{l+1}}$, allows for the derivation of the contribution, $\mathbf{r}_{\Delta\mathbf{x}_i^l \Delta\mathbf{x}_j^{l+1}}$, of neuron j, $\mathbf{x}_j^{l+1}$, to neuron i in the preceding layer, $\mathbf{x}_i^l$, using the chain rule with the following equation:

$$\mathbf{r}_{\Delta x_i^l \Delta x_j^{l+1}} = \mathbf{m}_{\Delta x_i^l \Delta x_j^{l+1}} \Delta\mathbf{x}_i^l$$

Once the DeepLIFT algorithm selects the references for input features, forward propagates those references through the network to calculate references and corresponding difference from reference values for all other neurons, then computes the relevant multipliers for each linear and non-linear activation between neurons, feature attribution can be performed for the FTSE100 CLOB DNN input features $\mathbf{x}$. A necessary adjustment for the softmax output layers utilised in the DNN architecture is to only consider the preceding linear layer $\mathbf{z}^L$ before the final softmax sigmoidal non-linearity is applied so as to not violate the summation to delta property (Shrikumar, 2017). Feature importance attribution requires the calculation of contribution scores for input feature neuron $\mathbf{x}_i$ to output neuron $\mathbf{y}_c$, $\mathbf{r}_{\Delta\mathbf{x}_i \Delta\mathbf{y}_c}$, which is determined using the chain rule for multipliers and normalised over all $c \in C$ classes to derive:

$$\mathbf{r}_{\Delta\mathbf{x}_i \Delta\mathbf{y}_c} = \mathbf{m}_{\Delta\mathbf{x}_i \Delta\mathbf{y}_c} \Delta\mathbf{x}_i \qquad \mathbf{r}'_{\Delta\mathbf{x}_i \Delta\mathbf{y}_c} = \mathbf{r}_{\Delta\mathbf{x}_i \Delta\mathbf{y}_c} - \frac{1}{C}\sum_{k=1}^{C} \mathbf{r}_{\Delta\mathbf{x}_i \Delta\mathbf{y}_k}$$

DeepLIFT provides an axiomatic feature attribution algorithm that allows the individual saliency, $\mathcal{FI}$, of the n-dimensional feature input vector for each Broker tuple DNN, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$, to be quantified.

4.1.5.5 Integrated Gradients

Sundararajan (2017) develops a computationally efficient backpropagation-based feature importance method for neural networks that studies how '*integrated gradients*' of scaled-down input feature counterfactuals provide insightful information regarding relative feature saliency. Integrated gradients (IG) provide an interesting model that combines elements of several previous backpropagation and input perturbation approaches introduced. Attribution for DNNs is measured using backpropagation to formulate the partial derivative, $\partial\mathbf{y}_c / \partial\mathbf{x}_i$, of output predictors, $\mathbf{y}_c$, w.r.t input features, $\mathbf{x}_i$, in a similar vein to Gradient*Input methods. However, IG also utilises the concept of a baseline feature value, $\mathbf{x}'_i$, similar to the reference points in DeepLIFT algorithms. At a high-level, IG integrates gradient values on perturbed input features over discrete steps in the continuous domain linearly connecting the input feature and its baseline, culminating in an estimated relative attribution measure for the feature, $\mathbf{r}_i$.

The IG method is built using an axiomatic framework to derive fundamental characteristics of feature attribution that are prevalent in the IG method, including the presence of two axioms – 'sensitivity' and 'implementation invariance' – which are relevant for the FTSE100 CLOB dataset. These axioms primarily address the saturation and threshold issue previously identified (Sundararajan, 2017). The *sensitivity* axiom defines valid models as those that assign a non-zero attribution value to features whose input value changes, relative to its baseline, with a corresponding, though not linear, shift in the predicted classifier output value. Methods that attribute importance solely from the value of the gradient w.r.t the feature will violate the sensitivity axiom. For example, a ReLU activated network will have zero gradients when the ReLU is activated in the negative region, thus, even if a shift in the input value correlates with a shift in the predictor value the gradient will compute zero and attribute no importance to the input feature, violating the sensitivity axiom. Defining the input feature in context to its baseline preserves the sensitivity axiom. The second axiom that IG maintains is *implementation invariance* that requires functionally equivalent neural networks to compute the same attribution for feature inputs. Sundararajan (2017) argues that IG methods of feature attribution meet these axioms. Further, they argue that whilst the DeepLIFT (Shrikumar, 2017) methods meet the sensitivity axiom through their use of baselines, they fail the implementation invariance axiom as the methods can yield different attributions for the same input and outputs from different neural network implementations.

Sundararajan (2017) propose a method for overcoming saturation issues when using gradients for feature importance by defining a series of $\alpha \in (0,1)$ scaled discrete perturbations interpolated between the input, $\mathbf{x}_i$, and baseline, $\mathbf{x}'_i$, feature value which serves as the counterfactual of the model for causal reasoning. Setting the baseline requires domain expertise to determine a position in the feature input space of the DNN where the impact on the output classifier predictor, $\mathbf{y}$, is neutral. Thus, the baseline value should attain a near-zero predictor value that conveys minimal information when signals of the output are backpropagated through the network. Baseline values should also be chosen in a way that they shift the input signal away from saturated areas.

IG attribution methods are applied to the FTSE100 CLOB DNNs, $\mathbf{y} = f(\mathbf{x}, \mathbf{w}, \mathbf{b})$, with feature importance derived from projecting the softmax output classifier, $\mathbf{y^c} \to [0,1]$, for $c \in C$, approximated by the mapping function f, onto the attached n-dimensional feature space $\mathbf{x} \in \mathbb{R}^n$. The method is used to develop an individual feature attribution, $\boldsymbol{r}^c_{\mathbf{x}_i}$, for input feature $\mathbf{x}_i$ with attached predictor class label c, baseline counterfactual $\mathbf{x}'_i$, and scalar variable α. To attain this objective, Sundararajan (2017) develops a series of '*interior gradients*', $\partial f\big(\mathbf{x}'_i + \alpha(\mathbf{x}_i - \mathbf{x}'_i)\big)$, initialised with α equal to zero, and attached baseline, $\mathbf{x}'_i$, before iteratively perturbing the feature value by a scalar factor over the domain between the original and baseline input values, expressed by the term $\alpha(\mathbf{x}_i - \mathbf{x}'_i)$. IG approaches compute a feature attribution metric, $\mathbf{r}^c_{\mathbf{x}_i}$, for input neuron i by taking the integral of the interior gradients along the path from the baseline input to the actual input value, such that:

$$\boldsymbol{r}^c_{\mathbf{x}_i} = (\mathbf{x}_i - \mathbf{x}'_i) \int_{\alpha=0}^{1} \frac{\partial f\big(\mathbf{x}' + \alpha(\mathbf{x} - \mathbf{x}')\big)}{\partial \mathbf{x}_i} d\alpha$$

IG provides a framework for measuring feature importance of the protagonist feature input $\mathbf{x}_i$ by building a representation of the collective impact of interior gradient counterfactual values using scalable parameters across the space. Intuitively, low scalar values of α should highlight relatively important features given that they have higher gradient magnitudes, though each scalar will provide only a single distributed representation. Sundararajan (2017) argues that the IG approach meets the defined axioms of completeness and sensitivity, given that the sum of all integrated gradients, or attribution values, equals the difference between classifier prediction values of the original input $\mathbf{y}^c(\mathbf{x})$ and baseline $\mathbf{y}^c(\mathbf{x}')$.

To maintain analytical tractability this chapter implements an approximation method for computing gradients as proposed by Sundararajan (2017). This requires a Riemman approximation of the integral using the accumulation of interior gradients at m over sufficiently small discrete and equal intervals along the linear path between the input and baseline counterfactuals, such that:

$$\mathbf{r}^{c,Approx}_{\mathbf{x}_i} = (\mathbf{x}_i - \mathbf{x}'_i)\left(\frac{1}{m}\right)\sum_{j=1}^{m} \frac{\partial f\left(\mathbf{x}'_i + \left(\frac{j}{m}\right)(\mathbf{x}_i - \mathbf{x}'_i)\right)}{\partial \mathbf{x}_i}$$

4.2 Backpropagation-based Feature Attribution Evaluation & Dataset Formulation

This section presents an empirical and qualitative evaluation of attribution methods in conjunction with a consideration of their theoretical limitations, with a particular focus on backpropagation-based approaches. The evaluation is performed to determine the optimal method for measuring the relevance metric, $\mathcal{FI}_i$, of LOB feature i, $\mathbf{x}_i$, when predicting trader action c, $\mathbf{y}_c$, for the FTSE100 CLOB ISIN-Date-Broker tuple, $\langle\mathcal{S},\mathcal{D},\mathcal{B}\rangle$, datasets. Categorizing market participants by the type of algorithmic trading strategy executed requires an understanding of the relevant features that are expected to act as inputs into their trading algorithms. Once the important LOB features for a market participant are determined, these traders can be clustered using an unsupervised learning algorithm to segment firms by their trading strategy.

Limitations of attribution models extend to their evaluation (Sundararajan, 2017). In particular, empirically quantifying a model's performance of correctly attributing feature importance may conflate errors in these measures that arise from both the incorrect specification of the DNN or the incorrect projection of the attribution from the output predictions to the input feature space. For robustness, several techniques are employed to evaluate attribution models. These include a visual inspection of feature saliency maps (Bach, 2015) though this method can induce bias given its non-statistical nature (Ancona, 2018), analysis of the change in 'log-odds' for output values when comparing a counterfactual of the correct output (Shrikumar, 2017), and a 'Sensitivity-n' test based on correlations between attributions and changes in output values (Ancona, 2018). The primary backpropagation-based feature attribution methods analysed include – Gradient of output w.r.t to input (Simonyan, 2014), Gradient*Input (Shrikumar, 2017), Layer-Wise Relevance Propagation (Bach, 2015), DeepLIFT (Shrikumar, 2017) and Integrated Gradients (Sundararajan, 2017). The backpropagation-based feature attribution evaluation is performed using three methods of saliency distributions, delta log-odd and Sensitivity-n in Section 4.2.1, 4.2.2 and 4.2.3, respectively. The evaluation indicates that the two axiomatic approaches of Integrated Gradients and DeepLIFT provide applicable feature attribution methods for the FTSE100 CLOB DNN tuple datasets, which are used to extract the feature attribution dataset used for clustering firms into different algorithmic trading strategies, as explained in Section 4.2.4

4.2.1 Saliency Distributions

Distributional properties and visualisations for individual raw feature saliency metrics are concurrently developed to inform a qualitative evaluation of each attribution method's applicability to the FTSE100 CLOB dataset. The importance scores for the five techniques – Gradient (G), Gradient*Input (GI), Layer-Wise Propagation Relevance (LRP), DeepLIFT (DL) and Integrated Gradients (IG) – are computed for each individual action of a brokers $\langle\mathcal{S},\mathcal{D},\mathcal{B}\rangle$ tuple dataset. Raw importance values are averaged across all actions, and normalised with Gaussian, $\mathcal{N}(0,1)$, to attain normalised importance metrics for feature i, $\mathcal{FI}_i$, which allows for positive scores to be interpreted as those features having positive attribution to a DNNs predicted output, with the reverse case for negative scores.

Feature importance distributions for nine separate raw features are presented in Figure 4.2.1 for the five attribution methods. Evidently, comparison of the distributional properties of each method indicates a strong similarity between four of the methods, with the Gradient method used in saliency mapping (Simonyan, 2014) diverging for several of the features studied. Distributions for most of the features have high density around zero with long tails extending into the positive region and relatively shorter tails in the negative region. The Gradient method seems to have a significantly different distribution for LOB features of the aggressive order arrival rate over ten events, OARA.E10, the volatility of the quoted spread, VOLS.QS.E5, the stochastic oscillator, SO.E5, Chaikin volatility, VOLC.E5, and latency arbitrage flags on the ask side, LAOA. One potential explanation for why distributions differ from the other backpropagation methods for these particular features is that the gradient method does not inculcate negative impacts from changes of the feature on the output. Thus, features with raw offset distributions are likely to have a less balanced positive versus negative impact, leading to the distributions skewing when taking the absolute value. Ancona (2018) provides a theoretical explanation for the divergence between pure Gradient and other backpropagation methods. They distinguish between attribution methods which attempt to explain how changes in the output are impacted by minor perturbations in the input feature, referred to as local methods, and those that measure the marginal effect on the output value when an input diverges from a baseline value, referred to as global methods. The Gradient method fits into the local attribution family given its measurement of saliency around the individual feature value, $\mathbf{x}_i$, whilst the other techniques are global methods which, to a degree, measure saliency across a spectrum between a baseline and the input feature, $\mathbf{x}_i - \mathbf{x}_i'$. The similarity in the four global backpropagation approaches indicates a positive result showing that these methods, which are rooted in sound theoretical literature, tend to have well-aligned attribution metrics.

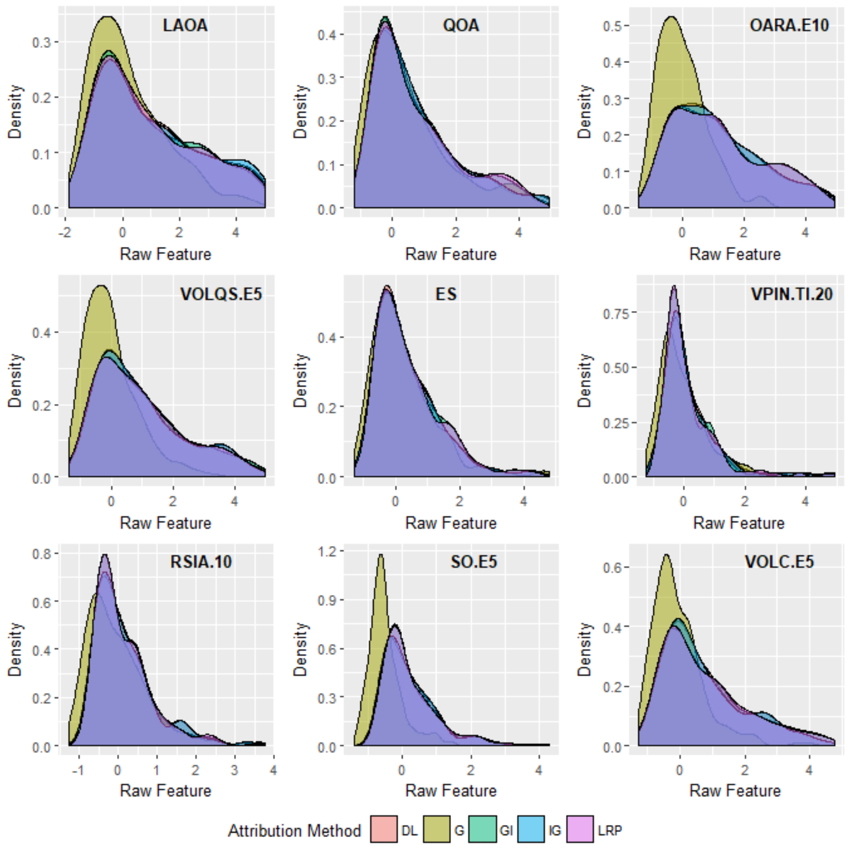

FIGURE 4.2.1 – Saliency distributions for nine LOB raw features compared across five backpropagation-based attribution methods. Note that the 'purple' in the graphs represents the combination of several overlapping density plots.

The alignment of saliency distributions for the four global backpropagation attribution methods derives from the underlying dominance of the partial derivative term, $\partial \mathbf{y}_c / \partial \mathbf{x}_i$, in each models' algorithms, and the use of baseline references. Further exploration of the interrelationship between different models' distributions involves a comparison of Kullback-Leibler (KL) Divergence distribution-wise asymmetric measures, $KL(P|Q) = \int p(x) \, ln \left(\frac{p(x)}{q(x)}\right) dx$, computed for each attribution method pair (P, Q).

Results are tabulated in a matrix form and visualised in Figure 4.2.2. Note that methods on the left of the figure represent the P distribution whilst the top method is the Q distribution.

The first result to note is the high KL Divergence between Gradient attribution methods and competing backpropagation-based methods. The large purple dots in the first row for each of the feature plots represent a high KL divergence between the Gradient method and competing methods, with all KL values being greater than one across the sample analysed. Divergence values greater than one indicate that the distributional properties diverge significantly between comparative methods, which in the context of information theory deems that Gradient distributions offer no 'information' regarding the distribution of other model distributions. This result aligns directly with the visual inspection of saliency distributions performed in the previous qualitative analysis where the low level of relative entropy between Gradient and comparative distributions was evident in the non-aligned density plots.

The second interesting result is the low KL Divergence between the Gradient*Input and Layer-Wise Relevance Propagation pair and the Integrated Gradients and DeepLIFT pair, for the nine features analysed. These results align with theoretical work performed by Ancona (2018) who present a unified framework for current backpropagation-based feature attribution models that investigate conditions of equivalence between various methods. They prove that the Layer-wise Relevance Propagation method is equivalent to the Gradient*Input when the non-linear activation function employed by the DNN is a ReLU and the DNN employs no additive biases. The DNNs trained in this chapter have no bias terms and utilise the exponential linear unit (ELU), an extension to ReLU with similar activation and sparsity properties, which explains the similarity between LRP and GI attribution distributions. Furthermore, the relationship between DeepLIFT and Integrated Gradients has also been explored in the literature (Ancona, 2018; Sundararajan, 2017). Both attribution methods measure the partial derivative, $\partial \mathbf{y}_c / \partial \mathbf{x}_i$, by using the difference from a reference or baseline input value, whilst also meeting the completeness axiom (Sundararajan, 2017) as attribution values sum to the difference between the output value calculated for the actual and baseline input features. Thus, both methods compute some attribution measure $\mathcal{FI}_i$ based on the gradient over the continuous domain between the input and baseline. The low KL Divergence for all features studied is indicative of the commonality between the DeepLIFT (DL) and Integrated Gradient (IG) approach to attribution in one instance, and the Layer-wise Relevance Propagation (LRP) and Gradient*Input (GI) in a second instance. KL Divergence metrics for all nine features sit within a range from 0.004 to 0.012 for $KL(DL|IG)$ and a range of 0.02 to 0.04 for $KL(GI|LRP)$.

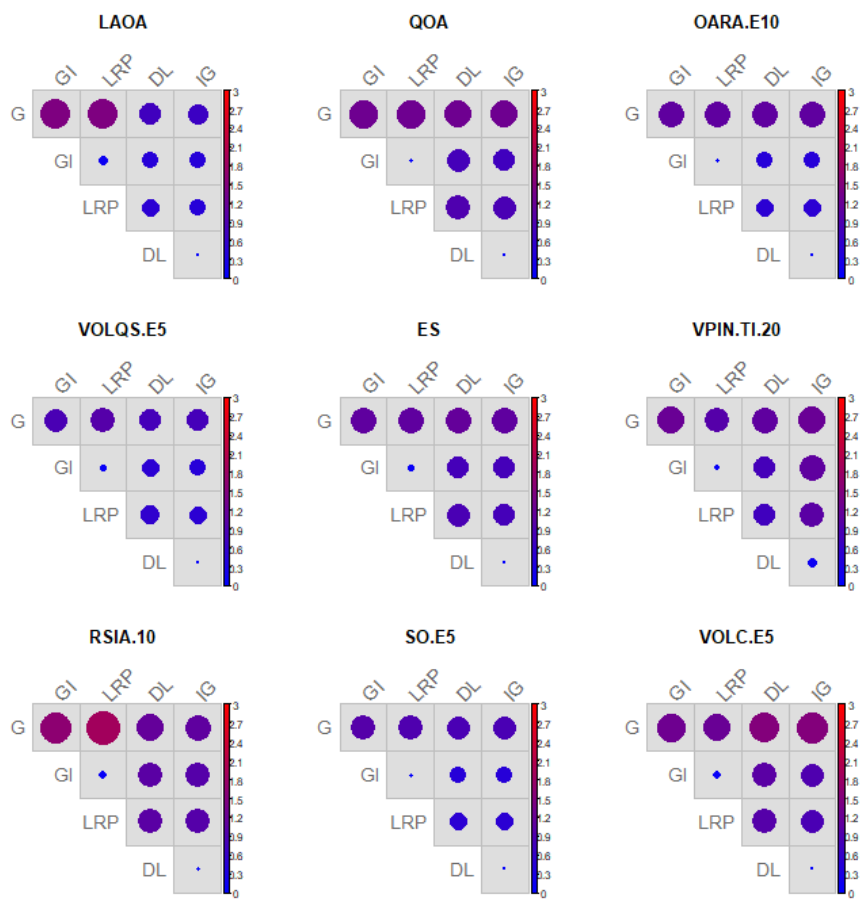

Figure 4.2.2 – Kullback-Leibler Divergence distribution, $KL(P|Q)$, for each attribution method pair (P, Q), analysed over nine raw LOB features. Y-axis represents P distributions and x-axis Q distributions.

The qualitative analysis presented in this section provides evidence towards the assertion that using a basic measure of the output partial derivative w.r.t input may not be an applicable feature attribution technique. Further, the results of the density distribution and KL Divergence analysis indicates a high-level of commonality between the four other attribution methods tested – Layer-wise Relevance Propagation, Gradient*Input, Integrated Gradients and DeepLIFT – which aligns with the development of theoretical frameworks of attribution methods that have found strong relations and equivalence between the different formulations (Ancona, 2018).

4.2.2 Delta Log-Odds

Evaluation of feature attribution methods can be performed by comparing the log-odds of a DNN prediction when all features are included and one when a subset of features are extricated from the feature space. This evaluation technique involves several steps. First, the feature importance score, $\mathcal{FI}_{i_a}$, related to DNN input feature $\mathbf{x}_i$ and attached to actual output class c_a are computed for all feature elements $i \in n$ in the n-dimensional input feature vector $\mathbf{x}$. Next, a confusion class c_b is selected and the same process is performed to calculate $\mathcal{FI}_{i_b}$. A feature subset is then selected by taking features with the lowest 25% of feature importance divergences between the actual and confusion class, $\mathcal{FI}_{i_{dif}} = \mathcal{FI}_{i_a} - \mathcal{FI}_{i_b}$. These features represent those determined by the attribution model to be of similar importance to the DNN for both classes. The log-odds evaluation technique occults these features, then propagates data through the network to calculate the log odds with a mask vector, of all zeros, applied to the selected features. An empirical comparison is made between the original log odds value which measures the difference or delta between predictions for the actual versus confusion class without any masking, and the occulted log odds value which performs the same computations but using input feature vectors with low $\mathcal{FI}_{i_{dif}}$ masked. The final result is a *delta log odds measure* for each observation in a dataset ISIN-Date-Broker tuple, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$. Intuitively, the evaluation metric compares how well the importance scores identify the correct class.

Delta log-odds evaluation metrics are computed for four of the backpropagation-based attribution methods across the 802 unique FTSE100 CLOB DNN tuples. Note that LRP is not considered in this experiment. This analysis is performed for one action only, limit orders at the best ask, LO.A.P0, with ask market orders, MO.A, as the confusion class. The subset of features chosen to occlude is 25%. Various other class pairs, including LO.B.B0-MO.B and LO.A.P0-LO.A.P1 and masking levels, of 15% and 20%, are tested for robustness with relatively similar results. The results for the delta log-odds analysis is graphically depicted in Figure 4.2.3. Higher log odds indicate that the attribution method assigns higher importance scores to features that are critical to the DNN differentiating between the *actual* and *confusion* class. Evidently, Integrated Gradients using 100 discrete interior gradients marginally outperforms DeepLIFT using the RevealCancel rule algorithm with an average delta log-odds value of 12.1 compared to 11.85 for DeepLIFT. Both methods have similar boxplot distributions with a whisker value, equal to 1.5 times the inter-quartile range, equalling approximately 9 in the bottom range for both methods, with upper range values of 23.2 and 24.8 for DeepLIFT and Integrated Gradients, respectively. Gradient*Input also performed relatively well with an average delta log odds value of 9.98 across all sample DNN tuples evaluated. The log odds between the actual and occluded DNNs when features are masked using the Gradient method did not appear to be influenced significantly, with an average change of only 0.56 over the datasets.

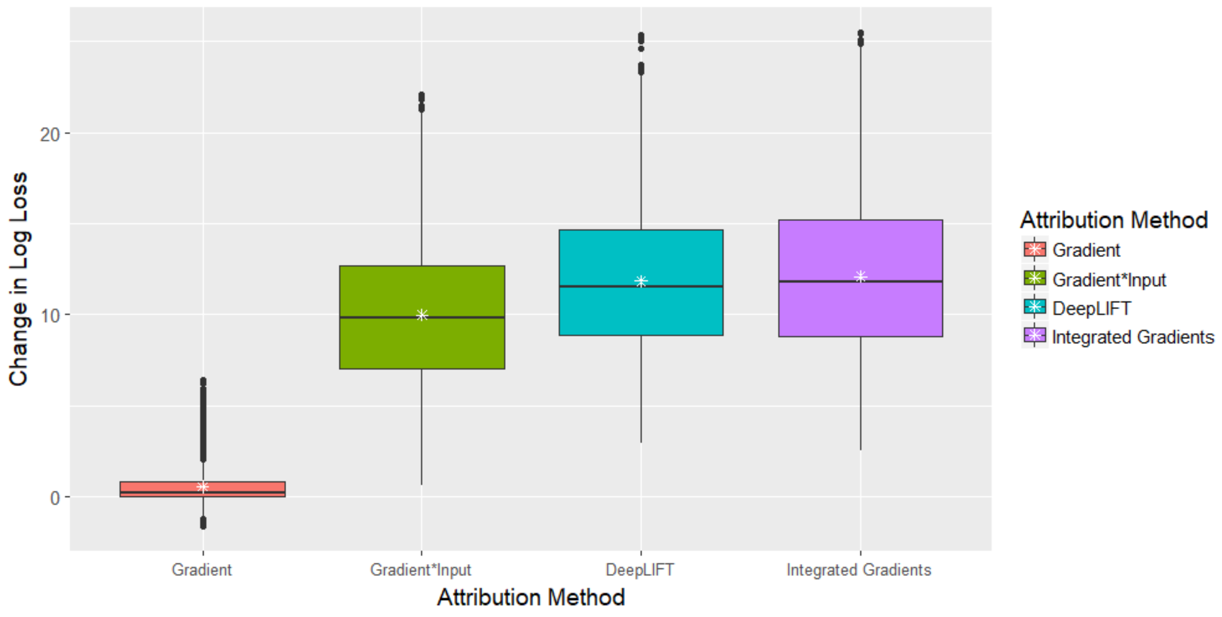

Figure 4.2.3 – Boxplot of delta log-loss for different feature attribution methods calculated over all $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$ tuple datasets and implemented using actions of limit order at best ask, LO.A.P0, as actual class and ask market order, MO.A, as confusion class. Mask is applied to 25% of features based on feature importance scores computed by each method.

4.2.3 Sensitivity-n

Axiomatic frameworks for feature importance have widely described the principle that the total raw non-normalised attribution for a subset of n features, $R_n^M = \sum_{i \in n} \mathcal{FI}_i^M$, for attribution method M, should equal the change in output value when those features are removed, masked or occulted, $A_n = \mathbf{y}_c - \mathbf{y}_c[\mathbf{x}_{i \in n} = 0]$, based on the axiom referred to in the literature as completeness (Sundararajan, 2017), summation to delta (Shrikumar, 2017) and Sensitivity-n (Ancona, 2018). This section analyses the five backpropagation attribution methods through an analytically tractable comparative framework using a metric that generalises the sensitivity-n axiom, allowing for the methods to be evaluated against one another to determine the utility of employing them on the FTSE100 CLOB datasets. The Sensitivity-n measure developed has similar attributes to delta log odds but derives from a different mathematical formulation (Ancona, 2018).

The Sensitivity-n evaluation technique samples a random subset of n features uniformly from the dataset, without assumptions regarding correlations. These features are then masked before feeding the input vector through the network and computing the deviation from the actual to new output value, or *delta output*, of the occulted DNN, A. Pearson correlation coefficients are measured between the delta output values and the total attribution of the n masked features, empirically determined for each attribution method M, culminating in the Sensitivity-n metric:

$$\rho_n^M = corr(R_n^M, A_n)$$

$$where\ R^M = \sum_{i \in n} \mathcal{FI}_i^M \quad and\ A_n = \mathbf{y}_c - \mathbf{y}_c[\mathbf{x}_{i \in n} = 0]$$

Sensitivity-n metrics are computed for all five backpropagation attribution methods, M, using random feature samples of n cardinality across the range (0,100) taken from the 323 features used in the optimised DNN. Metrics are evaluated for each DNN and averaged over all FTSE100 CLOB DNN tuples $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$. Attribution methods with higher Sensitivity-n values indicate that the importance scores have a stronger correlation with the impact on the output value when the features are masked. Thus, the attribution method with high Sensitivity-n more accurately attributes the importance of features in modelling the output. When n is zero the correlation will equal one, with correlations decreasing as the number of features masked increase, that is, a higher n is used.

Results are graphically reported in Figure 4.2.4. The monotonic decrease in Sensitivity-n values as the occluded sample subset size n increases aligns with the findings of Ancona (2018), given the logical assertion that DNNs trained without salient features produce more disparate predictions of the output. Evidently, the Integrated Gradient method performs best at feature attribution for the FTSE100 CLOB dataset based on the Sensitivity-n metric. Integrated Gradient attribution values have a strong positive correlation with the impact that these features have on the output value. Thus, when a subset of features n with summed attribution R^M are extricated from the DNN, the model's output predictor delta, A_n, shifts in a similar direction and magnitude as the attribution R^M. In other words, changes to an input feature value have a strong corresponding influence on the output when the Integrated Gradient feature importance values are high. Integrated Gradient methods maintain an average Sensitivity-n of 0.976 when a subset of 10 features are removed, which falls to an average of 0.812 when a sample n of 100 features is used. Both Integrated Gradient and DeepLIFT models tend to perform similarly over different cardinality of n, with DeepLIFT maintaining a correlation of 0.793 between summed attributions and delta output when n is 100. Layer-wise Relevance Propagation and Gradient*Input similarly attain correlated results with both measures performing relatively well until the cardinality of 10 features, at which point the correlation results tend to decrease sharply as more features are occulted from the dataset. These methods attain Sensitivity-n metrics of 0.663 and 0.631, respectively, when n equals 100 features.

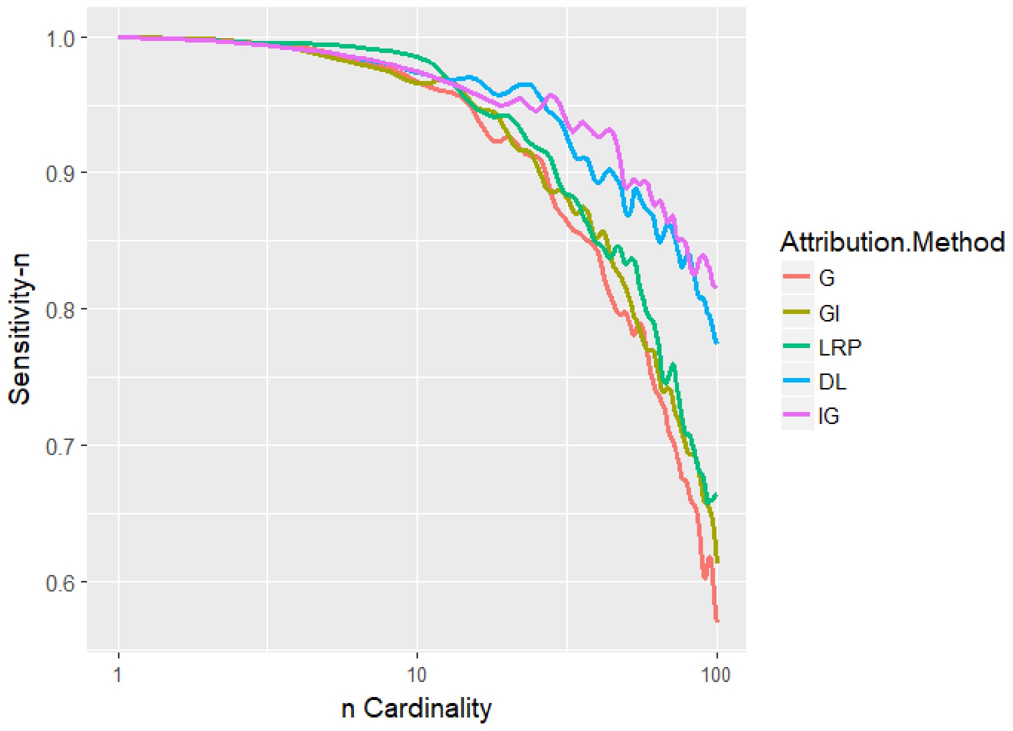

Figure 4.2.4 – Sensitivity-n metric computed for different attribution methods over n cardinality (0,100). The attribution methods analysed include Gradient (G), Gradient*Input (GI), Layer-wise Relevance Propagation (LRP), DeepLIFT (DL) and Integrated Gradients (IG). Cardinality refers to the number of features occluded from the dataset.

Evidently, both the axiomatic feature attribution methods of DeepLIFT and Integrated Gradients attain stable and relatively accurate measures of feature saliency illustrated by the relatively high Sensitivity-n metrics as the cardinality increases. These results align with the analysis performed in the previous saliency distribution and delta log-odds sections. Essentially, both these attribution methods provide unique but similar interpretations of how to solve the same problem of feature attribution in the FTSE100 CLOB dataset.

There are several primary similarities between the DeepLIFT and Integrated Gradients models that amplify their success. Both apply a backpropagation-based attribution approach with a focus on the partial derivative of the output w.r.t the input, $\partial \mathbf{y}_c / \partial \mathbf{x}_i$,. Furthermore, they take a global approach (Ancona, 2018) to quantify the marginal impact on the DNN output predictor, $\mathbf{y}_c$, from a minor perturbation in the input feature vector relative to a baseline or reference value, $\mathbf{x}_i - \mathbf{x}'_i$. The two techniques also solve the saturation problem (Shrikumar, 2016) endemic to previous backpropagation attribution methods. Both methods were also built using an axiomatic framework (Ancona, 2018) that measures the feature saliency values holistically in a way that meets various axioms theoretically important to feature attribution. Finally, both methods have additive properties (Lundberg, 2017) that allow individual raw input feature scores to be combined using mathematical operations. In the context of the algorithmic trader strategy identification procedure performed in the section after next, the additive property allows for various raw features to be amalgamated in a way that provides interpretability, compactness and explanatory power of the predicted LOB feature inputs for an individual Broker tuple, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$, trading algorithm, serving as the basis for segmenting firms by the algorithmic trading strategy being executed.

4.2.4 Feature Attribution Dataset

Integrated Gradient (Sundararajan, 2017) and DeepLIFT RevealCancel rule (Shrikumar, 2017) attribution methods are employed to quantify the degree of saliency or feature importance, $\mathcal{FI} = [\mathbf{r}_1, \dots, \mathbf{r}_n] \in \mathbb{R}^n$, for each of the n features, $\mathbf{x} = [\mathbf{x}_1, \dots, \mathbf{x}_n] \in \mathbb{R}^n$, which serve as input into the 802 unique FTSE100 CLOB ISIN-Date-Broker tuple, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$, optimised Deep Neural Networks (DNN), represented by the non-linear mapping function $\mathbf{y} = f(\mathbf{x}, \mathbf{w}, \mathbf{b})$.

The resulting dataset includes a feature importance, $\mathcal{FI}_i$, value for 323 LOB features used to train a DNN for a single broker $\mathcal{B}$ on a specific ISIN $\mathcal{S}$ and trading date $\mathcal{D}$. Feature importance values provide a machine learning-modelled insight into the transitional dynamics of UK equity markets, identifying LOB phenomena, $\mathbf{x}$, that have good explanatory power when predicting the next action a trader is going to execute, $\mathbf{y} \in \mathcal{A}$, given the current state. Furthermore, these importance values provide a measure of saliency for the LOB inputs into the trading algorithm of a market participant. Ultimately, an unsupervised machine learning model can be fitted to the importance values for each Broker tuple, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$, allowing them to be segmented into one of five algorithmic trading strategy categories – Automated Market Makers (AMM), Execution, Microstructural, Momentum, and Technical strategies.

Raw Integrated Gradient (IG) and DeepLIFT (DL) feature importance values are extracted from each sample action of the Broker tuple $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$ DNN. The raw values undergo a pre-processing procedure by summing the raw values over all actions the trader executes over the dataset, which is made possible due to the additive properties of both IG and DL (Lundberg, 2017). These additive properties enable feature importance values to be amalgamated using various frameworks and potentially could be used to measure importance values over even smaller intervals than the trading day used in this chapter, such as hours or minutes. For robustness, five separate optimised Broker tuple $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$ DNNs are trained and the IG and DL values are extracted and averaged over the five models. Furthermore, the raw feature importance values for each tuple are normalised with a Gaussian distribution, $\mathcal{N}(0,1)$, to have a mean zero centre and unit standard deviation. The pre-processing step culminates with a set of 323 normalised $\mathcal{FI}$ values for each Broker tuple that serve as input into the unsupervised machine learning clustering algorithms introduced in the next section to categorise algorithmic trading firms, which is performed in the next Section 4.3. The aggregate impact of these algorithmic trading strategies on UK equity market quality is then analysed in Section 4.4.

4.3 Algorithmic Trading Strategy Identification using Unsupervised Machine Learning

Experiments in this section utilise unsupervised machine learning techniques, specifically K-Means and its extensions, to cluster and classify each of the 802 multi-dimensional FTSE100 CLOB ISIN-Date-Broker DNN tuples, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$, into one of five defined strategy clusters based on empirical observations of feature attribution values, $\mathcal{FI}$, computed in the previous sections. Unsupervised learning algorithms find patterns in unstructured data using high-dimensional partitioning rules of the $\mathcal{FI}$ attribution values at the raw feature and grouped strategy level. Several competing unsupervised machine learning methods to K-Means, such as partitioning around medoids (PAM) and neural network-based self-organising maps (SOM) are considered. PAM executes a similar algorithm to K-Means, but rather minimises an arbitrary distance, generally absolute, rather than squared measures, when minimising the distance between centroids and data points to be clustered. Hence, PAM is more robust to outliers and noise, given that it does not penalise points by the square of their distance, which is less relevant for our dataset. SOMs are unsupervised learning methods that learn to visualise multi-dimensional datasets by using topological features to perform dimensionality reduction that scales the input feature vector into a lower dimensional space. Whilst they provide interesting two-dimensional representations of relationships between variables, we do not utilise this technique to partition data and define clusters, but rather rely on the K-Means unsupervised method.

This section introduces K-Means models capable of clustering traders by the five identified algorithmic trading strategies based on their raw feature importance scores. Four K-Means algorithms are defined in Section 4.3.1, before being evaluated for their ability to cluster the feature attribution data in Section 4.3.2. Finally, the most applicable method, Spherical K-Means, is applied to the data in section 4.3.3 resulting in the clustering of firms into five categories representing a specific algorithmic trading strategy. The algorithmic trading strategies are then empirically analysed for their impact on UK equity market quality.

4.3.1 K-Means Models

K-Means (MacQueen, 1967) is an unsupervised machine learning algorithm with the objective of clustering unlabelled n-dimensional quantitative input feature vector $\mathbf{x}_n$ into one of k clusters where $k \leq n$, with final cluster sets $\mathbf{K} = \{K_1, \ldots, K_k\}$. This chapter evaluates the applicability of the Hartigan-Wong K-Means algorithm (Hartigan, 1979) in clustering the FTSE100 CLOB feature attribution data. The algorithm minimises the squared Euclidean distance between data points and the cluster centre mean $\boldsymbol{\mu}_j$, which is equivalent to minimising the intra-class variance by reducing the sum of squared distance in n-dimensional space between partitioned data points and the clusters centre or centroid. K-Means iteratively assigns data points to the nearest cluster centroid in Euclidean space, $\|\mathbf{x} - \boldsymbol{\mu}_j\|^2$, before revaluating then updating centroids for each cluster. This process continues until the within-cluster variation between data points reaches an acceptable minimum. At this point the K-Means algorithm converges to an approximate solution, however the solution may converge at a non-optimal local minimum. Thus, ensemble methods or multiple repetitions of the algorithm may be required, selecting the iteration which minimises the within-cluster sum of squares (WCSS) K-Means objective function, defined as:

$$\underset{\mathbf{K}}{\arg\min} \sum_{j=1}^{k} \sum_{\mathbf{x}\in\mathbf{K}_j} \left\|\mathbf{x} - \boldsymbol{\mu}_j\right\|^2 = \underset{\mathbf{K}}{\arg\min} \sum_{j=1}^{k} |\mathbf{K}_j| Var(\mathbf{K}_j)$$

Cluster centre seeds are initialised using the K-Means++ algorithm before K-Means optimisation is performed (Arthur, 2007). This initialisation algorithm selects an initial centre at random before selecting a new data point centre based on their probability distribution and weighted by the square of the distance between the new data point and the initial centre. This process is performed iteratively until the k cluster centres data points have been assigned. By utilising a distance-based initialisation scheme for the centres, rather than a crude arbitrary selection like in the vanilla K-Means algorithm, the error of the model can be reduced (Arthur, 2007). K-Means provides an efficient approximation for the solution to the NP-hard problem of cluster partitioning of input data. However, vanilla K-Means algorithms tend to perform poorly for datasets with noisy variables which remain similar across multiple clusters (Buchta, 2012).

4.3.1.1 Sparse K-Means

Sparse K-Means (SKM) attempts to correct for the susceptibility of vanilla K-Means to split variables with similar attributes across different clusters when the dataset is inherently noisy, by performing adaptive feature selection during training of the algorithm and utilising weight penalties to allow sparse feature representations to be developed during clustering (Witten, 2010). The SKM model is most applicable for high-dimensional datasets given that the adaptive feature subset allows the algorithm to consider only the most relevant features, injecting sparse feature representations from which the algorithm learns the clusters, $\mathbf{K}$. Sparse K-Means requires a re-formulation of the WCSS minimisation problem of vanilla K-Means into an objective function that attempts to maximise the weighted between-cluster sum of squares (BCSS) subject to a series of weight constraints, $\mathbf{w}$. These weights are infused into the algorithm to drive sparsity in the complex dataset, comprising noisy data, by penalising outlier data points with low weights and adding excitatory signals with higher weights for features that fit well into a cluster. Weights are first initialised over the n LOB features to $1/\sqrt{n}$, before iterating through two alternating algorithms of (1) solving for the vanilla K-Means to optimise the centroids before (2) holding the cluster classes $\mathbf{K}$ fixed to maximise the BCSS with respect to weights $\mathbf{w}_j$ for each feature over all samples s, leading to:

$$\underset{\mathbf{K}}{\arg\min} \sum_{j=1}^{k} \sum_{\mathbf{x}\in\mathbf{K}_j} \left\|\mathbf{x} - \boldsymbol{\mu}_j\right\|^2 \quad (1)$$

$$\underset{\mathbf{w}}{\arg\max} \sum_{j=1}^{n} \mathbf{w}_j \sum_{i=1}^{s} \left\|\mathbf{x}_{ij} - \boldsymbol{\mu}_j\right\|^2 - \sum_{m=1}^{k} \sum_{\mathbf{x}\in\mathbf{K}_j} \left\|\mathbf{x} - \boldsymbol{\mu}_{jk}\right\|^2 \quad (2)$$

$$subject\ to\ \mathbf{w}_j \geq 0, \|\mathbf{w}\|^2 \leq 1, \|\mathbf{w}\| \leq s \quad \forall j$$

The algorithm culminates in the segmenting of samples into $\mathbf{K}$ clusters subject to corresponding weights $\mathbf{w}_j$ for each of the samples feature values $\mathbf{x}_{ij}$.

4.3.1.2 Spherical K-Means

Extended versions of the Spherical K-Means partitioning algorithm apply a fixed-point heuristic that iteratively assigns data points to their optimal cluster based on a cosine similarity distance metric, where the distance between the data point and a fixed prototype or cluster centroid is minimised, before then calculating the new optimal prototype or centroid for each cluster, analogous to the previously defined K-Means algorithm (Buchta, 2012). The algorithm employs a cosine dissimilarity distance metric, $D(\mathbf{x}_i, \mathbf{p}_j)$, which measures the angle between the i^{th} data point of the n-dimensional feature vector $\mathbf{x}_i$ and the prototype or centroid, $\mathbf{p}_j$, for $j \in \mathbf{K}$, in place of the Euclidean dissimilarity measure, $\|\mathbf{x} - \boldsymbol{\mu}_j\|^2$, used in vanilla K-Means. This implementation of the cosine similarity distance metric has been shown to attain improved clustering performance over Euclidean measures, with the intuition that both the direction and magnitude of feature values are critical elements in reassigning samples between clusters as opposed to merely its magnitude (Strehl, 2000). Thus, spherical K-Means performs clustering in a unit hypersphere with vectors normalised to unit length, $\|\mathbf{x}\| = 1$. Membership criterion variable $\mathbf{u}_{ij}^m$ is assigned the value one when the prototype falls within the class j otherwise zero, whilst the object weight $\mathbf{w}_i$ is a non-negative term weight representation that controls the importance of specific data points. Setting the criterion softness exponent parameter m greater than one allows for 'soft' spherical partitioning that relaxes the hard-spherical shape of the cluster (Buchta, 2012). The optimisation algorithm for variable prototypes sets the centroids, also referred to as prototypes, which minimises the following criterion function:

$$\arg\min_{\mathbf{p}} \sum\nolimits_{i \in n, j \in \mathbf{K}} \mathbf{w}_i \mathbf{u}_{ij}^m D(\mathbf{x}_i, \mathbf{p}_j)$$

$$where \quad D(x,p) = 1 - \cos(x,p) \qquad \mathbf{u}_{ik} = \begin{cases} 1 & if\ protype\ class\ equals\ j \\ 0 & otherwise \end{cases}$$

4.3.1.3 Hybrid Hierarchical K-Means

Hybrid Hierarchical K-Means clustering algorithms (Peterson, 2018; Nister, 2006) are a non-parametric approach that combines K-Means algorithms with hierarchical clustering to address the sensitivity to K-Means random initialisation, with the algorithm at risk of initialising into a weak region that results in more ambiguous clustering. Hierarchical K-Means clustering first develops a hierarchy of clusters that provide a series of k exemplar centroids for the dataset. These centroid data points, $\mathbf{p}_j$, defined for each of the k clusters, form the initial centroids of a vanilla K-Means algorithm that clusters spherical groups. Finally, an iterative optimisation procedure is performed to minimise the WCSS for cluster centroids allowing for general rather than merely spherical shapes, leading to the reassignment of samples into new clusters before the process is repeated. The primary deficiency in this method is that the number hierarchical clusters k is determined implicitly by the algorithm based on initial partitions of the data, thus, the algorithm is not robust to user selections of k. However, one can select a branching factor k that recursively clusters lower order sub-clusters of the data by combining branches of the clustering hierarchy. Ultimately, the algorithm performs quite similarly to vanilla K-Means, though it has been shown to accelerate clustering speed over K-Means, with no added risk of selecting sub-optimal centroids (Peterson, 2018).

4.3.2 Model Evaluation & Clustering Properties

K-Means models introduced in the previous section are evaluated for their applicability of solving the unsupervised machine learning problem of clustering the m FTSE100 CLOB ISIN-Date-Broker DNN tuples, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$, into one of five defined strategy clusters. The dataset for the m tuples is comprised of feature attribution values, $\mathcal{FI} \in \mathbb{R}^n$, for the n-dimensional feature vectors used to train the DNN. The resulting dataset is an $m \; x \; n$ matrix comprising 802 Broker tuples, m, and 323 LOB features, n, derived from the sample UK equity market data. This dataset is henceforth referred to as the *feature importance dataset*, or $\mathcal{FI}$ dataset for shorthand. Unsupervised K-Means clustering is used to cluster the 802 broker data vectors in the $\mathcal{FI}$ dataset into one of five algorithmic trading strategies defined in Section 2.3 – Automated Market Maker (AMM), Execution, Microstructural, Momentum and Technical strategies – based on the feature importance values calculated using Integrated Gradients and DeepLIFT.

This section first analyses the clustering tendency of the $\mathcal{FI}$ dataset, before testing whether the optimal number of clusters aligns with the objective of strategy clustering. Finally, a K-Means cluster validation and evaluation of the previously introduced model iterations is performed to determine the optimal K-Means partitioning model to use for the $\mathcal{FI}$ dataset.

4.3.2.1 Clustering Tendency

The additive properties of Integrated Gradient and DeepLIFT allows $\mathcal{FI}$ values to be summed over multiple raw features, $\mathbf{x}$, and group strategies, S, for each DNN tuple, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$, as explained in Section 4.2. Clustering tendency of the $\mathcal{FI}$ dataset is performed on both the raw features and grouped strategies to determine whether cluster analysis is in fact a feasible unsupervised machine learning technique for grouping different algorithmic trading firms based on their feature importance scores. The Hopkins statistic is a relevant statistical measure of clustering tendency and spatial randomness in a dataset, developed by comparing the $\mathcal{FI}$ dataset against a randomly generated dataset drawn from a uniform distribution (Lawson, 1990). The distance, $D_i(p_i, p_j)$, between each data point, $p_i \in \mathcal{FI}$, drawn from dataset $\mathcal{FI}$, and its nearest neighbour data point, p_j, is compared against a similar distance metric, $M_i(q_i, q_j)$, calculated for sampled artificial data points, $q_i \in \mathcal{Y}$, drawn randomly from a simulated uniformly distributed dataset $\mathcal{Y}$, which are measured against their closest real-dataset point, $q_j \in \mathcal{FI}$. The Hopkins statistic, H, is computed as a function of the average distance between real data points and the randomly generated data points, $\sum_{i=1}^{n} M_i(q_i, q_j)$, and the mean nearest neighbour distance in the actual $\mathcal{FI}$ dataset, $\sum_{i=1}^{n} D_i(p_i, p_j)$, such that:

$$H = \frac{\sum_{i=1}^{n} M_i(q_i, q_j)}{\sum_{i=1}^{n} M_i(q_i, q_j) + \sum_{i=1}^{n} D_i(p_i, p_j)}$$

The $\mathcal{FI}$ dataset clustered by raw individual features attains a Hopkins H statistic of 0.198 indicating that clusters are available in the $\mathcal{FI}$ dataset (Banerjee, 2004). The fact that the statistic is closer to zero and less than 0.5, statistically provides evidence against the null hypothesis, $H = 1$, that the dataset is uniformly distributed with no interpretable clusters. Thus, one can conclude that the $\mathcal{FI}$ dataset for raw feature values is clusterable and apply K-Means unsupervised machine learning techniques to partition the data. The Hopkins statistic for the $\mathcal{FI}$ dataset grouped by Strategy segments S is lower at 0.138, which is expected given the lower dimensionality of the dataset when features are grouped, making clusters easier to find, rather than attempting to separate a set of 323 raw features. The result indicates that amalgamating and normalising feature importance scores over algorithmic trading strategy groups improves the clusterability of the data, providing a useful alternative method to consider for clustering strategies rather than clustering based on raw feature importance values.

Visual inspection of the clustering tendency of the $\mathcal{FI}$ dataset is performed using a Visual Assessment of cluster Tendency (VAT) algorithm that projects an ordered dissimilarity image (ODI) using an ordered dissimilarity matrix (ODM). The ODM compares the distance between various data points or objects in the dataset. Distance in this context is a function of Euclidean distance between data points. Matrix ordering is completed using hierarchical clustering.

Figure 4.3.1 presents the ODI generated using the VAT algorithm for the raw feature dataset and when the feature dataset is grouped by strategies. Data points composed of the 323 feature importance scores for a single $\langle\mathcal{S},\mathcal{D},\mathcal{B}\rangle$ tuple data sample are represented on the diagonal. Given that these data samples are identical with a dissimilarity measure of zero, they are depicted as dark blue. The light shades of blue and white in both the figures indicates a higher level of dissimilarity between data points and a higher clustering tendency. As expected, amalgamating feature importance by the features relevant strategy for each DNN tuple illustrates a higher degree of dissimilarity, evident by the large number of cells with values over two, than when utilising raw feature values. This aligns with the results of the Hopkins statistical analysis. Evident for the figure, one can see clear degrees of separation and distance between the various Broker tuples, based on their feature importance values, indicating that the $\mathcal{FI}$ dataset can be sufficiently clustered.

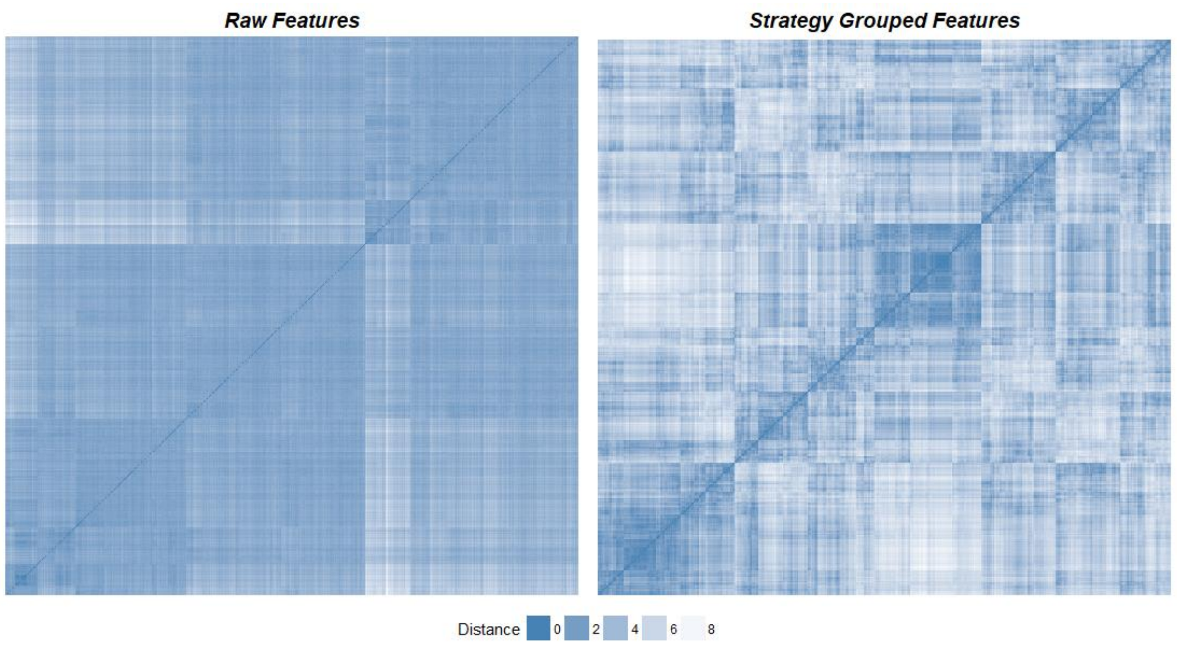

Figure 4.3.1 – Ordered Dissimilarity Image (ODI) for *raw features* (left image) and ODI for *features grouped by AT strategy* (right image), representing the distance between each unique feature attribution tuple $\langle\mathcal{S},\mathcal{D},\mathcal{B}\rangle$.

The low Hopkins statistics and visual inspection of the ODI indicates that the $\mathcal{FI}$ dataset has good clustering tendency, with data points for each Broker tuple separable in n-dimesional feature space.

4.3.2.2 Optimal Number of Clusters

Several methods are available in the literature for determining the optimal k or number of K-Means clusters given the underlying $\mathcal{FI}$ dataset to be partitioned. Optimal k requires a balance between high interpretability from compressing data into a minimal number of clusters and improvement in cluster accuracy and reduced intra-class variance when selecting a larger number of clusters. This section performs an optimal cluster analysis on the $\mathcal{FI}$ dataset with grouped strategy feature saliency values for all DNN tuples, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$. Methods for selecting optimal k can be categorised into (1) statistical testing methods including the Gap statistic, (2) direct methods which seek to optimise over a criterion, particularly, within cluster sum of squares (WCSS), which includes the elbow method, and (3) information criterion methods.

Gap Statistic (Tibshirani, 2001) methods are commonly used to choose an optimal k by selecting the smallest k that maximises the gap statistic, $Gap(k)$, up to the point where the increase in the gap statistic begins to plateau. Formally, this is defined as the cluster point where $Gap(k) \geq Gap(k+1) - s_{k+1}$, where s_{k+1} is one standard-error for the statistic, ensuring that model parsimony is balanced with the desire to maximise the gap statistic. The gap statistic is measured by comparing the log WCSS intra-cluster variation based on Euclidean distance for k clusters, $\log(W_k)$, and their expected values under a null uniform reference distribution calculated as the bootstrapped average intra-cluster variation from B random uniformly distributed reference datasets that have different numbers of clusters, such that:

$$Gap(k) = \frac{1}{B} \sum_{j=1}^{B} E^*\left[\log\left(W_{kj}\right)\right] - \log(W_k)$$

The gap statistic for the $\mathcal{FI}$ dataset is analysed with 100 reference datasets, $B = 100$, to extract robust results which are graphically depicted in Figure 4.3.2(a). The recommended number of clusters from the gap statistic is between 5 and 6 with a gap value of 0.41-0.42, at which point the increase in the gap statistic is less than s_{k+1}. Five clusters align well with the five algorithmic trading strategies analysed – AMM, Execution, Technical, Microstructural and Momentum. From the data, it become apparent that after nine clusters the gap statistic tends to plateau indicating that there is only a marginal benefit from increasing the number of clusters past this point, though at a cost of loss in interpretability of the K-Means models.

The 'elbow method' is a heuristic direct method used to confirm the interpretation of optimal clusters k taken from the gap statistic by plotting the percentage of variance or WCSS as a function of k. By selecting clusters that minimise WCSS, the data points will be compacted closely into regions, which themselves are separated in multi-dimensional space from other clusters or regions. The objective is to select the number of clusters at which point adding an additional cluster provides minimal gain in WCSS variance reduction and information, given the reduction in interpretability from adding the additional class. This point exists at the 'elbow' of the variance graph where the gradient tends to reduce. The gap statistic indicates that five clusters is an optimal choice with further evidence for this result provided in the elbow graph in Figure 4.3.2(b), which indicates that the information gained from the first five clusters is significant and the marginal gain of reduced WCSS from adding additional clusters may not be optimal.

Akaike information criteria (AIC) and Bayesian information criterion (BIC) for expectation-maximisation is an additional method for computing optimal k. Given that the K-Means model is similar to a Gaussian mixture model, a likelihood for the model can be estimated. The method selects models with the lowest BIC to compute the optimal number of clusters. BIC and AIC metrics are computed for the $\mathcal{FJ}$ dataset over clusters k of one to ten. AIC is grounded in information theory and is similar to BIC in that it provides an estimation of the relative quality of different statistical models such as K-Means models with varied number of clusters. AIC is calculated using a penalty for the number of parameters in the model, $2p$, whilst BIC uses the penalty $\ln(n)\,p$ where p is a fixed term. Both models have a similar formulation that apply the penalty to the conditional probability of the n-dimensional input vector $\mathbf{x}$ given the set of parameters, $\hat{\theta}$, that maximise the likelihood function for the model analysed. Figure 4.3.2(c) presents the BIC and AIC metrics for a K-Means model fitted to the $\mathcal{FJ}$ dataset as a function of k. In line with previous results, an 'elbow point' is evident around the five-cluster region where the marginal decrease in AIC as k increases begins to plateau. Furthermore, the BIC metric is minimised when k is around five before increasing in value as the number of clusters grows. Potential drawbacks of using BIC metrics to determine optimal clusters is that they are less capable of processing complex or high-dimensional datasets, though the $\mathcal{FJ}$ dataset is a relatively stable and simple dataset, with less than 300,000 data point elements in total. Analysis of the comparative benefits of using AIC versus BIC to approximate the optimal number of clusters has been mixed and depends on the assumptions of the model and dataset employed (Vrieze, 2012).

An additional analysis is completed by fitting a series of BIC K-Means algorithms with varied attributes in regard to how the clusters are initialised, the shape of the cluster, whether cluster volumes must be equal or can vary and the assigned orientation of the representation space. Results in Figure 4.3.2(d) are presented for eight common models, including an equal volume clusters with varied shape and identity orientation (EVI), equal volume clusters with same shape and orientation clusters (EEV) and variable volume clusters with equal shape and coordinate axes assigned orientation (VEI). One can see that the BIC for a range of models indicates an optimal number of clusters between five and ten, though most of the models offer only marginal improvement on the BIC after five clusters, providing further evidence that five clusters is a near optimal k.

$$AIC = 2p - 2\ln(\hat{L}) \qquad BIC = \ln(n)\,p - 2\ln(\hat{L}) \qquad \hat{L} = P(\mathbf{x}|\hat{\theta})$$

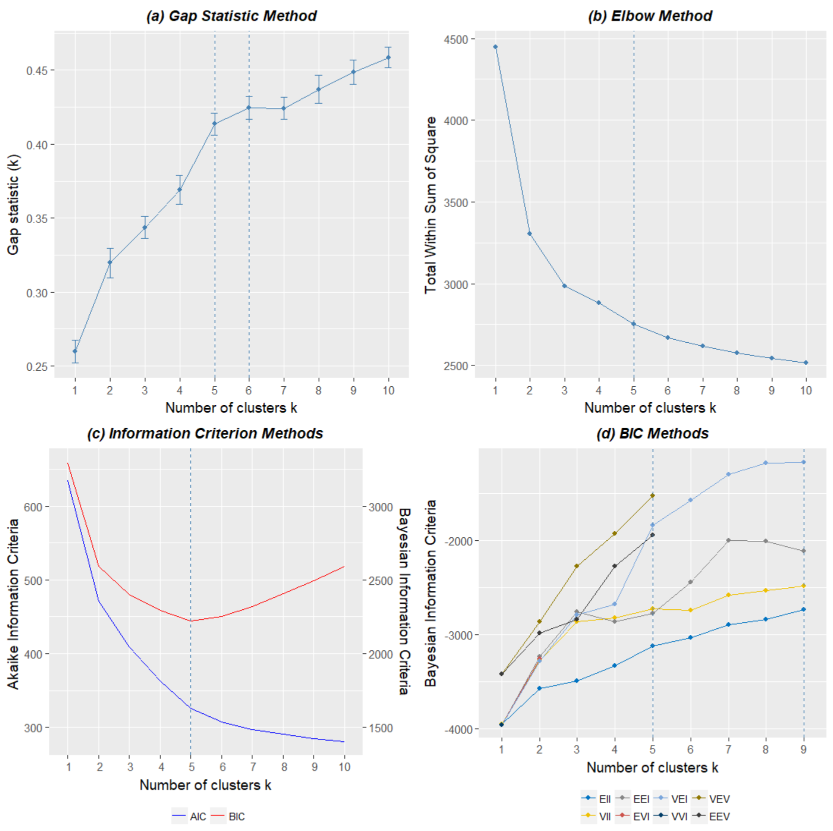

FIGURE 4.3.2 – Optimal number of clusters for the FTSE100 dataset using the (a) Gap-statistic method, (b) 'elbow' method, (c) Information Criterion Methods, and (d) various Bayesian Information Criterion Methods.

4.3.2.3 Cluster Validation for K-Means Partitioning

Cluster validation is a technique used to evaluate, compare and select the optimal K-Means based cluster partitioning model, defined in the Section 4.3.1, based on their capability of effectively separating the 802 DNN tuples, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$, into five cohesive and independent algorithmic trading strategy categories. Validation involves an evaluation of the K-Means models' ability to control the trade-off between maintaining distinct clusters and ensuring intra-cluster data points have low dissimilarity attributes. The K-Means models tested include the Hartigan-Wong K-Means algorithm (Hartigan, 1979), Sparse K-Means (Witten, 2010), Extended Spherical K-Means (Buchta, 2012), and a hybrid Hierarchical K-Means clustering algorithm (Peterson, 2018).

Several methods of internal cluster validation are applied to the K-Means algorithms to evaluate their internal clustering structure of the $\mathcal{FI}$ dataset by measuring both the compactness and separation of connections between clusters. Compactness requires cohesive clusters that have a low measure of within-cluster variation or distance between intra-cluster data points. Separation is a competing objective to compactness and requires clusters to be segregated in high-dimensional space by maximising the distance between cluster centroids and the outlier data points of the nearest neighbour cluster. Distance measures employed in this section are generally based on cosine cross-distances, $D(x, p) = 1 - \cos(x, p)$, between data points, x, and other points of interest, p, such as a nearest neighbour cluster centroid or outlier value.

Spherical K-Means (SKM) tends to perform best at partitioning the $\mathcal{FI}$ feature importance dataset into separate and distinct clusters of five algorithmic trading strategies, according to results of the comparative evaluation of different K-Means algorithms presented in Table 4.3.1. SKM attains the lowest within-cluster sum of squares (WCSS) with a value of 15.63 indicating SKM algorithms find a more optimal minimum of the K-means objective function during training than other comparative methods. Sparse K-Means (SK) performs poorly with a WCSS of 30.63. One possible explanation is the utilisation of complex and abstract representations of LOB features required for inputs into most algorithmic trading strategies. SK uses sparse representations and dynamic feature sub-setting to fit a K-Means algorithm that may perform well for high-dimensional data with minimal interaction effects between features, but performs poorly for splitting the FTSE100 CLOB $\mathcal{FI}$ dataset. Both vanilla K-Means (KM) and Hierarchical K-Means (HKM) perform relatively similar, in line with the literature (Peterson, 2018), attaining WCSS scores of 16.90 and 17.46, respectively, though both have higher WCSS than SKM. The fact that HKM is slightly less capable of minimising WCSS relative to the KM algorithm is expected given that HKM methods are highly sensitive to initialised values and are unlikely to partition the data as successful for two reasons. First, HKM is not as capable as K-Means at capturing critical information inherent in feature interactions. Second, the dataset lacks any single subset of feature variables capable of explaining an algorithmic trading strategy sufficiently, an agreeable property for well-functioning HKM partitions.

	K-Means Algorithm			
Metric	KM	SKM	SK	HKM
Min Cluster Size	145	140	72	149
Mean Cluster Diameter	0.77	0.71	0.80	0.80
Max Cluster Diameter	0.93	0.92	0.97	0.96
Mean Distance Within Clusters	0.15	0.12	0.20	0.16
Mean Distance Between Clusters	0.43	0.45	0.43	0.43
Within Cluster Sum of Squares	16.90	15.63	30.63	17.46
Mean Silhouette Width	0.42	0.43	0.21	0.42
Normalised Gamma (G)	0.52	0.53	0.48	0.52
Dunn Index (D)	1.36	1.61	1.04	1.36
Entropy of Cluster Distributions (E)	1.60	1.75	1.44	1.65
WB Ratio (WB)	0.36	0.32	0.46	0.36
Calinski-Harabasz Index (CH)	775.85	854.02	331.30	774.06

Table 4.3.1 – Cluster validation metrics comparing four K-Means algorithms applied to the feature importance dataset.

Additional statistical and heuristic measures are utilised in cluster validation to evaluate different K-Means models with the measures reported in Table 4.3.1. The Normalised Gamma metric, G, is a correlation coefficient that measures the dissimilarity between clusters (Halkidi, 2001). From the table one can note that SKM models attain the highest Normalised Gamma value of 0.53, showing that the clusters are better separated than other models. Dunn Index metrics, D, measures the minimum mean dissimilarity between two different clusters relative to the maximum mean within cluster dissimilarity across all clusters (Dunn, 1973). Valid clusters tend to have higher inter-cluster and lower intra-cluster dissimilarities leading to higher Dunn Index values. Following from previous results, SKM once again performs best with a Dunn Index value of 1.61, representing the fact that data points are compressed within clusters and that clusters are well separated. The entropy between cluster distributions is also measured to attain metric E, with results demonstrating that all models have relatively similar entropy measures, though once more SKM is marginally higher than comparative models. A ratio, WB, of the mean within cluster variance to the mean between cluster variance for each model is also calculated. The normalised inverse of these variables is additionally measured by the Calinski-Harabasz Index (Calinski, 1974), frequently referred to as the pseudo F Index. As expected, SKM models perform best with the lowest WB ratio of 0.32 and highest Pseudo F Index of 854.02. Evidently KM and HKM models, which are slight augmentation of the vanilla K-Means algorithm, perform relatively similar during cluster validation. Validation of the different models' compactness and separation are also studied.

Regarding the compactness of clusters, SKM attains the lowest mean and maximum cluster diameter of 0.71 and 0.92, respectively, among all K-Means models tested, compressing the data points within tighter regions in n-dimensional feature space. Tighter clusters indicate that the data points in the same cluster, representing the feature importance value for the 802 algorithmic trader tuple datasets, are relatively similar compared to KM and HKM. Thus, traders defined within each cluster are executing similar actions when faced with a given LOB state, indicating congruency in the strategies being executed. Furthermore, SKM attains a mean distance within clusters of 0.12 which is 25% lower than the nearest model, vanilla K-Means.

Cluster validation with respect to the structural separation of the data indicates that all models attain relatively similar mean distances between clusters indicating that the separation of centroids across the five models are comparable in terms of distance. SKM does attain the highest separation of 0.45 on average across clusters, only marginally better than the 0.43 attained by the other three comparative models. An additional method for validating intra-cluster consistency by measuring separation between clusters is to display silhouette plots (Rousseeuw, 1987). These plots provide an individual measure per data point i of the silhouette width, s_i, that compares its intra-cluster cohesion, measured as a function of the average distance b_i of each point i from other points in the cluster and the average separation or dissimilarity of i to the nearest neighbour cluster a_i, such that:

$$s_i = \frac{b_i - a_i}{\max\{a_i, b_i\}} \qquad s = \frac{1}{m}\sum_{i=1}^{m} s_i \qquad \textit{where } m \textit{ is the number of data points}$$

From Table 4.3.1, in conjunction with the set of Silhouette plots presented in Figure 4.3.3, one can see that SKM attains the highest mean silhouette width of 0.43 which is marginally higher than vanilla K-Means. Higher values of s indicate that data point clustering is correctly configured with data points well matched within-cluster and matched poorly to the nearest neighbour cluster. From the figure one can also note that the SKM silhouette plot shows that almost all data points for each cluster have a positive s_i value indicating well configured clustering and suitable selection of k, whereas for other K-Means models the density of negative silhouette values is slightly higher, which indicates data points are potentially incorrectly clustered. Once again, the SK model attains poor results with a mean silhouette width of 0.21 demonstrating that most data points fall between two or more clusters. Finally, narrower silhouette widths in the HKM algorithm potentially indicate that too few clusters have been selected which again reinforces the limitations of hierarchical-type models in partitioned data.

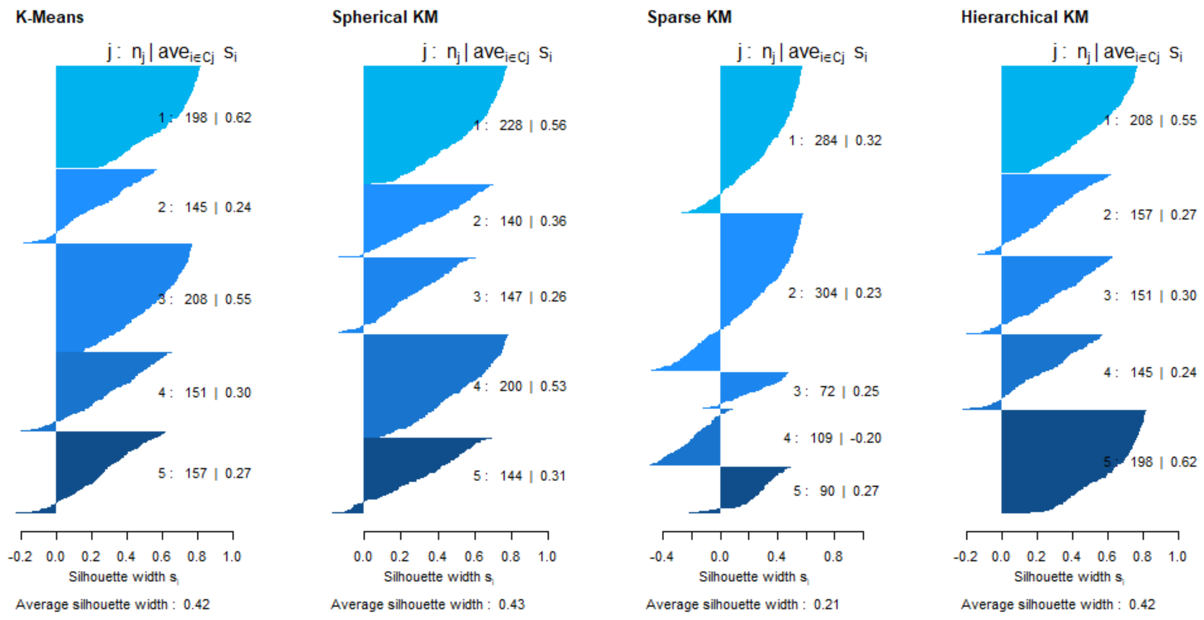

FIGURE 4.3.3 – Silhouette plots for each K-Means algorithm evaluated.

From the results presented in this model validation and evaluation section one can draw a conclusion that the FTSE100 CLOB $\mathcal{FI}$ dataset, composed of attribution measures for the 802 DNN tuples, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$, has a good clustering tendency with an optimal number of clusters of around five, aligning with the objective of this chapter to segment DNN tuples by the five trading strategies identified. Furthermore, Spherical K-Means models perform significantly better than comparative K-Means methods based on the cluster validation evaluation performed and therefore, represent the model utilised in segmenting algorithmic trading strategies in the next section.

4.3.3 Algorithmic Trading Strategy Identification

Spherical K-Means clustering is used to partition individual $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$ tuple observations in the $\mathcal{FI}$ dataset into five semi-homogenous algorithmic trading strategies – Automated Market Making, Execution, Technical Microstructural and Momentum strategies – based on structural patterns in the dataset. The clustering algorithm results in a strategy label for each tuple. The following sections evaluate the efficacy of the unsupervised K-Means algorithm at successfully portioning the $\mathcal{FI}$ dataset.

4.3.3.1 $\mathcal{FI}$ Structural Matrix

The $\mathcal{FI}$ dataset of 802 individual $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$ DNNs with attached feature attributions are well-partitioned into the five algorithmic trading strategies defined in Section 2.3 by using Spherical K-Means unsupervised machine learning algorithms to cluster the individual data points. The culmination of the clustering algorithm is an algorithmic trading strategy label attached to each DNN tuple $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$. Table 4.3.2 produces a summary statistic matrix for each of the five trading strategies listed in the left column, along with the number of tuple observations of the clustered strategy. The average $\mathcal{FI}$ percentage scores for normalised raw features related to specific algorithmic trading strategies, listed along the header of the matrix, are computed and averaged for each of the Broker tuples clustered into one strategies defined along the vertical axis. These scores represent the percentage of the trading strategy explained by normalised Integrated Gradient and DeepLIFT feature attribution values amalgamated for features relevant to the specific strategy. For example, the 19.5% value in the first row and fifth column is a representation of the saliency of Technical strategy features, such as stochastic oscillators and Bollinger Bands, for traders clustered using Spherical K-Means into the Automated Market Maker strategy segment.

The resulting table matrix provides evidence that the clustered algorithmic trading strategies are well defined principally by the relevant features identified as being important inputs into that strategy's algorithm. The separation of different strategies into semi-homogenous clusters based on LOB and financial market features is evidenced by the high weight, and dominance, of the input features relevant to that strategy in explaining trader actions on the LOB. This is represented by relatively high values along the diagonals vis-à-vis values on the horizontal axis. All diagonal values attain a minimum percentage of 30%, indicating that the strategies relevant features principally explain trader actions. Momentum, AMM and Microstructural strategy clusters tend to perform best at segmenting algorithmic traders by the features defined as important to these strategies in theory and academic literature. All three strategies attain a minimum 40% of feature importance from relevant strategy features.

		$\mathcal{FI}$ Value (%)				
Spherical K-Means Strategy	Count	AMM Features	Execution Features	Microstruct. Features	Momentum Features	Technical Features
AMM Strategies	144	**41.8%**	17.3%	8.1%	13.3%	19.5%
Execution Strategies	228	19.2%	**30.4%**	23.1%	1.2%	26.0%
Microstructural Strategies	140	19.8%	14.5%	**40.0%**	17.2%	7.5%
Momentum Strategies	200	19.7%	9.5%	10.7%	**48.1%**	12.0%
Technical Strategies	147	4.2%	21.3%	19.1%	24.6%	**30.7%**

Table 4.3.2 – Confusion Matrix with clustered strategies in the vertical column and the relevant feature importance scores for features that related to each individual strategy along the horizontal axis. Diagonals represent the percentage of total importance scores for features that relate to the clustered strategy.

Taking strategies partitioned into the Automated Market Maker's cluster as an example, these firms have an average normalised $\mathcal{FI}$ percentage value of 41.8% for features determined to be relevant inputs into AMM algorithms. These features include the traders relative quote offsets and positions, inventory positions and control, risk aversion and market-making relevant trading costs. AMM clustered trading firms $\mathcal{FI}$ importance values for the other non-AMM LOB features have relatively lower saliency than the AMM features, with only Technical strategy and Execution strategy features attaining $\mathcal{FI}$ attribution percentage values over 15%. Thus, firms conducting a primarily AMM strategy still place relative importance on some Execution features, such as market impact metrics or resilience measures, and Technical strategy relevant features, including price reversion, Chaikin volatility and stochastic oscillator features. These results directly align with a primary hypothesis of this thesis that firms conducting algorithmic trading over a multitudinous array of lit and dark equity LOBs no longer execute vanilla trading strategies based on traditional concepts, such as inventory-management market making. Rather traders conducting strategies such as market making infuse into their algorithms other features and variables, such as whether latency arbitrage opportunities are available across trading venues, whether short-term momentum signals are prevalent, or measures of the spread and depth of market liquidity, that may spur a certain action given the state of the LOB. Thus, in certain situations it becomes pertinent for market makers to perform an action that may be antithetical to theoretical tenants of the strategy, but ultimately aligns with the profit and business objectives of the firm, by placing importance in these situations on various other features that are conceptually more relevant for competing rather than traditional automated market making strategies.

Further analysis regarding the ability of Spherical K-Means to cluster firms by relevant strategy features requires the computation of a dissimilarity or confusion metric that represents the group of alternative strategy features that best explain traders' actions conducting a specific cluster strategy. Table 4.3.3 presents the results for the nearest alternative strategy measured by absolute percentage distance, dissimilarity importance values and difference between the strategies $\mathcal{FI}$ importance values for the cluster and alternative strategy. In line with the results above, AMM, Microstructural and Momentum strategies seem to be well clustered given that the features relevant for these particular strategies provide a good level of explanation for traders' actions and assumed inputs into their algorithms. These strategies have a minimum dissimilarity difference of 20%, representing a minimum 20% difference between the cluster and alternative strategy features on average. Evidently, these three strategies are very well clustered by their relevant features, for example, the actions on the LOB of algorithmic traders conducting Microstructural strategies may be heavily influence by quoted spread and limit order imbalances, Momentum strategies by high-frequency momentum and acceleration on the LOB, and AMM strategies by inventory and quote offset positions, among others.

Technical and Execution strategies on the other hand stand out with a dissimilarity difference measure of only 6.1% and 4.4%, respectively. From the data, Execution strategies commonly employ Technical strategy features, such as stochastic oscillators, Bollinger Bands, and various technical price and volume indicators. Intuitively, firms conducting Execution strategies take as input into their algorithms features such as Technical indicators when seeking to liquidate a client's parent order quickly and with minimal market impact. Technical analysis of market prices, particularly high-frequency indicators such as Relative Strength Index and Moving Average Convergence Divergence, will aid the Execution algorithms placement strategy.

Overall the results show good separation between the clustered strategies as evident by the high feature importance of relevant features to that strategy. However, the results also provide evidence towards the hypothesis that firms do not conduct any one pure traditional algorithmic trading strategy but rather generally execute some form of hybrid strategy skewed towards specific features of the clustered strategy.

Cluster Strategy	Alternative Strategy	$\mathcal{FI}$ Value	Dissimilarity	Difference
AMM	Technical	**41.8%**	19.5%	22.3%
Execution	Technical	**30.4%**	26.0%	4.4%
Microstructural	AMM	**40.0%**	20.8%	20.2%
Momentum	AMM	**48.1%**	19.7%	28.4%
Technical	Momentum	**30.7%**	24.6%	6.1%

Table 4.3.3 – Dissimilarity matrix for clustered strategies. Alternative strategy represents the strategy with the highest summed raw importance values different to the clustered strategy.

The mean $\mathcal{FI}$ feature importance value for the clustered strategy is compared against the confusion strategy for the 802 individual $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$ DNNs using a Welch two-sided statistical t-test to determine whether the means of the groups differ. Welch tests are deployed to account for potential differences in each groups variance (Ruxton, 2006), with the assumption of normality for each groups distribution. The null hypothesis that the means of the clustered and confusion feature importance values are equal is rejected at the 0.001% significance level. Results for the t-test are presented in Table 4.3.4 with the boxplot for cluster and dissimilarity strategy feature importance values shown in Figure 4.3.4. The test returns a t-value of 23.01 supporting the hypothesis that the difference in means is not equal to zero at the 0.001% confidence level, indicating a statistical difference between means. Evidently, Spherical K-Means clustering has resulted in a good separation of the strategies, with the order placement activity of traders conducting the cluster strategy primarily driven by the features relevant to that strategy. This is further confirmed by the divergence in feature importance values evident in the boxplot figure.

Welch Two Sided T-test	Value
Test statistic (T)	23.01***
Degrees of Freedom	1266

Table 4.3.4 – Two-sided Welch t-test comparing the difference between the mean of the clustered strategy related feature importance scores with the dissimilarity strategies feature importance scores. The test is performed across all strategies, stocks and dates in the original dataset. *, ** and *** correspond to statistical significance at 0.05, 0.01 and 0.001 levels respectively.

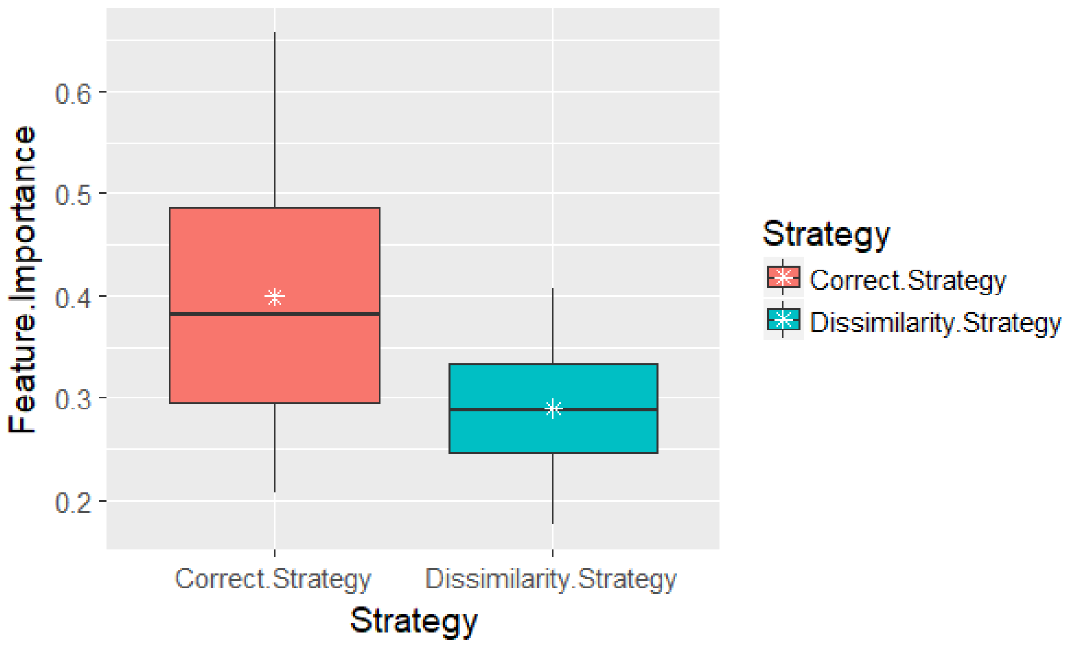

Figure 4.3.4 – Boxplots illustrating feature importance scores for the clustered strategy and dissimilarity strategy across all strategies, stocks and dates in the original dataset.

To confirm Spherical K-Means' separation of strategies, the spreads of the five algorithmic trading strategies across ISINs is evaluated. The results presented in Table 4.3.5 indicates that strategies are well partitioned into separately identifiable groups with proportional sizes to the specific ISIN's trading activity. All five ISINs studied in this chapter have a minimum of 19 Brokers categorised into each algorithmic trading strategy.

			Spherical k-Means clustered Algorithmic Trading Strategy				
ISIN	Dates	Count	AMM	Execution	Microstruct.	Momentum	Technical
GB0002875804	5	144	26	36	19	37	26
GB0005405286	5	178	37	57	25	35	24
GB0007980591	5	179	22	46	25	46	40
GB0009252882	5	170	29	39	32	38	32
GB00BH4HKS39	5	188	30	50	39	44	25

Table 4.3.5 – Number of clustered strategies for each ISIN over the five trading dates in the FTSE100 dataset.

4.3.3.2 Dimensionality Reduction & Cluster Visualisation

Following the strategy identification process, dimensionality reduction is performed to allow for cluster visualisation of the n-dimensional $\mathcal{FI}$ dataset in both two and three-dimensional space. Principal Component Analysis (PCA) is performed on the $\mathcal{FI}$ data to identify correlations between feature importance variables that allow an orthogonal linear transformation of the original data through a projection onto a smaller dimension of principal components. The number of principal components p must be less than or equal to the dimensionality of the dataset where n is the number of LOB raw or strategy features. PCA with both two and three dimensions is performed to reduce the $\mathcal{FI}$ feature set into interpretable dimensions. Transformation of the data requires the selection of principal components or dimensions that minimise the variance in the data, under the constraint that components are orthogonal to preceding adjacent components, with the process performed iteratively by adding principal components until p is equal to n and the variance is reduced to zero. The graphic in Figure 4.3.5 measures the reduction in variance as the number of principal components increases. The first principal component selected is that with the maximum variation on points projected from the original dataset onto a new principal p-dimensional axis, which is based on the direction and eigenvalues of the original features. There is a clear reduction in variance from 1.34 for one principal component to 0.58 when five principal components are included. These principal components are utilised to cluster original feature importance values of each clustered strategy into interpretable dimensions.

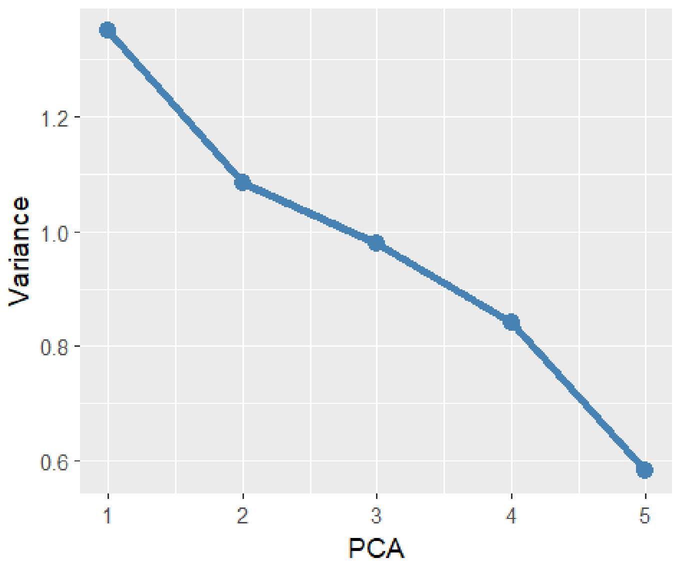

Figure 4.3.5 – Reductions in variance as the number of principal components increases.

Figure 4.3.6 represents a two-dimensional ellipsoid fit to the $\mathcal{FI}$ feature importance data with the horizontal axis being the first principal component that explains 36.4% of the $\mathcal{FI}$ data variance and the vertical axis representing the second principal component which explains 23.5% of variance. Each dimension can be viewed as eigenvectors of the covariance matrix between features of the dataset, with the two eigenvectors together representing less than 60% of the total information computed by summing all eigenvectors. The cluster plot of feature importance for each strategy projected into two-dimensions indicates that the strategies are relatively well-separated, except for the Momentum strategy that appears to cross the outlier barriers of the other strategies and reach towards their centroid values. Visualising the clusters by truncating the transformation into three principal components can be performed with a three-dimensional scatter plot, which is depicted for the AMM, Microstructural and Momentum strategies in Figure 4.3.7. This visualisation identifies more clearly how the strategies begin to show good separability and compression as the dimensions and principal components of the analysis increases. All three strategies occupy a separate segment of the three-dimensional principal component space with well-defined and comprehensible boundaries. Whilst PCA is a useful visualisation tool for confirming that strategies are well clustered, it must be noted that the factors of the PCA are based on the total variance in the $\mathcal{FI}$ dataset rather than other underlying statistical properties, thus limiting the interpretation of the analysis. To provide a more robust analysis, the next section studies the clustering ability of Spherical K-Means at the raw feature-level.

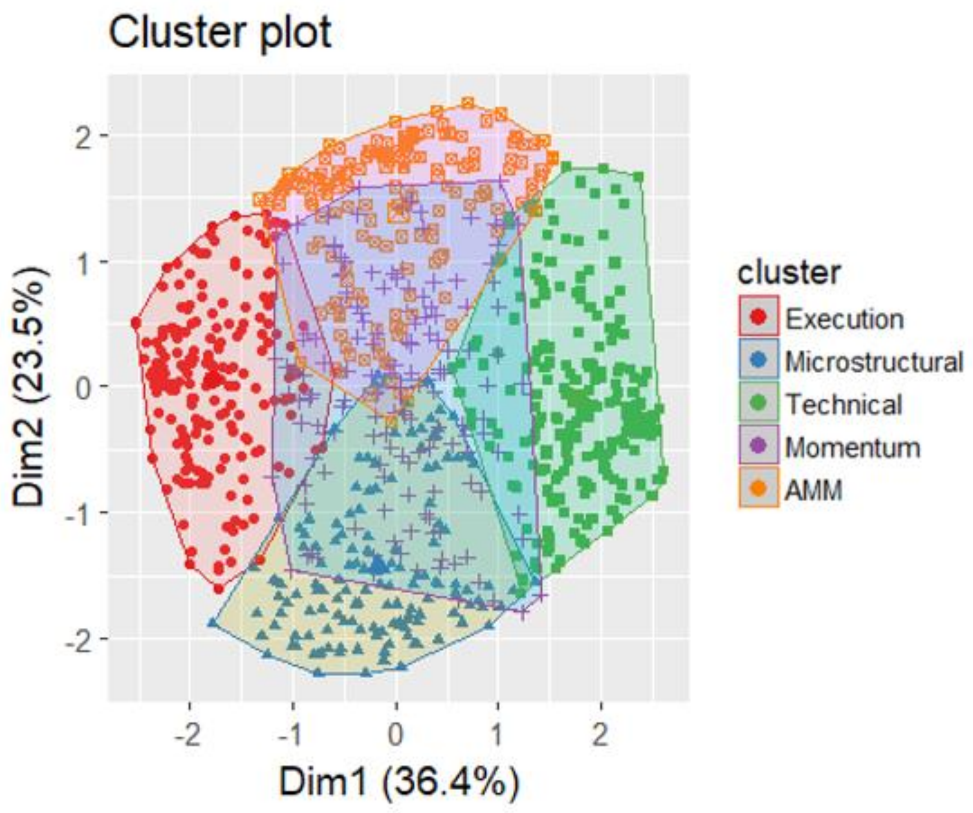

Figure 4.3.6 – Two-dimensional scatter plot of the two highest principal components, representing 59.5% of variance in the $\mathcal{FI}$ dataset. Dim refers to the relevant principal component. Scatter dots represent an individual firm clustered into one of five defined strategies.

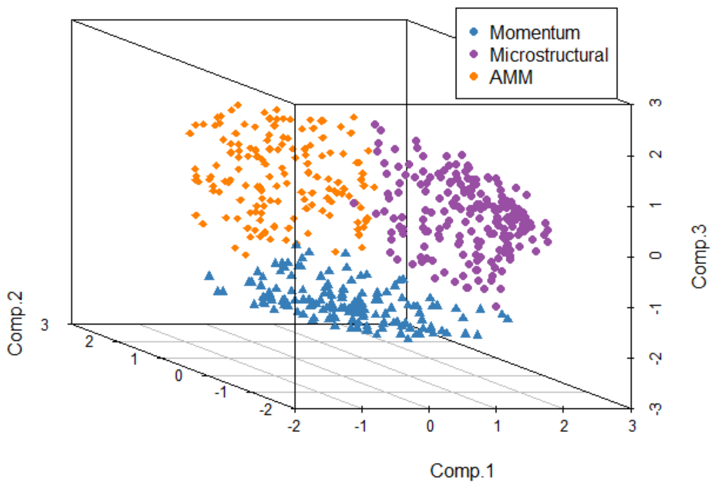

Figure 4.3.7 – Three-dimensional scatter plot of the three highest principal components. Comp refers to the relevant principal component. Scatter dots represents an individual firm clustered into one of three strategies – AMM, Momentum, and Microstructural.

4.3.3.3 Robust Analysis of Strategy Clustering

Spherical K-Means has been shown to cluster the AT strategies relatively successfully, though a deeper more robust analysis is performed in this section to confirm that the raw features attached to the individual AT strategies, as per the theory set out in Section 2.3 and feature definitions in Appendix A, corresponds with the feature importance values attained by firms clustered into those strategies. The primary approach is to compare the distributions of several features that are deemed as important to a specific AT strategy and analyse whether these raw features have high saliency values for firms clustered into that strategy. Both a qualitative analysis and statistical testing approach are taken.

Figure 4.3.8 graphs the density distributions for 10 features – 2 features defined as important for each of the five AT strategies assessed. The distributions for feature importance values, $\mathcal{FI}$, are compared for firms that are clustered by the Spherical K-Means algorithm into the *actual* strategy cluster related to that feature and firms that are clustered into an alternative strategy. For example, the first comparative distribution plot in the top left corner examines distributions of feature attribution values for quote offset on the ask features, QOA, separated by firms clustered as AMMs which are expected to have high attribution values versus firms clustered as one of the other strategies, including Execution, Microstructural, Momentum and Technical. The headings of each density plot indicate the relevant strategy and the attached raw feature value. Evidently, all distributions show a clear divergence between the labels with the raw feature assumed to be critical to the relevant AT strategy having feature importance values with distributions shifted to the right into the positive region. For example, the feature density plots for realised market impact, PIRA.X10, a critical LOB feature for firms attempting to liquidate a client order under an agency agreement, demonstrates that the average feature importance values for firms clustered in the Execution AT category are generally greater than zero, indicating that the features are important inputs into the firms DNN and assumed trading strategy. In contrast, the non-Execution AT clustered strategies have a distribution with a peak in the negative region demonstrating the lower relevance placed by these strategies on this particular measure of realised market impact. A similar relationship is prevalent in the other nine features studied.

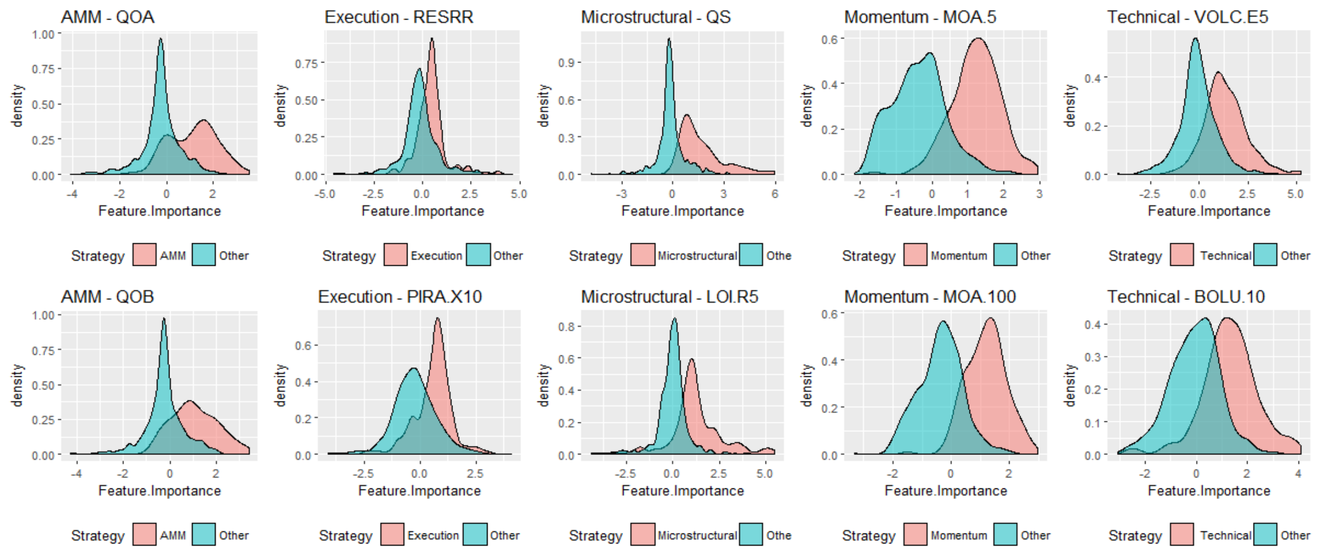

Figure 4.3.8 – Comparative density plots for ten LOB features based on feature importance scores. The red density plots represent feature importance values for firms clustered into the strategy to which the feature being studied most directly is related to. For example, quote offset on the ask side, QOA, relates to an AMM strategy, thus, red values represent feature importance values for QOA for firms clustered into the AMM strategy. Conversely, the green density plots represent feature importance values for all other firms.

Statistical hypothesis tests are applied to the dataset using a Welch t-test to compare whether the means of the two distributions are different across a range of features. That is, whether the average feature importance for firms clustered into the strategy relevant to the feature is significantly different to the average of the raw feature value for other firms. An analysis of all LOB features studied in this chapter found that 91.4% of the features had a higher feature importance value mean for the firms clustered as executing the strategy related to that feature compared to other AT firms. Furthermore, the difference in means is statistically significant at the 1% level for 72% of features and at the 5% level for an additional 7% of features. Table 4.3.6 reports the summary results of the analysis for the ten features studied in this section which all show a statistically significant divergence in means at the 0.001% level.

		$\mathcal{FI}$ Average		
Strategy	Raw Feature	Clustered Strategy	Different Strategy	Test Statistic
AMM	QOB	1.140	-0.229	-14.449***
AMM	PIRA.X10	1.089	-0.219	-10.816***
Execution	RESRR	0.604	-0.218	-27.611***
Execution	LOI.R50	0.432	-0.129	-28.131***
Microstructural	QS	1.251	-0.048	-12.400***
Microstructural	MOA.100	1.695	-0.120	-15.197***
Momentum	MOA.5	1.207	-0.366	-14.820***
Momentum	BOLU.10	1.220	-0.370	-15.690***
Technical	VOLC.E5	1.321	-0.066	-8.575***
Technical	QOB	1.360	-0.074	-14.847***

Table 4.3.6 – Comparison of average feature importance scores for firms clustered into the strategy related to the feature being studied versus the average feature importance score for other firms. These values are reported in the first and second column, respectively. The third column reports a statistical Welch t-test comparing whether means of the two are significantly different from zero. The test is performed across the sample securities and dates in the original dataset. *, ** and *** correspond to statistical significance at 0.05, 0.01 and 0.001 levels respectively.

This section has provided explicit evidence that the Spherical K-Means clustering algorithm has successfully partitioned the individual $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$ tuple observations in the $\mathcal{FI}$ dataset into five semi-homogenous algorithmic trading strategy categories. These strategies are shown to have high feature importance values for features that are assumed to serve as algorithmic inputs when executing those strategies. The output of this section is a strategy label for each individual Broker tuple that defines whether that firm is primarily conducting either an Automated Market Maker, Execution, Microstructural, Momentum or Technical strategy. Following the strategy segmentation process, we can now quantify the aggregate impact of each individual algorithmic trading strategy on UK equity market quality.

4.4 Impact of Algorithmic Trading on Market Quality

The contemporary symposium of public and regulatory discourse debating the impact of machine-driven algorithmic trading (AT) on the quality of financial markets is endemically fragmented. Dichotomised debate within public, regulator, industry and academic domains has failed to reach a broad consensus on the future role such actors should play in financial markets, an increasingly critical component of real-economy wealth creation laying fertile ground for deeper rational academic analysis. This section empirically examines the question of what impact different AT strategies have on various dimensions of UK equity market quality. Strategies are identified by modelling the relationship between trader actions and limit order book (LOB) features using Deep Neural Networks (DNNs), the axiomatic feature attribution methods of Integrated Gradients and DeepLIFT and Spherical K-Means clustering implemented in previous sections. Market quality on UK equity markets is generally assessed from a perspective of market efficiency and the degree of market integrity. Market efficiency broadly encompasses the domain of liquidity, price discovery and trading costs, whilst market integrity is conceptually rooted in whether the legal rules and regulations governing trading are being adhered to by market participants. Traditionally, academic literature on the subject of market quality has extended across the spectrum between the two pillars of efficiency and integrity, though recent research has found causal connections between the two (Aitken, 2018).

The primary focus of this section is the interrelationship between AT strategies and market efficiency on UK equity markets. Specifically, how the aggregate trading activity of firms conducting these strategies impacts attributes of market efficiency, including price discovery, market volatility, and LOB liquidity. Price discovery is a characteristic of efficient markets that reflect financial information implicitly in the market price of a security, with this information impounded quickly into the price once available (Fama, 1998). Efficient markets generally require both high levels of liquidity, allowing traders to transact large quantities in a security efficiently, low levels of volatility, to infuse trust in traders' ability to manage risk when trading on secondary markets, and low transaction costs, allowing firms to manage positions and risk. Interrelationships between the different market quality metrics are also explored. Specifically, the decomposition of price discovery into information and volatility components (Benos, 2012) is necessary to understand how noise in the form of excess volatility and information impounded into markets interrelates. Analysis in this section utilises the FCA FTSE100 CLOB dataset introduced in Section 3.3 that covers all millisecond-stamped messages between traders and the four main UK equity venues over the period of June 2015 for five FTSE100 stocks. The analysis is performed using an event-time framework which aligns with the predominate temporal conceptions of competitive AT strategies (Gomber, 2013). Market quality is assessed based on trading activity on the London Stock Exchange (LSE), the largest UK market in terms of volume, to control for noise in market quality variables cross-venues.

Impact from algorithmic and high-frequency trading (HFT) on markets has been the focus of recent investigations by regulators. MiFID II regulations implemented in 2018 place additional requirements on AT firms within a framework aiming to improve transparency in EU financial markets. Recent initiatives by the Financial Conduct Authority (FCA) have looked into whether HFTs exploit their advantage in speed and technology, finding mixed results regarding how HFTs anticipate order flow (Aquilina, 2016), and their role during periods of market stress when circuit breakers are triggered (Bercich, 2017). A report published in 2014 by the European Securities and Markets Authority (ESMA) analysed the level of HFT activity in EU equity markets without making significant conclusions regarding the impact of this activity on market quality (ESMA, 2014). The Australian Securities and Investment Commission (ASIC) undertook two recent reviews studying the role of HFT and dark liquidity in Australian financial markets (ASIC, 2013; ASIC, 2015). Both reviews found that the regulatory framework surrounding HFT and AT in Australia is robust with market participants generally able to adapt to the demands from faster electronic markets. These reviews instigated changes to market integrity rules to improve transparency in dark markets and minimise the delineation of fiduciary responsibilities when firms are acting as principals in an agency arrangement with retail investors. In 2015 the US Commodity Future Trading Commission (CFTC) proposed Regulation Automated Trading that would apply additional transparency requirements, risk safeguards, and source code access requirements on firms conducting automated trading on futures exchanges. The regulatory debate surrounding the role and impact of AT is continually evolving.

This thesis adds to the body of literature on the topic of algorithmic trading and market quality in electronic markets by studying AT from a new perspective, focusing on the specific algorithmic strategy executed by the firm rather than some nebulous definition of what constitutes ‘high-frequency trading’. Furthermore, this section focuses predominantly on the role of AT strategies at the ‘limit order’ level, that is, how traders interact with the LOB by placing limit orders as opposed to at the ‘transaction’ level that analyses traders’ executions only. Causal links between AT and market quality are difficult to identify due to the issue of endogeneity. The direction of causality in the relationship between AT activity and market quality has multifarious elements that cannot be disentangled in the absence of a market-wide exogenous shock that isolates the impact from only one variable. Thus, the results from the impact of AT on market quality analysis must be considered in the context of this issue, however, several methods are employed to overcome endogeneity so as to derive inferences on causality.

The impact from these distinct AT strategies is assessed using a robust econometric framework through the dimensions of how AT impacts LOB liquidity in Section 4.4.1, short-term volatility in Section 4.4.2, and the price discovery process in Section 4.4.3.

4.4.1 Liquidity

The liquidity of a financial instrument is ostensibly represented by the current set of resting orders available for immediate execution on the LOB. Despite its amorphic nature, liquidity plays a critical role infusing trust in secondary financial markets. Liquid LOBs allow market participants to execute trades quickly, cheaply and completely, serving as a fundamental element driving capital raising activity in primary markets. Traditional inventory and information-based models of liquidity focussed on how market makers provided liquidity at the best bid-ask spread depending on individual inventory levels and measures of adverse selection, respectively. The fast-paced evolution in financial markets instigated by evolutions in electronic trading, LOB matching systems, and direct market access models to exchanges, has facilitated the rise of high-frequency markets with implications for how liquidity is viewed by market participants and studied by academics. Literature on the interrelationship between trading and liquidity has increasingly diverged from traditional models and focussed on conceptions of liquidity relevant for high-frequency markets (Subrahmanyam, 2016). With this context, the following sections analyse liquidity provision primarily at the limit order-level with less focus on liquidity at the transaction level.

Stability in the supply of liquidity on the LOB in the form of passive order submissions is required for financial markets to form and function. Liquidity is also an important function of efficient asset pricing and trading costs with a natural negative correlation between stock prices and liquidity levels (Pastor, 2003). Evaporation of market liquidity caused by structural imbalances in the supply and funding of liquidity was shown to be a precursor to the GFC in 2008 (Brunnermeier, 2009). The multi-faceted nature of liquidity leads to a definition generally taken from the perspective of the market participant for which the concept has relevance. For example, for institutional investors liquidity relates to the cost, ease and timeliness of transacting in an asset (Pastor, 2003). Dimensions of liquidity at the microstructural level can generally serve as a proxy for the quoted best bid-offer (BBO) spread, the level of depth at the BBO, and the level of market resiliency. Quoted spreads at the BBO manifestly represent the cost of buying and selling a single share at a point in time whilst market depth indicates the theoretical limitations on the share volume that can be transacted at the BBO immediately.

Several studies have utilised event-study methodologies to analyse how the introduction of low-latency trading mechanisms and electronic trading systems have impacted market liquidity. Results have predominately shown a rise in liquidity, increased trading volumes, reduced risk of adverse selection and tighter quoted spreads when low-latency technology was introduced on exchanges spawning beneficial outcomes for market participants (Riordan, 2012; Hasbrouck, 2013). The growth of AT in equity markets has also had a significant impact on the correlation structure of liquidity in financial markets. Auto-correlations that previously existed over hours were shown to be occurring over periods of minutes or seconds, generating short-term volatility (Smith, 2010).

Additional research has examined the role of AT and HFT firms in supplying liquidity during periods of market stress. Groth (2011) provides evidence against the assertion that HFTs withdraw liquidity during periods of high volatility. Additionally, Brogaard (2010) employs an exchange HFT-identifier for NASDAQ securities over the period 2007-2009, finding that HFT liquidity supply improves vis-à-vis non-HFT traders during market stress periods or shocks to the system in the form of new information from earnings announcements. Additionally, Brogaard (2010) finds that HFTs are present at the BBO approximately 50% of the trading day. Similar results were reported by Hasbrouck (2013) using the same NASDAQ dataset and a simultaneous equations methodology, providing evidence that low spreads and high depth were correlated with increased HFT participation in trading.

Trading behaviour associated with placing limit orders at or within the BBO to reduce spreads, increase market depth and profit from price disequilibrium is aligned with improved market quality, reduced transaction costs from tighter spreads, and lower risk of adverse price impact from trading. Understanding the liquidity provision activities of firms conducting different AT strategies via limit order submissions and cancellations in order-driven markets is critical to assessing their overall impact on market quality. Academic research has largely aligned with the view that HFT and AT activity induce tighter quoted spreads (Malinova, 2013; Jovanovic, 2016). Both Jarnecic (2010) and Hendershott (2013) analyse FTSE100 stocks and DAX30, respectively, finding that HFT and AT activity improve market quality by increasing liquidity supply during profitable opportunities when quoted spreads widen. Zhang (2011) and Carrion (2013) confirm these results by utilising a NASDAQ dataset to analyse the impact of HFT passive and aggressive intra-day behaviour on liquidity metrics. They find increased HFT passive activity when quoted spreads are wide and more aggressive trading when they are narrow.

Several hypotheses of why the prevalence of AT activity in modern markets has contributed to smaller spreads are presented. First, Menkveld (2013) argues that AT strategies have largely emulated the traditional role of market makers on high-frequency markets. Lower operational costs of these new market makers allow for a competitive environment that ferments tighter quoted spreads. Second, reductions in market frictions driven by the evolution in trading technology have had correlative effects on firms cost of monitoring limit orders and executing transactions (Hasbrouck, 2013). Firms utilising AT systems are able to supply more liquidity to LOBs driven by a lower risk of being adversely selected stemming from their capability of updating quotes over minuscule time periods, reducing monitoring and friction costs (Hendershott, 2013; Jovanovic, 2016). In a more recent and relevant study, Subramanyam (2016) provides a comprehensive analysis of HFT limit order activity. The authors measure HFT liquidity provision metrics for HFT and non-HFT market participants on NASDAQ LOBs, including order sizes, spreads and participation during periods of market stress. The authors find that HFT liquidity providers overall enhance the quality of US equity markets.

4.4.1.1 Liquidity Supplying Characteristics of Algorithmic Trading Strategies

Liquidity in this section is explored along the dimensions of tightness in quoted spread and immediacy from available market depth at the inside spread. These ex-ante market quality metrics arise naturally from LOB data, and have significant implications for explicit trading costs, immediacy and possible size of execution when trading in financial securities. In the following parts of this section, the liquidity characteristics of AT strategies are studied at the limit-order level.

Descriptive statistics for the average volume submitted, cancelled and traded by firms grouped by AT strategies are presented in Table 4.4.1. Metrics are measured as an average for each strategy over intervals of 10 minutes throughout the trading day for the five securities studied in this chapter. Limit orders for these metrics are studied only at the BBO representing limit orders submitted with a higher probability of execution then if placed in a lower price queue several ticks from the BBO. The results indicate that firms conducting AMM AT strategies have the highest volume of submissions, with 88,193 shares submitted at the BBO on average every 10-minute period representing 31.2% of all order submissions. As one would expect AMM firms also cancel the largest volume, 79,031 shares on average, which can be logically deduced from the common finding that in fast-paced LOB environments high order submission to cancellation rates tend to prevail (Kirilenko, 2017). Hence, there is a strong positive correlation of 0.82 between order submission and cancellation volumes across all traders in the sample. Despite the large volume of limit orders submitted by AMMs, they only execute the second highest average volume of trades per period, with Momentum strategies 17,886 shares per ten-minute period, representing 33% of all transactions completed on average. Technical strategies tend to have the lowest level of submissions, cancellations and trades on average.

Strategy	Submissions Mean (SD)	Cancellations Mean (SD)	Trades Mean (SD)
AMM	88,193 (43,605)	79,031 (39,442)	11,724 (8,038)
Execution	42,092 (30,184)	35,207 (25,608)	9,483 (7,497)
Microstructural	52,543 (33,104)	44,467 (27,566)	8,296 (6,325)
Momentum	73,674 (44,574)	61,576 (37,962)	17,886 (12,209)
Technical	26,193 (21,096)	21,610 (17,428)	6,778 (5,956)

Table 4.4.1 – Limit order volume submitted, cancelled and traded on average over 10-minute periods across the sample dataset. SD refers to standard deviation.

Table 4.4.2 provides evidence of which firms are more likely to cancel BBO orders, shown in Panel A, and execute BBO orders, shown in Panel B. From Panel A it is evident that AMM AT firms have the highest execution ratio, at 46.6%, followed by Microstructural AT strategies, with a ratio of 46.5%. Cancellation ratios represent the volume of orders cancelled at the BBO as a percentage of total volume submitted and cancelled at the BBO, averaged over sample securities in the dataset. Technical strategies have the lowest cancellation ratio of 41.37% which is less than the mean ratio of all firms by 1.6% and statistically significant at the 1% level. Execution ratios are reported in Panel B with the metrics computed as the volume executed by each AT strategy as a percentage of total volume submitted at the BBO. Technical, Execution and Momentum strategies all attain higher execution ratios than the mean with a significance level of at least 5%, reporting average ratios of 28.5%, 25.4% and 24.8%, respectively, over the stocks studied. Note that these ratios are calculated based only on limit orders submitted at the BBO and are significantly less when considering orders placed at all price points by these traders. The primary reason for focusing on the BBO is its relevance in high-frequency markets where the cost of placing limit orders on liquid LOBs is very low, deeming that liquidity supplied at the BBO queue is paramount to quantifying the manifest impact of AT on market quality. The low execution ratios of AMM and Microstructural strategies align with the result from Panel A showing relatively higher cancellation ratios for these firms, though the difference from mean execution ratio of other firms is not statistically significant.

	Cancellation Ratio (Panel A)		Execution Ratio (Panel B)	
Strategy	**Mean (SD)**	**Difference (T)**	**Mean (SD)**	**Difference (T)**
AMM	0.4655 (0.1054)	0.0029 (0.14)	0.2073 (0.2062)	-0.0163 (-0.39)
Execution	0.4498 (0.0583)	-0.0186 (-1.61)	0.2537 (0.1119)	0.0628** (2.64)
Microstructural	0.4652 (0.1153)	0.0030 (0.12)	0.1996 (0.1145)	-0.0072 (-0.34)
Momentum	0.4518 (0.0258)	-0.0166** (-3.09)	0.2476 (0.0915)	0.0566*** (3.90)
Technical	0.4137 (0.0792)	-0.0554** (-3.21)	0.2851 (0.1370)	0.0932** (3.13)

Table 4.4.2 – Limit order cancellation ratios (Panel A) and execution ratios (Panel B). Cancellation ratio is the average ratio of limit order volume cancelled at BBO prices to total cancellation and submission volume at BBO prices. Execution ratios refer to average volume traded to volume submitted at the BBO. Differences are calculated as strategy ratio minus non-strategy ratio. Two-sided T-tests are performed. *, **, and *** indicate statistical significance at the 0.5, 0.01, and 0.001 levels, respectively.

Recent criticism from regulators regarding the high degree of order placements followed by immediate cancellation precipitated action taken by trading venues to curb this behaviour.[22] The MiFID II RTS 9 which took effect in 2018 now requires trading venues to set a *cap* on OTR's for individual securities which firms must adhere to each trading day. The results reported in Table 4.4.2 show that the low dispersion between cancellation ratios for the different trader groups indicates some degree of homogeneity in the order cancellation strategies of each group. Only AMM and Microstructural firms have a higher cancellation ratio than the average ratio for non-strategy firms, and these results are not statistically significant at the 5% level. These results provide some evidence against the utility of placing restrictions on firms OTRs given that no specific strategy seems to be abusing the market and cancelling significantly more orders, for given volumes executed, than other competing strategies. As a caveat, it must be noted that small trading firms conducting less than 1,000 actions in a trading day for a given security are not captured in this analysis and, as a result, further investigation is required as to whether these firms may be abusing market integrity with very high OTRs. Additionally, the analysis focuses only on the BBO and no limit orders placed at lower queue priority price points.

An analysis of the size of limit order submissions by AT strategy groups is summarised in Table 4.4.3. The average order size is relatively stable across the different AT strategies. Technical strategies tend to submit higher orders with an average size of 2,145 shares, though these results are volatile across the sample stocks and not statistically significant. The statistical tests performed do indicate that the Momentum average order size of 2,002 shares is higher than the mean sample size by 8.3%, a statistically significant result at the 5% level. Only orders placed at the BBO are considered. There appears to be a connection between the execution ratios reported previously and the order size, with Execution, Momentum and Technical strategies having the highest order sizes, with average order sizes over 2,000 shares, and also the highest execution ratios.

[22] See Securities Exchange Act Release No. 34-61358, 75 FR 3594, 3606 (January 21, 2010).

	Order Size	
Strategy	**Mean (SD)**	**Difference (T)**
AMM	1955.27 (1825.45)	120.20 (0.67)
Execution	2041.79 (2176.44)	206.72 (0.95)
Microstructural	1848.04 (1772.68)	-47.07 (-0.53)
Momentum	2001.59 (1771.13)	166.52* (2.40)
Technical	2145.19 (2313.39)	440.89 (1.19)

Table 4.4.3 – Limit order size is the average volume for orders submitted at the BBO. Differences are calculated as strategy order size minus non-strategy order size. Two-sided T-tests are performed. *, **, and *** indicate statistical significance at the 0.5, 0.01, and 0.001 levels, respectively.

Limit order survival times for orders placed at the BBO weighted by the volume size of the order is reported for all five AT strategies in Table 4.4.4. Panel A reports the mean and standard deviation of order survival times for all limit orders submitted at the BBO, that is, how many seconds the order stayed on the LOB for, in addition to the difference of the specific strategy from the population mean. A commensurate statistical t-test is performed to determine whether the means of the trader and non-trader population are statistically different. Similarly, Panel B reports the statistics for limit orders that were executed against as a *transaction* whilst Panel C reports order survival times for orders that were *cancelled*. All order times are reported in seconds. An interesting result is the small order survival time across all strategies with the mean survival time ranging from 17.97 seconds for Microstructural strategies to 13.73 seconds for Execution strategies. A common result across the three panels is the low survival time for orders placed by Execution strategies. Orders placed by Execution firms remain on the LOB for 29.6% less time than the average trader, a result significant at the 5% level. This result is highlighted further by the significantly lower survival time of only 8.67 seconds for orders that are executed. Intuitively, one would expect Execution algorithms to place orders at the BBO when the price meets their intrinsic value of the stock for which they are trading, given that these firms generally trade in one direction, long or short, in a stock for periods of the day in order to meet a client's objective of liquidating or building a stock position. The results presented in this section provide evidence against the assertion that certain types of predatory HFT strategies provide only transitory or phantom liquidity, given the higher survival times of strategies that one could argue as HFT such as AMM, Momentum and Microstructural, versus Execution algorithms which are likely to be run by institutional investors such as banks or asset managers.

	Survival Time All (Panel A)		Survival Time Trade (Panel B)		Survival Time Cancel (Panel C)	
Strategy	Mean (SD)	Difference (T)	Mean (SD)	Difference (T)	Mean (SD)	Difference (T)
AMM	16.0838 (7.8268)	-0.0308 (-0.18)	8.6843 (10.1972)	-0.2963 (-1.32)	17.6211 (8.2048)	-0.0523 (-0.35)
Execution	13.7298 (7.7083)	-0.2962* (-1.82)	8.6667 (5.0883)	-0.2981** (-3.68)	15.6043 (9.8834)	-0.2600 (-1.28)
Microstructural	17.9765 (16.4219)	0.1619 (0.56)	12.7659 (17.0919)	0.1268 (0.47)	19.2722 (16.3134)	0.0948 (0.35)
Momentum	16.5730 (7.4481)	0.0243 (0.16)	13.9262 (12.2683)	0.2483 (1.42)	17.9511 (7.2034)	-0.0183 (-0.18)
Technical	17.8178 (13.2083)	0.1601 (0.70)	14.0800 (11.4133)	0.2439 (1.18)	20.6734 (13.9502)	0.2604 (1.15)

Table 4.4.4 – Limit order survival times refer to the length of time (in seconds) an order placed at the BBO remains on the LOB. Panel A refers to all orders placed at the BBO, Panel B refers to orders at the BBO that are traded against, with Panel C referring to orders that left the LOB by being cancelled. Differences are calculated as strategy order survival time minus non-strategy survival time. Two-sided T-tests are performed. *, **, and *** indicate statistical significance at the 0.5, 0.01, and 0.001 levels, respectively.

Figure 4.4.1 graphically presents the average quoted spread to tick ratio (STR) for each AT strategy over minute periods using rolling averages over the past five-minute observations. STRs are measured through the trading day between 9 am and 4 pm to align with the conventions used in this chapter. One can note the clear differentiation between AMM and Momentum strategies that maintain significantly tighter spreads on average throughout the trading day than the other three strategies, with the ratio stable around three ticks between the best bid and ask of that trader group. Volatility in the STR is evident for all strategies around 1 pm, likely due to new information entering the UK markets when trading begins in the US. This volatility is also evident for Microstructural and Technical strategies at the 2:30 pm mark and AMM strategies around the 3 pm when other markets begin trading. An increase in a firm's STR is correlated with the perception of higher levels of informed trading on markets. Therefore, firms with a less technologically advanced AT systems and higher latency speeds may increase their spreads when a large burst of information enters the markets so as to not be adversely selected by firms with higher information processing capacity that can interpret and trade off that new information quicker. This potentially explains the widening of Technical and Execution strategies spreads at certain times of the day, given that they commonly have less technologically capable systems as firms conducting higher-frequency forms of trading.

Execution strategies tend to have a high average STR throughout the day though this begins to reduce in the last hour or two of trading. One potential explanation could be the requirement for these firms to liquidate or buy inventory to complete an order or meet client demands for liquidity. Similar patterns are evident in the Momentum and AMM strategies, with some of these firms possibly conducting hybrid strategies with both agency and proprietary trading components.

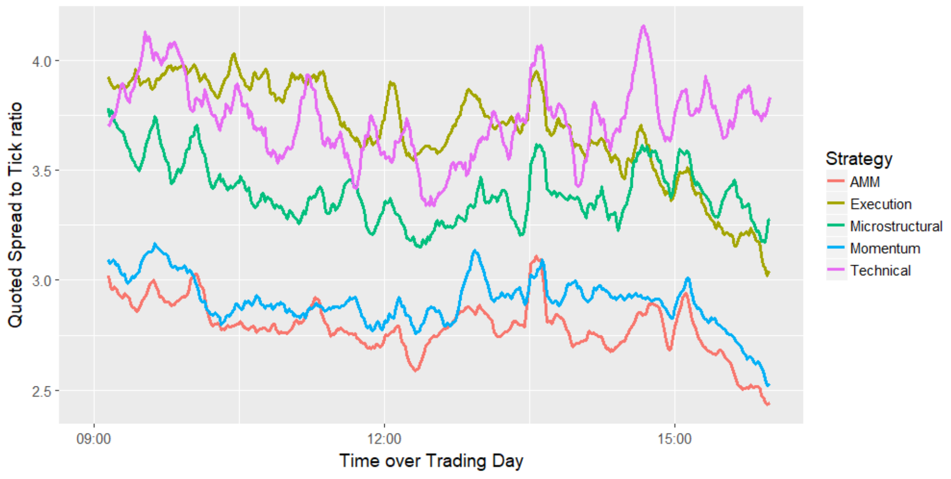

Figure 4.4.1 – Average 5-minute rolling Quoted Spread to Tick (QST) ratio for all firms conducting a specific AT strategy. Measured over the course of the trading day in one-minute increments between 9 am and 4 pm.

The shape of AT strategies LOB order distributions is illustrated in Figure 4.4.2. Each line represents the average time-weighted LOB market depth distribution, measured in shares, for traders conducting a specific AT strategy across all price queues up to five ticks from the BBO. All AT strategies tend to maintain a similar order distribution of variable magnitudes, with lower market depth at the BBO, then depth increasing up to price levels approximately three ticks from the BBO, P3, before reducing depth for prices further from P3. Momentum strategies on average maintain the highest quantum of market depth at the BBO, with 5107 shares available to be matched against by an incoming order on either side of the LOB. As expected AMMs also maintain relatively deeper liquidity positions on the BBO queues though on average their market depth is 9.9% less than Momentum strategies. However, AMM firms transact 34.4% less volume than Momentum firms, as evident in Table 4.4.1. Another interesting result is the relative steepness of Execution LOB distributions when compared against Microstructural strategies. Execution firms maintain smaller levels of depth at the BBO, though this increases in proportion to Microstructural strategies as queues move further from the BBO, with Execution firms on average maintaining 20.3% deeper depth at price points five ticks from the BBO. One potential explanation for this behaviour could be the risk of adverse selection for Execution traders, who may not want to expose a large parent order to the market that risks a predatory response from higher-speed traders that could result in a larger price impact and implicit cost of trading the parent order.

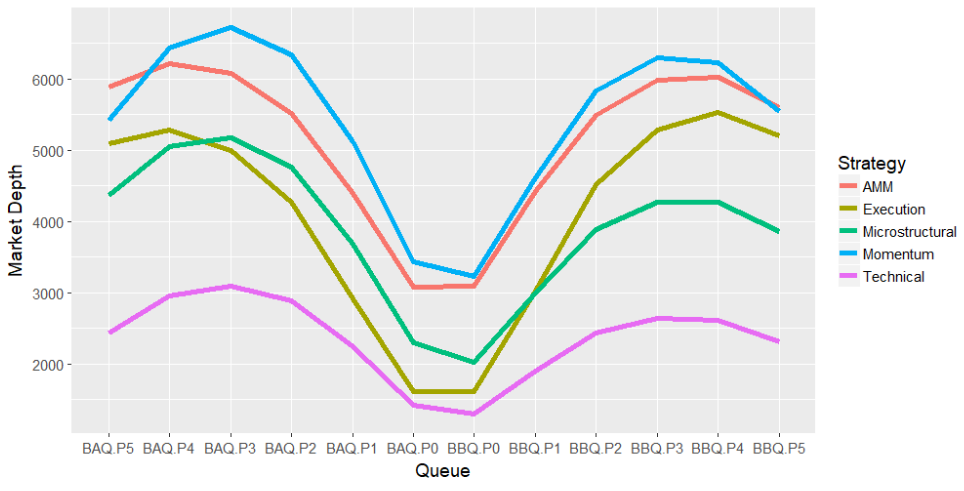

Figure 4.4.2 – Time-weighted average LOB market depth order distributions for all strategies at price points at and within five ticks of the BBO. Market depth is measured in share volume.

4.4.2 Volatility

This section assesses the contemporaneous, dynamic, causal and fundamental relationship between the trading activity of different AT strategies and market disruptive intra-day price volatility. A low volatility price environment is a central tenant of well-functioning and efficient financial markets that motivates the congenial participation of traders in markets given their desire to control the risk of investments. Volatile markets can potentially lead to rational risk-averse investors exiting the market during periods of high volatility (Kirilenko, 2017). Volatility arises naturally on LOBs when new information is introduced into the market resulting in a fundamental shift in the intrinsic equilibrium value of a security, with prices fluctuating as traders interact and execute transactions on the LOB. In the context of financial market trading, volatility can be viewed as a transaction cost given the risk that market forces move prices adversely to orders placed on the LOB, leading to market makers potentially increasing the spreads they offer institutional investors, making trading more expensive (Harris, 2003).

Price volatility during period t in this chapter is measured as the standard deviation in intra-day logarithmic midprice (MP) returns, R_t, using an event clock conception of t. Mathematically, the metric is defined as the standard deviation of logarithmic returns over period T, $SD(R)\sqrt{T}$. Separate metrics for realised price volatility, VOL_t^R, which utilises MP period returns over t, $R_i^R = \log\left(\frac{MP_t}{MP_{t-1}}\right)$, and high-low volatility, VOL_t^{HL}, which takes high-low logarithmic returns, $R_i^{HL} = \log\left(\frac{MP_t^H}{MP_t^L}\right)$, are tested. Realised returns are the predominant academic standard for measuring volatility (Andersen, 2003), whilst using logarithmic returns has also been shown to materially reduce the dispersion of measurement errors (Alizadeh, 2002).

Modelling the relationship between AT activity and stock volatility requires consideration of both the theoretical relationship between the variables and a set of statistical issues that arise due to endogeneity in the dataset. One potential theorem of this relationship is that AT activity is materially influenced by levels of price volatility given its impact on incentives for traders to take profitable positions, especially short-term, in the market. A dichotomic logic on this continuum would infer that the strategic discourse of AT firms themselves drives market volatility. The primary issue that arises when analysing the relationship is the problem of endogeneity that occurs when independent variables are serially correlated with the errors. In time series data this is generally due to auto-correlated errors or simultaneous causality between dependent and independent variables. Specifically, for chronological clock-based analysis the effect of multiple trades or quote updates at the BBO over short time periods can blur the causal relationship between AT activity and volatility. Utilising lagged values of volatility as instrumental variables given its auto-regressive nature could overcome this issue. However, finding a comparative instrument for AT activity to quell endogeneity issues is difficult to attain given its low auto-correlation. Rather, employing an event clock framework and utilising statistical techniques presented in Section 4.4.2.4 are used to overcome the endogeneity issue.

Several studies have assessed the impact from AT, or more specifically HFT, on intra-day price volatility, finding rather divergent results. The lack of a ubiquitous academic consensus regarding the relationship between AT and volatility stems from both the plethora of research methodologies and trader categorisation models employed. Generally, research can be categorised into one sub-category arguing the virtues of algorithmic trading, concurring with the view that AT activity reduces volatility, whilst a second sub-category highlights the negative role that AT has by making markets more volatile.

Research presenting evidence of a positive impact from AT on volatility is assessed. When price movements are transitory, AT activity conducive to market quality will supply liquidity and trade in the direction against these forces, whilst correlating trading in the direction of permanent price changes. Brogaard (2010) utilises an exchange-identified dataset of 26 HFT firms trading in 120 NASDAQ and BATS stocks over 2008-2009 to assess the impact of HFT on market quality. The study finds HFTs that aggressively supply liquidity trade primarily during periods of low market volatility. An extended study on the impact from the exogenous event defined by the 2008 short sale ban in US stocks, asserted that HFTs dampen volatility during periods of market stress (Brogaard, 2017). Hagstromer (2013) categorises AT into traders conducting automated market making and opportunistic strategies on the NASDAQ-OMX Stockholm exchange and use tick size changes as an exogenous instrument to conclude that both forms of AT mitigate volatility. These authors argue against treating AT firms as a homogenous group for the purpose of analysing market quality and formulating regulatory policy, a concept which is implemented in this thesis. Chaboud (2014) find a correlation between various AT firms trading common exchange rate permutations (USD, EUR, YEN) on foreign exchange markets, indicating a propensity in the market for herding behaviour between computer-executed algorithmic strategies. However, despite the correlation evidence of a negative causal link is found between the level of AT activity and market volatility indicating that trading by automated algorithms subdues price movements. Brogaard (2014) analyse the impact from HFT trading on the volatility of NASDAQ equity securities by employing a state space model to differentiate between permanent and transitory price movements. The study finds that HFTs generally trade against transitory pricing errors and in the direction of permanent price changes, determining that HFT activity quells volatile price movements. Hendershott (2013) analyses how the informational content from AT trades impacts the stability of prices for DAX30 composite stocks over three trading weeks in 2007-2008. No evidence of a fundamental relationship between volatility and AT activity is found in this study. Using a similar dataset and methodology, Groth (2011) studies the contemporaneous relationship between volatility and HFT activity, finding that these traders supply liquidity during stressful conditions with high market volatility. Menkveld (2013) analyses the Dutch equity market during a period in 2007 when an HFT firm first entered the market, finding no statistical impact from the new entrant on realised volatility.

Evidence supporting a negative impact on market quality arising from AT activity driving volatility is equally available. Benos (2012) study a UK FTSE 100 dataset of four stocks over one week of trading and employ a VAR methodology that finds a relationship between aggressive HFT activity and excess volatility. However, this negative impact should be understood in the context of HFTs role in the wider price discovery process. HFTs may have a high contribution to noise, or excess volatility, though their information-to-noise contribution ratio is higher than non-HFT firms indicating that whilst a proportion of their trading activity represents noise, a higher proportion has significant informational content that improves price discovery and market quality overall. Boehmer (2015) provides additional evidence towards the finding that AT activity speeds up price discovery and improves overall market quality at the cost of heightened excess volatility. The study finds this result to be robust for different markets using sample securities in 42 markets analysed over the period of 2001 to 2011. Several studies have found that HFTs exacerbate market volatility when they withdraw from markets, dump inventory, or increase in activity (Huh, 2014; Easley, 2011).

The contention that liquidity evaporates when HFTs exit markets during periods of stress has been studied extensively. These events are commonly referred to as *flash crashes* epitomised by the May 2010 'Flash Crash' on US futures markets that led to contagion of volatility across global markets.[23] An SEC investigation of the event concluded that whilst HFTs did not initiate the flash crash, they compounded the volatility in the market by contributing to an erosion of finite and rapidly diminishing liquidity during the event that resulted in the ensuing disorder in markets (SEC, 2010). Kirilenko (2017) provides a seminal analysis of the event to determine its primary cause, finding that whilst HFTs provided liquidity to fundamental sellers during the initial drop in value, they may also have had a role in accelerating the volatile conditions as they began to sell inventories during the price drop, thus, compounding the growing demand for liquidity.

4.4.2.1 Contemporaneous Relationship

Despite the evidence that AT and HFT activity contributes positively to liquidity provision (Hendershott, 2011) and more broadly to the price discovery process (Brogaard, 2010), criticism of HFT activity has fixated on systematic risk from these traders withdrawing liquidity during periods of extreme volatility where algorithms risk management protocols override the traditional AT strategies implementation (Kirilenko, 2017; SEC, 2010). An initial assessment of the direct contemporaneous relationship between different AT strategies and volatility is performed by comparing the abnormal fraction of trading volume executed by traders conducting a specific strategy and realised volatility over one-minute periods. A similar methodology is implemented by Brogaard (2010) and provides some evidence of how AT firms operate during periods of varying volatility.

The contemporaneous relationship between AT strategies and volatility is visually depicted by comparing the abnormal fraction of an AT strategy activity, s_t^{ABN}, for strategies $s \in S$ against abnormal volatility levels, VOL_t^{ABN}, which is bucketed into ten bins of increasing volatility. Abnormal volatility is measured as the percentage difference between actual period t, $VOL_{i,t}$, and average, $\overline{VOL_i}$, price volatility weighted by stock standard deviation, σ_i, for a specific stock i. Volatility levels are then ranked from lowest to highest before being placed into the ten groups of size N_p with similar levels of abnormal volatility levels. The anomalistic level of AT strategy activity is calculated for each period as the percentage difference in AT activity, $s_{i,t}$, from the average activity level, $\bar{s}_i$, for stock i across all time periods t. Abnormal volatility and activity levels are defined as:

$$VOL_{i,t}^{ABN} = \frac{VOL_{i,t} - \overline{VOL_i}}{\overline{VOL_i}} * \frac{1}{\sigma_i}$$

$$s_p^{ABN} = \sum\nolimits_{VOL_{i,t}^{ABN} \cap p} \frac{1}{N_p} \left(\frac{s_{i,t} - \bar{s}_i}{\bar{s}_i} \right)$$

[23] The 'Flash Crash' term pertains to the series of events that occurred from 2:32-3:08pm on 6/5/10 that spurred significant turbulence on market globally, resulting in a 1010.14 point decrease in the Dow Jones Index, the second largest intra-day point deviation in the indexes history.

Figure 4.4.3 provides a visual representation of the abnormal AT activity values s_p^{ABN} for each of the p deciles on the vertical axis against buckets of increasing realised volatility values, $VOL_j^{ABN} \cap p$, on the horizontal axis, for each of the five AT strategies studied. Thus, one can determine whether strategies have abnormally high activity during periods of stressful market conditions. Volatility during the high decile periods is on average 48.4% higher, whilst it is 34% lower as measured by the mid-quote price realised volatility during average market conditions. The figure demonstrates that a contemporaneous relationship exists between realised volatility and both AMM and Microstructural strategies, though in opposite directions. Firms conducting AMM strategies tend to interact more with the order book during periods of low volatility, submitting 6.4% extra orders on average at or within the BBO during event periods in the lowest decile of volatility. Conversely, Microstructural strategies are placing 6.5% fewer orders during these periods on a relative scale. The results are mirrored for the high volatility periods where AMM firms decrease activity by 12.6% and Microstructural increase by 7.5%, with all metrics measured on a relative scale averaged across the sample securities and dates.

The contemporaneous relationship between strategy activity and volatility is not as pronounced for the other three strategies. However, there appears to be a slight negative relationship between abnormal AT activity and volatility for AT firms conducting either Momentum or Execution strategies. Execution firms in particular appear to reduce trading activity during periods of higher market stress as evident by the 5.1% decrease in activity from an average relative baseline during the high volatility periods.

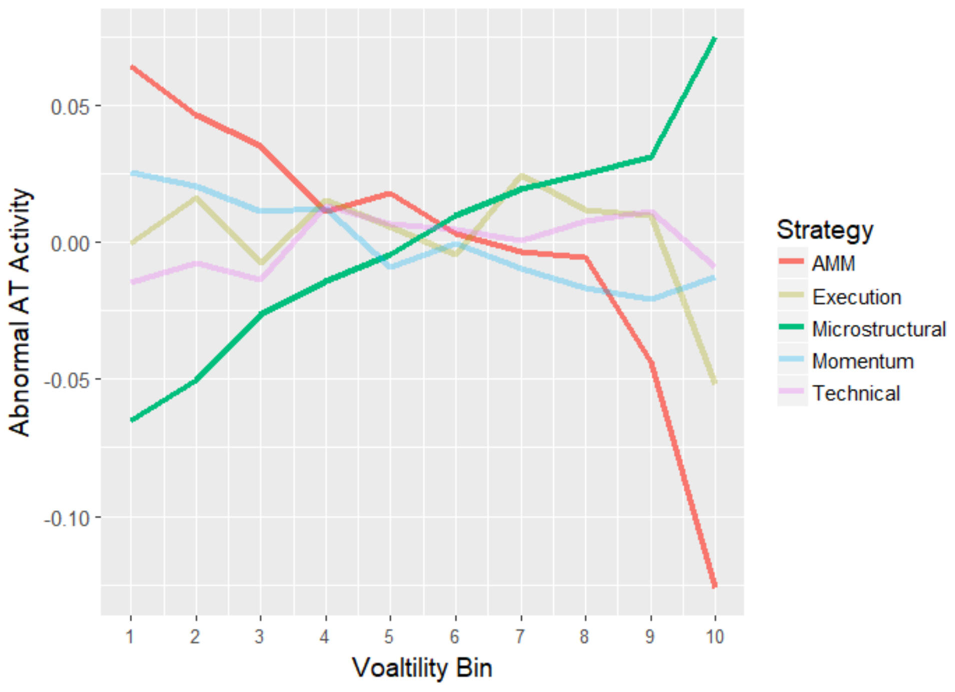

Figure 4.4.3 – Contemporaneous relationship between AT activity and volatility for the five trading strategies analysed. Abnormal AT activity for each strategy is measured as the percentage difference for a particular strategy from their baseline average activity levels using one-minute periods of analysis. Abnormal volatility is computed in a similar manner before observations are binned into deciles and ranked in ascending order.

This section provides an understanding of the contemporaneous relationship between AT strategy activity and volatility. There is a pronounced correlation between current period volatility and both Microstructural and AMM strategy activity, though in reverse directions. However, this analysis does not infer any causal, statistical or fundamental relationship between the variables given the likely endogeneity of the AT activity with respect to volatility.

4.4.2.2 Dynamic Relationship

The issue of whether AT firms exit markets following periods of high volatility is a fundamental question for regulators and market practitioners. This section analyses the dynamic relationship between AT firms conducting a specific strategy and market volatility by developing a finite distributed lag model. OLS regression is performed to assess how AT firms executing strategy s respond to price volatility in the previous period, $VOL_{i,t-1}$, based on their net liquidity volume for stock i in period t, $NL_{i,t}^{s}$, measured by bid minus ask volume at the best bid-ask (BBO) prices during t. In line with the event-clock methodology used in this chapter, periods of t are set to 500 events which equates to approximately 30 to 60 second periods on average. Dummy variables are included in the OLS regression for the AT strategy, D^{s}, and for the top 1% of periods with the highest levels of volatility, D^{HL}. Interaction terms are added to quantify the strategy-specific impact of volatility on AT firms' net liquidity supply. Following Subramanyam (2016), all continuous variables are standardised with mean zero and unit variance by individual stock i. Furthermore, the set of control variables, δ, are incorporated into the OLS regressions. These control variables include a set of indicators that may affect volatility in the macroeconomic context, including the FTSE Volatility Index (VFTSE), and a set of stock-specific control variables, including firm market capitalisation to proxy for stock liquidity, and the inverse price ratio as a proxy for transaction costs. The regression testing the relationship between AT firms' liquidity provision and volatility is specified as:

$$NL_{i,t}^{s} = \alpha_0 + VOL_{i,t-1} + D^{s} + D^{HV} + D^{s} * VOL_{i,t-1} + D^{s} * VOL_{i,t-1} * D^{HV} + \delta + \epsilon_{i,t}$$

Table 4.4.5 presents the OLS regression results for the distributed lag model testing the dynamic relationship between AT activity and volatility. The results demonstrate that the coefficient for lagged volatility, $VOL_{i,t-1}$, is positive and significant for all AT strategies implying that there is a net increase in liquidity supply following a volatile period for all strategies. The variable most of interest is the interaction effect between the strategy dummy variable and lagged volatility, $D^{s} * VOL_{i,t-1}$. This measure provides a *relative* measure of how AT firms respond with limit orders following a period of volatility. The negative and significant coefficient for AMM indicates that these firms reduce their net liquidity supply by 3.5% following a one standard deviation increase in volatility compared to non-AMM traders. Conversely, Momentum strategies have a significant positive coefficient implying that these firm increase their net liquidity supply by 3% relative to other traders following a similar increase of one standard deviation in volatility. Thus, while in absolute terms traders seem to be trading more in periods after higher volatility, we can draw a different conclusion when studying behaviour in relative terms using the lagged OLS regression. These results align with theoretical and empirical research on AMMs. Empirically, AMM-type AT firms have been found to provide more liquidity in days with stable returns (Anand, 2016) and tend to withdraw from market-making during high volatility periods (Easley, 2011).

The interaction term for extreme volatility periods, $D^{s} * VOL_{i,t-1} * D^{HV}$, has negative coefficients for all AT strategies. This implies that strategies are responding to extreme volatility periods by decreasing net liquidity supply or withdrawing liquidity from the market. This relationship is statistically significant for Execution and Momentum strategies at the 5% level and Microstructural strategies at the 1% level. Evidently, following one-minute periods where volatility is in the top 1% highest volatility bracket, Momentum strategies are expected to decrease net liquidity supply by 3.5% and Execution strategies by 2.7%. The decrease in Microstructural net liquidity supply has a higher magnitude of 4%. These results indicate that whilst Momentum strategies generally increase net liquidity in a period following volatility, this is not the case after periods of *extreme* volatility where traders tend to reduce the passive liquidity supply resting on the LOB.

Variables	AMM	Execution	Microstructural	Momentum	Technical
VOL_{t-1}	0.079*** (10.56)	0.073*** (9.82)	0.075*** (10.07)	0.070*** (9.38)	0.071*** (9.54)
D^s	-0.002 (-0.19)	0.001 (0.13)	-0.001 (-0.12)	0.003 (0.24)	-0.002 (-0.18)
$D^s * VOL_{i,t-1}$	-0.035** (-1.96)	0.007 (0.4)	-0.011 (-0.6)	0.030** (1.67)	-0.003 (-0.16)
$D^s * VOL_{i,t-1} * D^{HV}$	-0.015 (-0.95)	-0.0271* (-1.78)	-0.040** (-2.61)	-0.035* (-2.31)	-0.014 (-0/86)
δ_{MV}	-0.002 (-0.14)	-0.001 (-0.12)	-0.001 (-0.15)	-0.001 (-0.13)	-0.000 (-0.09)
δ_{VFTSE}	0.001** (2.53)	0.000** (2.5)	0.001** (2.05)	0.001** (2.42)	0.000*** (2.71)
δ_{IP}	0.023 (0.14)	0.022 (0.13)	0.025 (0.15)	0.024 (0.14)	0.020 (0.12)
α_0	-1.443** (-2.53)	-1.435** (-2.5)	-1.167** (-2.05)	-1.383** (-2.42)	-1.592*** (-2.71)

TABLE 4.4.5 – The table reports the coefficient estimates for the distributed lag OLS regression with the dependent variable being net liquidity supply, $NL_{i,t}^{s}$ of limit order submissions minus deletions at best bid-ask (BBO) prices for strategy s, stock i, in period t. The t-statistics are presented in parentheses and derived from Newey-West corrected standard errors. *, ** and *** correspond to statistical significance at 0.05, 0.01 and 0.001 levels respectively. Regressions are performed by aggregating activity of traders conducting the same AT strategy and is measured over the UK LSE LOB for five securities in June 2015. The lagged OLS regression estimated is: $NL_{i,t}^{s} = \alpha_0 + \sum_{i=1}^{5} D^i + VOL_{i,t-1} + D^s + D^s * VOL_{i,t-1} + D^s * VOL_{i,t-1} * D^{HV} + \delta + \epsilon_{i,t}$. The regression measures how net liquidity, $NL_{i,t}^{s}$, responds to price volatility in the previous period, $VOL_{i,t-1}$. Dummy variables are included in the OLS regression for the AT strategy, D^s, and variable D^{HL} that identifies 1% of periods with the highest levels of volatility. The set of control variables, δ, are incorporated into the OLS regressions. These control variables include a set of indicators that may affect volatility in the macroeconomic context, including the VFTSE Index, δ_{VFTSE}, and a set of stock-specific control variables, including firm market capitalisation to proxy for stock liquidity, δ_{MV}, and the inverse, δ_{IP}, to proxy for estimated transaction costs.

4.4.2.3 Causal Relationship

Robustness checks of the causal relationship between AT strategies' activity and market volatility is performed using a Granger non-causality panel regression test (Granger, 1969; Dumitrescu, 2012) that identifies statistical causality when lagged variables of x exhibit a causal effect by increasing the accuracy of predicting current values of y. It is necessary to contextualise the semantics of 'causality' commonly viewed as a corollary of a statistical relationship between two variables. Variable x is said to Granger cause variable y when lagged values of variable x precedes y to a defined significance level, indicating that lagged variables of x provide information regarding the future value of y that drives its value. Granger causality only allows for one to reject *no* causality rather than draw inference of whether one variable 'causes' the other. The model setup in this section tests for bi-directional Granger causality at the stock level between AT net liquidity supply levels for all different strategies, $NL_{i,t}^{s}$, and market volatility, $VOL_{i,t}$, in stock i over periods event-time t comprising 100 relevant LOB events. These variables are all defined in the previous section. Granger causality tests are performed by estimating the following equations for market volatility and net liquidity supply using a vector auto-regression VAR(n) framework:

$$VOL_{i,t} = \alpha_0 + \sum_{j=1}^{\mathrm{n}} \alpha_{1ij} NL_{i,t-j}^{s} + \sum_{j=1}^{\mathrm{n}} \alpha_{2ij} VOL_{i,t-j} + \varepsilon_{i,t}$$

$$NL_{i,t}^{s} = \beta_0 + \sum_{j=1}^{\mathrm{n}} \beta_{1ij} VOL_{i,t-\mathrm{j}} + \sum_{j=1}^{\mathrm{n}} \beta_{2ij} NL_{i,t-j}^{s} + \mu_{i,t}$$

The error disturbance terms $\varepsilon_{i,t}$ and $\mu_{i,t}$ represent variations in volatility and AT activity levels, respectively, that are not captured by lagged independent and control variables. Granger causality is tested over $n = 5$ evenly spaced period lags for each independent variable. Null hypotheses of no Granger causality between AT activity and volatility are developed at the stock-level:

$H_{0\mathrm{A}}: \alpha_0 = \alpha_1 = \alpha_2 = 0$; AT activity levels do not Granger cause volatility.

$H_{0\mathrm{B}}: \beta_0 = \beta_1 = \beta_2 = 0$; Volatility levels do not Granger cause AT activity.

Panel Granger Causality Tests are applied in this chapter to account for the stock-specific effects that allow for a holistic interpretation of Granger causality between the AT activity and volatility variables (Dumitrescu, 2012). Rejecting the null hypothesis $H_{0\mathrm{A}}$ and failing to reject the second null hypothesis $H_{0\mathrm{B}}$ demonstrates that lagged AT strategy net liquidity supply variables, $s_{i,t}$, can be used to predict current period volatility, $VOL_{i,t}$. The opposite is true for the reverse case.

The Panel Granger Causality Test utilises a Z-Tilde, $\tilde{Z}$, test statistic, which incorporates parametric Wald and F-tests performed implicitly, to test for Granger causality within the panel of stocks as a whole. The process to test the null hypothesis of whether lagged coefficients of independent variables are statistically different from zero across the panel data first requires performing F-tests for each stock, in a similar vein to original Granger causality, before calculating an average standard adjusted Wald statistic, $\bar{W}$. The Wald test statistic, W, is a function of the maximum likelihood estimate of the independent variable, $\bar{x}$, the proposed variable value, x_0, and variance matrix, $var(\bar{x})$, and draws from a Chi distribution, $W \sim \chi_k^2$. Based on the assumption that Wald statistics are i.i.d across all stocks, the standardised Z-Tilde, $\tilde{Z}$, test statistic can be computed as a function of the average Wald statistic, the number of panel samples, and observations. This leads to a final Z-Tilde statistic from which Panel Granger Causality tests can be executed. The Panel Granger test is only capable of detecting causality at the panel-level of multiple stocks (Lopez, 2017), rather than at the individual stock-level.

Bi-directional Panel Granger Causality Tests using five auto-regressive lags are tabulated in Table 4.4.6. Standardized Z-Tilde statistics are reported with the attached significance level for the five individual AT strategies along the horizontal axis and two relevant bidirectional tests along the vertical axis. Note that the null H_{0A} is the hypothesis that lagged net liquidity variables for each strategy do not provide information for the current volatility levels, that is net liquidity changes do *not* Granger cause volatility. The opposite case is true for H_{0B}.

The key result is that the H_{0B} is rejected for all strategies and thus, the hypothesis that volatility in the preceding periods does *not* influence current activity levels of AT strategies can be rejected. Lagged volatility may provide useful information to predict whether AT firms are going to shift their net liquidity supply. This aligns directly with results in the previous sections that found correlations and statistical relationships between lagged and contemporaneous volatility and increases in AT activity. Furthermore, the null hypothesis H_{0A} that lagged net liquidity fluctuations of Execution and Momentum strategies does not Granger cause current period volatility is not rejected. Both strategies have Z-Tilde values of 0.84 and 1.49, respectively. The results can be interpreted as indicating that lagged net liquidity supply variables for firms conducting Execution and Momentum strategies do not provide statistically significant information regarding future volatility, across the sample securities studied.

Variables	AMM	Execution	Microstructural	Momentum	Technical
$VOL_t \sim NL_i^s$ (H_{0A})	3.67***	0.84	7.58***	1.49	1.99*
$NL_i^s \sim VOL_t$ (H_{0B})	10.80***	4.21***	4.86***	3.86***	6.03***

Table 4.4.6 – This table summarises results for bidirectional Granger causality between volatility, VOL_i and net liquidity supply, NL_i^s, for the different AT strategies. *, ** and *** correspond to statistical significance at 0.05, 0.01 and 0.001 levels respectively.

4.4.2.4 Fundamental Relationship

Previous sections provide evidence that a contemporaneous, dynamic and statistical relationship exists between different types of AT activity and market volatility for the sample analysed. However, it is necessary to understand the fundamental relationship between the two variables given the endogeneity issue where OLS coefficients may be biased, as previously discussed. Traditional methods to overcome the issue of AT activity and volatility being endogenous variables include performing an instrumental variable analysis (Chaboud, 2014) or testing around market-wide exogenous shocks (Caivano, 2015; Brogaard, 2017) to obtain unbiased estimates of the coefficient variables. However, given that there is no natural instrument for AT due to its near-zero auto-correlation, these methods are not an option. Rather, this section extends the method used by Brogaard (2010) to overcome some of the issues caused by the endogeneity problem by developing a hypothetical time series of prices that assume the specific AT strategy being analysed was not prevalent in the market. One primary difference between the approach utilised in this chapter is the focus on limit orders, that is, activity on the LOB rather than just transaction data.

Testing the impact of AT strategies on volatility using this framework requires a comparison between two price series. One price series maintains the behaviours of all traders in the environment throughout the trading day modelling the *actual price path*, whilst a second *alternative price path* extricates a specific trading strategies' activity from the daily quotes as if the traders had not participated on the LOB at all. Realised volatility is calculated for both price paths for each individual AT strategy and a statistical test is performed to drive assertions regarding the fundamental relationship between AT activity and volatility. Table 4.4.7 reports the results of the analysis for each strategy. The first two columns compute the mean and standard deviation of volatility for *alternative* price paths when the limit orders at the BBO placed by traders executing these strategies are annulled from the dataset. The third column refers to the difference between the actual and alternative price path volatility. The difference in means between the two paths is statistically analysed using a paired t-test that incorporates Newey-West corrected errors to adjust for auto-correlation. The results demonstrate that all five AT strategies have a statistically significant fundamental positive impact on volatility at the 0.001% significance level when comparing the alternative hypothetical price path of no AT activity against the actual price path. AMM strategies appear to have the strongest dampening effect on volatility, such that when they are removed from the dataset, volatility increases on average by 2.95%. On the opposite side of the spectrum, Momentum strategy activity also reduces volatility levels though with a smaller magnitude of only 0.19%, despite being the largest category by the number of orders placed at the BBO. The results indicate that AMMs and Execution algorithms are two trading strategies that significantly dampen volatility.

Strategy	Mean (SD)	Difference
AMM	0.2858 (0.0330)	2.95%***
Execution	0.2830 (0.0349)	1.93%***
Microstructural	0.2797 (0.0404)	0.75%***
Momentum	0.2781 (0.0308)	0.19%***
Technical	0.2788 (0.0364)	0.41%***

Table 4.4.7 – Fundamental impact of AT strategies on price volatility through a statistical comparison of actual and alternative hypothetical price paths when the specific AT strategy is annulled from the trading day dataset. Difference is measured between the two price paths in percentage points. *, ** and *** correspond to statistical significance at 0.05, 0.01 and 0.001 levels respectively.

4.4.3 Price Discovery & Efficiency

This section compares the impact of firms conducting different AT strategies on price discovery and efficiency. Price discovery refers to the ability of markets to impound information into prices effectively whilst price efficiency defines the level of informational content available in prices. Modern high-frequency financial markets increasingly rely on firms executing automated trading algorithms to aide in the price formation process by supplying liquidity and transacting on the limit order book (LOB). Incorporating new public and private information efficiently into prices is a critical component of price discovery in well-functioning markets (O'Hara, 2003; Brogaard, 2014). Theoretically, price discovery and efficiency are improved, and institutional investors benefit, when informed trading by AT firms is performed in the direction against transitory pricing errors caused by a significant demand for liquidity, and in the direction of permanent price movements (Kyle, 1985). Intuitively, the price formation process is also aided by AT firms capable of detecting price anomalies and placing limit orders in a way that reverts prices back to their fair value. Regulators have an important role in creating a level playing field that places legal limitations on trading certain sets of private information and ensuring firms have equal opportunity to access timely and holistic market data.[24]

Academic research into the relationship between algorithmic or high-frequency trading and price discovery is assessed. Brogaard (2014) analyses the impact of HFTs on price discovery and price efficiency using a state-space model that decomposes price innovations into a permanent component that represents information and a transitory component that represents transitory volatility. This process is conducted using a Vector Autoregression (VAR) framework developed by Hasbrouck (1991, 1995) and

[24] MiFID II (RTS 10) requires trading venues to provide fair and non-discriminatory fee structures and services.

employed in various academic studies to decompose the components of price impact (Benos, 2012). Brogaard (2014) finds that HFTs contribute to price efficiency by conducting informed liquidity demanding trading in the direction of permanent trends and opposite to transitory pricing errors, attaining revenues in excess of the spread and trading fees on average. Additionally, Brogaard (2014) argues that whilst HFTs demanding liquidity impose adverse selection costs on non-HFT firms trading with transitory errors, there is a net positive externality from improved efficiency of prices as aggressive HFT activity reverts prices to their fair value. These results align with Hendershott (2013) and Chaboud (2014) who use a regulator-identifier for HFT activity and study the Deutsche Bourse equity and foreign exchange markets, respectively, finding evidence that HFT and ATs improve price efficiency. Similarly, the adoption of faster trading technology in German equity markets was examined by Riordan (2012) with the increased capacity to update limit orders over short time frames resulting in the supply-demand liquidity dynamics of the market changing in a way that allowed for faster price discovery. Benos (2012) utilises the VAR methodology and variance decomposition framework, similar to Brogaard (2014), to decouple the price efficiency and noise component of market impact, from which the contributory impact on each component from aggressive HFT activity can be measured. The authors found that aggressive HFTs tend to have a greater contribution to both components than passive HFTs.

The following analysis is conducted to assess the impact of different AT strategies on the price discovery process. First, impulse response functions are developed to measure the price impact induced from AT strategy activity, serving as a proxy for the level of private information impounded into different trading strategies' transactions and limit order actions. Next, a VAR Information Share methodology, involving a Choleski variance decomposition technique, is performed to parse the price impact from AT strategy activity into *informational* and *noise* components to quantify the contribution of different strategies to price efficiency. Finally, a dynamic linear model with a recursive Kalman filter is fitted to the data to estimate the permanent and transitory components of the efficient price attributed to each AT strategy.

4.4.3.1 Impulse Response Functions & Market Impact

Impulse response functions provide an econometric framework for assessing the role of various AT strategies in improving price discovery on UK equity markets. Quantifying the level of private information infused into actions executed by AT firms is performed using an event-time adaptation of Hasbrouck's k^{th} order vector auto-regression (VAR) methodology (Hasbrouck, 1991, 1993, 1995). VAR models can be used to ascertain the magnitude of price responses to information from various channels, particularly public and trader-specific sources. These price responses are generally referred to as the permanent price impact of a trader. Similar methodologies have been applied by Hendershott (2013), Brogaard (2010) and Benos (2012) to assess the impact on price discovery from HFT, AT, macroeconomic news, or some combination of all three.

Permanent price impact arising from the trading activity in period t attributed to AT firms conducting strategy s serves as a proxy for the informational content of limit orders placed or trades executed by these traders on the LOB. This proxy can be viewed as a derivative of the traders' level of private information and contribution to price discovery. The set of five AT strategies studied in this chapter are analysed, including Automated Market Maker (AMM), Execution (EX), Microstructural (MS), Momentum (MO) and Technical (TE) strategies, representing components $s \in S$. The multivariate stochastic VAR(k) models the relationship between endogenous variables of log midpoint returns, R_t, and signed order flows, O_t, using k lags of each variable. OLS regression is used to test the significance of the coefficients. For the purpose of this analysis, order flows are measured for each individual AT strategy s, O_t^s, taking a positive value when these traders execute an aggressive buy and a negative value when executing a negative sell. The VAR model is specified as:

$$R_t = \sum_{i=1}^{k} \alpha_i^s R_{t-i} + \sum_{i=1}^{k} \sum_{s \in S} \beta_i^s O_{t-i}^s + \varepsilon_{0t} \qquad O_t^s = \sum_{i=1}^{k} \delta_i^s R_{t-i} + \sum_{i=1}^{k} \sum_{s \in S} \vartheta_i^s O_{t-i}^s + \varepsilon_{st}$$

The return equation estimates the period t excess midpoint return, R_t, as a function of both lagged endogenous order flow and return variables, allowing temporal interdependencies between variables to be captured by the model. Error terms for each equation, ε_{it}, have expected mean zero, with contemporaneous variance-covariance matrix, $E(\boldsymbol{\varepsilon}_t \boldsymbol{\varepsilon}_t') = \Omega$. The matrix Ω has non-diagonal covariance components given that the order-flow equations are contemporaneously correlated to some extent. Implicit within the VAR model is the assumption that causality between returns and order flow is unidirectional, given that the excess return independent variables can be contemporaneously influenced by order flow variables, though not vice versa, when depth at the BBO is extinguished by an incoming aggressive order (Benos, 2012).

A VAR(5) model is estimated for strategies $s \in S$ using five lags over the trading period of 9 am to 4 pm to align with the contextual settings of this chapter. For example, testing for the price impact of AT firms executing AMM strategies requires the estimation of a VAR model using regressions for the return, R_t^s, order flow for AMMs, O_t^{AM}, and order flow for non-AMMs. By amalgamating the price impact measures of all firms conducting a specific AT strategy, one can draw inferences regarding the impact of that strategy on the price discovery process. To do so, impulse response functions are developed by first inverting the VAR model and applying a Choleski decomposition of the variance-covariance matrix, Ω, that results in the following vector moving average (VMA) model:

$$\begin{bmatrix} R_t^s \\ \vdots \\ O_t^s \end{bmatrix} = \begin{bmatrix} R^R(L) & \cdots & S^R(L) \\ \vdots & \ddots & \vdots \\ R^S(L) & \cdots & S^S(L) \end{bmatrix} \begin{bmatrix} \varepsilon_{0t} \\ \vdots \\ \varepsilon_{St} \end{bmatrix}$$

Decomposition into VMA form results in mutually orthogonal error terms, $\boldsymbol{\varepsilon}_t \boldsymbol{\varepsilon}_t' = \boldsymbol{I}$, that models causal impulses or shocks to one error term ε_t without being concerned with its correlation to other error terms. Permanent price impact from aggressive trades by firms conducting different types of AT strategies is estimated by the impulse response functions for the AT strategy, $s^R(L)$, where the T-period lag polynomial can be expressed in the form $s^R(L) = \sum_{i=0}^{T} s_i^R L^i$. Each impulse response function estimates the informational content and permanent impact of a trade innovation, or unexpected portion of the trade, on the future price midpoint following an aggressive trade executed by traders conducting strategy s.

Results for the impulse response functions of each trading strategy over 50 event periods for the VAR(5) model are graphically depicted in Figure 4.4.4. Impulse response functions for all lagged events are cumulatively summed over the event period for all individual trading strategies. Shaded areas represent 90% confidence intervals which are determined from a bootstrap distribution, ψ, based on the cumulated impulse response function's VAR coefficients (Benos, 2012). The y-axis is transformed to represent the impact of impulses using a basis point (bp) metric which can be interpreted as the response of the midpoint prices in bps after a transaction has been executed by a trader conducting the relevant AT strategy. For robustness, impulse response functions were developed over both 100 and 300 event time frames with results comparable to the analysis conducted over the 50-event time frames.

The results indicate that Momentum strategies have a significant long-term impact on prices, collectively contributing more to price discovery and information than other AT strategies. Over the 50-event period, the average price impact over the stocks studied is 2.2 bps when firms conducting Momentum strategies execute a trade, with Execution strategies also contributing significant levels of information with an average price impact of 0.96 bps. The higher price impact from Momentum strategies correlates with the higher level of aggressive trading activity, given that Momentum strategies execute 23.3% of aggressive trades whilst Execution strategies are only 16.6% of the time the aggressor, averaged across all stocks. However, the proportional price impact from Momentum strategies is significantly higher than other trader types. The source of the private information impounded in Momentum strategies trades may come in the form of a faster imputation of real-time LOB data into their trading algorithms, potentially due to more advanced low-latency algorithmic trading technology. These findings can be considered in the context of Riordan (2012) and Brogaard (2014) who found a statistical difference between the informational content of HFT and non-HFTs trades, with HFT contributing more to price discovery. Intuitively, the relatively low long-term price impact from Microstructural and Technical strategies stems from their algorithmic inputs of micro and longer-term LOB phenomena, respectively, that offers less informational content to the price discovery process.

Short-term price impact from Momentum strategies immediately following a trade innovation are highest at 1.19 bps on average over the stocks analysed. Technical strategies have the smallest temporary price impact of only 0.4 bps, however, these strategies represent only 7.9% of aggressive trades. Interestingly, AMMs have the second largest short-term impact from a price innovation of 0.99 bps which is 23% higher than Execution strategies. However, over the 50-event period the Execution strategies price impact remains stable whilst the permanent price impact from AMM trades gradually recedes. This indicates that prices may be overreacting to trades by AMMs with the informational content of these traders reassessed after the immediate impact.

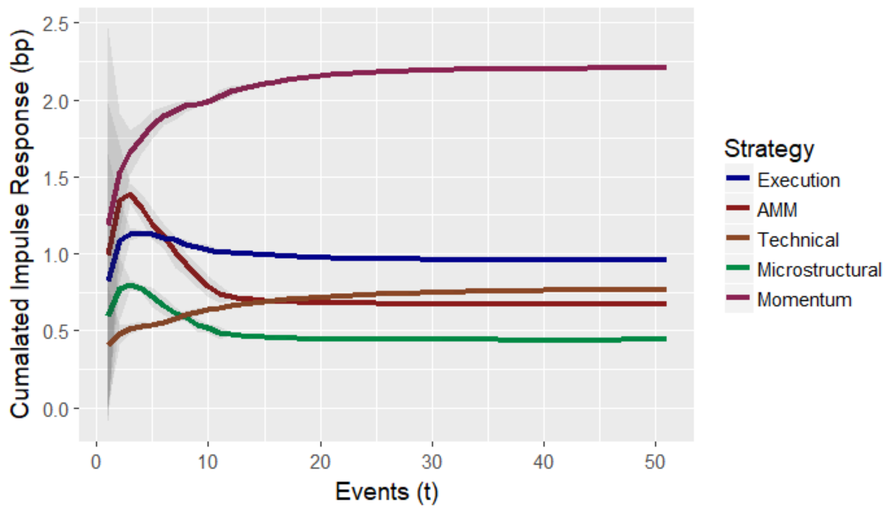

Figure 4.4.4 – Impulse response functions for different AT strategies for all trades and quote updates on the LSE at the BBO. Individual VAR models are used to estimate the cumulative price impact response from an impulse represented by an aggressive execution by traders conducting relevant strategy s. Results are presented for 50 lagged events and averaged over five securities analysed within a week period. Confidence intervals represented by the shaded areas are set at 90% using a bootstrap distribution estimated over 500 iterations.

4.4.3.2 Information Share Analysis

Price formation on markets can further be examined by analysing the contribution of trades executed and limit orders placed by different AT strategies to the price discovery process, rather than merely the permanent price impact as analysed in the previous section. Hasbrouck (1995) suggests a VAR 'Information Share' methodology coupled with a variance decomposition technique to parse price impact from AT strategy activity into *informational* and *noise* components. Information can be categorised as 'public', which is available to all traders and requires minimal analysis, or 'private', which may also be public but requires processing and analysis in order to make an informed action. The Information Share methodology has been employed to a degree by Benos (2012), Brogaard (2014) and Hendershott (2013) to analyse the relative contribution of HFT and non-HFT traders to price discovery. We extend these methods into the domain of AT by quantifying the contribution of different AT strategies to information and noise in the price discovery process.

The Information Share methodology assumes that observed midpoint prices, MP_t, have both a permanent and transitory component. The permanent efficient price component from information can be defined as MP_t^*, which follows a random walk, ϱ_t. The transitory non-persistent stationary price component from noise, κ_t, is modelled as residual noise with no long-term impact on prices, such that $\lim_{h \to \infty} E(\kappa_{t+h}) = 0$, leading to:

$$MP_t = MP_t^* + \kappa_t$$

$$MP_t^* = MP_{t-1}^* + \varrho_t \quad \varrho_t \sim iid\left(0, \sigma_\varrho^2\right) \quad where\ E(\varrho_t) = 0, E(\varrho_t^2) = \sigma_\varrho^2\ and\ E\left(\varrho_i \varrho_j\right) = 0 \quad i \neq j$$

Decomposition of the observed price allows for midpoint returns to be expressed by $R_t = MP_t - MP_{t-1} = \Delta P_t^* + \Delta\kappa_t$. The permanent price impact from information is represented by the term ΔP_t^* and is estimated using the VAR model specified in the previous section. Permanent impact terms have an approximate *information* variance σ_ϱ^2, with deviations from the stochastic random walk representing *noise* and incorporated into the transitory impact $\Delta\kappa_t$ with corresponding noise variance σ_κ^2. Variance decompositions for both the efficient information price and transitory noise are performed separately.

The contribution of different AT strategies to *information* variance, σ_ϱ^2, is estimated by decomposing the variance into contributions from public information, $\left(\sum_{i=0}^{\infty} R_i^R\right)\sigma_{\varepsilon 0}^2$, and private information, $\left(\sum_{i=0}^{\infty} s_i^R\right)\sigma_{\varepsilon s}^2$, for each individual trading strategy $s \in S$. This process extends from the VAR and related VMA specified in the previous section. The VMA requires a Choleski decomposition of the VARs variance-covariance matrix, based on the assumption that error terms in the matrix are mutually orthogonal with unit variance. Extending from these models, the permanent price impact, ΔP_t^*, and its variance, σ_ϱ^2, can be decomposed as follows for all strategies S to proxy for each AT strategies contribution to price discovery:

$$\Delta P_t^* = \left(\sum_{i=0}^{\infty} R_i^R\right)\varepsilon_{0t} + \sum\nolimits_{s \in S}\left(\sum_{i=0}^{\infty} s_i^R\right)\varepsilon_{st} \qquad \sigma_\varrho^2 = \left(\sum_{i=0}^{\infty} R_i^R\right)^2 \sigma_{\varepsilon 0}^2 + \sum\nolimits_{s \in S}\left(\sum_{i=0}^{\infty} s_i^R\right)^2 \sigma_{\varepsilon s}^2$$

Price discovery is ascribed to individual AT strategies by decomposing the efficient price variance, σ_ϱ^2, into components of public and strategy-specific private information through the estimation of contribution ratios. After fitting the VAR model, performing VMA transformations and estimating impulse response functions over 50 lags, the informational contribution ratio for each component can be expressed as an average fraction of the price impact information variance for public and trader-specific information:

$$\frac{\left(\sum_{i=1}^{50} R_i^R\right)^2 \sigma_{\varepsilon 0}^2}{\sigma_\varrho^2}, \frac{\left(\sum_{i=1}^{50} AM_i^R\right)^2 \sigma_{\varepsilon 1}^2}{\sigma_\varrho^2}, \frac{\left(\sum_{i=1}^{50} EX_i^R\right)^2 \sigma_{\varepsilon 2}^2}{\sigma_\varrho^2}, \frac{\left(\sum_{i=1}^{50} MS_i^R\right)^2 \sigma_{\varepsilon 3}^2}{\sigma_\varrho^2}, \frac{\left(\sum_{i=1}^{50} MO_i^R\right)^2 \sigma_{\varepsilon 4}^2}{\sigma_\varrho^2}, \frac{\left(\sum_{i=1}^{50} TE_i^R\right)^2 \sigma_{\varepsilon 5}^2}{\sigma_\varrho^2}$$

Decomposing individual AT strategy contributions to the transitory *noise* variance, σ_κ^2, of the efficient price noise component, κ_t, is performed to complement the information variance decomposition in the above to gain an overall view of how different traders impact information efficiency and price discovery on UK equity markets. The transitory efficient price is assumed to follow a moving average process from the VMA residuals ε estimated using VAR and VMA models built in the previous section (Benos, 2012; Hasbrouck, 1993). Thus, transitory prices can be expressed as a function of the residual errors, ε, coefficients for excess returns, α_i^s, and individual strategy $s \in S$ order flows, β_i^s. Following from Hasbrouck (1993) and Benos (2012), by combining the equations estimated for the VAR and VMA, and treating residuals as a moving average process, the transitory noise price component, κ_t, with parameters, α_i^s and β_i^s, can be expressed as:

$$\kappa_t = \sum_{i=0}^{\infty} \alpha_i^s \varepsilon_{0t} + \sum_{i}^{\infty} \sum_{s \in S} \beta_i^s \varepsilon_{st}$$

$$\alpha_i^s = \left(\sum_{i=0}^{\infty} R_i^R\right) \qquad \beta_i^s = \left(\sum_{i=0}^{\infty} s_i^R\right)$$

Variance decomposition of transitory variance σ_ϱ^2 is performed by using a lower bound for the transitory standard deviation, σ_ϱ, such that:

$$\sigma_\kappa^2 = \left\{\sum_{i=0}^{\infty} \sum_{s \in S} ((\alpha_i^s)^2 + (\beta_i^s)^2)\right\}$$

Similar to the information decomposition, the contribution of different AT strategies and public information to transitory noise variance can be formulated as a series of noise contribution ratios by dividing both sides of the above equation by the noise variance and estimating components using expected values of the 50-lag impulse response function developed in the previous section. The resulting decomposition expresses the contribution of non-trading sources, $\left(\sum_{i=0}^{50} (\alpha_i^s)^2\right)$, and individual trading strategies s, $\left(\sum_{i=0}^{50} (\beta_i^s)^2\right)$ to noise.

Results presented in Table 4.4.8 compare the contributions from each AT strategy to both information and noise variance using contribution ratios computed from the decomposition of their respective variances. A corresponding information to noise ratio is also calculated and tested for statistical significance.

Strategy	Information Mean (SD)	Noise Mean (SD)	Information-to-Nosie Ratio (T)
Public Information	0.1441 (0.0208)	0.1312 (0.0321)	1.0986
AMM	0.1476 (0.0486)	0.1230 (0.0342)	1.1993*
Execution	0.1432 (0.0319)	0.1702 (0.0248)	0.8412
Microstructural	0.0528 (0.0145)	0.0871 (0.0187)	0.6064**
Momentum	0.4573 (0.0432)	0.4172 (0.0341)	1.0960*
Technical	0.0548 (0.0051)	0.0722 (0.0067)	0.7585

Table 4.4.8 – Each strategy is decomposed into variance and noise contributions, with the information-to-noise (ITR) ratio then computed. This first requires an estimation of the VAR model defined in Section 4.4.3.1 for each stock in the sample over the period of analysis. The VMA form is then defined using a Choleski decomposition. The observed midpoint price, MP_t, is decomposed into an efficient permanent component, MP_t^*, and transitory component, κ_t. Variance decomposition for information, σ_ϱ^2, and noise, σ_κ^2, is then performed to compute the fraction of each component attributable to the specific AT strategy and public information. A t-test is performed to test whether the ITR is significantly different from zero. *, ** and *** correspond to statistical significance at 0.05, 0.01 and 0.001 levels respectively.

The table presents the information contribution ratios representing the impact of different information sources on price discovery. Momentum strategies contribute 45.7% of the total efficient price information variance averaged over the five stocks analysed. This result is consistent with the price impact metric developed using impulse response functions in the previous sections that found Momentum strategies to have the largest permanent price impact. Furthermore, Momentum strategies' contributions are significantly higher than their 23.3% share of aggressive trading, indicating that each trade contributes nearly twice as much information in relative terms. Momentum strategies are shown to contribute more information than the next three components combined, with public non-transaction related information and private trader-specific information from AMM and Execution strategies each contributing between 14-15% of information in the price discovery process. The 14.4% level of public information contribution is in line with results from Benos (2012) who studies a similar dataset over an earlier period. Given that all five stocks studied in this chapter are positioned at the higher end of the FTSE100 market capitalisation rankings, the intuition behind the high level of non-trader information is that these stocks have significant coverage by the public, deeming that information regarding their fundamental value infuses into the price from non-trader domains.

Additionally, Momentum strategies contribute predominately to the noise variance, σ_ϱ^2, when the variance is modelled as a moving average process of the previously expressed VMA model's residuals (Hasbrouck, 1993). The noise contribution of 41.7% by Momentum strategies is followed by Execution strategies which contribute 17% to the variance of the efficient prices noise component and public information which represents 13.1% of the total noise component.

The overall role of AT strategies in price discovery and informational efficiency can be computed by comparing the information and noise contributions to derive an information-to-noise (ITN) ratio metric. A t-test is performed to determine whether each strategies ITN is significantly different from one using data across samples and dates for each AT strategy group. The ITN ratio for Momentum strategies, public information and AMM are all above 1 with AMM attaining the highest ratio of 1.20. This implies that AMMs provide more information than noise in their interactions on the LOB. Microstructural strategies have the lowest ITN of 0.61, statistically significant at the 0.001% level. This indicates that in aggregate firms conducting Microstructural strategies infuse significantly less information than noise through their actions on the LOB. Intuitively, firms that utilise an order placement strategy based on LOB features are not contributing new information to the price discovery process, but rather trading based off micro-period changes in LOB dynamics.

The results from the VAR decomposition analysis can be compared against studies performed by Hendershott (2011) and Brogaard (2014) who find that HFTs explain a larger proportion of the variance in returns than non-HFTs. Concurrently, they find that HFTs also contribute to noise, with Hendershott (2011) deeming 33% of noise variance as attributable to HFT, resulting in an ITN significantly lower than one. In contrast, Benos (2012) measures the ITN of HFTs to be 1.13 on average over four stocks studied on the UK equity market, using a similar dataset over an earlier period. It appears possible that the HFT dataset used by Benos (2012) may flag certain firms conducting AMM and Momentum strategies, possibly others, as HFTs given their higher ITN in the study performed in this section.

4.4.3.3 Dynamic Linear Model Analysis

Modelling the role of different AT strategies in the price discovery can also be performed using an event-time Dynamic Linear Model (DLM) methodology as an alternative to VAR models (Brogaard, 2014). DLMs, also referred to as state space models, utilise a recursive Kalman filter as an estimation algorithm with the initial state vector formulated using an asymptotically unbiased Bayesian maximum likelihood estimation. The primary difference with VAR Models employed by Hasbrouck (1991, 1993) is that DLM state space models are infinite lag auto-regressive models, therefore the lag structure does not need to be truncated (Hendershott, 2011). One similar attribute maintained in DLMs is the requirement to perform a decomposition of the efficient price into permanent and transitory components in a similar vein to Hasbrouck's (1995) Information Share methodology.

DLMs maintain the Vector Autoregression model assumption that the efficient log mid-quote price can be decomposed into permanent and transitory components, $MP_t = MP_t^* + \kappa_t$. The efficient price captures permanent price increments, MP_t^*, and a transitory component that represents residual noise, κ_t. Furthermore, the permanent efficient price component can be modelled as a martingale, $MP_t^* = MP_{t-1}^* + \varepsilon_t$, where $\varepsilon_t \sim iid(0, \sigma_\varepsilon^2)$. Based on Hendershott (2011) and Brogaard (2014), the martingale term ε_t can be specified as a function of the surprise innovation, $\widetilde{O_{i,t}^s}$, in the trading activity of AT strategy firm s. This surprise innovation is estimated as the residual of a VAR(k) model of the period t net order flow of bid minus ask volume, $O_{i,t}^s$, which is regressed against k lagged variables of itself, with ten lags used for this experiment. After the VAR model is fitted and residuals estimated, a DLM state space model is deployed to estimate the permanent price impact, $\varepsilon_{i,t}^s$, for stock i and AT strategy s, specified as:

$$\varepsilon_{i,t}^s = \alpha_i^s \widetilde{O_{i,t}^s} + \mu_{i,t}^s$$

As identified, $\widetilde{O_{i,t}^s}$ represents the surprise innovation of the signed order flow, $O_{i,t}^s$, of AT strategy s for stock i which is computed as the residuals of the VAR(5) model. Price movements that arise independent of the trading activity of AT firms is represented by the variable μ_t. Furthermore, the transitory pricing error component κ_t which is assumed to be stationary, is estimated by the DLM state space model using a single lagged period auto-regressive model of order flow variables for strategy s, $O_{i,t}^s$, giving:

$$\kappa_{i,t}^s = \varphi^s \kappa_{t-1}^s + \beta_i^s O_{i,t}^s + \omega_{i,t}^s$$

DLM state space models measure the impact of different AT strategies on price discovery. An individual model is built for each ISIN-Date pair using the maximum likelihood estimates of model parameters before applying a Kalman Filter and smoothing technique to decompose permanent and transitory price components (Brogaard, 2014; Menkveld, 2013). Each dataset encompasses a valid sample to estimate the state-space model, which requires the log mid-quote for each price innovation and the signed order flow volume for traders conducting specific AT strategies. Furthermore, the covariance between the permanent price impact error, $\mu_{i,t}^s$, and transitory pricing error, $\omega_{i,t}^s$, is assumed to be zero, that is, the errors are uncorrelated such that $Cov(\mu_{i,t}^s, \omega_{i,t}^s) = 0$.

Results for the DLM analysis are reported in Table 4.4.9. Panel A and B split the results into complementary permanent and transitory price components. In line with the results of previous sections, all trading strategies' order flow activity have a statistically significant positive correlation with permanent changes to the efficient price. ATs conducting Momentum strategies tend to induce the highest impact on the efficient price from trading activity, with a £10,000 positive order flow surprise, $\tilde{O}_{i,t}^{MO}$, predicted by the DLM state space model to have a 0.81 bps permanent price impact. High coefficients can also be viewed as a proxy for the level of informed trading. The result that Momentum and Execution strategies have the highest informational content impounded into their order flow aligns with the analysis performed in the previous Section 4.4.3.1 using impulse response functions.

Transitory price component coefficients, β^s, are negatively correlated with trading activity for all strategies except for Microstructural strategies. These coefficients proxy for the level of noise induced by traders on the LOB with negative values indicating that firms trade against noise or transitory pricing errors. Firms executing Technical and AMM strategies have the lowest transitory coefficients of -2.82 and -2.03, respectively, indicating that these firms trade in the direction opposite to transitory pricing errors and contribute the most to price efficiency. Evidently, Microstructural strategies have a positive transitory component coefficient which indicates that these firm on average trade with pricing errors, potentially compounding volatility and noise on the LOB. This aligns with the low information-to-noise ratio computed for Microstructural firms in the previous section when applying a VAR Information decomposition method. The coefficients for the lagged auto-regressive error term, φ^s, in the transitory pricing error model are stable in the 1.06 to 1.28 range on average.

In traditional market-making models risk-neutral AMMs supply liquidity on the LOB and trade when matched against by informed traders with superior private information (Cartea, 2018). Traders consistent with this behaviour would have high DLM regression coefficients for the surprise innovation term, α_i^s, and low coefficients for the order flow term in the transitory equation, β_i^s. From the results it appears that AT firms clustered by the Spherical K-Means algorithm into the AMM category on average meet the behaviour of AMMs as expected from the theoretical risk-neutral models, attaining a low transitory coefficient and a high surprise innovation coefficient.

Microstructural strategies' positive transitory equation order flow coefficient, φ^s, indicates that these strategies may be trading with temporary price disconnections from equilibrium which has three potential explanations. First, traders behaving in this manner will generally be performing some form of risk management procedure, such as liquidating a position in a short time period. In these cases, it becomes necessary to trade quickly, even if that means trading in the direction of only temporary price changes. A second explanation may be that firms conducting Microstructural strategies are potentially being adversely selected by more informed traders. Thirdly, a nefarious explanation could be that some Microstructural strategies are engaging in market manipulation. Both the SEC (2010) and Brogaard (2014) explain a situation where a 'proprietary trading firm' places an order to establish a new BBO price, before matching the order with themselves from the opposite direction. This behaviour is related to multiple manipulative trading strategies. However, an analysis of the average ratio of trades executed against the same broker to those executed against a different broker indicates that Microstructural strategies ratio have a ratio in the mid-range of the five AT strategies being analysed, indicating the nefarious explanation may not be the case.

	Permanent (Panel A)		Transitory (Panel B)		
Strategy	α^s *Bps/£10,000*	$\sigma(\widetilde{O^s_{i,t}})^2$ *£10,000*	φ^s	β^s *Bps/£10,000*	$\sigma(O^s_{i,t})^2$ *£10,000*
AMM	0.65*** (18.71)	0.19	1.28	-2.01* (-2.04)	0.21
Execution	0.76*** (24.68)	0.13	1.27	-1.74 (-1.23)	0.14
Microstructural	0.67*** (11.67)	0.11	1.18	0.45 (0.41)	0.12
Momentum	0.81*** (20.17)	0.20	1.06	-0.07 (-0.01)	0.22
Technical	0.75*** (14.13)	0.06	1.09	-2.82 (-1.24)	0.37

Table 4.4.9 – Permanent (Panel A) and Transitory (Panel B) price components of the Dynamic Linear Model (DLM) with Kalman filter estimated over all sample securities. *, ** and *** for T-stat in brackets correspond to statistical significance at 0.05, 0.01 and 0.001 levels.

4.5 Conclusion

In this chapter, we utilise feature attribution methods to assess the internal dynamics of each DNN trained for a specific ISIN-Date-Broker tuple, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$, that allows for the extraction of feature importance values for individual algorithmic traders. This enabled firms to be clustered by the algorithmic trading strategy being executed using deep and machine learning methods before evaluating the aggregate impact of each strategy on UK equity market quality.

Backpropagation-based feature attribution methods of DeepLIFT (Shrikumar, 2017) and Integrated Gradients (Sundararajan, 2017) are employed to quantify the microstructural LOB feature importance values for each of the 802 unique FTSE100 CLOB ISIN-Date-Broker tuple, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$, DNN models. We first analyse and assess various distance correlation, synaptic weights, input perturbation and backpropagation attribution methods. The analysis indicates that Integrated Gradients and DeepLIFT have the highest degree of accuracy when computing feature importance metrics based on the relative attribution analysis measures of delta-log odds (Shrikumar, 2017) and Sensitivity-n (Acona, 2018). Feature attribution methods are employed to bridge the disparity between interpretability and performance innate to DNNs, commonly viewed as 'blackbox' models, allowing practitioners to interact with and understand the traditionally opaque internal dynamics of the model. An assumption of this thesis is that LOB features manifestly represent algorithmic inputs into a firm's trading strategy. Intuitively, there exists an inextricable connection between the state of the LOB environment that an algorithmic trader exists within and the underlying strategy for conducting profitable actions in that environment. Certain systemic elements of the LOB environment drive traders' behaviour more than others. Quantifying the degree to which certain features motivate a firm to perform a certain action on the LOB provides a basis for determining what type of algorithmic trading strategy that firm is employing. This chapter untangles the complex interdependencies between LOB features using Integrated Gradients and DeepLIFT methods, resulting in interpretable statistical representations of feature saliency for each unique tuple, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$, summarised in the $\mathcal{FI}$ described in Section 4.2.4.

Unsupervised Spherical K-Means algorithms (Buchta, 2012) are used next to cluster unique trader tuples, $\langle \mathcal{S}, \mathcal{D}, \mathcal{B} \rangle$, into one of five algorithmic trading strategies – Automated Market Making (AMM), Execution, Microstructural, Momentum and Technical strategies - based on their feature importance values, $\mathcal{FI}$. We employ an unsupervised machine learning method to infer structural patterns in the feature importance datasets without the aid of any response variable or class label. The dataset is shown to have a good clustering tendency with an optimal number of clusters, five, that matches the number of trading strategies being analysed. We perform cluster validation to confirm that the Spherical K-Means is the optimal K-Means algorithm for separating the FTSE100 feature attribution dataset. The result of this procedure is the partitioning of algorithmic trading firms into the five interpretable, exclusive and distinct strategy clusters, underpinned by a robust analysis and visualisation of the clustered firms. We find evidence that firms clustered into specific algorithmic trading strategy groups have higher feature importance values for features that are specific and relevant to that particular strategy.

Finally, we extend the contemporary symposium of public and regulatory discourse debating the impact of machine-driven algorithmic trading on the quality of financial markets by approaching the subject from a perspective that places the algorithmic trading *strategy* as the unit of analysis rather than the firm *type*. The aggregate impact of each algorithmic trading strategy on UK equity market quality was empirically assessed along dimensions of liquidity, volatility and price discovery.

Assessing the liquidity characteristics of each algorithmic trading strategy, we find that Momentum and AMM strategies provide the largest proportion of liquidity on UK equity markets, partaking in 33% of transactions and submitting 31.2% of orders at the BBO, respectively. These two strategies also have the tightest aggregate quoted spreads and market depth on average throughout the trading day. This analysis underpins the importance of Momentum and AMM strategies to the efficient functioning of markets given the critical role of liquidity as an essential pillar of market quality.

Assessing the relationship between algorithmic trading and volatility requires consideration of the relationship along contemporaneous, dynamic and fundamental dimensions. We find that Microstructural strategies have a positive and AMM strategies a negative contemporaneous relationship between the activity level of each strategy and realised volatility on the LOB.

Studying the dynamic relationship between the variables using an OLS regression with interaction effects we find that AMMs decrease their net liquidity supply by 3.5% and Momentum strategies increase their net liquidity supply by 3% relative to other traders, following a one standard deviation increase in volatility. These results align with theoretical and empirical research on AMMs. Empirically, AMM-type firms have been found to provide more liquidity in days with stable returns (Anand, 2016) and tend to withdraw from market-making during high volatility periods (Easley, 2011). However, this behaviour of Momentum strategies supplying net liquidity following volatile trading periods does not extend to periods of 'extreme' volatility. All firms tend to decrease net liquidity supply following periods when volatility is extreme, though Momentum and Microstructural strategies decrease net liquidity supply significantly more than the other strategies analysed.

We also analyse the fundamental relationship between algorithmic trading and volatility by performing an experiment comparing *actual* price paths with *alternative* price paths that would exist if a specific trading strategy was not implemented on the LOB, attempting to control for endogeneity. We find that firms conducting AMM strategies on the LOB lead to a 2.95% reduction in volatility, the largest decrease among the strategies analysed, whilst deploying Momentum strategies on the LOB reduces volatility by only 0.19% on average, the lowest among the strategies analysed. One can conclude that each strategy reacts to, and induces, volatility in a unique way, neither of which is seen as particularly harmful to market quality. However, we note that the role of Microstructural strategies in inducing volatility and their withdrawal from the market during periods of stress requires further investigation given that the results indicate they significantly reduce their net liquidity supply following periods of volatility.

Assessing the interrelationship between algorithmic trading strategies and the price discovery process requires consideration of how these traders contribute both information and noise during price formation through their actions. To assess this relationship, we utilise impulse response functions, a vector autoregression Information Share methodology (Hasbrouck, 1995) and Dynamic Linear Models.

We first compute impulse response functions for each strategy to quantify the permanent and temporary price impact induced by each algorithmic trading strategies' activity. Price impact can serve as a proxy for the level of private information infused into these strategies' transactions and LOB actions. The results indicate that Momentum strategies have the largest long-term impact on prices collectively contributing more to price discovery than other strategies, though these firms do represent the largest group of traders by activity. Over a 50-event period the average price impact for the stocks studied is 2.2 bps when firms conducting Momentum strategies execute a trade, with Execution strategies also contributing significant levels of information with an average price impact of 0.96 bps. Short-term temporary price impact from Momentum strategies immediately following a trade innovation are highest at 1.19 bps on average for the stocks analysed. Technical strategies have the smallest temporary price impact of 0.4 bps, however, these strategies represent only 7.9% of aggressive trades.

A vector autoregression (VAR) Information Share methodology, developed by Hasbrouck (1995), coupled with a Choleski variance decomposition technique is employed next to parse price impact from the activity of each algorithmic trading strategy activity into *informational* and *noise* components. The results indicate that Momentum strategies contribute 45.7% of the total efficient price information variance and 41.7% of the noise variance, on average, across the sample securities. However, in relative terms, firms conducting AMM strategies are shown to have the highest information-to-noise (ITR) contribution ratio of 1.20, whilst Momentum strategies have a ratio of 1.10. Microstructural strategies are shown to have the lowest ratio of 0.61. This implies that Momentum and AMMs provide more information than noise in their interactions on the LOB, with the reverse true for Microstructural strategies. This suggests that in aggregate, firms conducting Microstructural strategies infuse significantly less information than noise through their actions on the LOB. In line with expectations, firms conducting Microstructural strategies that utilise an order placement strategy based on microstructural LOB features are not contributing new information to the price discovery process, but rather trading based off micro-period changes in LOB dynamics.

An analysis of the impact of algorithmic trading strategies on the price discovery process is the final experiment performed. We use an event-time infinite lag auto-regressive Dynamic Linear Model (DLM) methodology with a recursive Kalman filter, as an alternative to the VAR methodology. The DLM state space model allows for an estimation of the permanent price impact and transitory pricing error for each algorithmic trading strategy. In respect to permanent price impact, the results indicate that algorithmic trading firms conducting Momentum strategies tend to have the highest impact on the efficient price, with a £10,000 positive order flow surprise predicted by the DLM state space model to have a 0.81 bps permanent price impact. Execution strategies have the second largest impact. One can view these large permanent price impact values as indicative of higher informational content impounded into their order flow, which serves as a proxy for the level of informed trading. In respect to transitory pricing errors, firms executing Technical and AMM strategies are shown to have the lowest transitory coefficients of -2.82 and -2.03, respectively, demonstrating that these firms trade in the direction opposite to transitory pricing errors and contribute the most to price efficiency. Microstructural strategies are found to have a positive transitory component coefficient, which indicates that these firms, on average, trade in the direction of pricing errors, potentially compounding volatility and noise on the LOB.

This chapter has highlighted the relative benefits and detriments of different algorithmic trading strategies for UK equity market quality. Primarily, we note that AMM and Momentum strategies tend to provide a significant portion of liquidity to UK lit order books. Furthermore, firms conducting Momentum strategies tend to increase their net liquidity supply following periods of volatility, whilst AMM strategies are found to reduce realised volatility through their presence on the LOB more than all other strategies. Finally, both strategies are shown to have a positive impact on the price discovery process with significant informational content impounded into their interactions with the LOB. Conversely, firms conducting Microstructural strategies have been found to exit the market by reducing their net liquidity supply, relative to other firms, following periods of induced market volatility. Furthermore, the noise contribution of Microstructural strategies is significantly higher than the informational content of their interactions with the LOB, indicating that they induce more noise than they provide information. This negative effect on price discovery and market quality overall is evident by the result that they trade in the direction of transitory pricing errors, compounding market volatility.

Accelerated adoption of advanced automated trading systems, technology and algorithms by market participants provides exigencies for regulators and academics to understand the dynamics of the contemporary trader ecosystem. Untangling the complexities of the trader ecosystem by extracting the various algorithmic trading strategies being conducted allows regulators to more accurately understand how markets function in their contemporary iteration. Regulators interested in viewing markets through the lens of the algorithmic trading *strategy* being deployed rather than the *type* of firm trading on financial markets can note these results with interest. Implementing this new paradigm of the trader ecosystem provides the tools for policymakers to develop policy prescriptions for modern markets using an empirical and diagnostic approach at the *strategy* level of the market, as exemplified in this chapter.

Chapter Five

5 Predicting Limit Order Book Dynamics using Deep Recurrent Reinforcement Learning

Evolution in financial markets and automated trading systems over the past decade has added increasing complexity to how one approaches the task of understanding, modelling and assessing the dynamics of the limit order book (LOB). Previous chapters have utilised deep learning techniques to model algorithmic trading strategies by solving for the non-linear relationship between a trader's actions or behaviour and a set of LOB features that explain the state of financial markets exposed to the trader. This section extends that analysis by considering how to model the complex non-linear dynamics of LOBs populated with algorithmic traders conducting various strategies.

The approach which we implement in this chapter is to model the dynamics of the LOB using deep learning methods, specifically Recurrent Neural Networks (RNN) and deep Recurrent Reinforcement Learning (RL). The RNN-type models defined in this section are capable of extracting highly abstract feature representations of the LOB state at any point in event-time, which incorporates temporal dependencies from past states intrinsically within the model (Graves, 2005; Cho, 2014). Thus, we perform iterative machine learning of patterns in the LOB by optimising a mapping function that relates the input feature explaining the state of the LOB at t, $\mathbf{x}^t$, to the prediction of how the LOB dynamics will evolve in the future, $\mathbf{y}^t$. Models in this section are trained on a sample of five securities traded during October 2017 which are extracted from the FTSE 100 UK equity data defined in Section 3.3. The objective is to predict the dynamics of the LOB over time. Specifically, three separate experiments are conducted to attain this goal. First, we predict the next trader action on the LOB at or within the best bid-ask price (BBO), which includes limit orders submitted either at or within the BBO to form a new midpoint price, or market orders that transact against opposite side liquidity. Next, the directional evolution of the LOB is studied by fitting a model to predict what the next price change of the midpoint will be using a joint distribution of the best bid-ask (Sirignano, 2016), that is, will the price move up or down. Finally, we use machine intelligence to deduce what the price of a security will be after x events. The unique approach taken in this chapter to modelling sequences of the LOB to predict future price dynamics extends on previous theoretical and empirical research in the deep learning community (Sirignano, 2016; Dixon, 2017).

This chapter focuses purely on the LOB as a multi-class queuing system that solves for the 'search' problem confronting traders seeking to buy or sell a security, as defined in Section 2.2. The previous sections have utilised features of the LOB inherent to each trader, however, we revert from this methodology by building a feature space using a spatial-temporal representation of the aggregated liquidity profile for a specific security extracted from the LOB. One can also view the LOB as a decentralised unitary system, founded on principles ascribed from the microstructural rules of trading venues hosting these LOBs, which amalgamates a disparate group of heterogeneous traders conducting primarily algorithmic trading strategies. The resulting inherent discontinuities, stochasticity and instabilities projected from the LOB into a trader's algorithm requires that trader to consider complex non-linear variable interactions, a high-dimensional action and feature space, and the near-infinite array of intrinsic trader-specific variables. This paradigm of trading makes modelling the various elements of contemporary high-frequency LOBs quite difficult, though of critical importance to regulators, policymakers and practitioners. Training a model capable of capturing these dynamics efficiently and effectively allows regulators and policymakers to understand what phenomena is driving aggregate market behaviour to ensure markets are operating correctly. Policymakers can draw inferences around how the LOB evolves and when combined with the results from the previous chapters, draw potential implications of the interrelationship between the LOB dynamics and the execution of various algorithmic trading strategies on the order book. This is consequential for regulators considering 'algorithm testing and compliance' procedures of paths of regulations, which has been raised recently by the CFTC in the proposed Regulation Automated Trading in 2015. Furthermore, utilising machine learning models as predictors of LOB dynamics can be used by traders conducting algorithmic trading strategies, as explored in previous chapters, to improve their order placement algorithms, adjusting orders in an efficient way that minimises adverse selection costs and maximises trading profits.

Academic research on modelling LOB dynamics has traditionally been dominated by theoretical equilibrium and stochastic models which have largely failed when empirically tested against real-world high-frequency data. Recently, the use of machine learning methods to perform the task of modelling LOB dynamics has been driven by the supply of data-driven optimisation techniques which make it possible to capture abstract non-linear representations in the data (Kercheval, 2015; Sirignano, 2016; Dixon, 2017). The application of such techniques is necessitated by the complex multi-dimensional nature of the market microstructure and trading environment exposed to traders when implementing an optimal algorithmic trading strategy. Further, shifting boundaries in the competitive environment manifesting in the dominance of ultra-low-latency systems (Boehmer, 2016) has driven significant investment in trading architecture and machine learning algorithms when conducting order placement strategies and operations on LOBs.

Deep Recurrent Neural Networks utilised in this section to model LOB dynamics are deep learning models that extend the temporal dimension of feedforward Deep Neural Networks (DNN) by incorporating biological foundations of cognitive memory into the neural network architecture using recurrent feedback connections. RNNs are a connectionist system that allows information to pass across sequential time steps, which allows temporal dependencies in the data to be captured and modelled. The impounding of a 'memory' component into the neural network solves for the implicit limitation in DNN models that assume independence among data samples. Memory is maintained in the network using an explicit recurrent hidden state representation that inculcates information about past states and sequence observations. Thus, we are able to model the sequential time-variant dynamics of the LOB data by learning a hierarchical system across the temporal manifold of past LOB states with significant flexibility and minimal mathematical constraints or assumptions. RNNs also allow contextual information of the LOB features to be incorporated into the model using a spatial-temporal dimension grounded in the RNNs explicit architectural feedback loops. Whilst RNNs are a generalist model with various architectural schematics, we focus predominately on simple Elman RNNs (Elman, 1990), Long-Short Term Memory (Hochreiter, 1997) and Gated Recurrent Unit (Cho, 2014), neural network architectures, with the infusion of an RNN approximator into a Deep Q-Network RL model also utilised.

Modelling LOB dynamics using Recurrent Neural Networks and Recurrent Reinforcement Learning is performed in this chapter to predict how prices evolve on LOBs. We first explore in Section 5.1 the vast academic literature that has analysed the question of how to model the LOB using various equilibrium, stochastic and machine learning models. Section 5.2 introduces the deep RNN as a dynamical system underpinned by neurophysiological conceptions of memory. As explained, these models are capable of modelling long-term temporal dependencies in the LOB data. The unique attributes of RNNs in modelling these dynamics are also explored, along with additional architectural components that improve the predictive capabilities of the models which build upon the DNN theory introduced in Section 3.1. Section 5.3 theoretically explains the deep reinforcement learning techniques employed to improve the RNN predictions of the LOB dynamics. In Section 5.4 we explain the data used for analysis which includes five sample securities traded over the month of October 2017, extracted from the FTSE100 dataset defined in Section 3.3. We first evaluate various design configurations, components and architecture of RNNs in Section 5.5 to develop a robust and efficient optimised model applicable for the FTSE100 dataset, which is defined in Section 5.6. Finally, RNNs with and without auxiliary reinforcement learning connections are then trained on the data to infer predictions with the results presented in Section 5.7. We perform three separate experiments to predict the next trader action at or within the best bid offer price (BBO), predict the next midpoint price change of the LOB, and finally to predict the future midpoint price of a security after x events.

5.1 Literature Review on Modelling Limit Order Books

Academic research focussed on developing analytically tractable microstructural models of limit order book (LOB) dynamics have traditionally adopted either an *equilibrium* or *stochastic* based approach to explain the evolution of the LOB. The debate on whether these approaches provide practical real-world relevance for solving issues such as optimal order placement or algorithmic trading strategies in contemporary high-frequency markets has been ongoing. Recently, *machine learning* approaches to modelling the dynamics of the LOB have arisen, driven by the vast data resources available to researchers and practitioners (Gould, 2013).

Theoretical equilibrium models of LOB dynamics maintain strong assumptions regarding unobservable parameters, such as trader utility, generally within a perfect-rationality framework. The dynamics of the LOB derive naturally as theoretical traders execute optimal order placement strategies given defined market conditions and constrictive modelling assumptions. Stochastic models of LOB dynamics generally populate theoretical markets with zero-intelligence (ZI) traders and allow for parameters to be modelled as stochastic processes and empirically estimated by relaxing certain auxiliary assumptions of the theoretical approaches. Specifically, stochastic models consider the aggregate behaviour of traders. For example, order arrivals, executions and cancellations can be modelled as stochastic processes that occur with probability sample from some distribution, commonly Poisson for order arrival rates (Cont, 2010). Thus, it becomes possible to empirically assess the validity and robustness of the dynamical model by testing against real-world high-frequency data. Whilst stochastic models provide a more analytically tractable method of analysing the dynamics of the LOB, both theoretical equilibrium and stochastic models are difficult to apply in fragmented high-frequency markets. This is primarily due to the complexities of algorithmic trading strategies and non-linear interrelationships and auto-correlations between features of the LOB, that simple models fail to encapsulate.

Within this context, data-driven machine learning models of LOB dynamics have been designed with minimal restrictive assumptions allowing the problem to be solved by developing complex abstract feature representation spaces that encapsulate the current state of the LOB, as opposed to those derived from theoretical models. Optimisation algorithms are applied to minimise some objective function w.r.t learnable parameters, generally using a gradient-descent based approach, culminating in a model capable of explaining real-world phenomena based on empirical evidence rather than theoretical principles that may result in a nebulous conception of the LOB not aligned with empirical facts. Machine learning models do not represent a panacea for solving financial modelling tasks with high degrees of complexity but rather serve as a structured method for modelling phenomena that require extraction of highly abstract patterns in multi-dimensional data not amenable to current statistics-based approaches.

In this section, we assess existing dynamic equilibrium, stochastic and machine learning models of LOB dynamics. Understanding the theoretical and practical implications of LOB models is rudimentary in our ability to more accurately formulate our own model of the LOB using deep learning methods.

5.1.1 Dynamic Equilibrium Models

Equilibrium-based approaches to modelling LOB dynamics were an initial attempt at building stylised theoretical models of trader behaviour that solved for the shape of liquidity on the LOB in equilibrium, subject to several confined auxiliary assumptions including perfect rationality, to ensure analytical tractability (Parlour, 1998; Foucault, 2005; Rosu, 2009). The practical utility from applying these theoretical models in complex real-world environments is minimal given the constraints imposed on interpretability by the unrealistic assumptions. Dynamic equilibrium models are a theoretical-based approach that present LOB dynamics as arising naturally from the optimisation of discrete choice optimal order placement trading strategies for individual traders. Strategy optimisation is performed by modelling the LOB as a dynamic risk-neutral multi-period multi-trader sequential game. The nature of sequential games dictates that traders arriving at the LOB do so in a strategic manner that incorporates into their decision-making process information from the actions of previous traders.

Foucault (1999) solves for the optimal order placement strategy by developing a game theory-based dynamic model of the LOB as a multi-step game where traders arrive sequentially and have a binary decision of whether to place a limit order or execute a market order against a resting limit order. If the order is unexecuted it otherwise expires, with games played through a continuous process until termination. The incoming trader uses an order placement cut-off strategy where the decision to submit either a market or limit order is split by a specific threshold value that is a function of the traders' private fundamental valuation of the asset and the current resting limit order price. Foucault (1999) finds that the primordial determinant of the trading decision is the asset's level of price volatility, with lower volatility reducing the probability of being picked-off by an informed trader and favouring the use of market orders given the lower order execution probability of limit orders. LOB dynamics evolve iteratively as traders enter and exit the market. Despite the tractability of the model, its ability to provide real-world interpretations for order flow and LOB dynamics is hindered by the constraints of the models' assumptions, particularly the expiration of the order after a single session and the random nature of the order arrival times.

The assumption of expiring orders is relaxed in Parlour (1998) which also models the LOB as a dynamic multi-step sequential game with similar binary order decisions but restricts the price-grid state space so that traders can only place limit orders at a single tick from the current price and orders are not able to be cancelled. Similar to Foucault (1999), the order placement decisions require a consideration of fill rates which endogenously depend on the state of the LOB, including the price and size of the current market depth, and the impact of the order on the incentives of future traders. Optimal trading strategies are also based on a cut-off strategy, though with consideration of both sides of the LOB, with the consequential cut-off decision being whether the estimated fill probability meets a specific threshold. The model predicts that when fill rates exceed this threshold then traders should submit a limit order, otherwise below the threshold indicates that the submission of a market order is more agreeable.

Dynamic equilibrium models of LOB dynamics and optimal endogenous order placement strategies were extended by Foucault (2005) utilises a multi-price grid to model the behaviours of strategic liquidity traders with asymmetric information and differing levels of patience. Several strong assumptions are built into the model, including sequential arrival, homogenous orders, order placement constrained to order submissions of limit orders at the BBO and market orders. A trader's decision of what order to place is modelled as a function of liquidity costs for immediacy versus delayed execution. Deriving a function and solving for the profit maximising order placement strategy leads the authors to find that the primordial determinants of the LOB dynamics are the order arrival rates and the proportion of traders that are patient across the market. They also show that a higher tick size has the effect of traders submitting relatively aggressive orders, which leads to tighter spreads and increased resiliency of markets to replenish liquidity at the best bid-ask spread. According to Foucault (2005), the order placement strategy involves a j-limit order, which creates a spread of $j \in \{0, \ldots, s-1\}$ ticks, for every possible spread $s \in \{1, \ldots, K\}$, and given tick size Δ. The optimal order placement trading strategy for trader i, $O_i(.)$, maximises the trader's profit determined by the traders' specific valuation of the asset, $\pi_i(j)$. The profit is positive if the price improvement, $j\Delta$, exceeds the waiting time cost function, $\delta_i T(j)$, which is composed of the time expected to execute the j-limit order, $T(j)$, and a trader-specific summation of the explicit and implicit market impact costs of executing a trade in monetary terms, δ_i, such that: $\max_{j\in\{0,\ldots s-1\}} \pi_i(j) \equiv j\Delta - \delta_i T(j)$. Solving the function for the optimal trading strategy requires an estimation of the endogenous expected execution waiting time which is a non-decreasing function in j such that traders are only able to attain a better execution price, a higher j, as the expected waiting times increase. The minimum spread that trader i is willing to place limit orders, referred to as the 'reservation spread' j_i^R, occurs at the point when the profit function, $\pi_i(j)$, is greater than zero. This reservation spread for trader i is a ceiling function of the traders total trading costs, δ_i, and the rate of executing orders per unit of time, μ, such that $j_i^R = ceiling[\delta_i/\mu\Delta]$. The model assumes that all other execution parameters are determined, that all orders are the same volume and that traders are heterogeneous with differing levels of patience and transaction costs. When a 'patient' traders' reservation spread, j_i^R, is less than the markets inside BBO spread, S, such that $j_i^R < S$, then the trader will avoid crossing the spread by placing a passive limit order at their reservation spread. However, when the inside BBO spread is less than the reservation spread, such that $j_i^R > S$, then the patient trader should place an aggressive market order that crosses the spread in order to avoid costs arising from the risk of not executing at the current price.

The assumption that arrival rates are time-invariant, that all orders are the same volume and that all orders that arrive on the order book lead to a spread reduction are impractical in real-life LOB's. However, the model forms a framework of how LOB dynamics are modelled using dynamic equilibrium models.

Dynamic equilibrium-based models of LOB dynamics where a trader has a binary choice between submitting a limit or market order have been extended in the case of asymmetric information where 'informed' traders with knowledge regarding the fundamental asset value are differentiated from those that are 'uninformed' (Kyle, 1985; Rosu, 2009). Several studies have also relaxed the assumption of endogenous orders but maintained the concept of continuous time and continuous pricing assumptions.

Goettler (2004) builds on Parlour's (1998) LOB theoretical work by considering in their model the continuous nature of the LOB as a Bayesian game with asymmetric information where traders randomly arrive at the LOB based on a Poisson stochastic sequential process. When traders arrive at the LOB they decide whether to purchase information regarding the common fundamental value of the asset to supplement their private valuation such that the trader becomes informed. Then they decide whether to place an order on one of multiple price points, then leave the market, returning once again based on an independent random Poisson stochastic process to either modify or cancel their active orders. The authors find that speculators, traders focusing purely on profit, are incentivised to purchase information and submit certain orders depending on the level of price volatility. Periods of lower fundamental price volatility favour a strategy of not buying information and submitting limit orders which increases the provision of liquidity on the LOB. However, Goettler's (2005) advance into real-world LOB dynamics comes at the cost of a lack of theoretical tractability of the model.

Rosu (2009) relaxes the constraint on strategic behaviour by traders in Foucault (2005) through a continuous-time dynamic perfect-rationality model of numerous anonymous and symmetrically informed liquidity traders arriving sequentially at the LOB. These traders are able to act strategically, with all actions available, by not just submitting limit or market orders, but also modifying and cancelling active orders, which allows for contemporaneous competition between traders in real-time. Rosu (2009) and Foucault (2005) both find that in equilibrium patient traders submit limit orders and impatient traders submit market orders, with implicit costs from the loss of utility waiting to execute an order a key determinant of the optimal trading strategy of whether to submit a limit or market order and the aggressiveness of the limit order.

Minimising market impact is a primary consideration in many theoretical perfect-rationality dynamic models of the LOB and optimal order placement strategies (Bouchaud, 2009; Gould, 2013). Bertsimas (1998) develop a random-walk model of the LOB prices, which is unrealistic in practice, to derive the optimal order placement strategy that minimises both explicit trading costs and implicit market impact costs. They show that under the assumption that exogenous information impacts the LOB prices, then the optimal strategy is to adjust trading quantities at each discrete-time period. Obizhaeva (2013) and Alfonsi (2010) extend the Bertsimas (1998) execution problem with a continuous-time model of the LOB. Alfonsi (2010) uses constraining assumptions and finds that the optimal strategy to minimise market impact costs involves submitting a large initial market order to stimulate liquidity, followed by small limit orders of the same size, and finally a large market order to complete the parent order. Further discussion on market impact models are presented in Section 2.3.5.2.

5.1.2 Stochastic Models

The general ambiguity and lack of real-world applications emanating from equilibrium-based models of LOB dynamics have led to contemporary market practitioners modelling the state dynamics of electronic LOB's using stochastic or machine learning approaches, which have greatly expanded our cognisance of the mathematical intricacies of LOBs.

Stochastic models of LOB dynamics model the statistical mechanics of the LOB using either discrete-time or continuous-time stochastic models driven by zero-intelligence (ZI) traders whose order flow is governed by random stochastic processes. This approach differs from equilibrium-based approaches that utilise some specific utility maximisation function to model LOB dynamics. In stochastic approaches, complex optimal trading strategies utilise real-time constructions of the LOB and stochastic models to derive specific information and predictions of certain events occurring based on the probability of that event, such as a shift in price when a queue at the best bid-ask is depleted, given the value of exogenous variables such as the current state of the LOB. The simplicity of stochastic LOB models' assumptions and constraints, whilst inconsistent with real-world situations, allows for the models' statistical outputs and predictive power to be quantified, evaluated, and tested empirically against real-world data, without the need for complex simulations, in the absence of auxiliary assumptions (Gould, 2013). Stochastic approaches are not generally trader-centric in that the complex strategic interactions between traders within a game-theoretical environment is not explicitly incorporated into the model as is the case with the equilibrium -based dynamic equilibrium models (Foucault, 2005; Rosu, 2009), but rather order flows are represented as stochastic random processes with statistical techniques used to analyse the dynamics of the LOB (Cont, 2010).

Several discrete-time stochastic models have sought to explain the dynamics of the LOB using discrete transitions of the LOB state rather than the analogue of continuous-time models which are more common and model the LOB using infinitely divisible parameters. Guo (2017) uses a correlated random walk discrete-time model of the LOB price dynamics that is similar to Brownian motion models but with mean-reversion properties, to solve for the optimal order placement strategy for one side of the LOB. The single-sided model of the LOB assumes that the spread is always one tick and prices only ever move one tick at a time. The model undergoes dimension reduction based on the correlated random walks generalised reflection principle. The best ask price at time t, $A_t = \sum_{i=1}^{t} X_i$, is assumed to follow a correlated random walk using parameter, X_i, as a Markov chain on $\{\pm 1\}$ with initial probability, $p = \mathbb{P}(X_1 = 1)$, for $p \in [0,1]$. The initial probability of the price dynamics, p, is a measure of market momentum or order imbalance at the LOB inside spread, and the probability of a limit order being executed is given by q. Choosing $p < \frac{1}{2}$ makes the model mean-reverting, which has been empirically proven using high-frequency datasets (Cont, 2013). The optimal order placement strategy solves the Markov decision problem with structure (A_t, X_t) where the cost of each action over each time step is solved recursively. The set of actions available to the trader at t, $A = \{Act^N, Act^L, Act^M\}$, have corresponding costs at $t = 1$ of $cost^N, cost^L, cost^M$, for no, limit, and market orders, respectively. Both $cost^L$ and $cost^N$ are linear functions relative to p, with $cost^M$ constant. The value function for purchasing a unit of shares by the end of the execution horizon at T, $V_t(X_t, a^t)$, is a function of the policy taken at t, a^t, such that the optimal policy, a^{t^*}, satisfies for any policy a^t, so that $V_t(X_t, a^{t^*}) \leq V_t(X_t, a^t)$.

The dynamic multi-stage model of Guo (2017) solves for the optimal trading strategy of whether to use a limit, market or no order based on two explicit threshold levels - at the intersection of the limit and market cost expression and the intersection of the limit and no order cost expression. The expressions and the intersections between them are a function of three variables - the mean-reversion scalar factor, the remaining trading time for the strategy and the level of market momentum. General conclusions of the model include that the optimal trading strategy increases in its level of conservativeness as the trading time decreases, limit orders are more commonly used when their transaction costs are lower, market orders are more common when the level of mean reversion is lower, and in general, the strategy is very sensitive to the estimates of the LOB model parameters.

Continues-time stochastic models of the LOB dynamics use stochastic elements of the trading process that evolves in continuous time within a ZI framework. The assumption of ZI traders and stochastic elements of the trading process provide market participants with analytical expressions of relevant real-time features of the LOB, that can then be estimated using historical data, and which have become exigent requirements of many low-latency high-frequency algorithmic trading order placement trading strategies, thus their relevance in contemporary financial markets. These models utilise queuing systems where the price and position of an order, the depth of the queue at a particular price, and the predicted waiting times within the queue, are key features used to explain the dynamics of the price-time priority matching LOB's at the microstructural level. Defining these features within the model are critical to facilitating optimal trading strategies with lower waiting times and higher execution probabilities.

Cont (2010) models the high-frequency dynamics of the LOB as a continuous-time multi-class queuing system that follows stochastic Markov processes to predict various transitional conditional probabilities of events of interest or changes in the LOB based on the current state of the LOB. Previous models that applied stochastic continuous-time models of the LOB utilised unconditional or steady-state distributions of LOB features as opposed to conditional probabilities, and also required the assumption that the relative prices of limit orders were uniformly distributed rather than based on power-law distributions (Bouchard, 2009). These continuous-time models built on more simplistic discrete-time models where traders reach the LOB in discrete time-steps (Maslov, 2000). The analytically tractable stochastic queuing model of the LOB as a continuous-time Markov chain, developed by Cont (2010), defines the LOB using an n-dimensional state space, encompassing all possible price points, and transition rate dynamics. The LOB evolves as order flow events occur - market and limit orders are submitted and limit orders are cancelled - at different rates based on a power-law depending on the order prices proximity to the bid-ask spread, an empirical feature of LOB's determined by Bouchaud (2002). These mutually independent order flow events are modelled as independent Poisson processes, though some studies have replaced the use of the Poisson process exponential distributions by using a Hawkes process to model arrival rate functions that place greater emphasis on recently arrived orders, which leads to order clusters more closely aligned with empirical findings (Toke, 2012). Practical real-world LOB's high-frequency time-series datasets can be used to estimate the parameters of the model, such as the average size and arrival rate functions of limit and market orders, and cancellation rate functions. This information from the LOB in conjunction with the LOB model can be used to predict the conditional probabilities of certain events occurring based on a given state of the LOB, such as the probability of the mid-price increasing versus decreasing based on the shape of the LOB. This process is conducted through either simulations or using conditional Laplace transformations of specific random variable, X, based on a certain event, A. Several of the hypothesis formulated by Cont (2010) fail when exposed to real-world high-frequency data, indicating that a data-driven machine learning model may be more applicable when modelling LOB dynamics.

Blanchet (2013) construct a continuous-time model of the full LOB dynamics as a multi-class queuing system with specific stylised features relevant in low-latency markets, such as, power-law tails, high cancellation rates, and fast order rates. Parameters are estimated empirically, such as the long-run distribution of LOB spreads and arrival rates. The authors model the distribution of orders on the LOB with power-law decaying tails, such that there is a functional relative correlation between orders close to the BBO and those further out. Furthermore, the cancellation rate for orders closer to the book are higher than those further out. Conflating these two components of the model leads to the postulation that in equilibrium the probability that an order is executed before it is cancelled is the same at all price points in the LOB at a single point in time. They use a two-dimensional Markov process model to draw a connection between the power-law tails of the LOB distribution, cancellation policies, and the asymptotic LOB regime, with the power-law tails from the distribution of returns. Their model can be calibrated using high-frequency datasets in order to estimate both price return distributions and order distributions within the LOB.

Cont (2013) builds a simplified one-dimensional reduced-form version of the Cont (2010) model by proposing a stochastic Markovian queuing model where the price dynamics of the LOB, including the arrivals of limit and market orders and cancellations, depend only on the best bid and ask queues and is driven primarily by bid-ask order flow imbalances. The reduced-form model, in smaller state space $\delta\mathbb{Z} \times \mathbb{N}^2$, aggregates all visible order flow into a consolidated LOB which consists of two interacting queues at the best bid price, P_t^B, with bid depth, V_t^B, and the best ask price, P_t^A, with ask depth, V_t^A, at time t, and with tick size, δ. All other orders outside the best bid-ask depicted as 'reservoirs' with distributions a function of the size of the best bid and ask queues. LOB events include the usual limit, market or cancellation orders. The prices of the best bid and ask change when a queue is depleted, with the new queue depth volume, which is not kept in memory, based on a stationary variable drawn from a certain distribution, f. The model also incorporates the arrival rates of new limit orders, λ, and the rate of limit order cancellations or market orders, $\mu + \theta$, which are driven by Poisson processes and can all be estimated from high-frequency datasets. Cont's (2013) Markovian model develops, in an endogenous manner, analytical expressions for relevant parameters of interest conditional on the state of the LOB, using only the arrival rate parameters and the order queue depth function f. These analytical expressions of LOB market properties include the probability of price changes, the distribution of time durations between price changes, price volatility, and the distribution and auto-correlations of price movements.

Gao (2018) extends the work on microscopic LOB models of Cont (2010) by modelling the stochastic LOB dynamics as a continuous-time Markov-chain (CTMC) with the shape of the LOB explained by a state space of finite series of price points with attached depth volumes. Cont (2010) effectively derives analytical expressions of the LOB by utilising Laplace transformations to measure steady-state conditional probabilities of events in CTMC. An interesting concept is to utilise transient probability distributions, by which the distribution of a system depends on time t, even though this distribution is likely to approach a steady-state distribution based on its initial stationary distribution as t approaches infinity. However, computing transient distributions for multi-dimensional CTMC's, given the potentially large n-dimensional state space, especially for small tick sizes, is computationally expensive. Thus, Gao's (2018) dynamical model of the transient behaviour for the full LOB utilises the theory of hydrodynamic limits to approximate Cont's (2010) high-dimensional CTMC model, in order to understand the transient behaviour of the CTMC and approximate the transient distribution of the LOB.

Guo (2015) analyses the dynamics of the LOB queuing system which is impacted by the series of market and limit order submission and cancellations placed with the trading exchange. They focus on how traders' queue positions impact the probability of the order being executed, which in previous studies was assumed constant (Cont, 2013), or as dependent on order arrival rates based on a homogeneous Poisson stochastic process (Cont, 2010). Guo (2015) models the queues under two-dimensional Brownian motion with the mean and covariance based on general technical conditions of ergodicity and stationarity of order arrival rates and sizes. They allow for flexibility in the assumptions involving order-arrival processes which can be modelled as Poisson or Hawkes processes, common to many LOB models (Large, 2007; Abergel, 2015), or other stochastic processes. A key component of the model is that once a queue is depleted, the new queue is randomly initialised using the stochastic process explained above, thus, managing a representation of the whole LOB is not required. Fluctuations in order positions follow a mean-reverting Gaussian process at a speed proportional to order submission and cancellation rates, where the mean-reverting level is the approximation of the evolution of the stochastic process, or the fluid limit, relative to the queue length modified by the intensity of limit, market and cancellation orders.

Maglaras (2015) models the financial market as a stochastic network that evolves a decentralised multi-class parallel queuing system, represented as individual trading exchanges, in order to solve for optimal routing and trading strategies and LOB dynamics in fragmented markets where traders are able to access multiple exchange LOB's. Traders are modelled as self-interested atomistic heterogeneous agents driven to place limit or market orders on a particular queue, in this case an exchange, based on its execution metrics, such as fill rates and probability of execution, conditions, such as best bid-ask prices and market depths, and rules of the exchange, such as liquidity rebates or trading fees. Trading firms are heterogeneous agents with their optimal trading and routing strategy dependent on their interpretation of the real-time information regarding the state of the market, including the execution metrics, conditions and rules of each exchange, in addition to their heterogeneous individual strategic parameters, including their level of patience, risk of execution delay, and fundamental price. Maglaras (2015) find that the equilibrium for the whole decentralised market meets the so-called multiplicative state space collapse property (Bramson, 1998), where the length of the queue incentivises new orders joining and attracting market orders to execute against current limit orders in the queue. Hence, it is the anticipated delay of waiting in the queue, related to the length of the depth, which motivates the optimal order placement decision of whether to submit limit orders or market orders and drives the dynamics of the LOB.

Chen (2017) considered the role of hidden orders in LOB dynamics building a dynamic programming multi-stage model of the LOB dynamics to optimise the limit order placement decision under certain market conditions. The model builds convexity assumptions for its parameters in certain situations to produce a closed-form analytical solution for the optimal trading strategy for risk-neutral traders seeking to trade a target volume. The optimal solution minimises execution costs by balancing the trade-off between lower execution probabilities from hidden limit orders and higher signalling risk from visible limit orders.

5.1.3 Machine-Learning Models

Research into the application of machine learning methods to model LOB dynamics has arisen in recent years. One could posit that the increased prevalence of the methods in the literature to solve the task of modelling the LOBs has been driven by the vast data available for consumption regarding events on cross-market LOBs and the wider financial market. Another potential reason is the general misalignment between the statistical assumptions of certain parameters in stochastic models of the LOB and the empirical evidence from high-frequency markets that compute divergent distributions from that expected by the models. Stochastic models are particularly difficult to apply in high-frequency markets where disparate algorithmic trading strategies can influence LOB distributions in contradictory manners for different reasons. Furthermore, the strong assumptions and prohibitive limitations of stochastic models detract from the application of these models by practitioners require them to be amenable to changing situations and able to run in an online environment. Thus, data-driven deep learning and machine learning models capable of untangling complex features by building abstract representations of the LOB are most pertinent for modelling the LOB dynamics in contemporary financial markets.

Modelling the evolution of the LOB by predicting future price movements using Recurrent Neural Networks and Reinforcement Learning is a key theme of this thesis. Machine learning models trained to predict dynamics of the LOB have been applied through the use of Support Vector Machines (Kercheval, 2015), Deep Neural Networks (Sirignano, 2016) and Recurrent Neural Networks (Dixon, 2017).

Kercheval (2015) develop a multi-class SVM that utilises a feature representation space composed of price and volume-based metrics of the LOB to predict how midpoint prices of the LOB will evolve given the state of the LOB. They use class labels of up, down and stationary for each vector of the LOB state as it evolves in real-time. The technique utilised to separate features into basic, time-insensitive and time-sensitive categories is interesting and merits consideration for integration into this thesis. Several time-sensitive features, which includes order acceleration, relative intensity measures and price derivatives w.r.t time, are incorporated into the feature set used to train RNNs in this chapter based on an event-time conception. Furthermore, feature selection using information gain criteria, whilst not commonly employed in neural networks given the embedded nature of the feature selection performed by the network, is tested for relevance in our setting.

Sirignano (2016) builds a predictive framework for how prices of the LOB will evolve using a ‘spatial’ DNN architecture that models the joint distribution of the bid and ask in future periods conditional on the current LOB distribution. The spatial DNN considers the whole local structure distribution of orders deep into the LOB, over 50 price points from the BBO, rather than truncating the LOB to within x ticks of the BBO as is the case in this chapter. For DNNs designed with ReLU activations, the author proves that the spatial DNN is well-posed with a positive mass as price points approach infinity. The study finds that the DNN outperforms non-linear models such as logistic regression and other naïve empirical models including SVMs. This thesis diverges from the spatial DNN employed in Sirignano (2016), which focussed on risk modelling that required the retention of information deep in the order book, by considering only the distribution of the LOB around the BBO prices. The truncated LOB state is developed given that our focus is on LOB dynamics and algorithmic trading over micro time-periods where the BBO and price queues nearby are most pertinent for the analysis performed.

RNNs have been previously utilised in the literature to model LOB dynamics (Dixon, 2017). Dixon (2017) models the LOB dynamics for E-Mini S&P500 futures as a sequence machine learning classification task solved by training an RNN to predict whether the midpoint of a security will move up, down or stay neutral, at any given point in time. The model is trained using LOB data for E-Mini S&P500 futures traded on the CME over the month of August 2016. RNNs impute features related to the LOB market depth and order flow for both sides of the LOB. The architecture employed in the study is a basic Elman RNN given that the authors find minimal benefit of using LSTMs to extract longer-term temporal dependencies. Dixon (2017) finds that predicting neutral price movements is significantly easier than determining when price changes will be up or down, which attained average F1 scores of 16.1% and 16.5, respectively. One issue with the methodology applied is the significant class imbalance problem of the classifier, with over 99% of samples classed as having a 'neutral' price change, which requires significant data augmentation and 'balancing' procedures that oversample these neutral price movements and can create instabilities and discontinuities in the recurrence of the neural network. This thesis extends the Dixon (2017) model to apply to equity markets using more expressive architectures capable of finding patterns along temporal dimensions, a more complex and abstract feature representation space, a differently defined task that predicts price changes and the *next sign* of a price change on the LOB in addition to performing other prediction tasks.

5.2 Deep Recurrent Neural Networks

Deep Recurrent Neural Networks (RNN) are dynamic deep learning models that extend the temporal dimension of feedforward Deep Neural Networks (DNN) by incorporating biological foundations of cognitive memory into the neural network architecture using recurrent feedback connections. RNNs are a connectionist system where information passes across sequential time steps, allowing temporal dependencies in the data to be captured and modelled. The impounding of a 'memory' component into the neural network solves for the implicit limitation in DNN models that assume independence among data samples. Thus, the sequential nature of limit order book (LOB) time series data is ostensibly suited to RNN models which are capable of learning a hierarchical system across the temporal manifold. In particular, modelling the dynamics of the LOB using RNN machine learning methods allows for significant flexibility with minimal mathematical constraints or assumptions, making these models capable of capturing the time-variant dynamics of the LOB system (Schafer, 2007).

The inherent flexible nature of RNNs derives from their ability to incorporate contextual information using a spatial-temporal dimension grounded in the models' explicit architectural feedback loops, allowing for time to be represented recursively. RNNs are a generalist model with various types of networks and architectures employed in the literature, though a common element involves the series of operations and transformations applied to a memory component. Memory is maintained in the network using an explicit recurrent distributed hidden state representation, $\mathbf{h}$, that impounds information about past states and sequence observations. Integrating past information into the current time context encapsulates relevant historical information, allowing temporal dependencies between data in various time epochs to be captured by the RNN. The hidden state memory, $\mathbf{h}$, is iteratively updated based on new information received in the form of an input feature vector in the current period t. In this way, RNNs are referred to as dynamical systems where information regarding the whole sequence is maintained in memory, allowing long-term dependencies to inform network predictions of an output value at time t.

In a similar vein to DNNs, recurrent networks employ optimisation algorithms to approximate the non-linear mapping function, $\mathbf{y} = f(\mathbf{x})$, that relates a set of sequential observations $\mathbf{x} = \{\mathbf{x}^1, \dots, \mathbf{x}^T\}$, with elemental vectors comprising features that explain the current observable phenomena within the system, to its corresponding class label, $\mathbf{y} = \{\mathbf{y}^1, \dots, \mathbf{y}^t\}$. As discussed, RNNs relax the assumption inherent in feed-forward DNNs that input and output vectors are independent of each other, by developing an internal state composed of sequences of inputs which are dynamically updated depending on the internal memory, $\mathbf{h}$, developed through previous computations of the sequence. The highly expressive nature of RNNs extend from the integration of this recursive internal memory component into its architecture and is evident by their ability to compute any representable function under certain assumptions (Siegelmann, 1995). These assumptions include full network connectivity, a finite number of recurrent connections, and sigmoidal activated neurons. In this way, RNN systems are Turing complete, capable of simulating any arbitrary Turing machine for a given input.

The next sections introduce the neurological framework that motivates the RNN memory component before explaining the conception of RNNs as a dynamical system and the importance of capturing temporal dependencies to build expressive memory representations. Building from this section, expressive and high capacity deep RNN architectures are theoretically defined with a specific focus on Elman RNNs (Elman, 1990), Long-Short Term Memory (Hochreiter, 1997) and Gated Recurrent Units (Cho, 2014). Finally, we introduce new concepts specific to RNN architectures that build upon the theoretical derivations from Section 3.1, with a particular focus on layer normalisation (Ba, 2016), recurrent dropout (Gal, 2016) and scaled exponential linear units (Klambauer, 2017).

5.2.1 Neurophysiological Conceptions of Memory

Recurrent connections along temporal dimensions inform the RNNs internal memory at $t, \mathbf{h}^t$, by providing a conduit for information pertaining to historical states to enter the network and inform the current context. Modelling machine intelligence memory components in RNN models draws significantly from neuroscience research, with specific roots in the quasi-analogous role of *working memory* in human neurology. Congruent with the theoretical foundations of hidden RNN states, working memory is a component of a human's cognitive system that performs operations and manipulations to new information entering the system. This new 'stimuli' motivates decision-making on what and how to execute an action, before storing the memory for future retention when required to perform actions relevant to that memory.

Memory is universally limited by capacity constraints. Several theories have been proposed to explain the capacity of memory to transiently maintain temporal dependency representations of history that provide contextual information for forthcoming decisions and actions.

An iterated multicomponent model of working memory (Baddeley, 2008) places at its radial point the 'central executive' which serves as a rule-based arbiter of how sensory memory propagates into both long-term memory systems and prioritises short-term decision making. Working memory regulates how short-term information is captured. The key concepts in this theory are selection and priority, that is, the working memory selects what stimuli need to be addressed, for example, what phenomena need to be impounded into memory, and the magnitude of response. Three additional components of working memory connect to both short and long-term memory. This includes the phonological store, which provides memory maintenance for speech-based information, the visuospatial sketchpad, which provides memory for spatial contexts (Baddeley, 2008), and an episodic buffer that acts as a flexible recurrent connection between long and short-term memory, in which operations are performed involving how information flows through the system.

Limited resource models (Palmer, 1990) do not apply quantifiable boundaries on memory capacity but rather model the qualitative capacity of memory as a system with limited resources to capture interesting phenomena, for example images or numbers, measured as a function of precision and quality. The theory posits that memory is a finite resource that is able to be shared over multiple representations which can be accessed concurrently. The quality of memory depends significantly on the resources allocated and level of noise inherent in certain internal representations of past stimuli that are coded into memory (Ma, 2014). Noise is a function of both explicit random corruptions in a perception of the stimuli being memorised and the number of stimuli imputed into memory, such that noise increases when the number of stimuli in memory reaches some threshold, reducing the quality of working memory learning new representations.

Flexible resource models (Bays, 2008) hypothesise diversity in memory resource capacity with non-critical information rescinded from the representational manifold. Recall precision is modelled on the assumption that a voluntary control component modulates the allocation of memory resources depending on the importance of the stimuli. Empirical studies have found evidence towards the hypothesis that memory resources are not distributed evenly but rather are regulated by the level of priority given to different items (Gorgoraptis, 2011). Decay is a pervasive concept in theoretical models of working memory, including resource models. Barrouillet (2004) introduced a resource-sharing model of working memory where decay occurs when memory representations are not maintained or retrieved over time. The quality of working memory regarding a representation decays when attention is switched to an alternative task which frays the original memory depending on the ‘cognitive load’ placed on the resource system by the new task.

Theoretical research into human cognition and working memory has both motivated and enhanced the design, architecture and configuration of RNNs trained to solve real-world problems. The task of modelling LOB dynamics is analogous to that of a human trader in the past that sought to predict how markets would evolve so as to inform their optimal limit order placement strategy. One can posit that the transfer of decision-making ‘central executive’ from human to algorithm is a manifestation of the pervading algorithmic trading dominated markets as explored in previous chapters of this thesis. Algorithmic traders have largely replaced human traders in the contemporary iteration of financial markets. Thus, algorithms or deep learning techniques such as RNNs are commonly used to solve the optimal order placement decision problem. This nexus between deep learning models of the LOB and algorithmic trading strategies used to solve for optimal order placement is explored throughout the rest of this chapter.

5.2.2 Dynamical Systems

RNNs are a representation of a dynamical system in which the state or memory of the system at discrete-time t, $\mathbf{h}^t$, undergoes an iterative updating procedure when new stimuli, δ, enters the system. For the purpose of this chapter, the sequential series of LOB input feature vectors, $\mathbf{x} = \{\mathbf{x}^1, \ldots, \mathbf{x}^T\}$, represent the set of external signals, δ, which flow into the RNN system. Updating the memory component of the RNN dynamical system is performed over a sequential time series in a recursive manner that utilises dynamic programming. The iterated hidden state, $\mathbf{h}^t$, is updated by applying a non-linear activation function, f, to the previous state, $\mathbf{h}^{t-1}$, and the set of external signals, δ, represented by the input feature vector for that time period, $\mathbf{x}^t$, such that:

$$\mathbf{h}^t = f(\mathbf{h}^{t-1}; \mathbf{x}^t) = f(f(\mathbf{h}^{t-2}; \mathbf{x}^{t-1}); \mathbf{x}^t)$$

Evidently, the current hidden state is defined as a recursion of itself, influenced only by its past value and new information entering the network in the form of input signals. Hidden states in RNNs generally have the architectural trait of being a fixed length vector. Thus, there exist mathematical limitations on the quantum of memory capable of being projected forward from previous states. As datasets iterate with time, $t \to T$, the hidden state expels information due to its fixed nature and potentially long historical time series. RNN architectures deviate in how they maintain this memory state and can be differentiated by the capacity and connectedness of this memory unit within the network. Recent innovations in RNN architectures have addressed the capacity constraints of RNNs to maintain long-term memory by externalising the memory component of the network into a storage bank that maintains very long-term dependencies and information (Graves, 2014).

5.2.3 Temporal Dependencies

In this chapter, an RNN is employed to model the LOB dynamics of UK equity markets, a process analogous to predicting the future state of a financial time series system. Long-term temporal dependencies are a ubiquitous characteristic of financial time series-based systems in that the current state of financial system derives from the historical actions and behaviour of agents within the system. RNNs capture long-term dependencies within the memory component of the network, $\mathbf{h}$, which is a function of the past states of the system. In the context of LOB dynamics, observations of previous states of the LOB, explained by the feature vector $\mathbf{x}$, in conjunction with the transitions between states, can be rationally expected to influence predictions regarding how the state will evolve between event time t and $t + 1$. Thus, highly expressive RNNs implicitly model temporal dependencies within their prediction of future state transitions. When temporal dependencies ameliorate over small intervals, the need for RNN architectures declines and the non-linear mapping function of the input-output relationship may be better modelled using a feedforward DNN.

The high-frequency nature of modern LOBs magnifies the exigencies of modelling the states of financial markets and LOBs in near real-time, with common algorithmic trading strategies employing ultra-low latency systems capable of processing stimuli and deciding on a commensurate response at the millisecond level (Boehmer, 2017). This evolutionary change in market dynamics and algorithmic trading systems has instigated a divagation from chronological conceptions of time in trading, with event-time cyclical clocks dominating the contemporary trading environment (Aldridge, 2013). Thus, whilst the temporal dependencies between LOB states may seem tenuous when considering that multiple trading actions may be executed over a single millisecond, the shift towards event-time clocks in trading has changed the nature of these dependencies to consider how states transition between single or multiple events. The time continuum has been flattened and stretched over a dynamic event space. For example, when modelling LOB dynamics, one would not look at dependencies between quoted spreads over intervals of seconds, but rather over intervals of either single events or a period of lagged observations in event-time. Furthermore, capturing temporal dependencies in high-frequency data is necessary given that patterns may exist in the data over micro-time frames as traders with slow algorithmic trading systems take longer to impute and process new stimuli entering the system.

RNNs withdraw from the assumption inherent in DNNs that input vectors are independent by integrating temporal dependencies within the models' architectures. Traditional methods of modelling temporal connections between non-concurrent events in finance using techniques such as Markov models and moving windows, can more efficiently be performed by a neural network with recurrent connections.

5.2.4 Deep Network Architectures

Deep recurrent neural network architectures control how information propagates through various layers of recurrent neurons that connect input feature vectors at t, $\mathbf{x}^t$, with their predictor output, $\hat{\mathbf{y}}^t$. The dynamical nature of RNNs introduces the concept of the hidden state, $\mathbf{h}^t$, that is recursively determined as a function of the prior transitory state, $\mathbf{h}^{t-1}$, and a set of external signals. The inherent recursive nature of RNN models allows past hidden states to be incorporated within a dynamical system that allows depth along temporal dimensions that is non-existent in standard feedforward DNNs. Depth in the context of this chapter also refers to the number of hidden recurrent *layers* in the network that apply a non-linearity, allowing a larger hierarchical function representation space with increasingly abstract LOB features, in a way that extends basic RNN architectures that have relatively shallow architectures (Salehinejad, 2018). RNNs are parameterised by weight matrices that are shared across temporal dimensions of the network.

Design configurations of the RNN architecture regulate how information propagates through the network in the form of a temporal input feature vector, $\mathbf{x}^t$, hidden states, $\mathbf{h}^{t-1}$, and during the backpropagation of error signals, $\boldsymbol{\delta}$, when learning. For deep networks, all hidden layers $l \in L$ are calculated in a recursive manner beginning at the input vector $\mathbf{x}^t$ that propagates through the network and terminating at the output layer L with a consequent prediction, $\hat{\mathbf{y}}^t$. The objective of robust deep RNN architectures is to formulate accurate temporal predictors, $\hat{\mathbf{y}}^t$, when measured against actual outputs, $\mathbf{y}^t$, within the framework of a cost function, $\mathcal{L}$. Networks consist of a multifarious array of synaptic weight connections between neurons in the model whose design and configuration depends on the RNN architecture deployed. Given their differentiable properties, the learnable parameters of the network are generally optimised using gradient-descent or second-order algorithms.

Constructing networks with deep architectures is predicated on the hypothesis that deeper structures increase the representation space of possible non-linear mapping functions capable of being learnt by a machine learning model (Bengio, 2009), improving the expressiveness and capacity of these models (Hermans, 2013; Graves, 2013; Pascanu, 2013). Deeper neural networks have been shown to further increase feature abstractions, disentangling underlying features critical for formulating better predictors (Glorot, 2011) and simpler data manifolds (Bengio, 2013). Empirical studies grounded in deep learning theory have provided evidence towards the hypothesis that deeper architectures improve the accuracy of RNNs (Pascanu, 2014), Convolutional Neural Networks (Goodfellow, 2013), and feedforward DNNs in general (He, 2015).

Depth can be added to the RNN model by either stacking L recurrent and fully connected hidden layers between the input and output layers, $\mathbf{x}^t \to \hat{\mathbf{y}}^t$, to build more abstract feature representations (Hermans, 2013), or by deepening the non-linear connections between hidden state transitions, $\mathbf{h}^{t-1} \to \mathbf{h}^t$, to enhance long-term dependencies (Pascanu, 2013). The ultimate objective of incorporating additional layers into the network is to improve the accuracy of the predictor, $\hat{\mathbf{y}}^t$. In this chapter, we focus on adding depth between the input and output layers, $\mathbf{x}^t \to \hat{\mathbf{y}}^t$, to allow variations in complex LOB features to be disentangled, providing more predictive power. For robustness, experiments are conducted to ascertain the benefits of adding layer connections between hidden state transitions, $\mathbf{h}^{t-1} \to \mathbf{h}^t$. The results indicate that the process is computationally prohibitive and overly complex without any commensurate improvement in the RNNs predictive ability.

RNN layers are trained iteratively during backpropagation using similar techniques as in feedforward DNN training. However, augmenting RNN structures with deeper layers requires significantly more computational resources due to the series of non-linear operations *between* states in both *temporal* and *spatial* dimensions during forward propagation. This also requires the computation of more complex differentials and memory-intensive storage of recursive signals during the dynamic programming component of backpropagation. The exponential increase in non-linearity operations applied between and across both time-dependent and input-dependent components of the RNN also magnifies the risk of vanishing or exploding gradients (Hochreiter, 1998) common to sigmoidal activated RNNs. The vanishing gradient stops the gradient from flowing backwards during training, stunting learning and slowing convergence (Graves, 2005, Graves 2008). In addition, the model will fail to learn long-term dependencies if gradients between hidden state layers saturate. These gradient issues are largely overcome by selecting a model architecture with an *internal* recursive component existent within each neuron of the network (Hochreiter, 1998). When deepening layers between hidden states, saturation issues are overcome by forming direct connections between previous hidden states and input components of the network (Pascanu, 2013), to restrict how and to what extent the non-linearity is applied. Later sections address these issues.

The next three sections provide an exposition of the RNN architectures employed in this chapter, including the Elman Recurrent Neural Network (Elman, 1990), the Long-Short Term Memory (Hochreiter, 1997), and the Gated Recurrent Unit RNN (Cho, 2014). All RNN architectures employ hidden layer activation functions, $\boldsymbol{\phi}$, to apply non-linearity to neurons in the network and an output activation function, $\boldsymbol{\varphi}$, that provides a probabilistic interpretation of the output value. Both activations must be differentiable over time. Activation functions commonly employed in RNN architectures from the sigmoidal family include hyperbolic TanH, $\phi_{TanH}(z)$, and sigmoid, $\phi_{Sigmoid}(z)$. Rectifier functions including ReLU, $\phi_{ReLU}(z)$, and ELU, $\phi_{ELU}(z)$, are commonly utilised in RNNs to allow more sparsity in neuron activations and gradients. Section 3.1.5 introduces, defines, explains and critiques common activation functions generally used in neural networks. Output activation functions, φ, generally take the form of a softmax to allow for a probabilistic interpretation of the output neurons.

5.2.4.1 Elman Recurrent Neural Network

Conventional deep recurrent networks in the form of Elman RNNs represent a baseline architecture from which more expressive contemporary networks have evolved (Elman, 1990). The Elman RNN is a non-linear dynamical system that takes as input an n-dimensional temporal feature vector at event time t, $\mathbf{x}^t = (x_1, \dots, x_n)$, that represents an element of the dataset sequence $(\mathbf{x}^1, \dots, \mathbf{x}^t, \dots, \mathbf{x}^T)$. Input vectors feed into and through a series of $l \in L$ feedforward fully-connected hidden layers composed of j hidden state neurons, $\mathbf{h}_l^t = (h_{l,1}^t, \dots, h_{l,j}^t)$, representing the dynamic memory component of the system which for simplicity is defined as vectors of the same length j for all layers. These hidden state neurons are connected to previous transitory hidden states, $\mathbf{h}_l^{t-1}$, through recurrent feedback loops over time. States initialise at value $\mathbf{h}^0$ which we set to zero. The objective of the deep RNN is to predict the output at time t, $\hat{\mathbf{y}}^t$, which can be compared to the relevant element in the actual sequence of outputs $(\mathbf{y}^1, \dots, \mathbf{y}^t, \dots, \mathbf{y}^T)$ to compute the cost function $\mathcal{L}$. Elman RNNs have been applied to forecasting LOB dynamics in US futures markets by Dixon (2017), modelling of gene regulatory network (Mandal, 2006) and for predicting missing time series variables by concatenating the RNN process with particle swarm optimisation to derive a hybrid optimisation algorithm (Cai, 2007).

For the special case of the first layer, $l = 1$, the state is composed of the input vector, $\mathbf{x}^t$, which represents an algorithmic trader's contemporaneous view of LOB features. In the case of the first hidden layer, hidden state neurons, $\mathbf{h}_1^t$, are computed by applying an activation function, ϕ_1, to the input activation value, $\mathbf{z}_1^t$, equal to the sum of the affine transformation of the input feature vector, $\mathbf{x}^t$, and learnable synaptic weight matrices connecting input and hidden state neurons, $\mathbf{w}_1$, with the affine transformation of the previous hidden state vector, $\mathbf{h}_1^{t-1}$, and weight matrices connecting the previous and current hidden state, $\mathbf{u}_1$, with a final bias term added, $\mathbf{b}_1$. In a similar vein, hidden state neurons in hidden layer l, $\mathbf{h}_l^t$, equal the output of the activation, ϕ_l, applied to the input activation value, $\mathbf{z}_l^t$. This input activation value is composed of the affine transform of the previous hidden layers hidden state output with the relevant weight matrix, $\mathbf{w}_l\mathbf{h}_{l-1}^t$, plus the previous hidden state vector of the same layer, $\mathbf{u}_l\mathbf{h}_l^{t-1}$, with the added bias term for the layer, $\mathbf{b}_l$. Finally, output predictor vectors at time t in final layer L, $\hat{\mathbf{y}}^t$, utilise the output activation function, φ, which is applied to the input activation value, $\mathbf{z}_L^t$, after taking the affine transformation of the previous layers hidden state, $\mathbf{h}_{L-1}^t$, and weight matrix connecting the previous adjacent and output layer, $\mathbf{w}_L$, plus bias term for the output layer, $\mathbf{b}_L$. In this chapter, softmax output activation functions are used to provide a normalised probabilistic interpretation of the predicted class of the RNN at time t.

The RNN is explained by the set of learnable parameters which can be concatenated into an expression, $\delta = (\mathbf{w}, \mathbf{u}, \mathbf{b})$. These parameters are optimised using gradient-descent methods. A schematic illustration of simple Elman RNNs is provided in Figure 5.2.1, with the relevant mathematical formulations for the input, hidden and output layers given as:

$$\begin{array}{lll}
\mathbf{z}_1^t \leftarrow \mathbf{w}_1\mathbf{x}^t + \mathbf{u}_1\mathbf{h}_1^{t-1} + \mathbf{b}_1 & \mathbf{h}_1^t \leftarrow \phi_1(\mathbf{z}_1^t) & (Input\ Layer) \\
\mathbf{z}_l^t \leftarrow \mathbf{w}_l\mathbf{h}_{l-1}^t + \mathbf{u}_l\mathbf{h}_l^{t-1} + \mathbf{b}_l & \mathbf{h}_l^t \leftarrow \phi_l(\mathbf{z}_l^t) & (Hidden\ Layer) \\
\hat{\mathbf{y}}^t \leftarrow \mathbf{w}_L\mathbf{h}_{L-1}^t + \mathbf{b}_L & \hat{\mathbf{y}}^t \leftarrow \varphi(\mathbf{z}_L^t) & (Output\ Layer)
\end{array}$$

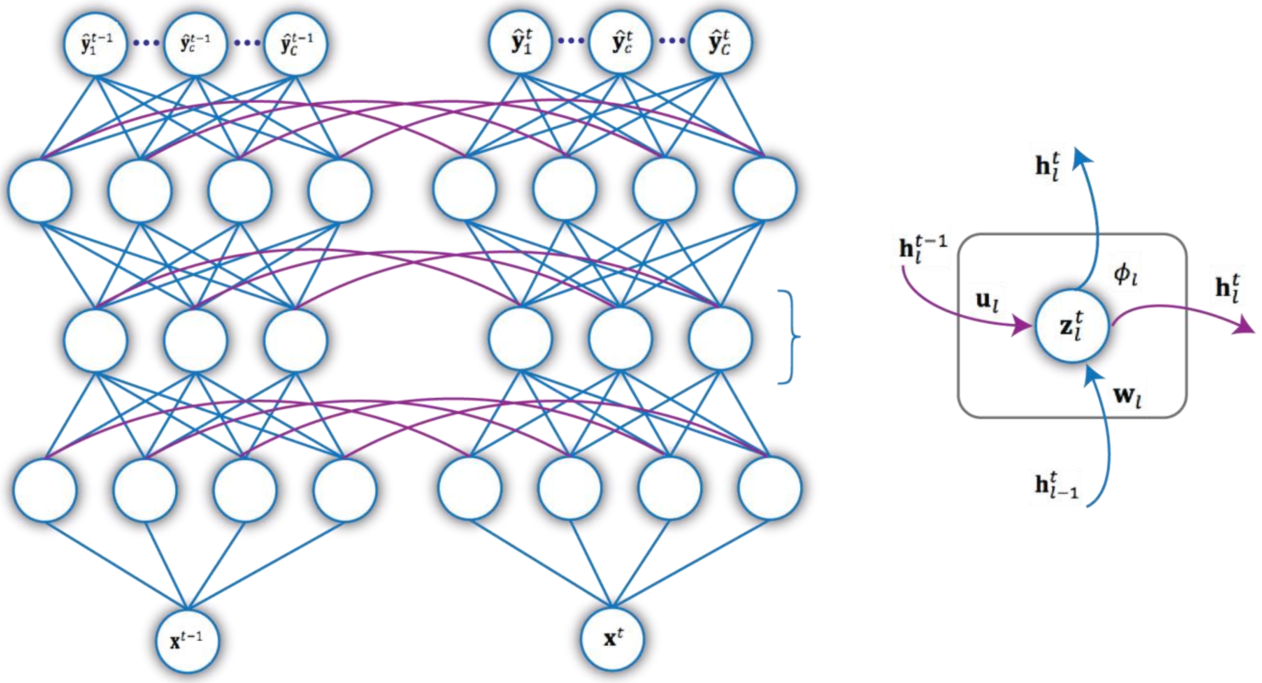

Figure 5.2.1 – Schematic illustration of the Elman RNN architecture for a cell in layer l, $\mathbf{h}_l^t \leftarrow \phi_l(\mathbf{z}_l^t)$, as a function of input from the previous layer in the current time period, $\mathbf{h}_{l-1}^t$, and from the same cell in the previous time period, $\mathbf{h}_l^{t-1}$, parametrised by their respective learnable weights $\mathbf{w}_l$ and $\mathbf{u}_l$, which are optimised during training. Note that the bias term, $\mathbf{b}_l$, is not shown for simplicity.

Conventional Elman RNNs are extremely expressive models but commonly fail during optimisation due to the tendency of gradients from long-term dependencies computed during training to suffer from the ‘vanishing gradient’ problem (Hochreiter, 1991; Bengio, 1994; Hochreiter, 1998). Gradients for sigmoidal-activated RNN data points further in the past from time t tend to have very small gradients caused by the multitude of multiplicative and exponential operations applied to hidden states, $\mathbf{h}$, over time steps to compute error signals during backpropagation when optimising the recurrent weight matrix, $\mathbf{u}$. This results in the magnitude of the gradient shrinking at an exponential rate over t. Hence, RNNs fail to incorporate long-range dependencies in the current time step t given that gradients further back in time become tiny and fail to propagate valuable historical information through the network.

Two primordial approaches to alleviating the vanishing gradient problem have centred on developing more stable optimisation algorithms than SGD and augmenting the non-linear architecture of the RNN to decrease the risk of gradients from past states saturating. Various second-order Hessian-free methods that derive information from the second derivative of the cost function have been deployed as an alternative learning algorithm to SGD (Martens, 2010). When the ratio of the first and second-order gradients is stable, one can draw information about the gradient’s past state from the second-order gradient, from which gradients further in the past can be stably propagated through the network during training (Martens, 2010). Furthermore, gradient descent optimisation algorithms that incorporate Nesterov momentum have also been applied with the objective of solving for vanishing gradients of past states (Sutskever, 2013). The method more commonly applied in the literature to solve the vanishing gradient problem is reparametrising the RNN model using a Long-Short Term Memory (Graves, 2005) or Gated Recurrent Unit (Cho, 2014) architecture rather than the more simplistic Elman RNN model. These models utilise SGD during learning but the inherent nature of the non-linear recurrent connections in their architecture result in past gradients flowing successfully through the network allowing for long-term dependencies to be incorporated into the RNN. Both these models are explored in following sections.

5.2.4.2 Long-Short Term Memory

Long-Short Term Memory (LSTM) RNNs were built in response to the limited capability of gradient descent-based optimisation algorithms to solve for the preponderance of Elman RNNs to suffer from vanishing gradient issues that led to a failure of these models to impound temporal dependencies during training (Hochreiter, 1997; Gers, 2000; Graves, 2005). The original LSTM architecture addressed the issue of vanishing gradients by incorporating an internal recurrent '*memory cell*' within each hidden layer neuron which regulates information flows through the network using soft multiplicative auxiliary '*gated connections'* (Hochreiter, 1997). Future extensions of the RNN included the addition of '*peephole'* connections (Gers, 2000) between the internal memory cell and the neurons gated connections, and the implementation of the backpropagation through time (BPTT) optimisation algorithm to train RNNs (Graves, 2005; Greff, 2017). This chapter utilises the RNN model architecture developed and employed by Graves (2005).

Memory cells, gated and peephole connections are an innovative design element in LSTMs. Memory cells allow the neurons within the network to maintain an iterative state over temporal dimensions, whilst non-linear gated connections control information flow internally within the LSTM. The practical value derived from using peephole connections to link the internal memory cell state to the LSTM gates when being updated, is that the RNN is capable of better measuring timing between different stimuli being received. LSTM networks have been applied to supervised machine learning problems in various domains successfully including text recognition (Graves, 2013), acoustic modelling (Sak, 2014), speech recognition (Graves, 2014) and machine translation (Sutskever, 2014).

LSTMs are employed to model LOB dynamics by incorporating temporal dependencies in contextual high-frequency data from prior states, $\mathbf{h}^{t-1}$, into the current state of the LOB, $\mathbf{h}^{t}$. This improves the model's predictions of how the LOB will evolve given this current state at t. LSTMs perform this function by developing conceptual LSTM blocks or units that are analogous to hidden state neurons in the simple Elman RNN, though with an internal recurrence inside the blocks. LSTM blocks maintain an internal 'memory cell' state, $\mathbf{m}$, an integral atomistic component of the block where information of past states is stored as memory. These LSTM blocks only take in information through connections from adjacent hidden and input layers or from connections to past hidden states, ultimately controlling how information is propagated through the network. Memory cells expand *across* temporal dimensions of the LSTM in the same vein as simple RNNs (Maass, 2007). The primary difference in approach occurs within the structure of the LSTM block which utilises a series of soft multiplicative 'gates' to regulate how data flows *through* the internal structure of the LSTM block and how it is coded into the internal memory cell, $\mathbf{m}_{l}^{t}$, as expressed for layer l at time t.

The architecturally inherent condition of vanishing and exploding gradients common to Elman RNNs is solved by LSTMs through the partial linearisation of connections between past hidden states, $\mathbf{h}^{t-1}$, which flow into LSTM blocks (Hochreiter, 1997). The Elman RNN applied a non-linear activation to the past hidden states that resulted in gradients of states further in the past quickly saturating, with no information regarding those states flowing forward, rescinding the temporal dependencies critical to RNN learning. LSTMs correct this issue by applying a self-recurrent connection to internal memory cells, $\mathbf{m}$, within the LSTM block and setting fixed weights, equal to one, for this connection. Furthermore, the risk of exploding gradients can be minimised by applying a truncation to the partial derivatives of the

RNNs updated hidden state w.r.t its past state, $\partial h^t / \partial h^{t-1}{}_{TR} = 1$. Hence, the gradient of the hidden state, $\mathbf{h}^t$, flows through self-recurrent loops through time along paths of long duration, ameliorating the risk of the gradient exploding or vanishing. Extensions to the architecture using peephole LSTMs employed in this chapter utilise a 'forget' gate (Gers, 2000) which results in the partial derivative, $\partial h^t / \partial h^{t-1} = \mathbf{f}^t$, where the forget gate value $\mathbf{f}^t$ controls how memory dissipates during learning and temporal iterations. When the memory state remains stable, perhaps due to irrelevant new stimuli entering the system, the memory cell *maintains* the gradients' error over this period rather than decaying the memory as is the case in Elman RNNs as gradients reduce with time. This allows LSTMs to retain past information from the memory cell over longer periods (Le, 2015). In addition to solving for vanishing gradients, the Graves (2005) LSTM RNN configuration provides a neurologically more realistic interpretation of hidden state memory by allowing memory to be expelled from the system slowly over time, rather than rarely as in the original LSTM (Hochreiter, 1997), or almost immediately as is the case for the Elman RNN (Elman, 1990). The result is an LSTM RNN capable of learning patterns from data with complex interdependencies across temporal dimensions by preserving signals and backpropagated errors over longer time periods than standard Elman RNNs, whilst minimising the vanishing gradient issues.

Deep LSTM RNNs employed in this chapter arrange each hidden layer of the network with depth L as a series of LSTM blocks. Information flows into the network in the form of an input feature vector received at t, $\mathbf{x}^t$, data from previous hidden states, $\mathbf{h}_1^{t-1}$, and previous states of the internal memory cell, $\mathbf{m}_1^{t-1}$, as expressed for layer $l = 1$. The internal memory cell represents useful information that is maintained from previous memory states. Evidently, there appears to be two 'memory' states, **h** and **m**, that together form the total amalgamated memory of the RNN. The external past hidden state, **h**, is conceptualised as a short-term memory element that influences LSTM information flows and learning over the recent history of t. Conversely, the memory cell element, **m**, incorporates longer-term information, with its prime objective being to alleviate the vanishing gradient problem by including a recurrent connection *internally* within the LSTM block (Graves, 2005).

LSTM blocks are composed of an internal data cell $\mathbf{d}_l^t$, three information regulation gates – an input, $\mathbf{i}_l^t$, forget, $\mathbf{f}_l^t$, and output, $\mathbf{o}_l^t$ gate - and an internal memory cell $\mathbf{m}_l^t$, which together control information movements into, within and out of the block. All gate vectors generally have the same dimension and shape of the hidden state. Evidently, following the introduction of gates into the LSTM architecture, the number of parameters in the RNN increases by a factor of four when compared to the simple Elman network (Mikolov, 2012). Elements within the LSTM blocks are connected through either input vectors from the previous adjacent layer, the previous hidden state, or internally between each other through peephole connections, which for clarity are defined by the synaptic weight terms $\mathbf{w}_{l,unit}, \mathbf{v}_{l,unit}, \mathbf{u}_{l,unit}$, respectively, for layer l. The subscript, $unit$, for the weights and biases in the networks relate to the internal unit of the LSTM from which information in the connection *flows forward.*

The exposition that follows explains the architecture and components of the LSTM RNN, with consideration given to how LSTM blocks operate in the first hidden layer of the network, $l = 1$, which can be extrapolated to all other hidden layers $l \in L$ as information propagates in a forward and fully-connected manner from the input to output layers of the network for time period t.

The data cell, $\mathbf{d}_1^t$, represents the raw information received by the network from the input feature vector at t, $\mathbf{x}^t$, and the previous hidden state, $\mathbf{h}_1^{t-1}$, parameterised by weights, $\mathbf{w}_{1,m}$, and, $\mathbf{u}_{1,m}$, respectively. The equation of the cell includes bias term, $\mathbf{b}_{1,m}$. A non-linear hyperbolic TanH function, ϕ_{TanH}, is applied to the input activation value to derive the updated data cell values. Data cell values have consequences for how the LSTM memory cell is updated and for what information flows from the LSTM block towards neurons in adjacent layers, as explained further below.

$$\mathbf{d}_1^t = \phi_{TanH}\left(\mathbf{w}_{1,m}\mathbf{x}^t + \mathbf{u}_{1,m}\mathbf{h}_1^{t-1} + \mathbf{b}_{1,m}\right)$$

The 'input' gate, $\mathbf{i}_1^t$, controls what information from the previous state memory cell, $\mathbf{m}_1^{t-1}$, will be updated given the new data received from the current feature vector, $\mathbf{x}^t$, and the previous hidden state, $\mathbf{h}_1^{t-1}$. Information flows are regulated by the learnable weight parameters for the input vectors, $\mathbf{w}_{1,i}$, the previous hidden state, $\mathbf{u}_{1,i}$, along with the weights connecting the previous memory cell with the input gate through the peephole connection, $\mathbf{v}_{1,i}$. Input gates generally employ a sigmoidal logistic activation, $\phi_{Sigmoid}$, to squash the output into a range between zero and one, with the value of the gate representing how 'important' it is for each of the internal memory cell elements to be updated given the new data received.

$$\mathbf{i}_1^t = \phi_{Sigmoid}\left(\mathbf{w}_{1,i}\mathbf{x}^t + \mathbf{u}_{1,i}\mathbf{h}_1^{t-1} + \mathbf{v}_{1,i}\mathbf{m}_1^{t-1} + \mathbf{b}_{1,i}\right)$$

In a similar manner, the 'forget' gate, $\mathbf{f}_1^t$, provides a blocking mechanism through a peephole connection to decay or terminate irrelevant internal memory cell elements, $\mathbf{m}_1^{t-1}$, with weights, $\mathbf{v}_{1,f}$. The forget gate represents memory elements that are no longer relevant for predicting the RNN classifier, given information signals flowing into the LSTM block from the current input feature vector, $\mathbf{x}^t$, parameterised by synaptic weights, $\mathbf{w}_{1,f}$, and the previous hidden state, $\mathbf{h}_1^{t-1}$, with weights, $\mathbf{u}_{1,f}$, and bias term, $\mathbf{b}_{1,f}$. The forget gate is manifestly a representation of what elements on the internal memory cell should be extricated from the LSTM block given the new data received from the feature vector and previous hidden state. Sigmoid activation functions, $\phi_{Sigmoid}$, are utilised in the forget gate.

$$\mathbf{f}_1^t = \phi_{Sigmoid}\left(\mathbf{w}_{1,f}\mathbf{x}^t + \mathbf{u}_{1,f}\mathbf{h}_1^{t-1} + \mathbf{v}_{1,f}\mathbf{m}_1^{t-1} + \mathbf{b}_{1,f}\right)$$

Following the computation of the values for the data cell, $\mathbf{d}_1^t$, input gate, $\mathbf{i}_1^t$, and forget gate, $\mathbf{f}_1^t$, one can now update the internal memory cell state of the LSTM unit iteratively at time t, $\mathbf{m}_1^{t-1} \rightarrow \mathbf{m}_1^t$. The new cell state, $\mathbf{m}_1^t$, simply forgets memory no longer required, $\mathbf{f}_1^t\mathbf{m}_1^{t-1}$, and incorporates new data into memory depending on its relevance, $\mathbf{i}_1^t\mathbf{d}_1^t$. The updated internal memory cell state is then connected across temporal dimensions to the same LSTM block in the next time period, $t+1$, in a continuous and iterative process until learning terminates. One can note the LSTM architecture applies an identity function to the memory cell, solving for the vanishing gradient problem due to the absence of any non-linear transfer between cells, allowing information from past states to contribute to the current memory update in a stable way that performs well during backpropagation.

$$\mathbf{m}_1^t \leftarrow \mathbf{f}_1^t\mathbf{m}_1^{t-1} + \mathbf{i}_1^t\mathbf{d}_1^t$$

The final requirement of the LSTM block is to compute both the output of the LSTM block, $\mathbf{o}_1^t$, and the updated hidden state for the block, $\mathbf{h}_1^t$, that is then incorporated along temporal dimensions into LSTM blocks in the next time iteration, $t+1$. The output gate amalgamates information from the input feature vector, $\mathbf{x}^t$, previous hidden state, $\mathbf{h}_1^{t-1}$, and updated memory cell, $\mathbf{m}_1^t$, parameterised by weights,

$\mathbf{w}_{1,o}$, $\mathbf{u}_{1,o}$, $\mathbf{v}_{1,o}$, respectively, with added bias term, $\mathbf{b}_{1,o}$. Non-linearity in the form of a sigmoid activation, $\phi_{Sigmoid}$, is applied. The output value, $\mathbf{o}_1^t$, implies the portion of the internal LSTM state that should be exposed to the external environment and represents the input value into LSTM blocks in the adjacent hidden layer deeper in the recurrent network. Once output values are computed they can be used to update the hidden state parameter, $\mathbf{h}_1^{t-1} \rightarrow \mathbf{h}_1^t$.

$$\mathbf{o}_1^t = \phi_{Sigmoid}\left(\mathbf{w}_{1,o}\mathbf{x}^t + \mathbf{u}_{1,o}\mathbf{h}_1^{t-1} + \mathbf{v}_{1,o}\mathbf{m}_1^t + \mathbf{b}_{1,o}\right)$$

$$\mathbf{h}_1^t = \mathbf{o}_1^t \phi_{TanH}(\mathbf{m}_1^t)$$

Hidden state output values, $\mathbf{h}_1^t$, continue propagating forward through the network layer-by-layer until the final output layer is reached at which point the output vector $\hat{\mathbf{y}}^t$ is predicted. The output layer computes the softmax of the previous layers hidden state output value resulting in the model's general prediction:

$$\hat{\mathbf{y}}^t = \varphi(\mathbf{h}_L^t)$$

A schematic illustration of the LSTM RNN is given in Figure 5.2.2 for an individual LSTM cell in layer l.

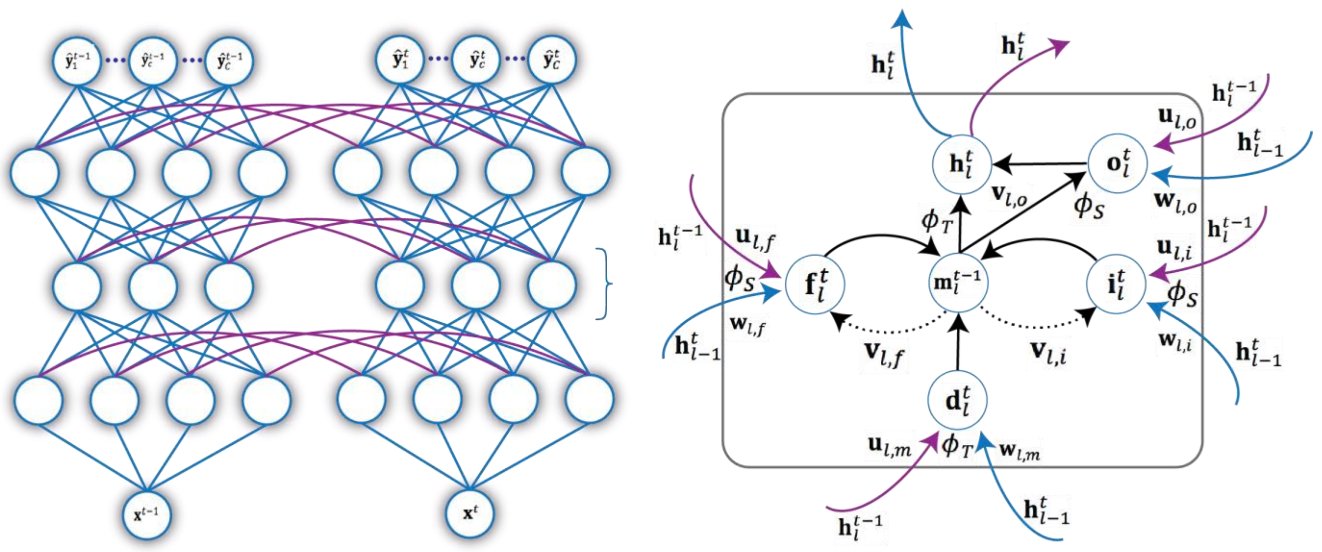

Figure 5.2.2 – Schematic illustration of the Long-Short Term Memory RNN architecture for a cell in layer l, $\mathbf{h}_l^t = \mathbf{o}_l^t\phi_{TanH}(\mathbf{m}_l^t)$. The diagram notes the relevant inputs into each gate and the commensurate activation function, where ϕ_S represents a sigmoid and ϕ_T a hyperbolic TanH activation. Purple lines indicate temporal connections between LSTM cells and blue lines indicate feedforward connections between stacked hidden layers in the LSTM RNN. Dashed black lines indicate peephole connections where gates are a function of the previous time period memory cell, $\mathbf{m}_l^{t-1}$, and the thick black lines represent weighted connections within the LSTM. Note that the memory cell is a recurrent unit that updates internally within the LSTM, $\mathbf{m}_l^t \leftarrow \mathbf{f}_l^t\mathbf{m}_l^{t-1} + \mathbf{i}_l^t\mathbf{d}_l^t$. Further note that the bias terms, $\mathbf{b}_l$, are not shown for simplicity.

LSTM RNNs are trained using backpropagation through time (BPTT) to optimise learnable parameters, $(\mathbf{w}, \mathbf{u}, \mathbf{v}, \mathbf{b})$, within a cost function framework, explained in later sections. Furthermore, it is noted that the application of sigmoidal and hyperbolic tangent activation functions, whilst theoretically tractable, is not a prerequisite and one can draw from the rectifier family of functions to perform this task, which we do in this chapter. Incorporating a recurrent memory cell within the neuronal architecture of the RNN by extending the Elman RNN results in the amelioration of the vanishing gradient problem and the maintenance of temporal dependencies within the model.

5.2.4.3 Gated Recurrent Units

Gated Recurrent Unit (GRU) RNNs utilise a simplified gated architecture that modulates information flows through the network and captures temporal dependencies over different time scales using an adaptive control mechanism rather than a separate memory cell as is the case with LSTMs (Cho, 2014; Chung, 2014). The GRU RNN operates in the same manner as an LSTM network though with the LSTM block replaced by a GRU with its own internal dynamics and weighted gating system. The reparameterisation process undergone in converting GRU to LSTM architectures results in a reduction of learnable parameters. Both networks focus on updating the hidden state, $\mathbf{h}$, for each unit or block in the architecture, based on information from both past states and forward propagated input feature vectors, though differ in their formulation of the state and its level of importance in maintaining memory.

GRUs fully expose their state to the network whilst LSTMs utilise memory cells to internalise the state and memory within each LSTM block, with exposure to the network adaptively controlled by their output gates (Cho, 2014). Both networks maintain temporal dependencies with past hidden states in a way that alleviates the vanishing gradient problem (Chung, 2014). GRU RNN architectures have been shown to perform competitively against LSTM architectures for solving various machine learning problems for temporal data in terms of generalisation power, convergence speed and training stability (Cho, 2014; Chung, 2014; Greff, 2017; Bahdanau, 2014). Other studies have noted the superiority of the GRU architecture over LSTMs in areas such as text and speech recognition (Jozefowicz, 2015).

GRUs can be viewed as neuronal components of an RNN arranged in layers which are fully connected in a sequential process from the input to output layer, similar to the LSTM. These units control how the input feature vector, $\mathbf{x}^t$, flows into the first layer of the network, before updating hidden states for each of the GRU units, $\mathbf{h}^t$, then propagated forwards through the network towards the output layer. For deep RNNs, this process continues iteratively and sequentially from the input towards the output layer through $l \in L$ hidden layers. The critical element of GRUs is their use of a two-gated system to control how this process is conducted. GRU units are analogous to LSTM blocks, though they deviate from the LSTM structure by extricating the implicit memory cell, eliminating peephole connections and reducing the number of gated connections to only two – an update gate, $\mathbf{a}^t$, and a reset gate, $\mathbf{r}^t$.

GRUs replace the memory cell with the LSTM system by maintaining memory explicitly through the hidden state elements, $\mathbf{h}^t$. The GRU RNN is designed in a way that the computed hidden state flows concurrently both through the network at time t to arrive at an output predictor, $\hat{\mathbf{y}}^t$, for input $\mathbf{x}^t$, and across temporal dimensions when the network is exposed to new information at $t+1$. The schematic representation of deep RNNs built with GRUs is presented in Figure 5.2.3. From the figure, it is evident that the hidden state is a critical element of the RNN given its temporal and dynamic linkages through the network. The holistic depiction of the GRU RNN architecture explains how information flows from the input feature vector towards the first hidden layer of the network where $l = 1$. For deeper networks employed in this chapter, the mathematical formulations can be extended for all $l \in L$ layers. Similar to the LSTM section, superscripts refer to the event time step in which the sample vector is received, and subscripts refer to the relevant layer and in the case of learnable weight parameters, the directional flow of information through the synaptic connections.

Information flows into GRU neuron in the form of an input feature vector received at t, $\mathbf{x}^t$, and a previous hidden state, $\mathbf{h}_l^{t-1}$. The hidden state is then updated, $\mathbf{h}_l^{t-1} \rightarrow \mathbf{h}_l^t$, based on information that flows through the reset and update gates of the GRU. The updated hidden state then serves as input into the next hidden layer and as the prior hidden state in the next time step. Hidden states are updated in GRUs depending on both the fully exposed previous hidden state, $\mathbf{h}_l^{t-1}$, and a candidate hidden state, $\tilde{\mathbf{h}}_l^t$, whose values are influenced by how the previous state propagates through the reset gate, $\mathbf{r}_l^t$. Ultimately, the update gate, $\mathbf{a}_l^t$, determines in what proportion the previous hidden state and candidate state influence the new value of the updated hidden state.

Reset gates for layer l at time t, $\mathbf{r}_l^t$, are employed by GRU networks to modulate how temporal dependencies explicitly defined in the prior hidden state, $\mathbf{h}_l^{t-1}$, are combined with new input feature data, $\mathbf{x}$, and maintained in memory. The intuition behind the reset gate is that when it activates with low values it implicitly 'shuts' information flowing from previous hidden states of the network, essentially deleting prior memory, analogous to how memory cells maintain critical information from past states of LSTM blocks. Hence, the reset gate executes the important neurologically-motivated task of forgetting past data from past hidden states, $\mathbf{h}_l^{t-i}$; $i > 0$. The reset gate for the first layer, $\mathbf{r}_1^t$, is computed by applying an additive operation to the affine transformation of the input vector and relevant connective weights between the input and the reset gate, $\mathbf{w}_{1,r}\mathbf{x}^t$, and the affine transformation of the previous hidden state and connecting weights between the hidden state and reset gate, $\mathbf{u}_{1,r}\mathbf{h}_1^{t-1}$, with an added bias term, $\mathbf{b}_{1,r}^t$. Sigmoid activations are generally then applied to the summed input activation value to limit the output within an interpretable range. The weight terms $\mathbf{w}$ and $\mathbf{u}$ control to what extent the prior state information should be maintained or extricated from memory given the new information that has just entered the network at t.

$$\mathbf{r}_1^t = \phi_{Sigmoid}\left(\mathbf{w}_{1,r}\mathbf{x}^t + \mathbf{u}_{1,r}\mathbf{h}_1^{t-1} + \mathbf{b}_{1,r}^t\right)$$

The reset gate is employed when setting the intermediate state's activation value, $\mathbf{h'}_1^t$, to adaptively control the quantum and level of prior state dependencies considered for inclusion in the updated hidden state at t. The updated intermediate state can be interpreted as a vector containing the critical information from prior states after forgetting past information no longer useful as determined by the reset gate. The intermediate state's activation is defined as:

$$\mathbf{h'}_1^t = \mathbf{r}_1^t\mathbf{h}_1^{t-1}$$

Hidden state values, $\mathbf{h}_1^t$, for GRUs are updated through a sequential process that requires the formulation of a candidate hidden state, $\tilde{\mathbf{h}}_1^t$, which serves as a proxy for what the new state should be and derives from the intermediate hidden state variable, $\mathbf{h'}_1^t$, previously calculated. Candidate hidden state values, $\tilde{\mathbf{h}}_1^t$, are computed as the sum of the affine transformation for both the input vector and connecting weights, $\mathbf{w}_{1,h}\mathbf{x}^t$, and the updated intermediate state vector and respective weights, $\mathbf{u}_{1,h}\mathbf{h'}_1^t$, plus the bias term, $\mathbf{b}_{1,h}$. A hyperbolic TanH activation non-linearity is applied to this input value.

$$\tilde{\mathbf{h}}_1^t = \phi_{Tanh}\left(\mathbf{w}_{1,h}\mathbf{x}^t + \mathbf{u}_{1,h}\mathbf{h'}_1^t + \mathbf{b}_{1,h}\right)$$

Both the candidate state, $\tilde{\mathbf{h}}_1^t$, and the prior hidden state vectors, $\mathbf{h}_1^{t-1}$, propagate forwards through the GRU cell and into the update gate vector, $\mathbf{a}_1^t$, which amalgamates the input and forget gates inherent in the LSTM architecture into a single element that controls changes to the memory of the hidden state neuron, $\mathbf{h}$. The procedure taken to compute the update gate for the first layer of the network, $\mathbf{a}_1^t$, involves a similar procedure to calculating input gates in LSTM blocks without any peephole connections. Update gates sum the affine transformation between the input feature vector and the synaptic weights connecting the input and first hidden layer update gate, $\mathbf{w}_{1,a}\mathbf{x}^t$, with the affine transformation of the previous hidden state and its connecting weight, $\mathbf{u}_{1,a}\mathbf{h}_1^{t-1}$, plus bias term, $\mathbf{b}_{1,a}^t$. Sigmoid activation functions transform the input activation value into a value between zero and one in a way that modulates the quantum of information flowing into the GRUs memory.

$$\mathbf{a}_1^t = \phi_{Sigmoid}\left(\mathbf{w}_{1,a}\mathbf{x}^t + \mathbf{u}_{1,a}\mathbf{h}_1^{t-1} + \mathbf{b}_{1,a}^t\right)$$

Hidden state values, $\mathbf{h}_1^t$, are then updated by applying a linear interpolation between the prior hidden state, $\mathbf{h}_1^{t-1}$, and the candidate hidden state, $\tilde{\mathbf{h}}_1^t$, with the proportion of each element determined by the update gate, $\mathbf{a}_1^t$.

$$\mathbf{h}_1^t = (1 - \mathbf{a}_1^t)\mathbf{h}_1^{t-1} + \mathbf{a}_1^t\tilde{\mathbf{h}}_1^t$$

Updated hidden states are a manifestation of the recurrent memory component for GRU RNNs. These values are propagated forwards through the network towards the output layer to help form a prediction, $\hat{\mathbf{y}}^t$, given the input features, and across the temporal dimensions of the network as a prior hidden state, $\mathbf{h}$, at time $t+1$. Once values are propagated layer-by-layer through the deep RNN, an output predictor, $\hat{\mathbf{y}}^t$, is computed and mapped to the input feature vector, using a softmax, φ, output activation function, resulting in the output prediction:

$$\hat{\mathbf{y}}^t = \varphi(\mathbf{o}_L^t)$$

A schematic illustration of the GRU RNN is given in Figure 5.2.3 for an individual GRU cell located in layer l.

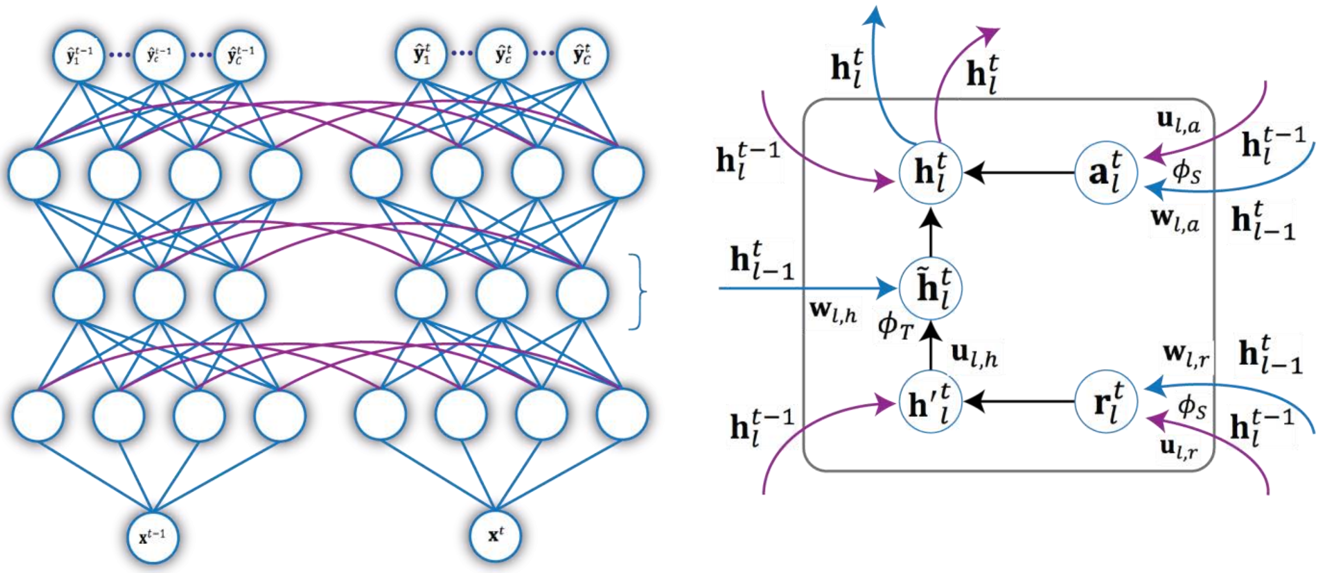

Figure 5.2.3 – Schematic illustration of the Gated Recurrent Unit RNN architecture for a cell in layer l, $\mathbf{h}_1^t = (1 - \mathbf{a}_1^t)\mathbf{h}_1^{t-1} + \mathbf{a}_1^t\tilde{\mathbf{h}}_1^t$. The diagram notes the relevant inputs into each gate and the commensurate activation function, where ϕ_S represents a sigmoid and ϕ_T a hyperbolic TanH activation. Purple lines indicate temporal connections between GRU cells and blue lines indicate feedforward connections between stacked hidden layers in the GRU RNN. Thick black lines represent weighted connections within the LSTM. Note that the bias terms, $\mathbf{b}_l$, are not shown for simplicity.

Both LSTM and GRU RNNs deviate from the traditional Elman RNN by maintaining temporal dependencies from prior hidden states and incorporating new relevant information into the system at t, as encapsulated by the memory components of each network. Memory is maintained using gated connections which are updated iteratively rather than through wholesale new computations of the hidden state as is the case with Elman RNNs. The primary difference between the architectures is the use of a memory cell in the LSTM that controls how memory is exposed in both the internal block using peephole connections and to the external system through output gates. In contrast, GRUs expose their hidden state or memory directly to the network in the form of an updated hidden state value, $\mathbf{h}$. A primordial innovation of the GRU is its combinatorial treatment of the input and forget gates of the LSTM block, which together are ingratiated into the candidate hidden state that enters the update gate of the GRU. This process involves information from previous hidden states in prior time steps being controlled at the update gate of the GRU, rather than through separate or disconnected processes as is the case in the LSTM block.

5.2.5 Network Optimisation

Optimisation of learnable weight and bias parameters for RNNs is predicated on a similar suite of techniques that are employed in feedforward DNNs that were explored in Section 3.1. The objective of RNN optimisation is to train a network that most accurately approximates the mapping function of the training dataset, $\mathbf{y}^t = f(\mathbf{x}^t, \boldsymbol{\vartheta})$. Functions need to be generalisable to real-world domains. Neural networks are trained by augmenting learnable parameters, $\boldsymbol{\vartheta}$, in a way that minimises the networks defined cost function, $\mathcal{L}$, which provides a framework for comparing the prediction of the model, $\hat{\mathbf{y}}^t$, with the actual output class, $\mathbf{y}^t$, over all classes $c \in C$.

Cost functions, $\mathcal{L}$, employed when modelling LOB dynamics with RNNs evaluate the accuracy of the model at predicting the correct future price change on the LOB at time t, $\mathbf{y}^t$, by comparing it against the RNN model's output prediction, $\hat{\mathbf{y}}^t$. Modelling LOB dynamics generally requires the prediction of a specific class, for example, whether the best bid price will increase or decrease over the period $t + \Delta t$. Thus, predicting a class from a fixed subset requires the use of softmax activation functions to allow a probabilistic interpretation of output predictors of the network. Several cost functions work well when comparing probability distributions of the actual output and RNN predictor, with Mean Square Error (MSE) relevant for output distributions with a Gaussian shape, and a cross-entropy (CE) cost function more applicable for modelling the loss from outputs with arbitrary distributions. Cost functions relevant to framing the neural network optimisation problem in this chapter are defined comprehensively in Section 3.1.6.

First-order gradient descent-based learning algorithms are utilised in this chapter as a greedy method to minimise the differentiable cost function through an iterative process of evaluating gradients of the learnable parameters w.r.t the loss function, $\nabla_{\boldsymbol{\delta}}\mathcal{L} = \left[\frac{\partial \mathcal{L}}{\partial \boldsymbol{\vartheta}_1} \quad \cdots \quad \frac{\partial \mathcal{L}}{\partial \boldsymbol{\vartheta}_\mathrm{n}}\right]$, before updating those parameters using a specific optimisation algorithm. Optimisation is performed using mini-batches of training samples, with the learnable parameters updated over each batch. The size of the mini-batch is selected to be large enough to ensure efficient use of computational resources and small enough to minimise the risk of too much noise impacting the convergence of the RNN. The full suite of optimisation algorithms and strategies common to the deep learning literature are set out in Section 3.1.8. One critical difference of optimising RNNs that diverges from DNN training procedures is the use of Backpropagation Through Time (BPTT) to calculate gradients during learning. Furthermore, RNNs utilise parameter sharing such that parameter matrices are maintained across temporal dimensions in the network. This requires the assumption of stationarity over the conditional probability distribution of parameter vector, $\boldsymbol{\delta}$, at time $t+1$ given the value of the variables at t.

5.2.5.1 Adaptive Moment Estimation Algorithms

Adaptive gradient descent-based optimisation algorithms use momentum and learning rate annealing techniques to improve the performance and computational efficiency vis-à-vis vanilla stochastic gradient descent (SGD). Momentum methods induce acceleration into the weight updates in the direction of a persistent reduction in the cost function error, depending on the 'friction' along the curvature of the error surface inherent in the gradients computed over time. Learning annealing is the process of dynamically adapting the learning rate parameter for a specific schedule or training iteration. Adaptive optimisation algorithms that use momentum and learning annealing are able to stabilise oscillations of weights during BPTT around a local minimum region by reducing the learning rate parameter and commensurate kinetic energy in the weight updates over iterations.

The primary optimisation algorithms employed in this chapter derive from Adaptive Moment Estimation (Adam), a first-order adaptive stochastic optimisation algorithm (Kingma, 2015). Adam maintains a decaying squared gradient learning annealing inversion, V_i^m, whilst augmenting and smoothing the gradient term by incorporating a momentum component, M_i^m, that takes the exponentially decaying mean of past iterations gradients whilst adding a bias correction to account for zero-centred vector initialisation. Both the learning rate annealing denominator and the gradient momentum term vectors are initialised towards zero and hence, it is necessary to counteract the biased moment estimates using bias-corrected estimates, $\hat{V}_i^m$, and, $\hat{M}_i^m$. This involves scaling both annealing and momentum variables by the exponentiate of their respective bias terms, $\beta_1, \beta_2 \in [0,1)$, depending on the number of iterations *m*, resulting in scale factors of $(\beta_1)^m$, and, $(\beta_1)^m$, respectively. The value of hyper-parameter bias term β_1 should be chosen to provide less relevance to weights further in the past, especially when the vector was initialised, than those calculated in more recent iterations. In addition, β_2 needs to be sufficiently large to drive convergence through decreased learning rates when the weights are near the local minimum as the neural network cements specific patterns and the gradients increase in sparsity (Kingma, 2015). Recent innovations by Dozat (2016) have introduced Nesterov Momentum components into the adaptive moment estimation algorithms.

5.2.5.2 Backpropagation Through Time

Recurrent neural networks in this chapter are trained to minimise errors in the cost function, $\mathcal{L}$, by applying a truncated backpropagation through time (BPTT) based learning algorithm to optimise the learnable parameters of the network (Werbos, 1990). Utilising BPTT allows for longer-term temporal dependencies to be maintained in the hidden state and memory cell component of the network that can be retained for future time periods. BPTT develops a closed-form solution for RNNs which allows gradient information to propagate through the network from cost function errors to update networks weight matrices $\mathbf{w}, \mathbf{u}, \mathbf{v}$. This procedure requires an *unfolding* of the RNNs layer structure over the temporal dimension axis $t \in t'$, redirecting recurrent connections within the network to form direct connections between hidden states. This results in a directed acyclic graph with weights inextricably linked to their respective layers, analogous to a feedforward DNN, from which gradients of the cost function w.r.t the network's learnable parameters can be computed. The chain rule and dynamic programming are applied iteratively using first-order optimisation techniques defined in Section 3.1.8. Truncated BPTT is also used to reduce the computational complexity of the learning process whilst still capturing long-term dependencies by limiting the event periods over which BPTT is applied during the backward pass. For the LOB dynamics modelled in this chapter truncating the algorithm is necessary given the extremely long sequences inherent in financial time series data. This technique of truncating the BPTT algorithm has been applied successfully for various machine learning tasks including natural language processing (Mikolov, 2012).

BPTT involves the calculation of partial derivatives of the cost function, $\mathcal{L}$, w.r.t learnable parameters, ϑ, including weights connecting adjacent layers in a deep RNN, $\mathbf{w}$, recurrent weights connected to past hidden states, $\mathbf{u}$, and internal weights for LSTM or GRUs, $\mathbf{v}$. In addition, the bias term, $\mathbf{b}$, is also updated using BPTT. Algorithm 5.1 defines the backpropagation pass specifically for an LSTM RNN following the unrolling of a network. The first procedure requires the computation of deltas for the hidden state output, $\delta\mathbf{h}^t = \frac{\partial\mathcal{L}}{\partial\mathbf{h}^t}$, and additionally for elements of the LSTM block. This process is performed iteratively in a backwards direction from the output towards the input, culminating in the computation of the input vector delta, $\delta\mathbf{x}^{\mathrm{t}}$. Once this process has been completed for all layers, $l \in L$, and over the truncated time periods unrolled by the RNN, $t \in t'$, gradients of the learnable parameters can be computed. Algorithm 5.1 defines the final partial derivates for the cost function w.r.t learnable parameters, $\frac{\partial\mathcal{L}}{\partial\vartheta}$. Adam optimisation is then applied to update the parameters.

Algorithm 5.1 – LSTM BPTT with ADAM

```
Initialise LSTM weight vectors w, u, v, and bias term b.
Define delta for hidden state $\delta\mathbf{h}^t = \frac{\partial\mathcal{L}}{\partial\mathbf{h}^t}$
  for each layer $l \in L$ and time $t \in t'$ do
    $\delta\mathbf{h}_{l+1}^t = \mathbf{u}_{l,m}\delta\mathbf{d}_l^{t+1} + \mathbf{u}_{l,i}\delta\mathbf{i}_l^{t+1} + \mathbf{u}_{l,f}\delta\mathbf{f}_l^{t+1} + \mathbf{u}_{l,o}\delta\mathbf{o}_l^{t+1} + \boldsymbol{\theta}^t$   where $\boldsymbol{\theta}^t$ is a vector of $\delta$ from $l+1$
    $\delta\mathbf{o}_l^t = \partial\mathbf{h}_l^t \cdot \tanh(\mathbf{m}_l^t) \cdot \mathbf{o}^t$
    $\delta\mathbf{m}_l^t = \delta\mathbf{h}_{l+1}^t \cdot \mathbf{o}^t \cdot \left(1 - \tanh^2(\mathbf{m}^t)\right) + \mathbf{v}_{l,i}\delta\mathbf{i}_l^{t+1} + \mathbf{v}_{l,f}\delta\mathbf{f}_l^{t+1} + \mathbf{v}_{l,o}\delta\mathbf{o}_l^t + (\delta\mathbf{m}^{t+1} \cdot \mathbf{f}^{t+1})$
    $\delta\mathbf{d}^t = \delta\mathbf{m}^t \cdot \mathbf{i}^t \cdot \left(1 - \tanh^2(\mathbf{d}^t)\right)$
    $\delta\mathbf{i}^t = \delta\mathbf{m}^t \cdot \mathbf{d}^t \cdot \mathbf{i}^t \cdot (1 - \mathbf{i}^t)$
    $\delta\mathbf{f}^t = \delta\mathbf{m}^t \cdot \mathbf{m}^{t-1} \cdot \mathbf{f}^t \cdot (1 - \mathbf{f}^t)$
    $\delta\mathbf{h}_l^t = \mathbf{w}_{l,m}\delta\mathbf{d}_l^t + \mathbf{w}_{l,i}\delta\mathbf{i}_l^t + \mathbf{w}_{l,f}\delta\mathbf{f}_l^t + \mathbf{w}_{l,o}\delta\mathbf{o}_l^t$   where $\mathbf{h}_l^t$ equals input vector $\mathbf{x}^{\mathrm{t}}$ for the first layer
  end for
Compute gradients for LSTM learnable parameters
  for each layer $l \in L$, gate $\mathbf{g} \in \{\mathbf{d}, \mathbf{i}, \mathbf{f}, \mathbf{h}\}$, and time $t \in t'$ do
    $\delta\mathbf{b}_{l,g} = \sum_{t\in t'} \delta\mathbf{g}_l^t$
    $\delta\mathbf{w}_{l,g} = \sum_{t\in t'} \langle \delta\mathbf{g}_l^t, \delta\mathbf{h}_l^t \rangle$   where $\mathbf{h}_l^t$ equals input vector $\mathbf{x}^{\mathrm{t}}$ for the first layer
    $\delta\mathbf{u}_{l,g} = \sum_{t\in t'} \langle \delta\mathbf{g}_l^{t+1}, \delta\mathbf{h}_{l+1}^t \rangle$   where $\mathbf{h}_{l+1}^t$ equals output vector $\hat{\mathbf{y}}^{\mathrm{t}}$ for the final layer
    $\delta\mathbf{v}_{l,i} = \sum_{t\in t'} \mathbf{m}^t \cdot \delta\mathbf{i}^{t+1}$
    $\delta\mathbf{v}_{l,f} = \sum_{t\in t'} \mathbf{m}^t \cdot \delta\mathbf{f}^{t+1}$
    $\delta\mathbf{v}_{l,o} = \sum_{t\in t'} \mathbf{m}^t \cdot \delta\mathbf{o}^t$
  end for
Update learnable parameters using ADAM optimisation algorithm
  for each layer $l \in L$ and parameter vector $\boldsymbol{\vartheta} = [\mathbf{b}, \mathbf{w}, \mathbf{u}, \mathbf{v}]$ comprising all elements $i$
    initialise $\mathbf{M}^0 \leftarrow 0, \mathbf{V}^0 \leftarrow 0, m \leftarrow 0$
    while $\boldsymbol{\vartheta}^m$ has not converged do
      $\mathbf{M}^m \leftarrow \beta_1\mathbf{M}^{m-1} + (1-\beta_1)\delta\boldsymbol{\vartheta}_i^m$
      $\mathbf{V}^m \leftarrow \beta_2\mathbf{V}^{m-1} + (1-\beta_2)(\delta\boldsymbol{\vartheta}_i^m)^2$
      $\widehat{\mathbf{M}}^m \leftarrow \frac{\mathbf{M}^m}{1-(\beta_1)^m}$
      $\widehat{\mathbf{V}}^m \leftarrow \frac{\mathbf{V}^m}{1-(\beta_2)^m}$
      $\boldsymbol{\vartheta}_i^{m+1} \leftarrow \boldsymbol{\vartheta}_i^m - \left(\frac{\alpha}{\sqrt{\widehat{\mathbf{V}}^m + \epsilon}} \cdot \widehat{\mathbf{M}}^m\right)$
    end while
    return learnable parameter $\boldsymbol{\vartheta}^{m+1}$
```

5.2.6 Layer Normalisation

Layer normalisation (LN) is a technique for reducing computational resources required to train RNNs (Ba, 2016). The method uses an approach analogous to batch normalisation (BN) (Ioffe, 2015) introduced in Section 3.1.11. However, rather than applying the normalisation to the *mini-batch* of training samples that inputs into the neural network during training, LN takes the summed inputs for a layer composed of recurrent neurons for a single input sample and normalises across *temporal* dimensions. Similar to BN, the normalisation procedure is performed before the non-linearity is applied through the activation function. Thus, LN is essentially a transposition of the BN algorithm with the normalisation process for

LN occurring at each time step. Most of the networks employed in this chapter utilise approximately 100-500 time step recursions during training which is similar to the number of samples, 128 to 256, used to train the DNNs in Chapter 3. Hence, the impact on computational time is fairly similar for each method.

LN was developed in response to the success of BN at improving computational speed whilst solving the problem of 'internal covariate shift' intrinsic to very deep feedforward networks. BN does not apply well to RNNs in general given that BN computes the statistics for each neuron as a moving average requiring the storing of mean and standard deviation statistics at each time period, which is difficult when training neural networks that model temporal dependencies across very large event-time periods (Cooijmans, 2016).

For an RNN with n neurons defined for layer l, one can define the hidden state for neuron i in that layer as $\mathbf{h}_{l,i}^{t}$, which is derived as the input vector, $\mathbf{x}^{t}$, at time t, propagates through the network layer-by-layer combining with other elements of the RNN in the process, before inputting in layer l. Hidden state neurons serve as both the input into the next sequential layer towards the output and as the hidden state across the linear temporal domain into the next time period. Layer normalisation is applied to recurrent layers of the network by normalising each hidden state neuron $\mathbf{h}_{l,i}^{t}$ by shifting the value by the *mean* of all hidden state neuron values in the *layer*, μ_l^t, and scaling that value by the standard deviation for that layer, σ_l^t, before the non-linearity, ϕ_l, is applied. Similar to BN, the LN technique applies an identity transformation to scale and shift the normalised activation input values by parameters, γ, and, β, respectively, which maintains the dimensions of the hidden state variable, leading to:

$$\mathbf{h}_{l,i}^{t} \leftarrow \phi_l\left(\frac{\gamma\left(\mathbf{h}_{l,i}^{t} - \mu_l^t\right)}{\sigma_l^t} + \beta\right)$$

$$\mu_l^t = \frac{1}{n}\sum_{i=1}^{n} \mathbf{h}_{l,i}^{t} \qquad \sigma_l^t = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{h}_{l,i}^{t} - \mu_l^t\right)^2}$$

The LN procedure results in faster computations during training and a more stable RNN particularly when there are complex inter-neuron connections as is generally the case with deep recurrent networks (Ba, 2016).

5.2.7 Recurrent Dropout

Regularisation techniques are commonly employed in RNNs given their tendency to overfit to the training data during optimisation. Several regularisation strategies were introduced for DNNs in Section 3.4.9 with the most applicable being *dropout* which prevents neurons from co-adapting during training by fitting a randomly generated mask that drops neurons in the network with probability p (Hinton, 2012b; Srivastava, 2014). Neurons extricated from the data manifold are not updated during training optimisation.

In this study, two forms of dropout are applied to the RNNs. *Standard* dropout is applied for neurons connected *between* the input, recurrent and output layers in the RNN and *within* the recurrent layer itself to the memory cell or hidden state vectors, depending on the RNN, using a form of *recurrent* dropout. Recent studies have demonstrated the deficiency in both network stability and predictive power when applying dropout naively to the *recurrent* connections in RNNs which has been attributed to the noise applied implicitly to recurrent connections (Pachitariu, 2013) and to the disruption caused to the internal dynamics of the RNN (Pham, 2014), when dropping random neurons. To solve for these issues, we employ a recurrent dropout method for RNNs introduced by Gal (2016) that largely averts the stability concerns when naively applying dropout to recurrent layers after each batch size has been fed into the network by using a Bayesian approximate inference technique that involves fitting the same mask for all neurons during a single epoch training iteration.

5.2.8 Self-Normalising Neural Networks

Scaled exponential linear units (SELU) are a non-linear activation function applied to 'self-normalising neural networks' (Klambauer, 2017) and are employed in this chapter when training RNN models of LOB dynamics. SELUs extend the activation functions holistically explored in Section 3.1.5, particularly the exponential linear unit (ELU) function which is part of the rectifier activation family (Clevert, 2015). ELUs activate in an analogous manner to traditional hard-zero bounded ReLU functions, though with a smooth negative region bounded by hyper-parameter a. This has the added impact of allowing the function to saturate, rather than be constant in the negative region, therefore inducing a noise robust deactivation state. The primary benefit of ELUs is that they perform a bias-shift correction that pushes the mean activation output value of a vanilla ReLU network towards zero mean and unit variance. This theoretically tractable property is conducive to neural networks as it results in tensors being propagated through the network in a way that maintains the activation distribution for *each layer*, that is, with mean zero and unit variance. However, whilst ELU corrects for the bias shift to a degree, the fixed parameter boundary of the ELU activated network still generates discontinuities between the activations passed between layers. Both layer normalisation (Ba, 2016) and batch normalisation (Ioffe, 2015) algorithms have been proposed to perform the normalisation process explicitly to correct for non-zero mean and unit variance activations, however, this process can be internalised within the network's non-linearity by utilising the SELU activation function. Klambauer (2017) utilises the Banach fixed-point theorem neuron to prove that SELU activation values converge to zero mean and unit variance. SELUs control for the zero mean through their property of having activation values in both positive and negative regions. Furthermore, they also control the shape of the saturation region which allows for the variance of the network activations to be regulated. Finally, the inclusion of a scalar, $\lambda > 1$, allows for the activation to modulate the influence of network layers closer and further from the input layer. The result is the SELU activation function that executed implicit self-normalisation. The form of the SELU is that of an ELU activation, multiplied by a scalar, λ. Furthermore, both the scalar and the negative bound hyper-parameter, α, are learnable parameters, with estimates derived in Klambauer (2017) provided with the activation function:

$$\phi_{SELU}(z) = \lambda \begin{cases} z & if\ z > 0 \\ \alpha(e^z - 1) & if\ z \leq 0 \end{cases} \qquad \lambda > 1 \qquad C = 1.6733,\ \lambda = 1.0507$$

5.3 Deep Reinforcement Learning

Deep reinforcement learning (RL) is a computational approach to designing machine intelligence which extrapolates human conceptions of learning and optimal behaviour through interactions with the environment. As machine intelligence systems begin to supersede human cognitive capabilities in the realm of market microstructure and trading, RL techniques present themselves as a useful model to understand how and why limit order books (LOB) evolve over time. Whilst the ability to acquire and apply knowledge is increasingly being outsourced to intelligent machines, the nature of how machines learn does not diverge significantly from humans. All humans are faced with an environment beyond their comprehension, there are almost an infinite series of situations one can find themselves in and decisions that can be made, all played out in real-time, yet over time humans form intelligence as they learn to acquire and utilise knowledge. The crux of this intelligence is gained by interacting with one's environment to test what is possible, garner information and knowledge, and then to formulate what is desirable. This concept is the foundation of deep reinforcement learning.

The following sections introduce the Markov Decision Process as the primary paradigm employed in RL. We define dynamic programming as the exact method to solve for optimal behaviour with an RL system using Bellman Optimality equations and off-policy Q-Learning, though we note that this technique is not computationally tractable for complex systems such as modelling LOB dynamics. Finally, we provide a brief exposition of more relevant model-free methods that utilise approximate RL before introducing the Deep Recurrent Q-Network (DRQN) employed in this chapter to predict the future price dynamics of LOBs.

5.3.1 Markov Decision Process & Dynamic Programming

LOB dynamics can be modelled as a stochastic optimal control problem in a dynamic system within a discrete event time framework in which traders make decisions to maximise some reward function. RL is the machine learning response to this type of *control* optimisation problem involving state-based agents making sequential decisions within a dynamic system analogous to a Markov Decision Processes (MDP) (Howard, 1960, Sutton, 1998).

Mathematically, MDP RL systems involve a state space, $\mathcal{S}$, and action space available to the agent in a certain state, $\mathcal{A}(s_t)$, in which agents receive information about the current state, $s_t \in \mathcal{S}$, and execute an available action, $a_t \in \mathcal{A}(s_t)$, across a sequence of discrete time periods $t \in T$. The state space represents the different configurations of the system and encompasses all observable and relevant features in the environment at present that are used by the agent to make a decision and which cannot be arbitrarily altered by the agent (Sutton, 1998). Hence, past observations of the environment, actions taken, and rewards received are not explicitly included within an RL algorithm, but rather the relevant historical components are captured and infused into the current state space.

The stochastic transition of the state from s_t towards s_{t+1} upon execution of an action, a_t, based on transition probability $\mathcal{P}_a$, presents the agent with a reward signal, $r_{t+1} \in \mathcal{R}$, where $\mathcal{R}$ represents the real value rewards available to the agent. The transition process is rooted in the logic that once an agent engages with the environment, the current state is dynamically transformed. The reward represents a scalar feedback signal which is a real value number that infers the current position of the agent. Reward signals define the task of the agent and as a result are computed external to the agent who is not able to arbitrarily alter them (Sutton, 1998). A discount factor, γ, can be applied to the reward function to differentiate between value attributed to past and present returns, avoiding the problem of infinite returns in a cyclic Markov process. Hence, the MDP for modelling an agent's decision process can be defined through a five-factor tuple, $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}_a, \mathcal{R}, \gamma)$.

Critical assumptions of MDPs include that the current state is completely observable and that it holistically captures all relevant information required by the agent to execute optimal actions, including knowledge garnered through past experiences. Mathematically, MDPs require that the conditional probability distribution of transitioning to successor states and receiving rewards given the current state and action space are independent of past states and actions, hence meet the Markov property. The distinguishing feature of RL systems is that no explicit representation of the correct behaviour or action to take is presented to the agent. Rather, through the agents' interaction with the environment they receive evaluative feedback signals from the actions they take which indicate how successful an action is and potential ways of making better decisions (Sutton, 2012).

The RL machine learning paradigm depicts agents as conducting a defined policy that maps states to actions, $\pi(a_t|s_t)$, informing agents optimal actions independent of the epoch period. The MDP $\mathcal{M}$ and policy π fully define agents behaviour with the objective of selecting a policy that maximises cumulative expected discount returns (Watkins, 1989; Sutton, 2012). At any point t in an MDP we calculate the cumulative discounted return from the current state and across an episode of k actions to the terminating time period T, based on the sequence of rewards r_i and discount factor γ, as $G_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1}$, where $\gamma \in [0,1]$ and $|r_k| < \infty : k > t$. Solving for the optimal policy requires the use of a state-action value or Q-function to represent the desirability of being in state s_t and taking action a_t before following policy π (Watkins, 1992). The Q-function can also be decomposed into both an immediate reward, r_{t+1}, and the discounted value of the successor state, $\gamma Q_\pi(s_{t+1}, a_{t+1})$, which allows for it to be reformulated using a recursive Bellman equation (Sutton, 1998), with the following consistency condition holding for all state values $s \in \mathcal{S}$ and any policy π:

$$Q_\pi(s_t, a_t) = \mathbb{E}\left[\sum_{k=0}^{T} \gamma^k r(s_{t+k}, a_{t+k}) \,|\pi\right] = \mathbb{E}[r_{t+1} + \gamma Q_\pi(s_{t+1}, a_{t+1})|\pi]$$

The mathematical structure of the Bellman Optimality equations inherently encapsulates the principle of optimality which presupposes that whatever state the agent is currently in they should take the best action and then *must* continue to act optimally afterwards (Bellman, 1957). Rational agents in RL systems with MDP settings select an optimal policy π^* which solves the MDP $\mathcal{M}$ and maximises the cumulative expected returns at any point for a given state s and expected transition probability, $\mathcal{P}(s_{t+1}|s_t, a_t)$, based on optimal decision rules which maximise the Q-function, $Q_{\pi^*}(s, a)$, giving:

$$Q_{\pi^*}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{\mathcal{P}(s_{t+1}|s_t, a_t)}\left[\max_{a_{t+1} \in \mathcal{A}(s_{t+1})} \{Q_{\pi^*}(s_{t+1}, a_{t+1})\}\right]$$

Dynamic programming (DP) algorithms are an RL method that used to solve for the optimal Q-function and policy if an agent has complete information regarding the elements of a discretised, stationary and fully known environment of the MDP environment $\mathcal{M}$ (Bellman, 1957; Watkins, 1989). DP involves breaking the overall RL problem into smaller sub-problems, solving for each problem, and then storing the solutions in a memory-based tabular data structure. This technique of *memoisation* attempts to overcome the 'curse of dimensionality' in which the rise in number of explanatory features exponentially increases the state space possibilities, requiring the caching of computationally expensive functions and processes into memory. DP requires the application of the Bellman Optimality equation to solve for the optimal policy π^* at convergence using iterative learning methods such as value iteration or generalised policy iteration.

It becomes infeasible to solve an RL problem in many real-world problems with DP when the assumptions of a fully known MDP are not met. Even for complex RL problems with fully known MDPs, the curse of dimensionality, where the number of variables required to be computed grows exponentially with the dimensionality of the state space, rendering the agent unable to solve for optimisation given the bounded computational constraints. To account for the shortcomings of DP, model-free and approximate methods of RL have been developed to truncate the computational requirements for solving complex problems. These methods draw their architectural roots in DP but relaxing the assumption that complete knowledge of the environment is required. Rather, agents learn by interacting with the environment to determine the optimal Q-function and hence the optimal policy for the agent to follow.

LOB dynamics in modern market microstructure environments have the capability of being explained through an MDP framework and solved for using RL methods. However, the complexity of this environment and the infeasible computational capability required for optimisation with basic control methods requires consideration of approximate RL methods. Furthermore, these methods can be amalgamated with Recurrent Neural Network (RNN) models to optimise an agents' policy using model-free Deep Q-Learning approximation methods, which is the key RL model deployed in this chapter.

5.3.2 Model-Free Reinforcement Learning

Model-free methods of control optimisation can be utilised to break Bellman's curse of dimensionality in an environment with unknown state transition dynamics when reward functions are non-stationary in nature, or in cases where the MDP is well defined it may be too computationally expensive to solve for optimal behaviour using DP. Agents learn by extracting samples of experience by interacting directly with the environment to formulate some value function and hence an optimal policy. The sample trajectories and expectations supplant the full reward function $\mathcal{R}$ and transition dynamics $\mathcal{P}$, and thus, do not require full knowledge of the MDP environment dynamics to solve for optimal behaviour.

Model-free reinforcement learning techniques approach the agents' control optimisation problem through a synchronous iterative process of *policy evaluation* and *policy improvement* based on sample trajectories. Policy evaluation involves performing backup and updates to an agents' Q-functions based on their freshest samples of experience. Secondly, exploratory ϵ-greedy policy improvement is performed to derive a better policy for the agent based on the updated Q-functions. This continuous cycle of policy evaluation and improvement is executed until convergence at the optimal Q-function and optimal policy.

In this chapter we, focus on off-policy Q-Learning methods to learn a deterministic optimal policy by sampling from episodic experiences of executing policies π_k, and using these samples to improve the Q-function using generalised policy iteration to better approximate π. Q-Learning is a model-free Temporal

Difference algorithm which is rooted in the same architecture of the SARSA algorithm (Rummery, 1994), but rather learns the optimal Q-function through off-policy learning (Watkins, 1989; Singh, 2000). Q-Learning begins by initialising an arbitrary Q-function function, Q_{π_k}, and updates the function based on an observed reward and state transition tuple $\langle s_t, a_t, s_{t+1}, r_{t+1}\rangle$, based on a learning rate α. The process of updating the function, $Q_{\pi_k}(s_t, a_t)$, involves taking the ϵ-greedy action, a_{t+1}, with respect to the successor state-action value function, $Q_t(s_{t+1}, a_{t+1})$:

$$Q_{\pi_k}(s_t, a_t) \leftarrow Q_{\pi_k}(s_t, a_t) + \alpha\left(r_{t+1} + \gamma\left[\max_{a_{t+1}\in\mathcal{A}(s_{t+1})} Q_t(s_{t+1}, a_{t+1})\right] - Q_{\pi_k}(s_t, a_t)\right)$$

This update is performed independently of the policy π_k being followed, though the policy will still determine how the agent moves through the system and what state-action pairs are visited and updated.

Once the policy evaluation is complete the agent performs ϵ-Greedy in the Limit with Infinite Exploration (GLIE) improvement by choosing the ϵ-greedy action with respect to the updated function, $Q_{\pi_k}(S_t, A_t)$, for each state-action pair. This procedure determines that the agent should perform exploration with probability ϵ by selecting a random one of m actions in the set $\mathcal{A}(s_t)$ which are available to the agent in state s_t. With probability, $1-\epsilon$, the agent exploits their current knowledge by acting greedily with respect to the state-action value function, Q_{π_k}, which involves selecting a policy from all available actions in the current state, $\mathcal{A}(s_t)$, which maximises the agents cumulative expected discounted rewards. The GLIE method requires the epsilon value to decay asymptotically to zero, $\epsilon \leftarrow \frac{1}{k}$, as more sample episodes are conducted, satisfying the Bellman conditions of optimality by ensuring that all state-action pairs are explored infinitely multiple times and that eventually the policy converges on a greedy policy.

Exploratory ϵ-GLIE policy improvement bridges the exploitation versus exploration dichotomy manifested in the competitive tension between the agents' desire to exploit current knowledge to maximise rewards, and their quest for broader knowledge through the exploration of the environment. An agent that follows a deterministic policy π_k will only view the state-action pairs for which the policy specifies, whilst there may be utility gained from learning the values of non-policy actions available to the agent. Conducting exploration ensures an agent transitions to states previously unvisited, and through interacting with new and diverse components of the environment the agent may acquire new knowledge and potentially increase the upper bound of rewards available. The ϵ-GLIE policy improvement leads to the new policy π_{k+1} for state s_t of:

$$\pi_{k+1}(s_t) \leftarrow \epsilon - GLIE\left(Q_{\pi_k}(s_t, a_t)\right) \quad where\ a_t = \begin{cases} \arg\max_{a_t\in\mathcal{A}(s_t)} Q_{\pi_k}(s_t, a_t) & with\ probability\ 1-\epsilon_k \\ random\ a_t \in \mathcal{A} & with\ probability\ \epsilon_k \end{cases} \quad \epsilon \leftarrow \frac{1}{k}$$

Q-Learning algorithms converge asymptotically to the optimal state-action value function, Q_{π^*}, under the assumptions that states are visited infinitely often which is achieved using episodic GLIE ϵ-greedy policy improvement (Singh, 2000).

5.3.3 Deep Recurrent Q-Networks

Modern LOBs are a highly complex system explained by a near infinite state space of potential financial variables and an action space composed of a multifarious array of order types that can be placed at minute price point constrained only by the trading venues tick size. Modelling this complex system can be performed using approximate RL in the form of a Deep Recurrent Q-Network (DRQN). This model requires a divagation from the tabular representation of state and action values utilised in Q-Learning to one where Q-function values are approximated using a Recurrent Neural Network (RNN) in a quasi neuro-dynamic programming framework. Various iterations of DRQN models have been successfully applied to generate music (Jacques, 2017), whilst standard Deep Q-Networks have been used to attain and surpass human-level abilities when playing computer games (van Hasselt, 2016).

In complex systems, Q-functions may be approximated using value function approximation by generalising the state-action space rather than calling directly into a large cached state-action tabular database. Value function approximation uses supervised learning methods such as neural networks by taking examples of desirable value functions and extrapolating these to develop a general non-linear approximation of the entire value function. Function approximators must account for problems where the data is non-stationary and samples are not independent and identically distributed, given that the probability distribution of the random variables in the model change with time. The advantage for using value function approximations is that multiple state or state-action pairs with very similar attributes where the difference between them has only a minuscule impact on the value of being in that state-action pair, can be compacted together into a single value function, allowing for scalability in very complex RL systems such as LOBs.

Q-function or state-action value function approximation estimates the true value function, $Q_\pi(s,a)$, for policy π, using a parametric function approximator, $\hat{Q}_\pi(s,a,\mathbf{w}) \approx Q_\pi(s,a)$, with parameter weight vector $\mathbf{w}$. The true Q-function, Q_π, represents the actual discounted cumulative reward from being in state s, taking action a, and then following policy π. A DRQN parameterised by $\mathbf{w}$ can be trained to find an estimate of the current policy Q-function that is closest to the optimal policy Q-function such that, $\hat{Q}_\pi(s,a,\mathbf{w}) \approx Q_{\pi^*}(s,a)$. The weights of the DRQN are optimised through this process using stochastic gradient descent (SGD) or adaptive gradient descent optimisation techniques presented in Section 3.1.8 and Section 5.2.5 by minimising the mean squared error (MSE) cost function, $\mathcal{L}(\mathbf{w})$, with respect to the parameter of weights $\mathbf{w}$, over a distribution of inputs. The MSE for weight $\mathbf{w}$ estimates the sum of the squared difference between the true Q-function, $Q_\pi(s,a)$, and the value function approximator, $\hat{Q}_\pi(s,a,\mathbf{w})$, giving:

$$\mathcal{L}(\mathbf{w}) = \mathbb{E}_\pi\left[\left(r_{t+1} + \gamma\left[\max_{a_{t+1}\in\mathcal{A}(s_{t+1})} Q_t(s_{t+1},a_{t+1})\right] - \hat{Q}_\pi(S,A,\mathbf{w})\right)^2\right]$$

In deep RL, using value function approximation and DRQN to approximate the Q-function attempts to find the set of weights $\mathbf{w}^*$ that minimises the MSE at convergence, such that:

$$\mathcal{L}(\mathbf{w}^*) \leq \mathcal{L}(\mathbf{w}) \qquad \forall \mathbf{w}$$

Several extensions and techniques pertinent to DRQNs have been extended in recent times. This includes including use of experience replay (Mnih, 2015), double Q-learning methods (van Hasselt, 2016) and RL tuner algorithms (Jacques, 2017).

5.4 Data

Recurrent Neural Networks employed in this chapter utilise data extracted from the FTSE100 consolidated limit order book dataset introduced in Chapter 3. The unique dataset consolidates trading exchange reporting data provided by the Financial Conduct Authority (FCA) for all key lit and dark limit order books (LOB) which trade UK equity securities. The panoptic nature of the consolidated LOB dataset allows for 97% of trading activity over the period 2015-2017 for FTSE100 sample securities to be captured.[25] At any point in time before a trader executes an action, also referred to as an event, one can develop a representation of the state of the combined LOB exposed to that trader. All sample securities undergo continuous trading during the period of 8 am to 4:30 pm, with no trading halts or disruptions to continuous trading occurring across the period of analysis. Similar datasets using past trading periods have been used by Bercich (2017), Benos (2012) and Aquilina (2016).

In this chapter, we focus predominately on the LOB dynamics of the London Stock Exchange for five random FTSE 100 securities traded during October 2017. Additional features of the state of financial markets are garnered based on the activity of traders on other lit and dark order books trading these securities. The period of analysis, stocks and sample size have been chosen to maintain a level of robustness and accuracy in the analysis, whilst working within the constraints of available computational resources. The dataset totals several billion messages exchanged between traders and venues and is cumulatively over four terabytes in size.

The dataset employed captures all messages between market participants and four primary UK exchange lit and three dark-pools – London Stock Exchange (XLON), BATS Europe Lit BXE (BATE), BATS Europe CXE (CHIX), Turquoise Integrated Lit Order Book (TRQX), Bats Europe BXE Dark order Book (BATD), Bats Europe CXE Dark Order Book (CHID) & Turquoise Plato Dark Order Book (TRQM). Raw unfiltered message data and transaction reports for each of the order books undergo a cleaning and filtering process, before being amalgamated into a single FTSE100 CLOB. Orders placed on each exchange occur sequentially from the beginning of trading until all orders are cleared off the LOB at the end of the trading day. Information pertaining to the Broker submitting a message to the exchange is extricated from the data manifold given that we want to model the dynamics of the LOB independent of individual traders. The dataset includes intraday orders submitted, cancelled, amended, all transactions, and other residual messages at all levels of the LOB and is timestamped to the millisecond. Additional meta-data is provided in the form of the order type, for example limit or iceberg order, the initiator and aggressor of the trade, being either the bid or ask broker, and the period that the trade is submitted, during continuous trading or auctions.

Limit order book dynamics are modelled using an event time methodology that captures a state of the LOB and its relevant explanatory features at discrete points in event time. The LOB datasets for each FTSE100 security are combined over five-day periods. A training dataset is extracted from the first three days, a validation dataset over the fourth and a final testing dataset for the fifth complete trading day.

[25] Estimates were calculated for the period 1/1/15 to 31/12/17 for the FTSE 100 Index using data from fragmentation.fidessa.com.

Recurrent Neural Networks models are trained over the datasets to complete three separate prediction experiments:

A. Prediction of the next order submission executed at or within the BBO. Models are trained on datasets for orders submitted at these prices which represent impactful information regarding the dynamics of the LOB.
B. Prediction of the next midpoint price change direction using a dataset of all actions performed on the LOB within five ticks of the BBO.
C. Prediction at event period t of the future midpoint price of the securities at event time $t + n$.

The dataset for experiment A is comprised of all events when an order is submitted at or within the BBO. These actions include market orders that execute a trade, passive limit orders that tighten the spread and BBO limit orders that increase the market depth of the best price on either the bid or ask side. For experiment B the RNN model employed attempts to capture temporal dependencies between features that provide insight into the next midpoint price movement for the specific security. Specifically, we predict whether the next price change will be an increase or decrease. Midpoint price changes are less common than actions performed in the previous experiment, given that only orders submitted within the BBO, or order deletions at the BBO that deplete the queue on either the bid or ask side, cause the midpoint price to shift.

5.4.1 Features

Modelling complex non-linear LOB dynamics using an RNN to predict price movements requires a representation of the current state of the LOB, $\mathbf{x}$, at a given point in event time t. Feature engineering is performed to identify and generate a rich set of LOB features that one would expect to be integrated into the set of rules, calculations and operations underlying the algorithm of firms trading in UK equity markets. The generation of features follows a similar process to that conducted in Chapter 3 to model algorithmic trading behaviour. However, a key difference is that features relevant to specific traders need to be extricated from the data manifold given that these features are inherent to the trader, thus, they are not visible to the external environment. For example, each trader's current net inventory position, their LOB distributions, quote positions, recent order history, and many other features that were used to model these traders' behaviours using a Deep Neural Network, are not used in this chapter.

A rich *subset* of LOB features defined in Appendix A are extracted from the LOB over micro-time periods is used to train RNNs to model the LOB dynamics and predict price changes. This set of features is derived primarily from the LSE LOB and includes microstructural, momentum, technical, execution and probability of informed trading signals. In addition, signals from both dark pools and other lit trading venues that trade these UK equity securities are also incorporated. We only utilise features of the LOB that would be apparent to *all* traders, ignoring individual trader features that are important for understanding algorithmic trading but not the overall dynamics of the LOB.

Microstructural signals emanate from the inherent LOB dynamics of a security given the state of the market microstructure and trading environment itself. Algorithmic or high-frequency traders commonly interpolate these signals within their algorithms, several of which are employed in this chapter as features when modelling LOB dynamics. These features are derived from order flow auto-correlation, order aggressiveness, market depth, limit order imbalance, quoted spread, effective spread, and volatility signals. Order flow signals occur when market order signs have the statistical property of serial autocorrelation as they tend to persistently occur in the same direction over sequential trade events. Cartea (2018) notes the relevance of these signals in order placement strategies of market makers. Several features related to these signals are developed based on both the sign and volume of the order flow. Order aggressiveness signals measure the demand for immediacy of liquidity over a certain period, with these signals providing information on traders' expectation of future price moves (Pasquariello, 2007). Market depth features are developed to provide a reconstruction of the LOB state at multiple price levels from the BBO that proxies for liquidity in the market. Limit order imbalance features that measure the relative imbalance between bid and ask orders at multiple price points are employed as features to explain the LOB dynamics of a security. Several theoretical and empirical machine learning studies have found non-linear relationships between order imbalances and LOB dynamics (Sirignano, 2016; Cartea, 2018). Quoted spreads are an ex-post features used to quantify the distance between the best bid and ask, whilst effective spreads measure the distance between these prices when transactions are executed, proxying for trading costs. These features have invariably been assessed as proxies for the prevalence of informed trading (Ait-Sahalia, 2017), volatility (Glosten, 1988), and liquidity (Sarkissian, 2016), motivating their use in this chapter as features relevant to model LOB dynamics. Finally, volatility signals are impounded into the RNN using measures of realised volatility using logarithmic returns based on open and close prices and high and low prices.

Momentum signals are based on movements in the LOB dynamics over micro-time frames. Features relating to momentum signals are used to impound information relating to how the LOB evolves over temporal periods which can be exploited given the recurrent component of RNNs. Momentum features relate to how prices are changing over small periods in event-time. Acceleration features measure the rate of change in momentum signals using various window periods. In contrast to momentum and acceleration, which are based on both limit orders and transactions, volume features are derived only from trades by quantifying the total absolute number of shares traded on the LOB over certain window periods.

Technical indicator signals are derived from movements and patterns in LOB prices and volumes over a specific event-time period. The primary differentiator between momentum and technical signals is the time or event period of analysis, with momentum related to micro-time movements in the high-frequency data whilst technical indicators are generally based on short or medium-term fundamental trends or patterns in security prices and LOB dynamics to make predictions about price changes. Technical indicator signals can be segregated into price reversion, price pattern, and volatility signals. Price reversion signals indicate when the current security price has reached a bounded level, beyond which it is expected to mean revert back to its equilibrium value. This chapter utilises Bollinger band features to proxy for this boundary, with bands modelled by expectations regarding the continuance of trends in a stock over a short time-frame encapsulated by the level of buying or selling in a signed direction, until resistance levels are reached, after which prices are assumed to mean revert. Price pattern technical indicators employed as features include the Moving average convergence/divergence and Relative strength indicator. Both these features utilise various forms of exponential moving average measures of price changes to determine when securities are over or under priced. One would expect traders to react to these metrics by executing actions which respond to the relevant signal, explicitly influencing the dynamics of the LOB. Along a similar dimension, stochastic oscillator signals are technical indicators that measure trends, momentum and the speed of price movements and serve as features for modelling LOB price changes and dynamics. Finally, Chaikin volatility is another technical indicator that proxies for volatility on the LOB and incorporates a dynamic component by measuring how volatility rates are changing in event time.

Execution signals are derived from the optimisation process involved in buy-side firms executing large parent orders on behalf of their clients. The LOB dynamics can be modelled using features derived from these signals as they represent critical elements of execution trading strategies. In particular, the detection of a firm executing a large parent order has a consequential impact on how the dynamics of the LOB evolve as other traders respond by shifting their order distributions in a way that may increase the implicit and explicit costs of the trader liquidating or buying the parent order. Both Time-weighted and Volume-weighted average price features are developed to help predict how the LOB dynamics will evolve over time. As a basic example, when the VWAP of a security is higher than the current price, this may motivate traders executing a VWAP strategy to purchase more of the security to reduce their VWAP, with a commensurate effect on how the LOB order distributions evolve. Features relating to implementation shortfall measures are also derived to proxy for the degree of slippage in the market using various transaction cost metrics. Of more relevance for LOB state movements is the level of permanent and transitory price impact for a security, in addition to the resiliency of the LOB for that security. A wide range of features are developed to quantify price impact and resiliency. First, regressions are developed using models built by Almgren (2005) to decompose the permanent, temporary and realised components of price impact, in addition to a measure for market resiliency, using various moving window sizes. These measures quantify the commensurate impact on midpoint prices from transactions and limit orders, separately. Second, a transient impact model (Gatheral, 2012) with a decay component is employed to measure the non-linear temporary market impact from order book events. Third, a history dependent impact model (Eisler, 2011) is used in conjunction with an ‘influence kernel’ to model the price impact dynamics depending on how certain orders influence the LOB dynamics. Finally, a comprehensive set of market resiliency features are developed using vector auto-regression models impulse response functions (Lo, 2015; Kempf, 2015). Note that execution features are generally derived using some form of moving window to allow for features to quantify both the longer-form impact in addition to impacts over micro-periods more relevant for LOB dynamics in high-frequency markets.

Probability of informed trading (PIN) signals quantify the probability that one’s counterparty to a trade possesses asymmetric superior information (Easley, 1996). The presence of informed trading on the LOB creates an adverse selection risk for firms transacting against those with superior information. Various PIN and Volume Probability of Informed Trading (VPIN) metrics are developed. PIN features are derived from a model of strategic interactions on the LOB between a heterogeneous population of informed and uninformed traders with different information sets. Features for PIN are developed using various periods of analysis, and separate features for informed trading predicated on transactions and an extended metric that measures informed trading using limit orders being placed which has relevance for high-frequency markets. VPIN adapts PIN to the high-frequency nature of modern markets which are increasingly based on volume clocks by measuring the real-time level of informed trading, an indicator for order flow toxicity, in a given security (Easley, 2012). VPIN takes a value between zero and one with values closer to zero reflecting a well-balanced market for the security, whilst higher values closer to one indicate a significant imbalance in the market and higher probability of order flow toxicity. High order flow toxicity is manifest in securities with high VPIN levels and can serve as a proxy for the probability that liquidity is fleeting in nature and may evaporate causing significant market volatility. Several features are developed for VPIN using various window sizes and procedures for measuring volume order imbalances.

Non-primary market and Dark Pool signals are also derived based on trading on the lit and dark order books of BATS, Chi-X and Turquoise for the relevant securities in the sample. We develop features for the RNNs trained in this chapter relating to trading volumes, order flows and auto-correlations with the primary market of the non-LSE order books to aid in predictions of how the LOB dynamics of the LSE will evolve. By analysing how the order book dynamics are shifting on non-LSE order books, one can assume that phenomena extracted from this analysis will have an impact to some degree on the LSE order book.

5.5 Model Design & Evaluation

The objective of this chapter is to implement an optimised Recurrent Neural Network (RNN) architecture capable of modelling the limit order book (LOB) dynamics of UK equity markets. RNNs are highly expressive models capable of building abstract hierarchical feature representations that capture temporal dependencies between LOB phenomena in event-time. The inherent modular nature of the RNN architecture allows various models to be tested to determine configuration settings which best improve the performance of modelling LOB dynamics. We traverse the infinite model variant space by implementing a heuristic grid search experimental design to first select then evaluate various elements of the RNN architecture. The results aggregated over the experiments performed will inform the *optimal* configuration of the RNN model. A recent innovation in optimal network evaluation that we consider is automated search techniques which have recently been proposed as a way of discovering novel network elements that improve performance by using reinforcement-learning based search methods (Ramachandran, 2018).

In this section, an evaluation of various RNN architectures is performed within a comparative framework to isolate the beneficial design elements, components, configurations and hyper-parameters of the neural network that more accurately relate inputs to outputs. This follows the consensus scientific approach adopted by deep learning practitioners which involves empirically evaluating architectures to determine the optimal model design. The focus of this section is an evaluation of the RNN models when trained on the FTSE100 dataset for all sample securities to predict the next action at the LOB, defined as Experiment A in the previous section. The RNN design evaluation involves an Ablation Analysis (Fawcett, 2016) procedure that initialises with an expert-defined 'source' RNN configuration before analysing the impact on the RNN's performance from iteratively modifying parameter settings that lead to a 'target' model configuration. We first posit a source RNN, referred to as the 'baseline' model in this instance. The baseline model defines the configuration, individual components and hyper-parameters of a neural network, providing a basis from which single components can be augmented to test for their individual impact on performance.

Baseline RNN models use techniques and best practices available in the deep learning literature (Greff, 2017; Lipton, 2015), before evaluating each element of the network including its configuration, components, hyper-parameters and network architecture. The evaluation is designed to inform improvements to the RNN's design that result in a more accurate *optimised* model of the LOB dynamics. The *baseline model* developed in this chapter is a deep recurrent neural network composed of Gated Recurrent Units (GRU) (Cho, 2014) with two-layers comprising 32 GRU neurons each. The model activates neurons in the hidden layers using Exponential Linear Units (ELU) bounded in the negative region with alpha α equal to 0.1 (Clevert, 2015), whilst output layers use softmax activations. The network is optimised using an Adam adaptive optimisation algorithm with learning rate a of 0.01, β_1 of 0.9, β_2 of 0.95, and schedule decay of zero (Kingma, 2015). The baseline RNN is trained with mini-batches of 256 samples, within a cross-entropy cost function framework. Additionally, layer normalisation (Ba, 2016), dropout layers (Srivastava, 2014) and recurrent dropout layers (Gal, 2016) are attached to each hidden layer with dropout rates of 0.2 for all neuronal connections. Networks are initialised using orthogonal weight matrix initialisation (Saxe, 2014). Furthermore, the RNN uses a 'lookback' period of 150 events which captures temporal dependencies along this dimension. Finally, the data is split using three days of trading for the training dataset, one for the validation dataset and one for the test dataset, with early stopping techniques employed.

Experimental evaluation of the network configuration is performed by perturbing single components of the model before testing the impact of the elemental augmentation on the RNN's ability to accurately model LOB dynamics in a way that generalises well to unseen data. Results of this analysis help derive recommendations for how the baseline RNN's design can be improved and optimised to explain movements in the LOB. Design *components* and *configurations* are evaluated along three themes. First, the architecture of the model is evaluated by testing the choice of recurrent neuronal architecture, activation function, bias terms, dropout regularisation and lookback periods. Next, the dimensionality and depth of the RNN is evaluated to determine the most effective network design. Finally, the optimisation process when training the RNN is evaluated, including the effectiveness of cost functions, optimisation algorithms, and batch size parameters. For each of these components a general experimental evaluation is first performed to determine the best element and hyper-parameter settings using a grid search, to build the optimal RNN that maximises the performance of predicting LOB dynamics.

5.5.1 Architecture Evaluation

This section first evaluates the performance of various RNN neuronal architectures at predicting the LOB dynamics of UK equity markets using the FTSE100 dataset. Three models introduced in Section 5.2.4 are evaluated including the basic Elman RNN (Elman, 1990), the Long-Short Term Memory (Hochreiter, 1997) and the Gated Recurrent Unit (Cho, 2014). The simple Elman RNN, the LSTM and the GRU used in this section are designed the same way as the baseline model. LSTMs and GRUs implement an innovative RNN design with recurrent connections within the neurons memory components that solve for the failure of Elman RNNs to impound temporal dependencies, as gradients of past states 'vanish', leading to information failing to backpropagate through the network when updating learnable parameters. The LSTM architecture evaluated in this section follows Graves (2005) by utilising a series of gated connections, a recurrent internal memory cell and peepholes connections between the memory cell and internal auxiliary gates. Forget gates are initialised to one (Sutskever, 2013) and internal memory cells and hidden states to zero. GRU RNNs simplify the LSTMs' gated architecture and uses an adaptive control mechanism, rather than an internal memory cell as is the case with LSTMs, to modulate how information propagates through the network (Cho, 2014; Chung, 2014). Similar to LSTMs, the deep GRU tested follows the 'baseline' model configuration and the initialisation and training process of Cho (2014).

The evaluation of the neuronal architecture is reported in Table 5.5.1. The results are explicit in that the GRU neurons work best at predicting the next trader action at or within the BBO for the FTSE100 dataset. RNN models trained with GRU neurons and maintaining all other baseline configurations attain the highest inference accuracy of 63.53%. This follows from their relatively higher predictive accuracy in relation to the training set, where they average a 1.58% higher training accuracy than LSTM and the validation dataset in which GRUs outperform the nearest model, Elman RNNs, by 0.54% on average. Interestingly, the GRU models take longer to train than the LSTMs which is counterintuitive given that LSTMs have more trainable parameters in the form of more complex gating connections. As expected, the Elman RNN is trained the quickest on a per epoch basis, taking 40% less time to perform an epoch than RNNs trained with GRU neurons.

For robustness, the LSTM was analysed when individual gates of the blocks were removed which saw a significant reduction in performance for the RNN. Omitting the forget gate from the LSTM architecture has the largest negative impact on results whilst removing the output gate actually has a relatively minor impact as the hidden state propagates directly from the memory cell before undergoing a non-linear activation, which aligns with the empirical literature (Jozefowicz, 2015; Greff, 2017).

GRU RNNs trained on the FTSE100 dataset present the most effective neuronal architecture for capturing abstract representations and temporal dependencies necessary to predict how the LOB evolves over time. Given the results, we maintain the GRU architecture in the optimised model.

RNN Models	Training Set		Validation Set		Test Set		Speed (Sec)
Architecture	**Accuracy**	**Loss**	**Accuracy**	**Loss**	**Accuracy**	**Loss**	**Per Epoch**
Elman RNN	59.31%	0.973	60.96%	0.920	59.24%	0.942	60.4
LSTM	60.42%	0.967	60.55%	0.928	60.43%	0.921	83.4
GRU	62.00%	0.911	61.50%	0.911	63.53%	0.831	84.9

Table 5.5.1 – Evaluation of Elman RNN, LSTM and GRU architectures trained on the FTSE100 evaluation dataset. Each RNN is trained using relevant elements of the baseline model, just with the neuronal architecture augmented. Performance is assessed along dimensions of predictive accuracy during training, validation and inference, in addition to computational speed.

The evaluation of the non-linear differentiable activation function applied to the inter-layer connections for the temporal data samples is performed next. Activation functions referred to in this section relate to the operations performed to *outputs* of RNN neuronal units that feed into the next layer of the deep RNN, thus, they do not represent the activations performed *within* the RNN neuron itself. The design of a suitable hidden layer activation function, ϕ, applicable for RNNs is an active research area with no definitive theoretical principles. The flexible nature of the RNN architecture allows for the designer to choose one of multiple pre-defined activation functions or use a heuristic function tailored to a given task. Considerations of the function's design include its impact on (1) the speed of training the model, (2) the stability of the learning process and (3) the ability to correctly and efficiently propagate errors through the network during learning. The generic hyperbolic TanH activation is compared against three rectifier activation functions – Rectifier Linear Unit (ReLU) (Nair, 2010), Exponential Linear Unit (ELU) (Clevert, 2015), and Scaled Exponential Linear Unit (SELU) (Klambauer, 2017). ReLUs were designed as a piece-wise linear identity function with a zero bound in the negative region to alleviate the risk of vanishing gradients in networks using sigmoidal activations (Hochreiter, 1998). ELUs improved on the ReLU by allowing negative activation using a hard-boundary in the negative region, performing a bias-shift correction to mean centre neurons' activations relative to ReLU, with the exponential smoothing component creating a saturation plateau that allows for a noise robust deactivation state. SELU are self-normalising activations that are similar to ELU with an added scalar component that is learnt by the model to efficiently develop activations that are normalised, centred with mean zero and unit standard deviation, throughout the layers of the network.

The RNNs trained on the FTSE100 dataset to model predictions of the next trader order submission at or within the BBO responded best to networks designed with SELU activation functions. Results are presented in Table 5.5.2. Baseline RNNs with SELU attained the highest test set accuracy of 65.87% with a test loss of 0.877, showing strong predictive and generalisation power. ELU activated networks attained the second highest test accuracy of 63.07% and training accuracy of 61.79%. The results indicate that all activations allow for expressive models to be developed that are capable of finding a vast representation space of learnable functions. The SELU activation has the unique property of performing normalisation implicitly within the activation function by controlling the variance of the network through the modulation of the saturation region, whilst shifting the mean activation by augmenting the learnable negative region boundary α of the network. The results highlight the ability of the SELU activation scalar λ to perturb the influence of layers across the network in a way that allows for normalised activation neurons deep into the network. The scalar learned by the network of 1.08 is in line with the finding of Klambauer (2017) who provide a value of approximately 1.05 as the efficient scalar for optimised networks. The success of the TanH activated network, attaining a test accuracy only 4.73% less than SELU is interesting given the risk of saturated gradients in the deep GRU RNN architectures when activations approach the asymptotes. One explanation for the respectable results of TanH activations is the large sample size of the FTSE100 dataset that has shown to improve convergence and generalisation abilities for neural networks with sigmoidal-activated neurons (Daqi, 2003). TanH networks are also trained relatively quickly, with a training time approximately 3% lower than SELU networks which is expected given the additional complexities of computing the SELU parameters. The low dispersion of accuracy levels for networks trained with different activations aligns with the findings from the ablation analysis performed using DNNs in Chapter 3 to model algorithmic trading strategies. Given the higher performance of SELU we utilise this activation function in the optimised model.

RNN Models	Training Set		Validation Set		Test Set		Speed (Sec)
Activation	**Accuracy**	**Loss**	**Accuracy**	**Loss**	**Accuracy**	**Loss**	**Per Epoch**
TanH	61.77%	0.894	61.74%	0.897	61.14%	0.811	85.9
ReLU	61.60%	0.900	61.87%	0.901	61.28%	0.864	87.8
ELU	61.79%	0.902	61.90%	0.904	63.07%	0.873	87.8
SELU	61.95%	0.911	61.91%	0.905	65.87%	0.877	88.5

TABLE 5.5.2 – Evaluation of RNN activation functions including sigmoidal TanH and rectifier ReLU, ELU and SELU functions. Performance is assessed along dimensions of predictive accuracy during training, validation and inference, in addition to computational speed.

RNNs were also evaluated for the utility derived from including a learnable bias term, $\mathbf{b}$, within the network architecture. Bias terms allow the transfer function in RNNs to be shifted by bias term $\mathbf{b}$ before undergoing non-linearity when passed through the activation function. This provides the RNN with a trainable constant whilst maintaining the shape of the transfer function. The complexity of learnable parameters updated at each iteration is reduced for RNNs when compared to DNNs with very deep architectures, given that weights and biases of RNNs are shared across temporal dimensions. Hence, the question of whether to include a bias term has practical implications for the generalisation capability of networks that utilise recurrent connections. Results of the evaluation are reported in Table 5.5.3. For the sample securities analysed, baseline RNNs trained with a bias term attained an average test accuracy of 61.88%, whilst models trained without a bias term had an accuracy of 62.11%. Furthermore, there was minimal distinction between the models without the bias term computing an average training and validation accuracy 0.11% better for each than networks with the bias term.

The purpose of this chapter is to model LOB dynamics by predicting how prices evolve given the state observed by heterogeneous traders, defined through a series of LOB features. Thus, the addition of a bias term introduces a parameter that provides no utility to this analysis given that the term is likely to bias the synaptic weights connecting neurons. As an additional note, the inclusion of layer normalisation layers in our network placate the need for including a bias term given the inherent normalising nature of these layers that implicitly applies bias adjustments based on the covariance of the neuron's input (Ba, 2016). In summary, there is no evidence that including bias terms in an optimal RNN is necessary given the nature of the machine learning problem this chapter seeks to solve and the stronger performance of RNNs trained with no bias terms.

RNN Models	Training Set		Validation Set		Test Set		Speed (Sec)
Bias Neuron	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss	Per Epoch
True	61.56%	0.910	61.54%	0.908	61.88%	0.874	85.7
False	61.67%	0.896	61.43%	0.913	62.11%	0.893	85.0

Table 5.5.3 – Evaluation of performance when including or not including bias neurons in the RNN architecture. Performance is assessed along dimensions of predictive accuracy during training, validation and inference, in addition to computational speed.

The regularisation technique of dropout (Srivastava, 2014) and recurrent dropout (Gal, 2016) with various dropout rates, p, are evaluated for their impact on the accuracy and generalisation of the RNNs trained on the FTSE100 dataset to predict the next trader submission at or within the BBO. RNNs trained on complex datasets such as the FTSE100 data tend to overfit the non-linear mapping function to the training dataset. *Standard dropout* is implemented between hidden layers of recurrent neurons to prevent neurons developing complex neuronal co-adaptations between abstract feature representations learnt during network training that leads to overfitting and poor generalisation. Furthermore, we use *recurrent dropout* by randomly dropping connections along the temporal dimension of the RNN.

Results for the dropout evaluation are presented in Figure 5.5.1 and Table 5.5.4. The graphs on the left of the figure analyse the training and generalisation accuracy for networks trained with standard dropout rates of 0, 0.2, and 0.6, whilst the graphs on the right evaluate recurrent dropout using the same dropout rates. Evidently, dropout performs well at breaking co-adaptation patterns in the data, reducing the risk of overfitting, through the application of a random dropout mask to both the input and temporal connections in the RNN. One can note from the results that networks trained with standard dropout with p 0.2 attain the highest training, validation and test accuracy of 63.18%, 61.69%, and 64.17%, on average across the samples after 20 epochs of training. Networks trained with higher dropout rates of 0.6 fail to converge to an agreeable region of the error surface, resulting in a test accuracy of only 59.92%, nearly 5% lower than networks with 0.2 dropout. Interestingly, RNNs trained with no standard dropout perform relatively well with a validation accuracy only 0.98% less on average than networks trained with 0.2 dropout rates. The empirical results suggest that applying standard dropout to the hidden layer input connections of the RNN alleviates some of the concerns regarding overfitting, a common issue in deep networks. These results align with research inferring that dropout applied to non-recurrent connections can be effective by itself even when not applied to the non-dynamic or recurrent components of the RNN (Bayer, 2013; Pham, 2014; Zaremba, 2014).

Implementing dropout layers in recurrent connections of the RNN is evaluated next. This thesis utilises techniques based on Bayesian approximation inference developed by Gal (2016) and random dropout to the memory component of the network along temporal dimensions used by Moon (2015) to minimise the risk of dropout in recurrent connections causing instabilities in the RNN model. The results of this analysis are presented, along with the standard dropout results, in Figure 5.5.1 and Table 5.5.4. Minor improvements of 0.24% in test accuracy on average are noted by applying recurrent dropouts of 0.2 against networks trained with no recurrent dropout layers. Networks trained with 0.6 dropout seem to extricate too many recurrent connections from the parameter manifold, perhaps breaking neuronal co-adaptations but also failing to allow the network to learn representable functions and patterns in the data with a decent degree of accuracy. The results demonstrate the potential negative implications from naively implementing recurrent dropout.

RNN Models		Training Set		Validation Set		Test Set		Speed (Sec)
Dropout	Rate	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss	Per Epoch
Standard	0.0	61.71%	0.896	61.63%	0.896	62.12%	0.888	86.6
Standard	0.2	63.13%	0.876	62.61%	0.879	64.17%	0.845	85.4
Standard	0.6	57.91%	0.984	58.82%	0.959	59.92%	0.920	86.9
Recurrent	0.0	62.65%	0.885	60.40%	0.937	63.34%	0.855	86.5
Recurrent	0.2	63.18%	0.880	61.69%	0.913	63.58%	0.829	86.2
Recurrent	0.6	61.86%	0.901	60.25%	0.931	58.00%	0.917	86.3

Table 5.5.4 – Evaluation of applying dropout layers with varied dropout rates to the *standard* input components of the network (input to hidden to output connections), and the *recurrent* components of the network (memory components along temporal connections). Note that a *recurrent* dropout rate of 0.2 (baseline model) is held constant for all the *standard* dropout experiments, while for the *recurrent* experiment a *standard* dropout rate of 0.2 is maintained. Performance is assessed along dimensions of predictive accuracy during training, validation and inference, in addition to computational speed.

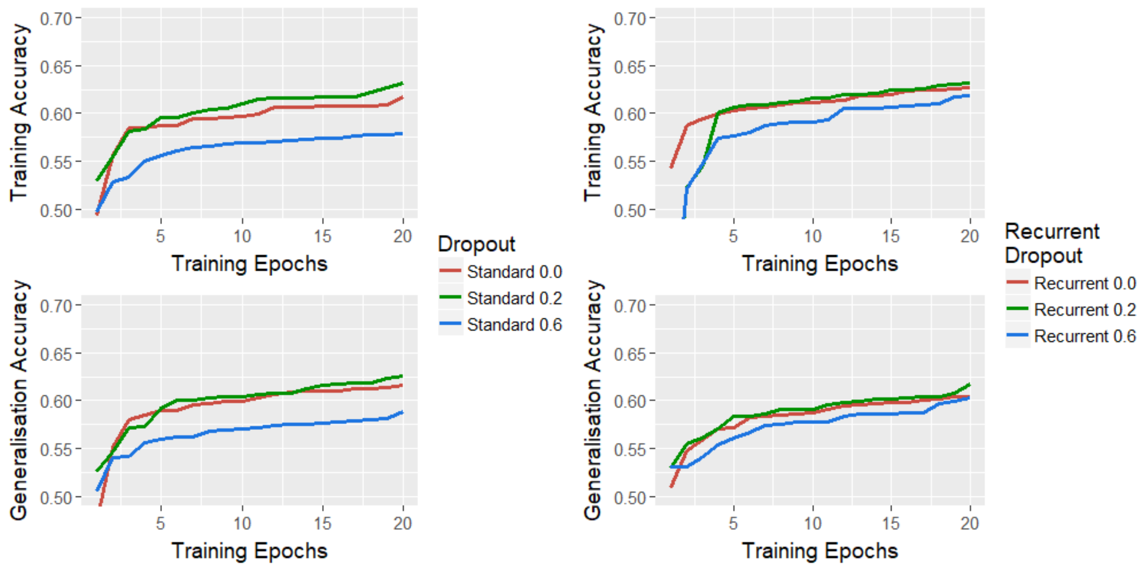

Figure 5.5.1 – Comparative performance of dropout masks for standard input and recurrent connections. Figures on the left measure the training accuracy and generalisation accuracy when augmenting standard dropout rates, whilst figures on the right measure accuracy when the recurrent dropout rate is augmented.

The lookback period represents the period length in event-time over which the RNN maintains the 'state' of the network. Temporal dependencies stored within the RNN are inextricably linked to the lookback period. Whilst the RNN implicitly learns from all samples in the network, given that the hidden state and memory in an Elman, GRU or LSTM network is iteratively updated over event periods, the lookback parameter influences the *magnitude* of the impact from dependencies in the distant past. This section evaluates the baseline RNN trained with a lookback period of 150 events by augmenting this number and training a new RNN to determine how performance is impacted. The lookback periods are evaluated using 100, 150, 200 and 400 events.

From the results reported in Table 5.5.5, one can see that utilising the past 200 events to update the hidden state of the RNN appears to result in a better trained network than the alternatives. Test accuracy reaches 65.25% on average with a commensurate categorical cross-entropy loss of 0.8. However, RNNs trained with lookback periods of 150 have the best training accuracy, whilst the validation accuracy is highest for RNNs trained with 100 period lookbacks. Evidently, the results when using a lookback of 400 events detracts from the performance of the network, potentially inducing noise into the optimisation process from temporal dependencies too far in the past to have any *significant* impact on current events. Thus, RNNs trained with a lookback period between 100 to 200 events appears to be the most effective at capturing temporal representations from which one can predict the next trader action at or within the BBO. We select the median value of 150 periods as the relevant lookback period for the baseline model.

RNN Models	Training Set		Validation Set		Test Set	
Lookback	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss
100	59.22%	0.880	62.72%	1.028	62.66%	0.888
150	61.29%	0.893	61.79%	0.932	64.98%	0.806
200	60.80%	0.908	61.69%	0.957	65.25%	0.800
400	57.62%	0.960	59.13%	0.941	59.87%	0.944

Table 5.5.5 – Evaluation of lookback period lengths representing the number of events incorporated into each temporal training iteration of the RNN. Performance is assessed along dimensions of predictive accuracy during training, validation and inference.

5.5.2 Depth & Dimensionality Evaluation

RNNs with deeper and wider architectures are capable of learning more abstract hierarchical representations of both the feature and hypothesis space given the higher capacity in the network. This section evaluates the performance of RNNs trained with various layer and neuron configurations to draw inferences of how the depth and dimensionality of the network can be improved when training networks to predict price dynamics of the FTSE100 dataset. Modern RNNs can be trained and optimised with a very high number of parameters as networks become deeper and wider, though with the trade-off of higher computational cost, proneness to overfitting, and risk of poor generalisation to unseen data. Theoretically, a fixed size RNN with recurrent connections is capable of computing any representable function under certain assumptions, including full network connectivity, a finite number of recurrent connections, and sigmoidal activated neurons (Siegelmann, 1995). In this way, RNN systems are Turing complete, capable of simulating any arbitrary Turing machine for a given input. However, building an RNN approximator that meets all these properties is impractical in real-world applications and rarely possible in experimental domains. Thus, developing deep and wide networks is both practical, computationally possible, and may result in more effective models that can generalise outside the training data. This section presents an evaluation of RNNs' capabilities of learning more complex functions and improve predictive performance as architectures become deeper and wider.

The depth of the RNN is evaluated using the baseline model with augmented configurations of one, two, three, four and five hidden layers, whilst maintaining the GRU architecture and other facets of the baseline. Performance results for training accuracy, generalisation, testing and convergence are reported in Table 5.5.6. Interestingly, the results follow an inverted u-shape with RNNs trained with one hidden layer of recurrent neurons attaining an average test accuracy of 52.78%, networks with three layers attaining an increased average 66.8% and network with five layers a 61.87% test accuracy. The depth efficiency of RNNs is referred to as an expressive power given that a reduction in layers results in an exponential decrease in the parameters and capacity of the network. Thus, shallower networks with fewer parameters seem to do well at memorising relationships in the data causing gross overfitting to the training dataset, evident by the single layer training accuracy of 58.73% being significantly higher than the validation accuracy of 52.87% at convergence. These shallow networks are not able to develop a complex representation space that allows an algorithm to learn general patterns and relationships applicable for making predictions in the real-world domain, being unable to compete with the expansive generalisation abilities and predictive power of deeper networks that can learn more abstract features, as evident in the results presented. Models trained with three layers seem to attain the best results with the highest test accuracy and validation accuracy of 62.71%. However, as prescribed by the u-shape in the results, five-layer networks whilst still exhibiting good convergence properties do not attain the accuracy levels of the three-layer network. One potential explanation could be that the level of abstraction required, along both temporal dimensions and vertically through the network from input to hidden to output layers, has a specific threshold beyond which the vast majority of hypothesis space of interest has already been formulated. Depth and computational cost are positively correlated as per the results in Table 5.5.6. Single layer RNNs take on average 60.90 seconds to complete one training epoch, though they converge to a sub-optimal solution, whilst five-layer networks take 23% longer to complete the same training objective with the difference that they output a more optimal mapping function solution.

RNN Models	Training Set		Validation Set		Test Set		Speed (Sec)
Depth (Layers)	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss	Per Epoch
One	58.73%	0.978	52.87%	1.084	52.78%	1.046	79.1
Two	56.74%	0.960	61.49%	0.903	66.55%	0.811	83.7
Three	62.27%	0.958	62.71%	0.907	66.80%	0.807	87.3
Four	56.22%	0.968	61.40%	0.892	65.73%	0.813	96.6
Five	58.46%	1.009	60.26%	0.912	61.87%	0.912	96.9

Table 5.5.6 – Evaluation of the depth in the baseline RNN models averaged across evaluation sample datasets. GRU baseline RNNs are trained with varied number of layers, each layer has 32 recurrent neurons. Performance is assessed along dimensions of predictive accuracy during training, validation and inference, in addition to computational speed.

Widening the layers of the network by adding GRU neuronal parameters to the hidden layers of the RNN architecture, has been shown to increase the capacity of the network, improving predictive results for some machine translation tasks (Graves, 2013; Liang, 2015). This section evaluates the utility of re-configuring the dimensionality of the hidden layers of the baseline model of the GRU RNN architecture by analysing networks trained with 1, 8, 16, 32, 64, 128 and 256 recurrent neurons in each hidden layer.

Validation curves presented in Figure 5.5.2 and results in Table 5.5.7 indicate a similar relationship between dimensionality and accuracy that was evident in the depth analysis, that is, an inverted u-shape between the increased capacity and predictive power. Three-layer GRU RNNs trained on the FTSE100 dataset attain the highest test accuracy of 66.08% when trained with 64 neurons on average across the evaluation sample datasets. As one would expect, RNNs trained with a single GRU unit in each hidden layer are not expressive enough to capture a large hypothesis space of learnable functions, evident by the test accuracy of 46.72%. Interestingly, the wider networks of 256 recurrent neurons are not as competitive as the 64 neuron networks, attaining a lower training, validation and test accuracy. Furthermore, these wider networks take 76% longer to cycle through a single epoch of the training dataset than networks built with 64 GRU cells.

RNN Models	Training Set		Validation Set		Test Set		Speed (Sec)
Width (Neurons)	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss	Per Epoch
1	43.09%	1.143	45.63%	1.084	46.72%	1.068	36.9
8	59.22%	0.941	60.62%	0.921	59.80%	0.918	39.1
16	57.27%	0.968	61.57%	0.898	62.81%	0.864	76.4
32	58.55%	0.972	60.34%	0.919	65.91%	0.814	83.9
64	60.64%	0.902	63.50%	0.895	66.08%	0.841	91.9
128	58.56%	0.945	60.87%	0.905	63.10%	0.856	111.1
256	55.89%	0.984	57.61%	0.949	63.38%	0.863	161.2

Table 5.5.7 – Evaluation of width in the RNN model architecture averaged across evaluation sample datasets. GRU baseline RNNs are trained with varied number of neurons per layer. Performance is assessed along dimensions of predictive accuracy during training, validation and inference, in addition to computational speed.

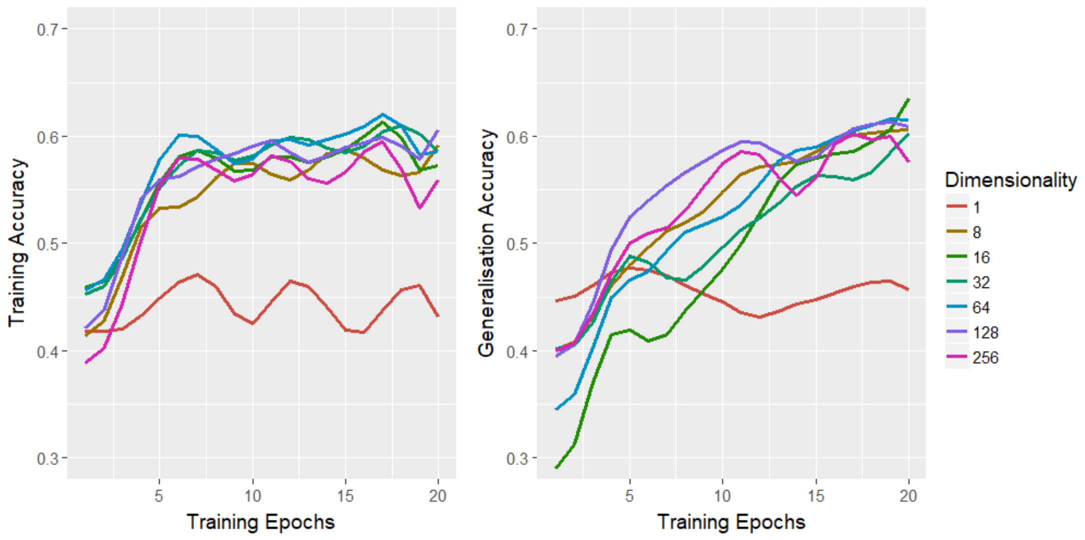

Figure 5.5.2 – Comparative performance of RNNs trained with a different number of neurons per hidden recurrent layer. Training and generalisation accuracy measures performance during training on the training and validation dataset, respectively.

The results in this section demonstrate that an equilibrium point exists when configuring an RNNs depth and width that is agreeable with the learning properties of the FTSE100 dataset. This is evident by the inverted u-shape between the capacity of the network, influenced by the size of the configurable parameters, and the predictive and generalisation power of the RNN. Minimal theoretical and empirical work has analysed the influence of depth and dimensionality on the performance of RNNs, with the number of neurons and layers generally tuned using some expert-defined heuristic. For the optimised model it seems pertinent to increase the depth of the RNN to three layers and increase the dimensionality to 64 recurrent units to extract more complex abstractions from the features without widening the functions representation space too large so as to cause instability in the model.

5.5.3 Network Optimisation Evaluation

Predicting the actions of a disparate set of traders on the LOB using RNN GRU models is a complex task given the various actions a trader can take depending on the state of the market and LOB. Critical to the RNN employed in this chapter to model this relationship is the choice of a robust cost function that accurately measures the errors of the RNN predictions, to inform efficient updates to the internal learnable parameters through the optimisation algorithm. Cost functions provide a framework to compare the distributions of the training output values of the model with the true output values. This requires an easily differentiable mathematical form that allows gradient descent optimisation algorithms to efficiently update network weights in the direction of good representational areas during iterative training. This section performs an ablation evaluation of the baseline RNN GRU architecture by modifying the cost function used. The empirical performance of RNN models trained using categorical cross-entropy (CE) cost functions is compared against those trained with a Log CosH and a Mean Square Error cost function. The formulas of the comparative cost functions are outlined in Section 3.1.6.

The results of the analysis are reported in Table 5.5.8. Evidently, the baseline RNN models that utilise a cross-entropy cost function to model the loss of the network perform marginally better than networks that use Log CosH or Mean Square Error. Cross-entropy configured RNNs attain an average test accuracy of 61.07% which is a 2-3% improvement on the other cost functions analysed. Furthermore, the baseline validation accuracy of 62.04% improves on RNNs trained with Log CosH, which attain an average 61.55% validation accuracy and mean square error with an average 61.87% accuracy. Neither of the cost functions analysed struggle to learn effectively or spur convergence issues. The use of cross-entropy cost functions to train neural network models is prevalent in the deep learning literature (Janocha, 2017), given that it provides an interpretable error that motivates the optimisation algorithm to minimise divergence between the true and predictor output probability distributions. RNNs trained using a cross-entropy cost function framework attain the highest training accuracy, with good convergence properties. However, this does come at a minimal cost of slower computation with the average training per epoch speed of 87.8 seconds.

One significant advantage of the cross-entropy vis-à-vis the mean square error cost function is its inherent differentiable structure which omits the derivative of the activation function in hidden layers, $\phi'(\mathbf{z})$. For cross-entropy this term gets cancelled out when applying the chain rule during learning. Thus, whilst both cross-entropy and mean square error cost functions are positive for all real value inputs, and the cost functions tend towards zero as the predictors closely approximate the actual output values, cross-entropy has the added advantage of propagating larger errors through the network resulting in faster learning and convergence (Goodfellow, 2016). Evidently, the selection of categorical cross-entropy as a cost function is consistent with developing an optimal RNN that accurately predicts the LOB dynamics. Furthermore, the cross-entropy cost function is conducive to propagating errors efficiently backwards through temporal dimensions when applying backpropagation through time algorithms. The optimised model maintains the cross-entropy cost function.

RNN Models	Training Set		Validation Set		Test Set		Speed (Sec)
Cost Function	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss	Per Epoch
Log CosH	61.10%	0.060	61.55%	0.908	58.62%	0.063	85.8
Mean Square Error	61.83%	0.128	61.87%	0.128	58.42%	0.134	85.7
Cross-Entropy	61.90%	0.917	62.04%	0.059	61.07%	0.875	87.7

TABLE 5.5.8 – Evaluation of RNN cost functions applied using the baseline GRU architecture. Performance is assessed along dimensions of predictive accuracy during training, validation and inference, in addition to computational speed.

Evaluation of the RNN optimisation algorithm is performed next. Comparisons between various first-order adaptive gradient descent-based optimisation algorithms including AdaDelta (Zeiler, 2012), Adaptive Moment Estimation (Adam) (Kingma, 2015), AdaMax (Kingma, 2015) and Nesterov Adam (NAdam) (Dozat, 2016) are compared against vanilla stochastic gradient descent (SGD) and Nesterov Accelerated Gradient (NAG) (Nesterov, 1983; Sutskever, 2013). Theory and empirical literature on each model are presented in Section 3.1.8. Each optimisation first undergoes a grid search hyper-parameter evaluation to determine the most effective design of each algorithm at solving the machine learning problem. Results from the comparative evaluation are summarised in Table 5.5.9, with generalisation curves for the training accuracy and generalisation accuracy of the algorithms depicted in Figure 5.5.3.

Interestingly, vanilla SGD trained with a learning rate α of 0.1 appears to train the RNN with decent predictive accuracy though only converging with a test accuracy 11.39% less than the best performing algorithm, AdaMax. Vanilla SGD also takes significantly longer to reach a convergence state which is expected due to the inherent architectural prerequisite that the fixed non-adaptive learning rate is set low to allow it to search over the error surface before converging slowly and inefficiently towards a local minimum (Bengio, 2012). For comparison purposes, a non-adaptive Nesterov momentum optimisation algorithm is fitted to the RNN which attains a slight improvement over the vanilla SGD but is not competitive with the adaptive algorithms. The multi-dimensional feature space derived from the FTSE100 LOB dataset indicates that the cross-entropy cost function will likely have a complex error surface. Applying Nesterov momentum with the Nesterov accelerated gradient (NAG) optimisation algorithm (Nesterov, 1983; Sutskever, 2013) allows for this error surface to be traversed with higher efficiency than vanilla SGD. The results indicate that the addition of the momentum term and performing a Nesterov correction to the gradient perturbation of the learnable parameters improves the accuracy of the predictions vis-à-vis vanilla SGD.

Adam and its two extensions, AdaMax and NAdam, attain superior training, generalisation and convergence performance for FTSE100 CLOB RNNs trained on the evaluation dataset compared to vanilla SGD and other adaptive algorithms. Adaptive optimisation algorithms with dynamic learning rates have been shown to be most effective for training models with sparse input feature representations (Ruder, 2017), a trait of our FTSE100 CLOB dataset and the generated features. Adam-type optimisation algorithms conduct natural annealing using an inversion parameter that takes the decaying squared historical gradients' second moment over a given period, $\hat{V}_i^m$, which is smoothed and augmented by incorporating a momentum component, $\widehat{M}_i^m$, that takes the exponentially decaying mean of past iterations' first moment gradients, which are both bias-corrected given they are initialised towards zero. All three iterations of the adaptive moment algorithms developed by Kingma (2015) – Adam, NAdam and AdaMax – attain similar performance with validation accuracy all over 60% at convergence or after 20 epochs, whichever occurs first.

Evidently, AdaMax optimisation performs most effectively at modelling the LOB dynamics for the FTSE100 dataset. Models trained on the FTSE100 dataset using AdaMax attain an average test accuracy of 64.38%, validation accuracy of 61.66% and test accuracy of 61.54%. This motivates the decision to utilise AdaMax in the optimised model with optimised hyper-parameters to improve training accuracy and predictive performance.

RNN Models		Training Set		Validation Set		Test Set		Speed (Sec)
Optimisation	Hyper-Parameters	Accuracy	Loss	Accuracy	Loss	Accuracy	Loss	Per Epoch
SGD	$\alpha = 0.1$	57.76%	0.989	54.73%	1.009	52.99%	1.010	85.5
Nesterov Mom.	$\alpha = 0.1, \gamma = 0.8$	57.74%	0.963	59.17%	0.938	55.97%	0.959	85.8
AdaDelta	$\alpha = 2, \gamma = 0.99$	54.91%	1.028	54.94%	1.037	53.98%	1.038	87.3
Adam	$\alpha, \beta_1, \beta_2 = 0.1, 0.9, 0.95$	61.03%	0.908	60.52%	0.958	60.01%	0.925	87.2
NAdam	$\alpha, \beta_1, \beta_2 = 0.1, 0.9, 0.95$	61.29%	0.874	60.52%	0.958	61.80%	0.923	85.4
AdaMax	$\alpha, \beta_1, \beta_2 = 0.1, 0.9, 0.95$	61.54%	0.927	61.66%	0.928	64.38%	0.838	85.9

Table 5.5.9 – Evaluation of optimisation algorithms using a comparison between various optimisers trained with hyper-parameters tuned using a basic grid search. Performance is assessed along dimensions of predictive accuracy during training, validation and inference, in addition to computational speed.

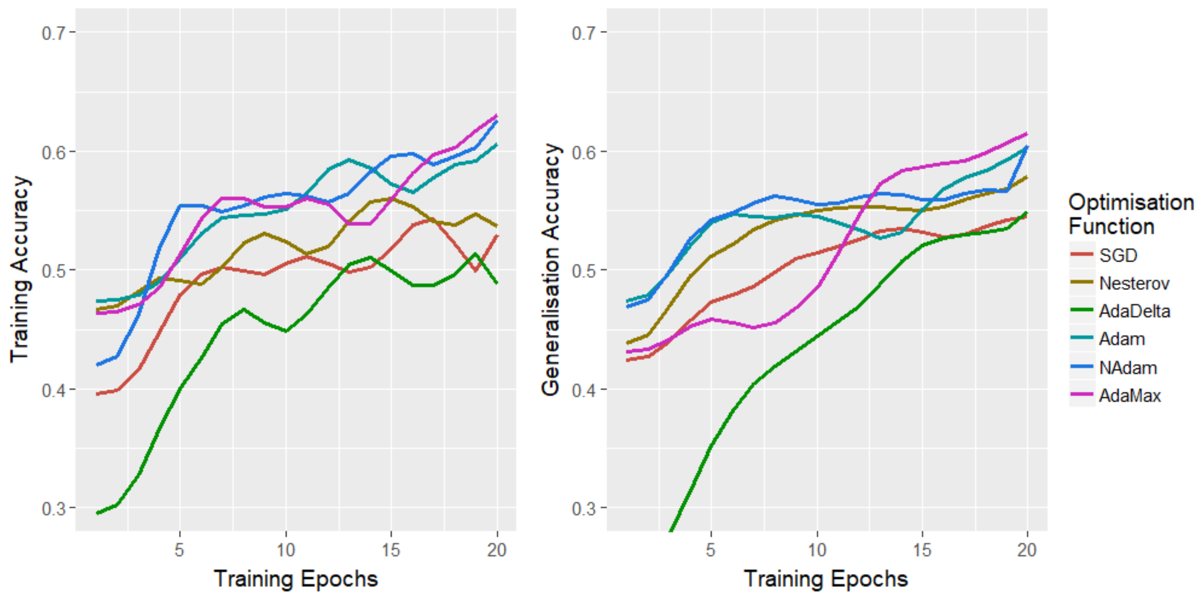

Figure 5.5.3 – Graphical comparison of training accuracy and generalisation accuracy for RNNs trained using augmented optimisation algorithms. Note that the hyper-parameters selected for each optimisation algorithm are defined in Table 5.5.9.

The final experiment evaluates the performance of RNNs trained with various batch sizes. Optimal mini-batch sizes generally differ depending on the machine learning problem, input feature dataset structure or learning objective, which motivates this evaluation. RNNs are evaluated by maintaining the baseline architecture and augmenting the batch size parameter values to 128, 512, 1024 and 2048 mini-batch sample sizes. These hyper-parameter values represent common batch sizes used in deep learning literature (Bengio, 2012; Krizhevsky, 2012; He, 2015) and follow the heuristic of using exponents n with base 2, that is, 2^n. The evaluation utilises mini-batch methods of gradient descent optimisation, rather than purely single-sample stochastic or batch gradient descent methods, given their computational simplicity, stable convergence properties and regularising effect on the network.

Results of the evaluation reported in Table 5.5.10 show that of the batch sizes evaluated, RNNs trained to model the LOB dynamics as defined in Experiment A, attain the highest predictive performance with a batch size of 1024. RNNs trained with mini-batches of 1024 samples attain a test accuracy of 62.39% and validation accuracy of 61.27%, the highest of all models analysed. These results are in line with expectations that the performance of an RNN as a function of the mini-batch size generally follows an inverted u-shape. This phenomenon occurs as RNNs trained with very large mini-batches generally converge to sharp minimisers, with larger positive eigenvalues of the training function (Keskar, 2017). Thus, mini-batches that are too large result in the optimisation algorithm focussing on regions of the error surface closest to the initialised weights, leading to a sub-optimal solution. Networks trained with large mini-batches also find it difficult to escape saddle points (Dauphin, 2014). In contrast, very small batch sizes induce too much stochasticity into the optimisation process, leading to extreme updates of the weights that create instability in the network. There exists an equilibrium mini-batch size that allows the RNN to converge to flatter minimisers, with small positive eigenvalues, due to inherent noise in the gradients (Keskar, 2017). Noise in these networks trained with an equilibrium mini-batch size acts as a quasi-regulariser that enables a wider and potentially gainful explorative search over the error surface given the induced oscillations in the optimiser algorithm from the stochasticity inherent in samples. Batch sizes of 1024 are used for the optimised model.

RNN Models	Training Set		Validation Set		Test Set	
Batch Size	**Accuracy**	**Loss**	**Accuracy**	**Loss**	**Accuracy**	**Loss**
128	59.75%	0.876	59.27%	0.924	58.11%	0.894
512	60.53%	0.888	60.47%	0.933	60.41%	0.881
1024	61.46%	0.875	61.27%	0.923	62.39%	0.832
2048	60.70%	0.879	59.98%	0.938	61.72%	0.084

Table 5.5.10 – Evaluation of RNN baseline models trained with various mini-batch sizes. Performance is assessed along dimensions of predictive accuracy during training, validation and inference.

5.6 Optimised Recurrent Neural Network

Following from the evaluation conducted in the previous section, we extract insights from the ablation analysis to design and configure an *optimised* Recurrent Neural Network (RNN) architecture for modelling the dynamics of the LOB using the FTSE100 dataset. We use this model to conduct three experiments that predict the LOB dynamics. All RNN models built and experiments performed in this section are programmed using TensorFlow v1.4 (Abadi, 2016) and cuDNN v7.0 (Chetlur, 2014) deep learning engines and primitive libraries. TensorFlow provides a comprehensive open-source deep learning library consisting of Python modules that allow for the efficient construction of computational graphs, which are capable of distributed training and processing in parallel across multiple CPU or GPUs. Computational data flow graphs represent the set of mathematical operations, manipulations and functions performed on and between nodes comprised of tensors, or multi-dimensional data arrays, which for example, can represent an optimisation algorithm. TensorFlow is ubiquitous within practitioner and academic domains for developing and training RNNs, primarily given its ability to perform extensive array manipulations efficiently. Training of the RNN is conducted on an Intel i7 CPU using 32GB RAM and a NIVIDA GPU.

The optimised RNN utilises a Gated Recurrent Unit (GRU) architecture (Cho, 2014) with three interconnected layers comprising 64 recurrent neurons in each to ensure a high degree of feature abstraction and a vast representation space of possible non-linear functions. No bias terms neurons are included. Scaled Exponential Linear Unit activation functions (Klambauer, 2017) are used to apply non-linearity when passing data between layers in the RNN. The output layer of the RNN utilises Softmax activation functions which is pertinent given the categorical nature of the FTSE100 CLOB classifiers. Layer normalisation is applied to each hidden layer with e^{-3} epsilon ϵ, beta β initialised to zero, and gamma γ initialised to one, with no regularisation applied to the LN layers' parameter weights (Ba, 2016). Recurrent dropout layers are also applied to the recurrent elements of the network (Gal, 2016), whilst standard dropout layers are masked over input neuronal recurrent connections between layers (Srivastava, 2014). Both the recurrent and standard dropout layers maintain the same dropout probability p of 0.2 for all layers, following the theoretical benefits of symmetrical dropout masks suggested by Gal (2016). Models are trained using AdaMax, an iteration of the Adaptive Moments Estimation algorithm (Kingma, 2015) with a learning rate α of 0.1, and bias terms β_1 of 0.9 and β_2 of 0.95, with no scheduled decay. The neural network is trained to fit a complex non-linear mapping function, $\mathbf{y}^t = f(\mathbf{x}^t, \mathbf{u}, \mathbf{v}, \mathbf{w})$, by minimising the cross-entropy cost function, $\mathcal{L}^{CE}(\mathbf{u}, \mathbf{v}, \mathbf{w})$, with mini-batch sizes of 1024 for each temporal iteration. Furthermore, a lookback period of 150 events is encapsulated in the model's training process to capture shorter-term temporal dependencies between event periods.

Truncated backpropagation through time is performed during optimisation. RNN hidden state neurons are initialised to zero with kernel weights initialised using the random orthogonal matrix technique with scaled gain factor g of 0.5 (Saxe, 2014). RNNs are trained over 50 epochs. Early stopping techniques are applied using systematic rules that require a trade-off between reducing training time and improving generalisation (Prechelt, 2012). A stopping criterion that halts training when the validation accuracy does not improve after a patience level of 10 epochs is utilised. Note that for the third experiment, Experiment C, a linear output activation and mean square error cost function is applied given the regression nature of the machine learning problem.

RNN models of the LOB dynamics are developed for each separate prediction experiment as defined in Section 5.4. For the purposes of developing the RNN predictive models, a stratified dataset is developed that uses three days of data to train the network, one to validate and one to test, resulting in the partitioning of the FTSE100 data for each security into a training, validation and testing dataset. The experiments are performed over five sample securities extracted from the FTSE100 dataset over the month of October 2017. Training of the RNN is conducted to optimise the learnable parameters, $\boldsymbol{\vartheta} = \{\mathbf{w}, \mathbf{u}, \mathbf{v}\}$, of the network in a way that best approximates the non-linear mapping function, $\mathbf{y} = f(\boldsymbol{\vartheta}, \mathbf{x})$, where $\mathbf{y}$ is the phenomenon of interest, the LOB midprice of the security, and $\mathbf{x}$ is the set of market microstructural features, defined in Section 5.4.1, that are extracted from the LOB to best explain the future price dynamics of the order book.

5.7 Experimental Results

This section presents the empirical results for predicting the LOB dynamics using the optimised RNN model as defined in the previous section. Specifically, we perform three experiments to predict the next order submission at or within the BBO (Experiment A), to predict the next midpoint price movement on the LOB (Experiment B) and to predict the midpoint price at some future period in event time (Experiment C). This section predominately focuses on measuring the predictive performance of the models by computing direct and proxy measures of accuracy, that is, the RNN models' ability to correctly predict the LOB dynamics. Sirignano (2016) suggests that accuracy may be a biased measure when interpreting the models' performance for financial time series data given the significant noise inherent in the financial system. However, we note that the author focussed more on modelling LOB dynamics from a risk perspective by considering order distributions deep into the order book, whilst this chapter concentrates primarily on modelling these dynamics for their relevance to algorithmic trading.

Experiments are performed by first training an RNN for all sample securities before evaluating the ability of the model to predict the LOB dynamics. For example, for Experiment A the variable of interest is the forthcoming trader action at or within the BBO given the state of the LOB. Thus, when evaluating results, a consideration is given to the 'quality' of the model at predicting these actions using various metrics for each of the four possible trader action classes – submit a market or immediate-or-cancel order at the ask, MO.A, at the bid, MO.A, or submit a limit order or iceberg order at the best ask, LO.P0,A, or at the bid, LO.P0.B.

Quality metrics are computed by comparing the positive class (P), which is the true class, and negative classes (N), which are all other incorrect classes. Positive classes may either be true positives (TP) such that they are correctly predicted or false negatives (FN) where a negative class is incorrectly predicted as the positive class. Additionally, negative classes can be broken up into false positives (FP) where the negative class is incorrectly predicted and a true negative (TN) where the negative class is correctly predicted. The following Figure 5.7.1 reiterates the quality metric components introduced in previous chapters.

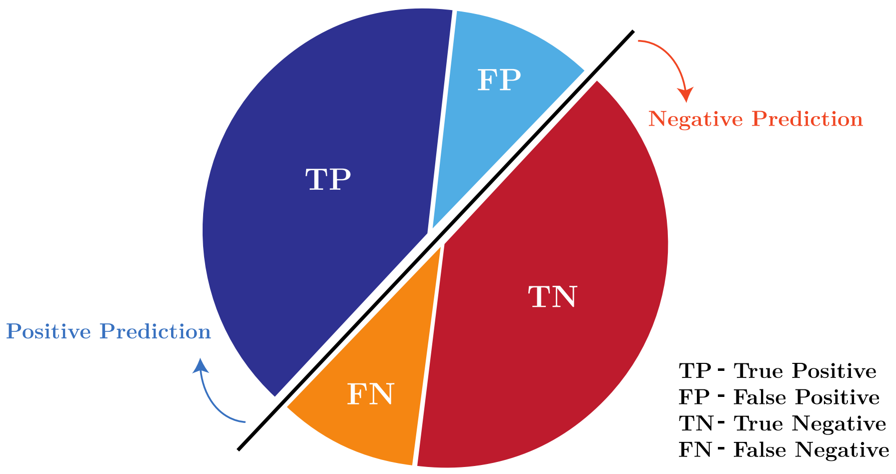

Figure 5.7.1 – Illustration of quality metric components for RNN models predicting limit order book dynamics.

5.7.1 Predicting Trader Actions at or Within the BBO

RNNs are trained to predict the next trader submission action at or within the BBO (Experiment A) with the results of the analysis presented in Table 5.7.1. Metrics are computed after exposing the optimised RNN models to the test dataset after training.

The prevalence metric measures the percentage of occurrence of each action over the sample datasets. Limit orders at or within the BBO are most common with bid and ask orders representing 42.73% and 41.69% of all actions, respectively. Orders that cross the spread as either a market order or immediate-or-cancel order total to 15.58% of all relevant orders in the dataset. Evidently, there is a class imbalance problem as orders are primarily submitted at the BBO with a lower quantum of orders actually transacting with opposite sided liquidity. We overcome this issue by calculating performance using a balanced accuracy metric which takes the average of the sensitivity and specificity measures (Velez, 2008). One can note from the results that the limit orders for the bid and ask are predicted with a high degree of balanced accuracy, averaging 72.65% and 73.36%, respectively, over the FTSE100 dataset. Market orders are more difficult to predict with an average balanced accuracy of 59.68% and 59.55% for ask and bid market orders, though this level is relatively higher than expected. It should be noted that the *observed* accuracy is significantly lower for market order actions but using the balanced metric we can infer a higher predictive accuracy by oversampling these less common actions.

RNN models for Experiment A are next evaluated for each class by their sensitivity and specificity quality metrics. Sensitivity measures the number of correct positive predictions as a percentage of the total number of positive reference classes in the dataset. Conversely, specificity measures the correct negative predictions as a percentage of all negative classes. The RNNs' predictions of the ask limit orders marginally outperform orders on the bid side in terms of sensitivity by 1.78% though has a lower specificity measure by 0.35% on average across the sample securities. The relatively higher sensitivity metric overall for the limit orders indicates that the RNN performs well at predicting BBO actions correctly, with a higher proportion of true positives to actual positive predictions. This result demonstrates that the model performs relatively well at determining when an order is placed in a way that adds liquidity to queues at the BBO, rather than executing a transaction by crossing the spread, which is an important element for a successful model of the LOB dynamics. Further compounding this positive view of the model is the sensitivity quality metric which indicates that market orders are highly successful at predicting true negatives as a proportion of all negatives, with over 94% sensitivity for both types of market orders. Thus, the RNN model correctly rejects observations of the LOB as being conducive to a market order, indicating it is relatively conservative at predicting when traders will cross the spread unless the dynamics of the LOB indicate there is a high probability of this occurring.

Precision metrics provide an overview of how well the RNN model performs at predicting the correct LOB action class without being significantly confused by an array of alternative classes. The result is in line with expectations in that limit orders attain a higher precision score than market orders. Limit orders at the best bid and ask have an average precision score of 68.38% and 67.69% whilst bid and ask market orders have lower precision scores of 30.81% and 27.32%, respectively. This result indicates that market orders are more easily confused with other types of orders studied, which is expected given that market orders are rarer and could easily be confused with a same side order within the BBO or an opposite side market order. One can also see that each limit and market order is on average primarily confused by orders on the same side but opposite type, which is common given the similarity in features driving these actions. Generally, the state of the LOB could be conducive to either a market or limit order on one side of the LOB depending on the algorithmic trading strategy being conducted. For example, a low latency trader conducting a market making or momentum type strategy that views a large order imbalance on the bid side of the order book may seek to place an order *within* the spread, using a limit order LO.P0.B, creating a new best bid. Conversely, a trader utilising an algorithmic trading system with higher latency, such as a trader conducting an execution strategy, may cross the spread immediately by placing a market order, MO.B, without risking the LOB dynamics evolving too quickly and missing an opportunity to buy shares at a fair price. Evidently, LOB dynamics specifically at the BBO are difficult to predict due to the complexities in the disparate algorithmic trader environment, which theoretical equilibrium or stochastic models find difficult to interpret, especially when analysing high-frequency data. Thus, the RNN model uses machine learning algorithms to understand and find patterns within these dynamics, without preconceived assumptions or constraints but rather focussing on the data imputed into the model to form a basis for predictions of how LOBs will evolve.

Finally, an F1 quality metric is calculated by taking the harmonic mean of the precision and sensitivity metrics for each action. The F1 quality metric score is adapted to the multi-class case by setting one class c to the positive, $\mathbf{y} = c$, and all other classes to the negative label. F1 metrics provide a measure of accuracy by weighting the importance of precision over sensitivity metrics using a beta factor, β, which is defined as one in this analysis. Aligning with the above results, the F1 score is highest for limit orders placed at or within the BBO. The performance for both limit order actions is comparable with an F1 score of 68.77% for bid and 69.28% for ask orders at or within the BBO. These scores decline rapidly for market orders with both bid and ask market orders attaining average F1 scores of approximately 26%. F1 scores convey the success of the RNN model at predicting the future trader action at or within the BBO given the current state of the LOB. We conclude that the model attains accuracy rates higher than expected for this task given the significant noise inherent when modelling LOB dynamics in high-frequency markets, as evident in the FTSE100 dataset. However, it must be noted that RNNs are trained *only* on observation samples for these relevant actions and exclude actions further outside the BBO.

	Experiment A (Trader Action)			
Quality Metric	LO.P0.A	LO.P0.B	MO.A	MO.B
Prevalence	41.69%	42.73%	7.81%	7.77%
Balanced Accuracy	73.36%	72.65%	59.68%	59.55%
Sensitivity	70.94%	69.16%	24.99%	23.56%
Specificity	75.79%	76.14%	94.37%	95.54%
Precision	67.69%	68.38%	27.32%	30.81%
F1	69.28%	68.77%	26.10%	26.70%
Balanced Confusability	MO.A	MO.B	LO.P0.A	LO.P0.B

TABLE 5.7.1 – Quality metrics for the trained RNNs to predict the next trader order submission action at or within the BBO (Experiment A). Metrics include prevalence, balanced accuracy, sensitivity, specificity, precision, F1 scores and balanced confusability class. The quality analysis is performed for four actions – MO.A, LO.P0.A, MO.B, LO.P0.B – with results aggregated over the five sample securities. Metrics are computed by first calculating the true positive (TP), false positive (FP), true negative (TN), and false negative (FN) values for each class. True predictions (T) are equal to TP plus FP, whilst Negative predictions (N) are equal to TN plus FN. Quality metrics are defined as: $Prevelance = \frac{Count\ Class}{Total\ Samples}$ $Balanced\ Accuracy = \frac{Sensitivity+Specificity}{2}, Sensitivity = \frac{TP}{P}, Specificity = \frac{TN}{N}, Precision = \frac{TP}{(TP+FP)},$

$F1 - Score = \frac{(1+\beta^2)*Precision*Sensitivity}{(\beta^2*Precision)+Sensitivity}\ where\ \beta = 1$

5.7.2 Predicting Midpoint Price Changes

Performance for RNN models trained to predict the next midpoint price change of each security (Experiment B) is analysed in this section with results reported in Table 5.7.2. Models are trained for each individual security with the quality metrics computed by exposing the optimised RNN models to the test dataset after training and averaging the results. This experiment involves a machine learning classification problem with two possible classes – the next midpoint price is a move 'up' or 'down'. The dataset utilised includes all orders placed at or within the BBO and up to five ticks from the BBO over the course of the sample period. Results are interpreted using accuracy metrics that measure the percentage of the samples that the model interprets as the correct class, which is a proxy for the RNN models' ability to reduce the cross-entropy loss.

From the results, the average RNN model accuracy on the test dataset for predicting the next midpoint price change is 59.44%, whilst for the validation and training dataset the accuracy is 58.84% and 59.43%, respectively. The predictive framework of the RNN appears to perform well given the market microstructural feature representation space extracted from the LOB, **x**, which includes features commonly used as inputs into algorithmic trading strategies. One can note that the prediction of the RNN is higher than 50% which would be expected under a naïve random model of prediction. Passing signals through recurrent connections appears to create a versatile storage of long-term dependencies into the memory of the RNN, allowing the model to interpret contextual and relevant information from the past to inform the prediction of the current LOB dynamics.

The machine learning problem in Experiment B reduces to a binary class problem when modelling whether the next midpoint price change will be higher or lower than the current value. Thus, the results reported measure quality metrics for the 'up' midpoint price movements which is analogous and orthogonal to the results for the 'down' price change. From the results, we can see that 'up' price movements represent 49.92% of the results, with a balanced accuracy of 59.44%. The specificity score of 61.16% is manifestly higher than the sensitivity score of 57.72%, indicating that the RNN model is better at predicting the negative class correctly, 'down' movements predicted in the midpoint price, than the class of interest, 'up' movements in the midpoint price. Both metrics are inherently inverse functions of one another, with one metric increasing inversely proportional to the other decreasing. The F1 metric of 58.7% is computed as the harmonic mean of precision and sensitivity. The relatively higher precision than sensitivity indicates that the RNN model has marginally higher false positives than false negatives when predicting the next midpoint price change on the LOB for the 'up' class. In conclusion, the trained RNN model provides a useful predictor of LOB dynamics by accurately forecasting the next midpoint price change for a significant majority of sample data points. It is important to note that the model is built using *sequential* data of messages exchanged between the LSE and trader participants. Thus, whilst the model can view the current state in event time it is unlikely that any trader is capable of doing so, given varied latency rates and periods of extremely high message flow.

	Experiment B (Midpoint Change)
Quality Metric	**Up**
Prevalence	49.92%
Balanced Accuracy	59.44%
Sensitivity	57.72%
Specificity	61.16%
Precision	59.70%
F1	58.70%

Table 5.7.2 – Quality metrics for the trained RNNs to predict the next midpoint price change (Experiment B). Metrics include prevalence, balanced accuracy, sensitivity, specificity, precision, and F1 scores. The quality analysis is performed for the action 'Up' which refers to the next midpoint price change on the LOB being an increase and is orthogonal to the 'Down' midpoint price change class. Results are aggregated over the five sample securities. Metrics are computed by first calculating the true positive (TP), false positive (FP), true negative (TN), and false negative (FN) values for each class. True predictions (T) are equal to TP plus FP, whilst Negative predictions (N) are equal to TN plus FN. Quality metrics are defined as: $Prevelance = \frac{Count\ Class}{Total\ Samples}$ $Balanced\ Accuracy = \frac{Sensitivity+Specificity}{2}, Sensitivity = \frac{TP}{P},$

$Specificity = \frac{TN}{N}, Precision = \frac{TP}{(TP+FP)}, F1 - Score = \frac{(1+\beta^2)*Precision*Sensitivity}{(\beta^2*Precision)+Sensitivity}$ $where\ \beta = 1.$

5.7.3 Predicting Future Prices

The final experiment performed in this section predicts the future midpoint price of the security after fixed event time horizon of $t + 20$ events conditional on the state of the LOB at t (Experiment C). Prediction of the midpoint price is analogous to predicting the future joint distribution of the best bid and ask prices as a function of the quoted spread (Sirignano, 2016). Two separate analyses are performed in this section. First, we analyse the predictive capabilities of the RNN model trained over the sample securities. Second, we use an approach that combines reinforcement learning (RL) and RNNs by improving the network using a simplified version of the RL 'tuner' algorithm introduced by Jacques

(2017). The technique utilises an off-policy stochastic optimal control method using a Deep Recurrent Q-Network (DRQN) augmented with a mean square error (MSE) based cost function. We slightly modify this approach to be relevant for a machine learning task using an action space composed of future changes in midpoint ticks as is the case with predicting future midpoint prices. See Section 5.3 for an explanation of how this model is implemented.

The first approach to Experiment C predicts the future midpoint prices using an RNN trained over 50 epochs for each sample security. Models converge on average to a mean square error loss of 0.34 for the training dataset, after either the termination epoch or if the optimisation is stopped early, and an average loss of 0.41 for the test dataset. The models converge on average after 34 epochs of training with each sample trained for approximately 8 hours on average. Performance is also measured by computing the difference between the model's prediction and the actual future midpoint price after 20 events, with the divergence between the two measured by the number of ticks. Over the sample period the average absolute value difference between the two values is 1.09 ticks, indicating that the prediction is on average incorrectly predicting the future price by more than one tick. The standard deviation of difference between the prediction and the actual midpoint price calculated over all the observations for each sample security is 1.05. Figure 5.7.2 illustrates the actual price path and the price path predicted by the model using a 20-event window for a single security over a single trading day encompassing 314,576 events, with midpoint prices reported as the number of half ticks from the opening price. Furthermore, the figure also demonstrates the difference between the actual and RNN predicted midpoint price over the event period 100,000 to 110,000. Both the results and the figure indicate that the performance of the RNN is relatively successful at predicting price points 20 events in the future. One must note that the 20-event period can encompass many events conducted in a single millisecond or several events spaced over a second period, with the model generally finding it more difficult to predict prices during high message throughput periods. Furthermore, it must be noted that the RNN is trained in an offline manner after the trading has been conducted.

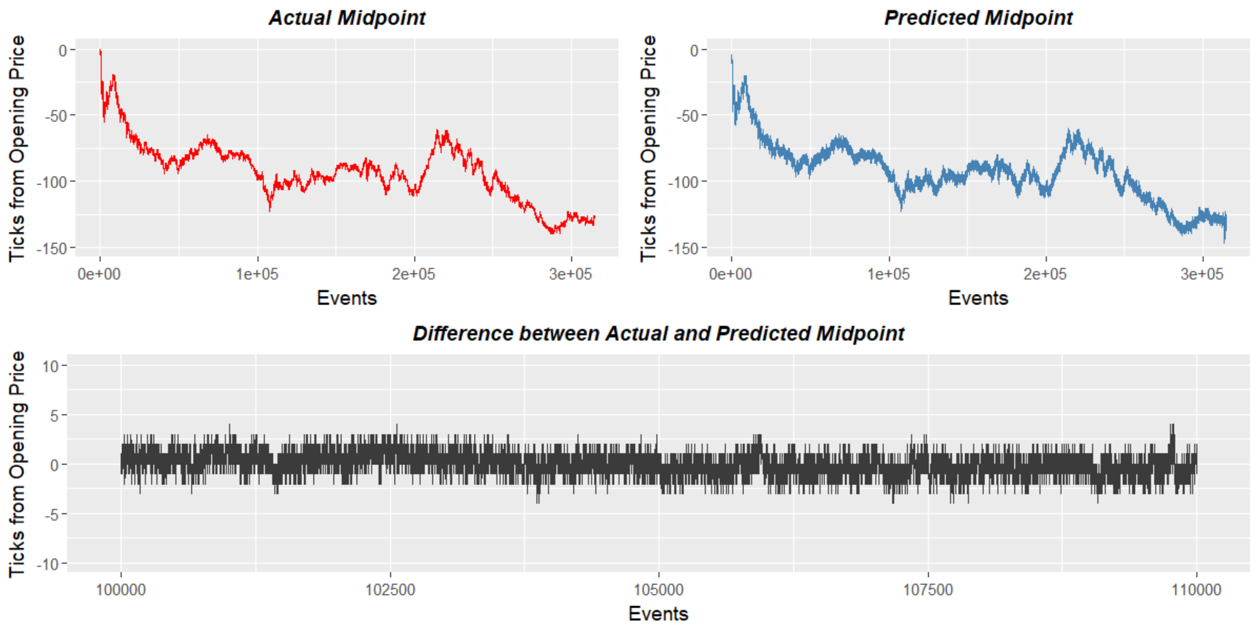

Figure 5.7.2 – Performance of the RNN model at predicting the future midpoint price of a sample security using a 20-event period. Actual price of a sample security measured by the number of ticks from the opening price over 314,576 events executed during a single trading day (top left). RNN predicted midpoint price 20 events in the future (top right). Difference between the actual and predicted price over the event period from 100,000 to 110,000 events for the sample security. All values are measured in half ticks.

The second approach using a simplified version of the Deep Q-Network RL 'tuner' (Jacques, 2017) attempts to improve on the RNN by imposing some form of structure to the features of the FTSE100 training dataset. We first train the RNN model on the FTSE100 time series dataset to predict the future midpoint prices of the sample securities using the optimised model architecture. We next train a Deep Q-Network (DQN) that utilises a trainer RNN to estimate the Q-function of the model, with the RL component including a reward function based on how well the model predicts the number of tick changes after 20 events. For the purpose of this experiment, the action space is composed of price points defined by the number of tick changes from the current price, which can be positive or negative.

The results of the analysis indicate that the DRQN RL tuner algorithm marginally improves on the performance of the RNN when predicting the future midpoint price of a security after 20 events. The model attains an average difference between the actual and predicted price of 0.74 half ticks which improves on the RNNs 1.09 half ticks. Furthermore, the standard deviation of the difference in the actual and prediction midpoint price reduces to 1.01 from the RNN model. All results are averaged over the five sample securities analysed. Figure 5.7.3 illustrates the actual and DRQN RL tuner predicted price path using a 20-event window for same security and trading day as analysed for the RNN, with midpoint prices reported as the number of half ticks from the opening price. Furthermore, the figure also shows the divergence between actual and predicted values for the same period of 100,000 to 110,000 events as the previous figure for the RNN. Evidently, from the results and figures presented, there is an improvement of the DRQN RL algorithm over the RNN, presenting a potential area for future development in the area of market microstructure and LOB dynamics research.

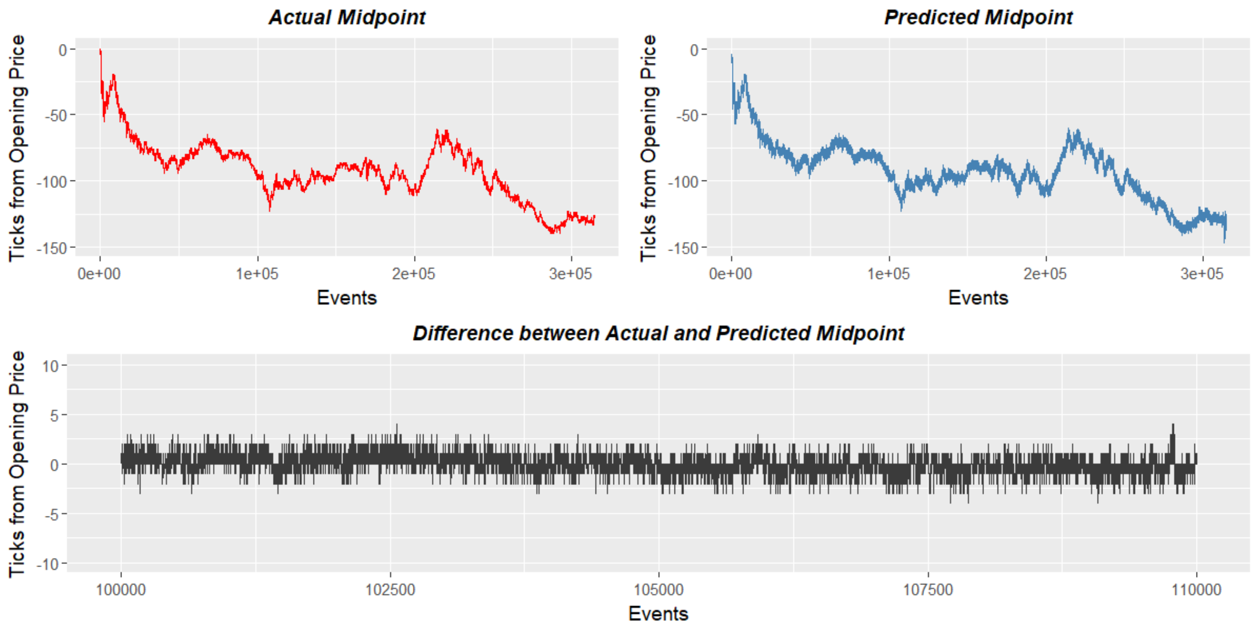

Figure 5.7.3 – Performance of the RL tuner Deep Q-Network with RNN model at predicting the future midpoint price of a sample security using a 20-event period. Actual price of a sample security measured by the number of ticks from the opening price over 314,576 events executed during a single trading day (top left). RNN predicted midpoint price 20 events in the future (top right). Difference between the actual and predicted price over the event period from 100,000 to 110,000 events for the sample security. All values are measured in half ticks.

5.8 Conclusion

In this chapter, we have developed a series of deep Recurrent Neural Network (RNN) and deep Reinforcement Learning (RL) models capable of predicting the future non-linear dynamics of the limit order book (LOB). These deep learning models untangle and extract complex features by building abstract temporal and spatial representations of the LOB state at any point in event-time, which intrinsically incorporate dependencies from past states. Performing time series predictions of the future state of the LOB using deep learning models required a consideration of the discontinuities, stochasticity and instabilities inherent in the LOB dynamics, given the complex set of interactions between heterogeneous traders. The plethora of academic literature that has attempted to model these complexities in the LOB are introduced in Section 5.1, with a distinction made between dynamic equilibrium, stochastic and machine-learning approaches to modelling the LOB dynamics. We argue that the vast data available in contemporary high-frequency markets, the failure of most generalisable assumptions regarding the LOB dynamics and the constrictive limitations of many stochastic models, rationalises the choice to model these dynamics using deep learning models.

A holistic exposition of relevant RNN and RL deep learning models, including their theoretical basis, mathematical formulation, design architecture and other elements are explored in Section 5.2 for RNNs and 5.3 for RL. Deep RNNs built and implemented in this section incorporate biological foundations of cognitive memory into their flexible neural network architecture using recurrent feedback connections. This allows temporal dependencies in the LOB history to be captured using an explicit recurrent distributed hidden state ‘memory’ representation that inculcates information about past states and sequence observations. Deep RL models serve as a computational approach to designing machine intelligence which extrapolates human conceptions of learning and optimal behaviour through interactions with the environment. As machine intelligence systems begin to supersede human intelligence in the realm of market microstructure, RL techniques present themselves as a useful model to understand how and why LOBs evolve over time. The dataset employed in this chapter is defined in Section 5.4, with a set of five FTSE 100 sample securities analysed during the period of October 2017.

An ablation analysis performed in Section 5.5 evaluates the various modular architectural schematics, configuration settings and design elements of an RNN trained to predict the LOB dynamics of the next trader action. This evaluation informs the development of an optimised model. Experimental evaluation of the network configuration is performed by perturbing single components of the model before testing the impact of the elemental augmentation on the RNN’s ability to accurately model LOB dynamics in a way that generalises well to unseen data. We performed a comparative analysis of the training, validation and generalisation performance of different design components and configurations along three dimensions. First, we assessed the performance of various architectural elements finding that a Gated Recurrent Unit RNN, with no bias term, that utilises a Scaled Exponential Unit activation function, with a 150-event lookback period, and with recurrent and standard dropout for each layer that use a masking neuron probability of 0.2, attained the best performance for the components analysed. Next, the dimensionality and depth of the RNN is evaluated, with the results indicating that a three-layer RNN with 64 neurons in each layer performed most effectively at modelling the LOB dynamics task. Finally, an evaluation of the optimisation process when training the RNN is performed, finding that RNNs trained with cross-entropy cost functions, AdaMax optimisation algorithms, and a batch size of

1024 samples led to the model achieving the highest training accuracy and predictive performance. The results from this analysis helped derive the optimised RNN model defined in Section 5.6 which is used to perform various prediction tasks to model LOB dynamics.

We next conducted in Section 5.7 three separate experiments including a prediction of the next trader action at or within the best bid-ask price (BBO) (Experiment A), the directional evolution of the LOB midpoint price (Experiment B), and a prediction of the midpoint price after x events. The unique approach taken in this chapter to modelling sequences of the LOB to predict future price dynamics extends previous theoretical and empirical research in the deep learning community (Sirignano, 2016; Dixon, 2017).

The results for Experiment A when predicting the next trader action at or within the BBO indicate that the RNN performs well at predicting the next submission of an order, but relatively poor at predicting the next marketable order. Limit orders for the bid and ask are predicted with a high degree of balanced accuracy, averaging 72.65% and 73.36%, respectively, over the FTSE100 dataset. Market orders are more difficult to predict with an average balanced accuracy of 59.68% and 59.55% for ask and bid market orders. It should be noted that the *observed* accuracy is significantly lower for market order actions but using the balanced metric we can infer a higher predictive accuracy by oversampling these less common actions. Evidently, one can argue that the optimised RNN performs well at predicting the next relevant trader action that has a commensurate impact on the dynamics of the LOB at the BBO.

Optimised RNNs also perform relatively well at predicting the next midpoint price change of each security (Experiment B). The average RNN model test dataset accuracy of 59.55% indicates that the model is predicting the future dynamics of the LOB with a higher accuracy than expected under a naïve random model. The long-term dependencies and complex feature abstractions developed by the RNN impute patterns into memory that provide contextual information for how the price is expected to change in the future. It must be noted that limitations do exist that detract from the applicability of employing this model in an online environment. In particular, the nature of the dataset, which is provided in a sequential structure, may differ from the state representation viewed by traders interacting with the LOB given the different latency rates between traders.

The final experiment to predict the future midpoint price after 20 events (Experiment C) demonstrated the utility of modelling LOB dynamics using a Deep Recurrent Q-Network (DRQN) that combines an RL agent with an RNN function approximator. The RNN individually attains an average difference between the models' prediction of the future price and the actual price of 1.09 half ticks over the sample dataset, with a trained mean square error loss of 0.41 for the test datasets. Utilising an off-policy RL 'tuner' algorithm using deep Recurrent Q-Networks augmented with a mean square error based cost function results in a reduction of the difference between the models' prediction and the actual future price to an average 0.74 half ticks.

This work represents the first attempt to integrate deep RNN models with deep RL to predict the future dynamics of the LOB. This new approach to modelling the complexities of contemporary high-frequency LOBs is of critical importance to both regulators, policymakers and traders conducting algorithmic trading strategies. Regulators want to understand the phenomena driving aggregate market behaviour to ensure markets are operating correctly. Utilising this approach, policymakers are able to make inferences regarding how the LOB evolves, which when combined with the results from the previous chapters allows regulators to draw potential implications of the interrelationship between the LOB dynamics and the execution of various algorithmic trading strategies on the order book. This bears interest for policymakers considering algorithmic testing procedures through potential future regulatory paths. For algorithmic traders, deep learning methodologies for predicting LOB dynamics enable these firms to improve their order placement algorithms, adjusting orders efficiently in order to minimise adverse selection costs, minimise implementation shortfall and improve best execution, whilst maximising profitability. The success of combining these two models to perform prediction tasks motivates the need for future research in this area that expands upon the techniques employed in this chapter.

Chapter Six

6 Conclusion

In this thesis, a deep learning framework has been developed to provide insight into the internal dynamics of algorithmic trading and modern equity limit order books (LOB). The transcendental advancements over the past decade in automated trading systems and theoretical models of deep learning have led to the rise of machines as the primordial arbiter of decision-making in financial markets. Consequently, markets have come to be dominated by algorithmic traders which has driven the demand from market practitioners and regulators for advanced tools and methodologies to comprehend the complex intricacies of this new paradigm of financial markets. The objective of this thesis is attained by developing deep learning models capable of defining algorithmic traders by the type of *strategy* the firm is conducting, analysing the *impact* of these strategies on UK equity market quality and modelling the *nexus* between algorithmic trading strategies and LOB dynamics.

In Chapter 3, we build and train an optimised Deep Neural Network (DNN) architecture to predict the behaviour of algorithmic traders given the state of the LOB and broader financial markets. Proprietary broker-identified data from the primary UK equity trading venues is captured and consolidated onto a single LOB to provide a panoptic view of the state of the market exposed to traders. This unique approach of amalgamating LOBs across trading venues allowed us to extract a holistic set of consolidated LOB features assumed to serve as algorithmic input into a firms trading strategy. Utilising a comprehensive ablation analysis, we were able to analyse the multifarious array of components, configurations, design elements and hyper-parameters of the DNN to derive a model capable of accurately predicting an algorithmic trading firms action on the LOB given the set of LOB features. Optimised models were trained with an average 6.9% generalisation error across 802 unique ISIN-Date-Broker tuple datasets for firms we define as algorithmic traders. Furthermore, the superiority of the DNN at modelling the relationship between LOB features and algorithmic trader actions was confirmed through a comparative analysis with other machine learning models. The results highlight the advantage of employing neurologically motivated deep learning models capable of finding complex non-linear patterns between abstract hierarchical features of the LOB and the actions executed by individual traders on the LOB, which are implicitly untangled by the DNNs deep neuronal connections.

Evidently, the deep learning methodology presented in this chapter allows regulators and market practitioners to more holistically comprehend how and why algorithmic traders interact with the LOB. Furthermore, using techniques developed and presented in this thesis, regulators can employ DNNs to ascertain intricate representations in equity market trading data that drive specific algorithmic trading strategies. This enables regulators to perform oversight functions and develop policy prescriptions relevant and necessary for contemporary financial markets. Future research can extend on the deep learning models developed in this thesis to further improve our understanding of how financial markets work, specifically what non-LOB financial phenomena drive algorithmic trading, the potential risks from herding behaviour on LOBs where traders execute strategies with lineated algorithmic input features, and questions of both the primary market and societal implications of secondary markets dominated by machines.

In Chapter 4, a new framework for segmenting algorithmic traders by the specific *strategy* being executed is developed before analysing the positive and negative *impact* of these strategies on UK equity market quality. We utilise axiomatic backpropagation-based feature attribution methods to assess the internal dynamics of a DNN trained in the previous chapter. By utilising data extracted from the feature importance analysis, we develop an understanding of the systemic elements of the LOB that *drives* algorithmic traders' actions and behaviour. Employing an unsupervised Spherical K-Means model, we are able to infer structural patterns in the feature importance datasets to categorise algorithmic traders as conducting one of five interpretable, exclusive and distinct strategies - Automated Market Making (AMM), Execution, Microstructural, Momentum and Technical strategies. Finally, the aggregate impact of each algorithmic trading strategy on UK equity market quality is empirically assessed along dimensions of liquidity, volatility and price discovery. Several insights are drawn from this analysis, including the net positive impact on market quality from AMM and Momentum strategies, who tend to provide a significant portion of liquidity to UK LOBs, dampen volatility and contribute positively to the price discovery process through the high-informational content impounded into their LOB order flow. Conversely, firms conducting Microstructural strategies in aggregate tend to reduce liquidity supply during periods of market volatility whilst contributing significantly less to price discovery.

The new paradigm of the equity market ecosystem explicated in this chapter places the algorithmic trading *strategy* as the tangible unit of analysis. This represents the most advanced and first explicit attempt to delineate the behaviour of traders conducting disparate operations into interpretable data-driven categories of algorithmic trading strategies. The accelerated dominance of machines in financial markets provides exigencies for regulators and academics to better understand the complex dynamics of the contemporary trader ecosystem. Untangling these complexities by extracting the various algorithmic trading strategies being conducted allows regulators to more accurately understand how markets function in their contemporary iteration. Implementing this new paradigm of the trader ecosystem provides the tools for policymakers to develop policy prescriptions for modern markets using an empirical and diagnostic approach at the *strategy* level. Further research can be extended to optimise the methodology employed in this chapter to more holistically model the trader environment. Specifically, the methodology developed is amenable in a way that allows for new strategies to be extracted. Regulators with a keen interest in understanding market manipulation algorithms and strategies can extend this framework to identify and analyse trading firms conducting strategies that negatively impact market quality, allowing for the formulation of stronger policies that placate the investor communities concern that certain traders have unfair competitive advantages.

In Chapter 5, we develop a deep learning model that utilises Deep Recurrent Neural Networks (RNN) and Reinforcement Learning (RL) to predict the future non-linear price dynamics of the LOB. The LOB is modelled as a complex dynamic and adaptive system populated by heterogeneous traders conducting disparate trading strategies. We extract hierarchical features by building abstract temporal and spatial representations of the LOB state at any point in event-time, using a recurrent memory component to intrinsically incorporate dependencies from past states. Utilising this deep learning framework, we perform experiments to predict the future state of the LOB, that is, predict the future actions of traders, midpoint prices, and price movements. An optimal RNN model is first developed through an ablation analysis that evaluates the ability of various architectural, design and network optimisation components to improve the generalisation power of the RNN when predicting future LOB dynamics. This optimised model trained on UK equity data is capable of predicting the next price change on the LOB with an average balanced accuracy of 59.55% across the securities analysed. However, there exist limitations in the models' applicability to real-world markets given the inherent divergence in firm's latency rates, and thus, state representations of the LOB, which are not factored into the model. Furthermore, an experiment performed to predict the future midpoint price after 20 events using an RNN model attains an average 1.09 half-tick error between the model prediction and the actual price. We next integrate an RNN as a function approximator into an augmented Deep Recurrent Q-Network (DRQN) RL model which results in a reduction in prediction error to 0.74 half ticks on average over the securities analysed.

The results presented in Chapter 5 provide evidence towards the capabilities of deep learning to develop highly abstract hierarchical feature representations of financial market phenomena that allow one to model the complexities of contemporary high-frequency LOBs with an acceptable degree of accuracy. When combined with models employed in Chapter 3 and Chapter 4, these new approaches to modelling algorithmic trading strategies and LOB dynamics presents regulators, policymakers and market practitioners with a framework for drawing inferences on the internal intricacies and interrelationships within modern financial markets. This bears significant interest for policymakers considering regulation in the form of 'algorithmic testing'. The deep learning methodology employed in this thesis provides a framework for testing the impact of current and potential algorithms on the quality of markets on which they are deployed. Furthermore, one can draw conclusions of how these algorithms interact with other market participants, including those conducting strategies conducive to market quality, to formulate a comprehensive view of what impact an algorithm is likely to have on the dynamics of the LOB and the quality of the market. The conclusions drawn in this thesis also provide tangible implications and opportunities for traders conducting complex algorithmic trading strategies that utilise machine learning models. Instituting a deep learning framework for modelling LOB dynamics with a large feature space allows these algorithmic traders to more accurately predict how markets will evolve over time. This allows firms to improve their order generation and placement algorithms, adjusting orders in a way that efficiently minimises market impact, the risk of adverse selection, implementation shortfall, and provides best execution for clients, with a global objective of maximising profitability.

One final area for future research in computational finance, algorithmic trading and LOB dynamics is the integration of deep learning models with expert human knowledge. Recent developments in deep learning and artificial intelligence have cemented the supremacy of machines, particularly in financial trading domains, when making functional decisions. This has been primarily driven by the theoretical advancements in optimisation algorithms and transfer of the feature engineering process from human to machine, with models trained using deep learning capable of learning feature abstractions difficult for humans to grasp. However, as researchers and practitioners traverse the long arc towards artificial general intelligence, there exist opportunities for augmenting deficiencies in current iterations of deep learning models by integrating additional forms of expert input. Thus, we propose that infusing expert domain knowledge in financial trading, provided by either regulators or market practitioners, into a deep Recurrent Q-Network, is an interesting area of future research.

We are optimistic that the deep learning framework developed in this thesis will empower regulators, practitioners and researchers with a new methodology and direction of how to approach algorithmic trading and understand the dynamics of the LOB in contemporary financial markets.

Appendix A – Limit Order Book & Trader-Specific Features

The set of limit order book (LOB) and trader-specific features used throughout this thesis are defined, explained and mathematically formulated in this Appendix. Features are split into the type of algorithmic trading strategy – Automated Market Making, Execution, Microstructural, Momentum and Technical strategies - to which they most commonly serve as algorithmic inputs into. Commonly used symbology in the equation include the price, P_t, volume, V_t, order, O_t, for period or event t. These symbols are also defined by whether they relate to a specific trader, x, a buy, B, or a sell, S. Furthermore, the best bid price, BBP_t, and best ask price, BAP_t, are commonly used, in addition to an exponential moving average (EMA) price, $AP_t = \delta(P_t) + (1-\delta)(P_{t-1})$, where $\delta = \frac{2}{k+1}$.

Automated Market Making

Feature	Name	Details	Calculation
QOB_t^x QOA_t^x	Quote Offset (Bid/Ask)	Percentage difference between current quote offset and time-weighted average quote offset of best quote from BBO (TQO measured over duration k).	$\frac{CQOB_t^x - TQOB_t^x(k)}{TQOB_t^x(k)}$ $\frac{CQOA_t^x - TQOA_t^x(k)}{TQOA_t^x(k)}$ $CQOB_t^x = BBP_t^m - BBP_t^x$ $CQOA_t^x = BAP_t^x - BAP_t^m$ $TQOB_t^x(k) = \frac{\sum_{i=1}^k CQOB_{t-i}^x}{k}$ $TQOA_t^x = \frac{\sum_{i=1}^k CQOA_{t-i}^x}{k}$
$QOVB_t^x$ $QOVA_t^x$	Volatility Normalised Quote Offset (Bid/Ask)	Quote offset normalised by k period historical volatility, σ, for both bid and ask prices.	$\frac{[CQOB_t^x + z\sigma(k)] - TQOB_t^x}{TQOB_t^x(k)}$ $\frac{[CQOA_t^x + z\sigma(k)] - TQOA_t^x}{TQOA_t^x(k)}$ $k = 1000$ events $z = 3$
$QOAB_t^x$ $QOAA_t^x$	Aggressiveness Normalised Quote Offset (Bid/Ask)	Quote offset normalised by k period aggressiveness levels, a, for both bid and ask orders.	$\frac{[CQOB_t^x + za(k)] - TQOB_t^x(k)}{TQOB_t^x(k)}$ $\frac{[CQOA_t^x + za(k)] - TQOA_t^x(k)}{TQOA_t^x(k)}$ $k = 500$ events $z = 1\%$
$INVN_t^x$	Net Inventory Ratio	Current ratio of cumulative net volume, buy volume $V_i^{x,B}$ minus sell volume $V_i^{x,S}$, to cumulative total volume traded by trader x.	$\frac{\sum_{i=0}^t \left(V_i^{x,B} - V_i^{x,S}\right)}{\sum_{i=0}^t \left(V_i^{x,B} + V_i^{x,S}\right)}$
$INVR_t^x$	Relative Inventory Position	Percentage deviation of current net inventory position from target net inventory ratio (end of day position).	$\frac{INVN_t^x - INVT_t^x}{INVT_t^x}$ $INVT_t^x = \frac{\sum_{i=0}^T \left(V_i^{x,B} - V_i^{x,S}\right)}{\sum_{i=0}^T \left(V_i^{x,B} + V_i^{x,S}\right)}$
$ESPT_t^x$ $ESAT_t^x$	Effective Half-Spread (Passive / Aggressive Orders)	Volume-weighted relative difference between price received by trader x, P_{t-i}^x, and midpoint, MP_{t-i}^x, over T. $ESPT_t^x$ is positive half-spread received by passive trader, and $ESAT_t^x$ is negative half-spread paid by aggressive trader.	$\frac{\sum_{i=0}^T \frac{\lvert P_{T-i}^x - MP_{T-i}^x\rvert}{MP_{T-i}^x} V_{T-i}^x}{\sum_{i=0}^T V_{T-i}^x}$ $\frac{-\sum_{i=0}^T \frac{\lvert P_{T-i}^x - MP_{T-i}^x\rvert}{MP_{T-i}^x} V_{T-i}^x}{\sum_{i=0}^T V_{T-i}^x}$
$CPAS_t^x$ $CAGG_t^x$	Trading Costs (Passive & Aggressive traders)	Total cost of trading for aggressive and passive traders. Passive trader costs include approximate rebates for trader x placing a passive limit orders at BBO on LOB which is subsequently executed against. Aggressive trader costs include total trading fees for placing aggressive order on exchange.	$ESPT_t^x(k) + REB_t$ $ESAT_t^x(k) + EC_t$ REB_t^x = BATE (0 bp), CHIX (0.15 bp), TRQX (0.15 bp), XLON (0 bp) EC_t^x = BATE (0.16 bp), CHIX (0.3 bp), TRQX (0.3 bp), XLON (0.45 bp)

$\boldsymbol{PINT_t(\alpha,\mu,\theta,k)}$ $\boldsymbol{PINM_t(\alpha,\mu,\theta,k)}$	Probability of Informed Trading (PIN) (Trades (T), Orders (O))	Measure of informed trading over period k based on probability of information event, a_t, and rates of informed, μ, and uninformed, θ, trader arrival rates, based on trading activity & order flow.	$\frac{\alpha_t^T\mu^T}{\alpha_t^T\mu^T+\theta_B^T+\theta_S^T}$ $\quad \frac{\alpha_t^O\mu^O}{\alpha_t^O\mu^O+\theta_B^O+\theta_S^O}$ $a_t,\beta \in [0,1] \quad \mu,\theta \in [0,\infty)^2$ $k = 10, 50, 100$ periods
$\boldsymbol{VPIN_t^{TR}(v,j,k)}$ $\boldsymbol{VPIN_t^{BVC}(v,j,k)}$ $\boldsymbol{VPIN_t^{TI}(v,j,k)}$	Volume-Synchronised Probability of Informed Trading (VPIN) (Tick, Bulk Volume Classification, True Initiator)	Rolling average of order flow trade imbalance based on volumes, $V_{i,l}$, calculated using the Tick Rule, Bulk Volume Classification rule, and the True Initiator rule, denoted by R, based on number of buckets k, number of bars, j and bucket volume size v.	$\frac{1}{k}\sum_{l=1}^{k} OFI_l\,(v,j)$ $OFI_l(v,j) = \frac{1}{v}\sum_{i=1}^{j}\left\lvert V_{i,l}^{S,R} - V_{i,l}^{B,R}\right\rvert$ $k = 10, 20, 50$ buckets $R = TR, BVC, TI$
$\boldsymbol{OARA_t(k)}$ $\boldsymbol{OARP_t(k)}$ $\boldsymbol{OARC_t(k)}$	Order Arrival Rates (Aggressive, Passive, Cancel orders)	Total ratio of orders for aggressive, O_t^{Agg}, passive, O_t^{Pass}, and cancel, O_t^{Can}, traders for period k relative to average number of orders, $O_T^{Ave,k}$.	$\frac{\sum_{i=0}^{k-1}\lvert O_{t-i}^{Agg}\rvert}{O_T^{Ave,k}}$ $\quad \frac{\sum_{i=0}^{k-1}\lvert O_{t-i}^{Pass}\rvert}{O_T^{Ave,k}}$ $\quad \frac{\sum_{i=0}^{T}\lvert O_{t-i}^{Can}\rvert}{O_T^{Ave,k}}$ $k = 10, 100, 1000, 10000$ events, $1, 5$ seconds
$\boldsymbol{OARA_t^x}$ $\boldsymbol{OARP_t^x}$ $\boldsymbol{OARC_t^x}$	Order Arrival Rates for trader x (Aggressive, Passive, Cancel orders)	Total ratio of trader x orders for aggressive, $O_t^{x,Agg}$, passive, $O_t^{x,Pass}$, and cancel, $O_t^{x,Can}$, traders to total number of orders, O_T^{All}.	$\frac{\sum_{i=0}^{T}\lvert O_{T-i}^{x,Agg}\rvert}{O_T^{All}}$ $\quad \frac{\sum_{i=0}^{T}\lvert O_{T-i}^{x,Pass}\rvert}{O_T^{All}}$ $\quad \frac{\sum_{i=0}^{T}\lvert O_{T-i}^{x,Can}\rvert}{O_T^{All}}$

Execution Strategies

Feature	Name	Details	Calculation
$\boldsymbol{POVI_t^x}$	Percentage of Volume (POV) Imbalance	Difference between current POV for trader x, $POVT_t^x$, and that traders end of day POV target, $POVX_t^x$.	$POVT_t^x - POVX_t^x$ $POVT_t^x = \frac{\sum_{k=1}^{t} V_{t-k}^x}{\sum_{k=1}^{t} V_{t-k}}$ $\quad POVX_t^x = \frac{\sum_{i=0}^{T} V_{T-i}^x}{\sum_{i=0}^{T} V_{T-i}}$
$\boldsymbol{TWAPMI_t(k)}$	Market Time Weighted Average Price (TWAP) Imbalance	Difference between current market average trade price at time t weighted over periods of length k (TWAPM) and market price.	$\frac{TWAPM_t - P_t}{P_t}$ $TWAPM_t(k) = \frac{1}{4t}\sum_{k=1}^{t} k(P_{t-k}^O + P_{t-k}^H + P_{t-k}^L + P_{t-k}^C)$
$\boldsymbol{TWAPTI_t^x(k)}$	Trader Time Weighted Average Price (TWAP) Imbalance	Difference between current trader x TWAP over periods of length k and market TWAP.	$\frac{TWAPT_t^x - TWAPM_t}{TWAPM_t}$ $TWAPT_t^x(k) = \frac{1}{t}\sum_{k=1}^{t} k(P_{t-k}^x)$
$\boldsymbol{VWAPMI_t(k)}$	Market Volume Weighted Average Price (VWAP) Imbalance	Difference between current market volume weighted average trade price at time t weighted over periods of length k (VWAPM) and market price.	$\frac{VWAPM_t(k) - P_t}{P_t}$ $VWAPM_t(k) = \frac{\sum_{k=1}^{t} P_{t-k}V_{t-k}}{\sum_{k=1}^{t} V_{t-k}}$
$\boldsymbol{VWAPTI_t^{x,I}(k)}$	Trader Volume Weighted Average Price (VWAP) Imbalance	Difference between current trader x VWAP (Buy/Sell) over periods of length k and market VWAP.	$\frac{VWAPT_t^{x,I} - VWAPM_t(k)}{VWAPM_t(k)}$ $VWAPT_t^x(k) = \frac{\sum_{k=1}^{t} P_{t-k}^x V_{t-k}^x}{\sum_{k=1}^{t} V_{t-k}^x}$ $\quad I = Buy\ /\ Sell$
$\boldsymbol{VWAPTD_t^x}$	Trader VWAP Difference	Difference between trader buy and sell VWAPs.	$VWAPT_t^{x,Sell} - VWAPT_t^{x,Buy}$
$\boldsymbol{VWAPTW_t^x}$	Weighted Trader VWAP Difference	Difference between trader buy and sell VWAPs, weighted by volumes	$\frac{\left(VWAPT_t^{x,Sell}V_t^{x,Sell}\right) - \left(VWAPT_t^{x,Buy}V_t^{x,Buy}\right)}{\left(V_t^{x,Sell} + V_t^{x,Buy}\right)}$

$\boldsymbol{PIPA_t(k)}$ $\boldsymbol{PIPAX_t^x(k)}$	Linear Permanent Market Impact (All Traders & Trader $\boldsymbol{x}$) (Almgren, 2005)	Regression model of permanent linear price impact from information leakage arising from the order execution over time. Requires estimation of coefficient, γ, using linear regression of $PIPA_t$ observable values. Hyper-parameters, α and δ estimated using the Marquardt-Nash optimisation algorithm that minimises residual sum of squares.	$PIPA_t = \gamma\sigma T sgn(X)\left\|\frac{X}{VT}\right\|^{\alpha}\left(\frac{\phi}{V}\right)^{\delta} + \eta \quad Regression$ $PIPAX_t^x = \gamma_x\sigma T sgn(X)\left\|\frac{X}{VT}\right\|^{\alpha}\left(\frac{\phi}{V}\right)^{\delta} + \eta \quad Regression$ $PIPA_t(k) = \frac{P_{t+k} - P_t}{P_t} \quad Observable$ $k = 10, 20, 30$ minutes $PIPA_t$ coefficient γ estimated with linear regression for all traders. $PIPAX_t^x$ coefficient γ_x estimated with linear regression for trader x.
$\boldsymbol{PIRA_t(k)}$ $\boldsymbol{PIRAX_t^x(k)}$	Linear Realised Market Impact (All Traders & Trader $\boldsymbol{x}$) (Almgren, 2005)	Regression model of realised price impact. Coefficient, ψ, estimated using linear regression of, $PIRA_t$, observable values. Hyper-parameter, β, estimated using the Marquardt-Nash optimisation algorithm that minimises residual sum of squares.	$PIRA_t = \frac{PIPA_t}{2} + \psi\sigma T sgn(X)\left\|\frac{X}{VT}\right\|^{\beta} + \eta \quad Regression$ $PIRAX_t^x = \frac{PIPA_t^x}{2} + \psi_x\sigma T sgn(X)\left\|\frac{X}{VT}\right\|^{\beta} + \eta \quad Regression$ $PIRA_t(k) = \frac{\bar{P} - P_t}{P_t} \quad Observable$ $k = 10, 20, 30$ minutes $PIRA_t$ coefficient γ estimated with linear regression for all traders. $PIRAX_t^x$ coefficient ψ_x estimated with linear regression for trader x.
$\boldsymbol{PITA_t(k)}$ $\boldsymbol{PITAX_t^x(k)}$	Linear Temporary Market Impact (All Traders & Trader $\boldsymbol{x}$) (Almgren, 2005)	Regression model of temporary price impact caused from the demand of liquidity from trading.	$PITA_t = PIRA_t - PIPA_t$ $PITAX_t^x = PIRAX_t^x - PIPAX_t^x$ $k = 10, 20, 30$ minutes
$\boldsymbol{PIGA_t(k)}$	Permanent Market Impact (Almgren, 2005)	Gamma in linear permanent market impact regression.	$\gamma\ from\ PIPA_t\ regression$ $k = 10$ minutes
$\boldsymbol{PIPRR_t(j,k,l)}$	Permanent Market Impact (GMM) / Rolling	GMM regression model of permanent price impact for whole market.	Permanent and temporary price impact and market resiliency estimated using GMM regression over k intervals of time length j: $P_k - MP_0 = \vartheta_k + PITR_k\sum_{i=0}^{k} RESR_i OI_i + PIPR_k\sum_{i=0}^{k} q_i + \varepsilon_k$ Signed order imbalance l^{th}-order autoregression model used to calculate errors q_k for k^{th} interval. $OI_k = \alpha_k + \sum_{i=0}^{m} \beta_i OI_i + q_k$ $j = 40$ minute periods $k = 10$ minute periods $l = 5$ peiod intervals Rolling regression performed over 40 minute periods.
$\boldsymbol{PITRR_t(j,k,l)}$	Temporary Market Impact (GMM) / Rolling	GMM regression model of temporary price impact for whole market.	
$\boldsymbol{RESRR_t(j,k,l)}$	Market Resiliency (GMM) / Rolling	GMM regression model of market resiliency.	
$\boldsymbol{PIIKBR_t^{\pi}(k)}$ $\boldsymbol{PIIKAR_t^{\pi}(k)}$	Influence Kernel (BBP, BAP) (Rolling) (Eisler, 2011)	Data-driven measure, $\mathcal{G}_{\pi_k,\pi_t}$, of the average market impact for specific event, π_t, based on correlation with other events, π_k. Events include order submissions and deletions, subdivided into whether they are inside or at the BBO. Market impact is analysed over k lagged periods.	$R_t = \Delta P_{\pi_t}^R \epsilon_t + \int_{k=1}^{t} \mathcal{G}_{\pi_k,\pi_t}(t-k)\,\epsilon_t + \eta_t$ $\Delta P_{\pi_t}^R = \langle \Delta P_t \| \pi_t = \pi \rangle$ $\mathcal{G}(k) = G(k+1) - G(k)$ $\epsilon_t = \begin{cases} 1 & for\ buys \\ -1 & for\ sells \end{cases}$ π = {Enter, Cancel} {Inside BBO, At BBO} $k = 10$ seconds, 1 minute
$\boldsymbol{RESDR_t(j,k)}$ $\boldsymbol{RESQR_t(j,k)}$	Market Resiliency (Spread & Depth) / Rolling	Mean reversion autoregression AR(j) model of change in spread, QS, and depth, MD, liquidity levels, $\Delta L_k^{QS/MD}$, with j lags of length k to estimate resiliency speed, ρ, of depth and spread replenishment.	$\Delta L_k^{QS/MD} = \alpha_k + \rho L_{k-1}^{QS/MD} + \sum_{i=1}^{j} \beta_i \Delta L_{k-i}^{QS/MD} + \varepsilon_t$ $\rho = RESDR_t\ and\ RESQR_t$ $j = 5$ lags $k = 1, 5$ minutes

$\boldsymbol{IRFS_t^{\mathcal{L},\mathcal{A}}(k,n)}$ $\boldsymbol{IRFD_t^{\mathcal{L},\mathcal{A}}(k,n)}$	Impulse response function (Spread / Depth)	Impulse response function, $\hat{R}(n,\delta)$, for shock matrix, $\delta_{\mathcal{L},\mathcal{A}}$, and lag n, estimated from k^{th} order vector auto regression (VAR(k)) of spread and depth liquduity levels, $L_t^{S/D}$.	$L_t^{S/D} = M_t + \sum_{i=1}^{t-1} A^i \mu_{t-i}$ $\hat{R}(n,\delta_{\mathcal{L},\mathcal{A}}) = \hat{A}^n \delta_{\mathcal{L},\mathcal{A}}$ $L_{T+h\vert T,\mathcal{L},\mathcal{A}} = A_1 L_{T+h-1\vert T,\mathcal{L},\mathcal{A}} + \cdots + A_k L_{T+h-k\vert T,\mathcal{L},\mathcal{A}}$ L_t^S = liqudity levels at time t μ_{t-i} = estimated errors from VAR(k) $\mathcal{L} = L^S, L^D$, $\mathcal{A} \in BBP0, BBP1, BAP0, BAP1$ $\delta_{\mathcal{L},\mathcal{A}}$ = shock matrix $k = 5\ lags\ of\ 100\ ms$ $n = 100\ lags\ of\ 100\ ms$ 50% and 90% replenishment levels
$\boldsymbol{CLCV_t^x}$ $\boldsymbol{CLCN_t^x}$	Cumulative Liquidity Charge (CLC) (Volume-Weighted, BP)	Cumulative difference between the current execution price, P_i^E, and the previous execution price, P_{i-1}.	$CLCV_t = \sum_{i=1}^{t} ILCV_i \quad CLCN_t = \sum_{i=1}^{t} ILCN_i$ $ILCV_i = \begin{cases} V_i^E(P_i^E - P_{i-1}) & if\ buying \\ V_i^E(P_{i-1} - P_i^E) & if\ selling \end{cases}$ $ILCN_i = \begin{cases} (P_i^E - P_{i-1}) & if\ buying \\ (P_{i-1} - P_i^E) & if\ selling \end{cases}$
$\boldsymbol{CLPV_t^x}$ $\boldsymbol{CLPN_t^x}$	Cumulative Liquidity Premium (CLP) (Volume-Weighted, BP)	Cumulative difference between best price on the side of the LOB opposite the trade and the execution price.	$CLPV_t = \sum_{i=1}^{t} ILPV_i \quad CLPN_t = \sum_{i=1}^{t} ILPN_i$ $ILPV_i = \begin{cases} \max\left(0, V_i^E(P_i^E - P_i^{AP})\right) & if\ buying \\ \max\left(0, V_i^E(P_i^{BP} - P_i^E)\right) & if\ selling \end{cases}$ $ILPN_i = \begin{cases} \max\left(0, (P_i^E - P_i^{AP})\right) & if\ buying \\ \max\left(0, (P_i^{BP} - P_i^E)\right) & if\ selling \end{cases}$
$\boldsymbol{CSCV_t^x}$ $\boldsymbol{CSCN_t^x}$	Cumulative Spread Charge (CSC) (Volume-Weighted, BP)	Minimum function of ILC minus ILP, for trade i, bounded by zero.	$CSCV_t = \sum_{i=1}^{t} ISCV_i \quad CSCN_t = \sum_{i=1}^{t} ISCN_i$ $ISCV_i = \min(ILCV_i - ILPV_i, 0)$ $ISCN_i = \min(ILCN_i - ILPN_i, 0)$
$\boldsymbol{TCV_t^x}$ $\boldsymbol{TCN_t^x}$	Timing Consequence (TC) (Volume-Weighted, BP)	Residual market impact or residual implementation shortfall after accounting for individual traders CLC.	$ISV_t = \frac{\sum_{i=1}^{t} V_i(P_i - P^A)}{\sum_{i=1}^{t} V_i} = CLCV_t^X + TCV$ $ISN_t = \frac{\sum_{i=1}^{t}(P_i - P^A)}{\sum_{i=1}^{t} V_i} = CLCN_t^X + TCN$
$\boldsymbol{QSTR_t^x}$	Trader Quoted Spread Ratio	Difference between trader $\boldsymbol{x}$ best ask price, BAP_t^x, and best bid price, BBP_t^x, divided by tick, δ.	$\frac{(BAP_t^x - BBP_t^x)}{\delta}$
$\boldsymbol{MDTR_t^x}$	Trader Market Depth	Total trader $\boldsymbol{x}$ depth at best bid, $V_t^{x,BBP}$, and best ask price, $V_t^{x,BAP}$.	$V_t^{x,BBP} + V_t^{x,BAP}$

Microstructural Strategies

Feature	Name	Details	Calculation
$\boldsymbol{OFS_t(k)}$ $\boldsymbol{OFV_t(k)}$	Order Flow (Signed, Volume)	Average signed and volume-weighted order flow of aggregated buyer minus seller-initiated trades over past k trades.	$\frac{1}{k}\sum_{i=0}^{k-1} S_{t-i} \qquad \frac{\sum_{i=0}^{k-1} S_{t-i}V_{t-i}}{\sum_{i=0}^{k-1} t - i}$ $k = 1, 5, 10, 50, 100$ trades
$\boldsymbol{OFVE_t(a)}$	Order Flow (EMA Volume)	Exponential moving average (EMA) of signed order flow volume, with degree of weighting decrease $\boldsymbol{a}$.	$\frac{2(OFV_t) + (a-1)(OFVE_{t-1})}{a+1}$ $a = 7\ (quarter), 3\ (third), 1\ (half)$
$\boldsymbol{MD_t(j,k)}$	Market depth	Total volume of limit orders placed at price point j over the period k.	$\frac{1}{k}\sum_{i=0}^{k-1} V_{t-i}^{j}$ $j = 1, 2, 5 \qquad k = 0, 1\ ms, 1\ sec$

$\boldsymbol{MDE_t(a)}$	Market depth (EMA)	Exponential moving average (EMA) of market depth at BBO, with degree of weighting decrease a.	$\frac{2(MD_t)+(a-1)(MDE_{t-1})}{a+1}$ $a = 7\ (quarter), 3\ (third), 1\ (half)$
$\boldsymbol{LOI_t(j,k)}$	Limit order imbalance	Order imbalance of LOB market depth for k lag periods and price points j ticks from the respective BBO prices.	$\frac{1}{k}\sum_{i=0}^{k-1}(MD_{t-i}^{B,j} - MD_{t-i}^{S,j})$ $j = 1, 2, 5$ $k = 0, 10, 100$ events, 1 ms, 5 ms, 1 second
$\boldsymbol{LOIE_t(a)}$	Limit order imbalance (EMA)	Exponential moving average (EMA) of limit order imbalance of BBO, with degree of weighting decrease a.	$\frac{2(LOI_t)+(a-1)(LOIE_{t-1})}{a+1}$ $a = 7\ (quarter), 3\ (third), 1\ (half)$
$\boldsymbol{AGGC_t(k)}$ $\boldsymbol{AGGV_t(k)}$	Aggressiveness Ratio (Count, Volume)	Count & Volume ratio of aggressive market orders to passive limit orders over past k events submitted at prices equal to or better than BBO.	$\frac{\sum_{i=0}^{k-1}\lvert S_{t-i}\rvert}{\sum_{i=0}^{k-1}\lvert O_{t-i} - S_{t-i}\rvert}$ $\frac{\sum_{i=0}^{k-1}\lvert S_{t-i}V_{t-i}\rvert}{\sum_{i=0}^{k-1}\lvert (O_{t-i} - S_{t-i})V_{t-i}\rvert}$ $k = 100, 1000, 10000$ events
$\boldsymbol{QS_t(k)}$	Quoted spread	Difference between the best ask and best bid price over the period k.	$\frac{1}{k}\sum_{i=0}^{k-1}P_{t-i}^{BA} - P_{t-i}^{BB}$ $k = 0, 1, 5, 10, 50, 1000$ events, 1 ms, 5 ms, 1 second
$\boldsymbol{QSE_t(a)}$	Quoted spread (EMA)	Exponential moving average (EMA) of quoted spread, with degree of weighting decrease a.	$\frac{2(QS_t)+(a-1)(QSE_{t-1})}{a+1}$ $a = 7\ (quarter), 3\ (third), 1\ (half)$
$\boldsymbol{ES_t(k)}$	Effective Spread (All traders)	Double volume-weighted percentage difference between trade price, P_{t-i}, and midpoint, MP_{t-i}, over k periods.	$\frac{2\sum_{i=0}^{k-1}\frac{\lvert P_{t-i} - MP_{t-i}\rvert}{MP_{t-i}}V_{t-i}}{\sum_{i=0}^{k-1}V_{t-i}}$ $k = 1, 5, 10, 50, 100, 1000$ trades
$\boldsymbol{VOLR_t(j,k)}$	Realised volatility	Standard deviation of j period logarithmic returns. Volatility is measured over length k periods.	$\sqrt{\frac{\sum_{i=0}^{k}R_{t-i}(j)^2}{k}}$ $R_t(j) = \ln\left(\frac{P_t}{P_{t-j}}\right)$ $j, k = (1,10), (1, 100), (5, 1000), (5, 10000)$ events
$\boldsymbol{VOLHL_t(j,k)}$	High-low volatility (Parkinson, 1980)	Standard deviation of j period max-min logarithmic returns using max prices, P_i^{max}, and min prices, P_i^{min}, corrected by inverse scalar $\sqrt{4\ ln2}$. Volatility is measured over length k periods.	$\sqrt{\frac{\sum_{i=0}^{k}R_{t-i}(j)^2}{4\ k\ ln2}}$ $R_t(j) = \ln\left(\frac{P_j^{max}}{P_j^{min}}\right)$ $j, k = (1,10), (1, 100), (5, 1000), (5, 10000)$ events
$\boldsymbol{VOLRS_t(k)}$	High-low-open-close volatility (Garman, 1980)	Volatility measure using max prices, P_i^{max}, min prices, P_i^{min}, close prices, P_i^{close}, and open prices, P_i^{open}, over period k.	$\sqrt{\frac{\sum_{i=0}^{k}\left(\ln\left(\frac{P_i^{max}}{P_i^{min}}\right)\right)\left(\ln\left(\frac{P_i^{max}}{P_i^{open}}\right)\right)+\left(\ln\left(\frac{P_i^{min}}{P_i^{close}}\right)\right)\left(\ln\left(\frac{P_i^{min}}{P_i^{open}}\right)\right)}{k}}$ $k = 10, 50, 100, 1000$ events
$\boldsymbol{VOLQS_t(k)}$	Quoted spread volatility	Difference between maximum quoted spread and minimum quoted spread in period k.	$QS_{t\in k}^{max} - QS_{t\in k}^{min}$ $k = 5, 20, 50$ events

Momentum Strategies

Feature	Name	Details	Calculation
$\boldsymbol{MO_t(k)}$	Momentum	Price change over last k periods.	$P_t - P_{t-k}$ $k = 2, 5, 10$ trades
$\boldsymbol{MOB_t(k)}$ $\boldsymbol{MOA_t(k)}$	Momentum (Bid, Ask)	Price change in BBP/BAP over last k periods.	$P_t^{BBP} - P_{t-k}^{BBP}$ $P_t^{BAP} - P_{t-k}^{BAP}$ $k = 5, 10, 100, 1000$ events

$\boldsymbol{AC_t(k)}$	Acceleration	Change in price momentum over last k periods.	$MO_t - MO_{t-k}$ $k = 5$ trades
$\boldsymbol{ACB_t(k)}$ $\boldsymbol{ACA_t(k)}$	Acceleration (Bid, Ask)	Change in bid/ask momentum over last k periods.	$MOB_t - MOB_{t-k} \quad MOA_t - MOA_{t-k}$ $k = 10$ events
$\boldsymbol{V_t(k)}$	Volume traded	Absolute average number of shares traded over last k periods.	$\frac{1}{k}\sum_{i=1}^{k}\lvert V_{t-i}\rvert$ $k = 5, 10, 50, 100, 1000$ events
$\boldsymbol{LAOB_t}$ $\boldsymbol{LAOA_t}$	Latency Arbitrage Opportunity (Bid / Ask)	Order at or better than BBP/BAP when (near) latency arbitrage opportunity available (Max global BBP/BAP greater than min global BAP/BBP).	$\begin{cases}1 & \text{if} \max(BBP_t^y) > \min(BBP_t^y) \mid O^{ENTER,BBP} \\ 0 & otherwise\end{cases}$ $\begin{cases}1 & \text{if} \max(BAP_t^y) > \min(BAP_t^y) \mid O^{ENTER,BAP} \\ 0 & otherwise\end{cases}$ BBP_t^y is the BBP for trading venue y BAP_t^y is the BAP for trading venue y

Technical Strategies

Feature	Name	Details	Calculation
$\boldsymbol{BOLU_t(j,a)}$ $\boldsymbol{BOLL_t(j,a)}$	Bollinger Bands (Upper, Lower)	Difference between current price and upper/lower Bollinger resistance level.	$\begin{cases}(P_t - [AP_t(a) + j\sigma])/\tau & if\ P_t > [AP_t(a) + j\sigma] \\ 0\ otherwise\end{cases}$ $\begin{cases}([AP_t(a) + js] - P_t)/\tau & if\ P_t < [AP_t(a) + js] \\ 0\ otherwise\end{cases}$ $j = 2, k = 5, 10, 50$ trades
$\boldsymbol{MACD_t(j,k)}$	Moving average convergence / divergence	Percentage difference between price EMA with long, k, and short, j, price lags. Based on trades.	$\frac{[AP_t(k) - AP_t(j)]}{AP_t(j)} \quad k > j$ $j, k = (5, 10), (10, 20), (15, 30)$ trades
$\boldsymbol{MACDB_t(j,k)}$ $\boldsymbol{MACDA_t(j,k)}$	Moving average convergence / divergence (Bid/Ask)	Percentage difference between BBP/BAP EMA with long, j, and short, k, price lags. Based on orders placed on the order book.	$\frac{[AP_t^{BBP}(k) - AP_t^{BBP}(j)]}{AP_t^{BBP}(j)} \quad \frac{[AP_t^{BAP}(k) - AP_t^{BAP}(j)]}{AP_t^{BAP}(j)} \quad k > j$ $j, k = (10, 20), (50, 100), (100, 200)$ events
$\boldsymbol{RSI_t(k)}$	Relative strength indicator (RSI)	Ratio of EMA in periods where price increases to EMA in periods when decreases. Based on trades executed on the order book.	$100 - \frac{AP_t(k, P^U)}{AP_t(k, P^D)}$ $k = 5, 10, 15$ trades
$\boldsymbol{RSIB_t(k)}$ $\boldsymbol{RSIA_t(k)}$	Relative strength indicator (RSI) (Bid/Ask)	Ratio of BBP/BAP EMA in periods where price increases to BBP/BAP EMA in periods when decreases. Based on orders placed on the order book.	$100 - \frac{AP_t^{BBP}(k, P^{U,BBP})}{AP_t^{BBP}(k, P^{D,BBP})} \quad 100 - \frac{AP_t^{BAP}(k, P^{U,BAP})}{AP_t^{BAP}(k, P^{D,BAP})}$ $k = 10, 50, 100, 500$ events
$\boldsymbol{SO_t(k)}$	Stochastic oscillator (SO)	Price change relative to the period minimum as a ratio of the price range over the total period of length k.	$\frac{P_t - \min(P_t^k)}{\max(P_t^k) - \min(P_t^k)}$ $k = 5, 10, 15$ trades
$\boldsymbol{SSO_t(j,k)}$	Slow stochastic oscillator (SSO)	Moving average over last j periods of stochastic oscillator.	$\frac{1}{j}\sum_{i=1}^{j} SO_{t-i}(k)$ $j = 5, \quad k = 5, 10, 15$ trades
$\boldsymbol{SOC_t(j,k)}$	Stochastic oscillator cross	Difference between SO and SSO.	$SO_t(k) - SSO_t(j, k)$ $j = 5, \ k = 5, 10, 15$ trades
$\boldsymbol{VOLC_t(j,k)}$	Chaikin volatility	Percentage change between current HLA_t and j lag HLA_t, where HLA_t is the EMA of the difference between max and min price over period k.	$\left(\frac{HLA_t(a,k)}{HLA_{t-j}(a,j,k)} - 1\right) * 100$ $HLA_t(a,j,k) = \frac{2\left(P_{t\in k}^{max} - P_{t\in k}^{min}\right) + (a-1)(HLA_{t-1})}{a+1}$ $j, k = (5, 100), (5, 1000), (10, 100), (10, 1000), (50, 1000)$ events

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467.*

Abergel, F., & Jedidi, A. (2015). Long-Time Behaviour of a Hawkes Process-Based Limit Order Book. *SIAM Journal on Financial Mathematics*, *6*(1), 1026-1043.

Agostinelli, F., Hoffman, M., Sadowski, P., & Baldi, P. (2015). Learning activation functions to improve deep neural networks. *International Conference on Learning Representations. arXiv preprint arXiv:1412.6830.*

Ahmad, F. K., Norwawi, N. M., Deris, S., & Othman, N. H. (2008). A review of feature selection techniques via gene expression profiles. In *Information Technology, 2008. ITSim 2008. International Symposium on* (Vol. 2, pp. 1-7). IEEE.

Aitken, M. J., Aspris, A., Foley, S., & Harris, F. H. D. B. (2018). Market fairness: The poor country cousin of market efficiency. *Journal of Business Ethics*, *147*(1), 5-23.

Ait-Sahalia, Y., & Sağlam, M. (2017). High frequency market making. *Working Paper. Princeton University.*

Alain, G., & Bengio, Y. (2014). What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, *15*(1), 3563-3593.

Aldridge, I. (2013). *High-frequency trading: a practical guide to algorithmic strategies and trading systems* (Vol. 604). John Wiley & Sons.

Alfonsi, A., Fruth, A., & Schied, A. (2010). Optimal execution strategies in limit order books with general shape functions. *Quantitative Finance*, *10*(2), 143-157.

Alizadeh, S., Brandt, M. W., & Diebold, F. X. (2002). Range-based estimation of stochastic volatility models. *The Journal of Finance*, *57*(3), 1047-1091.

Almgren, R., & Chriss, N. (2001). Optimal execution of portfolio transactions. *Journal of Risk*, *3*, 5-40.

Almgren, R., Thum, C., Hauptmann, E., & Li, H. (2005). Direct estimation of equity market impact. *Risk*, *18*(7), 5862.

Amihud, Y., & Mendelson, H. (1980). Dealership market: Market-making with inventory. *Journal of Financial Economics*, *8*(1), 31-53.

Amihud, Y., Mendelson, H., & Pedersen, L. H. (2006). Liquidity and asset prices. *Foundations and Trends® in Finance*, *1*(4), 269-364.

Anand, A., & Venkataraman, K. (2016). Market conditions, fragility, and the economics of market making. *Journal of Financial Economics*, *121*(2), 327-349.

Ancona, M., Ceolini, E., Oztireli, C., & Gross, M. (2018). Towards better understanding of gradient-based attribution methods for Deep Neural Networks. *International Conference on Learning Representations.*

Andersen, T. G., & Bondarenko, O. (2013). *Assessing Measures of Order Flow Toxicity via Perfect Trade Classification* (No. 2013-43). Department of Economics and Business Economics, Aarhus University.

Andersen, T. G., Bollerslev, T., Diebold, F. X., & Vega, C. (2003). Micro effects of macro announcements: Real-time price discovery in foreign exchange. *American Economic Review*, *93*(1), 38-62.

Aquilina, M., & Ysusi, C. (2016). Are high-frequency traders anticipating the order flow? Cross-venue evidence from the UK market. *FCA Occasional Paper No. 16.*

Arthur, D., & Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (pp. 1027-1035). Society for Industrial and Applied Mathematics.

Attwell, D., & Laughlin, S. B. (2001). An energy budget for signalling in the grey matter of the brain. *Journal of Cerebral Blood Flow & Metabolism*, *21*(10), 1133-1145.

Australian Securities and Investment Commission (ASIC). (2013). Dark liquidity and high-frequency trading. *Australian Securities and Investments Commission*, *84*.

Australian Securities and Investment Commission (ASIC). (2015). Review of High-Frequency Trading and Dark Liquidity. *Australian Securities and Investments Commission, 452*, 20.

Avellaneda, M., & Stoikov, S. (2008). High-frequency trading in a limit order book. *Quantitative Finance*, *8*(3), 217-224.

Axioglou, C., & Skouras, S. (2011). Markets change every day: Evidence from the memory of trade direction. *Journal of Empirical Finance*, *18*(3), 423-446.

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450.*

Ba, J., & Caruana, R. (2014). Do deep nets really need to be deep?. In *Advances in neural information processing systems* (pp. 2654-2662).

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K. R., & Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, *10*(7), e0130140.

Baddeley, A., Eysenck., Michael W., Anderson, M. (2009). *Memory.* New York: Psychology Press.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473.*

Banerjee, A., & Dave, R. N. (2004). Validating clusters using the Hopkins statistic. In *Fuzzy systems, 2004. Proceedings. 2004 IEEE international conference on* (Vol. 1, pp. 149-153). IEEE.

Baron, M., Brogaard, J., Hagströmer, B., & Kirilenko, A. (2017). Risk and return in high-frequency trading. *SSRN: 2433118.*

Barrouillet, P., Bernardin, S., & Camos, V. (2004). Time constraints and resource sharing in adults' working memory spans. *Journal of Experimental Psychology: General*, *133*(1), 83.

Bays, P. M., & Husain, M. (2008). Dynamic shifts of limited working memory resources in human vision. *Science*, *321*(5890), 851-854.

Beliakov, G., Kelarev, A., & Yearwood, J. (2011). Robust artificial neural networks and outlier detection. Technical report. *arXiv preprint arXiv:1110.0169.*

Belič, I. (2006). Neural networks and modelling in vacuum science. *Vacuum*, *80*(10), 1107-1122.

Bellman, R. (1957). *Dynamic Programming*. Princeton: Princeton University Press.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, *2*(1), 1-127.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade* (pp. 437-478). Berlin: Springer.

Bengio, Y., & LeCun, Y. (2007). Scaling learning algorithms towards AI. *Large-scale kernel machines*, *34*(5), 1-41.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, *35*(8), 1798-1828.

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *Advances in neural information processing systems* (pp. 153-160).

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, *5*(2), 157-166.

Benos, E., & Sagade, S. (2012). High-frequency trading behaviour and its impact on market quality: evidence from the UK equity market. *Bank of England. Quarterly Bulletin*, *52*(4), 370.

Bercich, J., Allan, M. (2017) Catching a falling knife: an analysis of trading halts. *FCA Insight Article.*

Bergstra, J., Komer, B., Eliasmith, C., & Warde-Farley, D. (2014). Preliminary evaluation of hyperopt algorithms on HPOLib. In *ICML workshop on AutoML.*

Berkowitz, S. A., Logue, D. E., & Noser, E. A. (1988). The total cost of transactions on the NYSE. *The Journal of Finance*, *43*(1), 97-112.

Biais, B., Hillion, P., & Spatt, C. (1995). An empirical analysis of the limit order book and the order flow in the Paris Bourse. *The Journal of Finance*, *50*(5), 1655-1689.

Biesiada, J., & Duch, W. (2007). Feature selection for high-dimensional data—a Pearson redundancy based filter. In *Computer recognition systems 2* (pp. 242-249). Springer, Berlin, Heidelberg.

Blanchet, J., & Chen, X. (2013). Continuous-time modelling of bid-ask spread and price dynamics in limit order books. *arXiv preprint arXiv:1310.1103.*

Boehmer, E., Fong, K., & Wu, J. (2015). International evidence on algorithmic trading. In *FMA 2015 annual meeting paper.*

Boehmer, E., Li, D., & Saar, G. (2017). The Competitive Landscape of High-Frequency Trading Firms. *The Review of Financial Studies.*

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory* (pp. 144-152). ACM.

Bottou, L., Curtis, F. E., & Nocedal, J. (2016). Optimization methods for large-scale machine learning. *arXiv preprint arXiv:1606.04838.*

Bouchaud, J. P., Gefen, Y., Potters, M., & Wyart, M. (2004). Fluctuations and response in financial markets: the subtle nature of 'random'price changes. *Quantitative finance*, *4*(2), 176-190.

Bouchaud, J. P., Kockelkoren, J., & Potters, M. (2006). Random walks, liquidity molasses and critical response in financial markets. *Quantitative finance*, *6*(02), 115-123.

Bouchaud, J. P., Mézard, M., & Potters, M. (2002). Statistical properties of stock order books: empirical results and models. *Quantitative finance*, *2*(4), 251-256.

Bouchaud, J.P., Farmer, J.D. and Lillo, F. (2009). How markets slowly digest changes in supply and demand. In Handbook of Financial Markets: Dynamics and Evolution. New York: Academic Press.

Bramson, M. (1998). State space collapse with application to heavy traffic limits for multiclass queueing networks. *Queueing Systems*, *30*(1-2), 89-140.

Branco, T., & Staras, K. (2009). The probability of neurotransmitter release: variability and feedback control at single synapses. *Nature Reviews Neuroscience*, *10*(5), 373.

Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123-140.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees*. Wadsworth & Brooks/Cole Advanced Books & Software. California: Pacific.

Brodersen, K. H., Ong, C. S., Stephan, K. E., & Buhmann, J. M. (2010). The balanced accuracy and its posterior distribution. In *Pattern recognition (ICPR), 2010 20th international conference on* (pp. 3121-3124). IEEE.

Brogaard, J. (2010). High frequency trading and its impact on market quality. *Northwestern University Kellogg School of Management Working Paper*, *66.*

Brogaard, J., Hendershott, T., & Riordan, R. (2014). High-frequency trading and price discovery. *The Review of Financial Studies*, *27*(8), 2267-2306.

Brogaard, J., Hendershott, T., & Riordan, R. (2017). High frequency trading and the 2008 short-sale ban. *Journal of Financial Economics*, *124*(1), 22-42.

Brunnermeier, M. K. (2009). Deciphering the liquidity and credit crunch 2007-2008. *Journal of Economic perspectives*, *23*(1), 77-100.

Buchta, C., Kober, M., Feinerer, I., & Hornik, K. (2012). Spherical k-means clustering. *Journal of Statistical Software*, *50*(10), 1-22.

Byrd, R. H., Chin, G. M., Nocedal, J., & Wu, Y. (2012). Sample size selection in optimization methods for machine learning. *Mathematical programming*, *134*(1), 127-155.

Cai, X., Zhang, N., Venayagamoorthy, G. K., & Wunsch II, D. C. (2007). Time series prediction with recurrent neural networks trained by a hybrid PSO–EA algorithm. *Neurocomputing*, *70*(13-15), 2342-2353.

Caivano, V. (2015). The Impact of High-Frequency Trading on Volatility. Evidence from the Italian Market. *CONSOB Working Papers No. 80.*

Caliński, T., & Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, *3*(1), 1-27.

Carmona, R., & Webster, K. (2012). High frequency market making. *arXiv preprint arXiv:1210.5781.*

Carrion, A. (2013). Very fast money: High-frequency trading on the NASDAQ. *Journal of Financial Markets*, *16*(4), 680-711.

Cartea, Á., Donnelly, R., & Jaimungal, S. (2018). Enhancing trading strategies with order book signals. *Applied Mathematical Finance*, 1-35.

Chaboud, A. P., Chiquoine, B., Hjalmarsson, E., & Vega, C. (2014). Rise of the machines: Algorithmic trading in the foreign exchange market. *The Journal of Finance*, *69*(5), 2045-2084.

Chang, Y. W., Hsieh, C. J., Chang, K. W., Ringgaard, M., & Lin, C. J. (2010). Training and testing low-degree polynomial data mappings via linear SVM. *Journal of Machine Learning Research*, 11, 1471-1490.

Chen, Y., Li, D., & Gao, X. (2017). Optimal Order Exposure in a Limit Order Market. *SSRN: 2938377.*

Cheney, N., Schrimpf, M., & Kreiman, G. (2017). On the Robustness of Convolutional Neural Networks to Internal Architecture and Weight Perturbations. *arXiv preprint arXiv:1703.08245.*

Cheng, E. (2017). Just 10% of trading is regular stock picking, JPMorgan estimates. Retrieved from https://www.cnbc.com/2017/06/13/death-of-the-human-investor-just-10-percent-of-trading-is-regular-stock-picking-jpmorgan-estimates.html.

Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., & Shelhamer, E. (2014). cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759.*

Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259.*

Choi, K., Fazekas, G., Cho, K., & Sandler, M. (2017). A tutorial on deep learning for music information retrieval. *arXiv preprint arXiv:1709.04396.*

Chollet, F. (2015). Keras. https://github.com/fchollet/keras.

Chordia, T., & Subrahmanyam, A. (2004). Order imbalance and individual stock returns: Theory and evidence. *Journal of Financial Economics*, *72*(3), 485-518.

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., & LeCun, Y. (2015). The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics* (pp. 192-204).

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modelling. *arXiv preprint arXiv:1412.3555.*

Clevert, D. A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289.*

Collobert, R., & Bengio, S. (2004). Links between perceptrons, MLPs and SVMs. In *Proceedings of the twenty-first international conference on Machine learning* (p. 23). ACM.

Cont, R., & De Larrard, A. (2013). Price dynamics in a Markovian limit order market. *SIAM Journal on Financial Mathematics*, *4*(1), 1-25.

Cont, R., Stoikov, S., & Talreja, R. (2010). A stochastic model for order book dynamics. *Operations research*, *58*(3), 549-563.

Cooijmans, T., Ballas, N., Laurent, C., Gülçehre, Ç., & Courville, A. (2016). Recurrent batch normalization. *arXiv preprint arXiv:1603.09025.*

Cootner, P. H. (1964). *The random character of stock market prices*. Cambridge: MIT Press.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*(3), 273-297.

Cowan, J. D., Neuman, J., & van Drongelen, W. (2016). Wilson–Cowan equations for neocortical dynamics. *The Journal of Mathematical Neuroscience*, *6*(1), 1.

Creswell, A., Arulkumaran, K., & Bharath, A. A. (2017). On denoising autoencoders trained to minimise binary cross-entropy. *arXiv preprint arXiv:1708.08487.*

Curato, G., Gatheral, J., & Lillo, F. (2017). Optimal execution with non-linear transient market impact. *Quantitative Finance*, *17*(1), 41-54.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, *2*(4), 303-314.

Dang, N. M. (2017). Optimal execution with transient impact. *Market Microstructure and Liquidity*, *3*(01), 1750008.

Daqi, G., & Genxing, Y. (2003). Influences of variable scales and activation functions on the performances of multilayer feedforward neural networks. *Pattern recognition*, *36*(4), 869-878.

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., & Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems* (pp. 2933-2941).

De Boer, P. T., Kroese, D. P., Mannor, S., & Rubinstein, R. Y. (2005). A tutorial on the cross-entropy method. *Annals of operations research*, *134*(1), 19-67.

Devarakonda, A., Naumov, M., & Garland, M. (2017). AdaBatch: Adaptive Batch Sizes for Training Deep Neural Networks. *arXiv preprint arXiv:1712.02029.*

Díaz-Uriarte, R., & De Andres, S. A. (2006). Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, *7*(1), 3.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International workshop on multiple classifier systems* (pp. 1-15). Berlin: Springer.

Directive 2004/39/EC of the European Parliament and of the Council of 21 April 2004 on markets in financial instruments amending Council Directives 85/611/EEC and 93/6/EEC and Directive 2000/12/EC of the European Parliament and of the Council and repealing Council Directive 93/22/EEC (MiFID I).

Directive 2014/65/EU on Markets in Financial Instruments (MiFID II).

Dixon, M. (2017). Sequence classification of the limit order book using recurrent neural networks. *arXiv preprint arXiv:1707.05642.*

Douglas, R. J., Koch, C., Mahowald, M., Martin, K. A., & Suarez, H. H. (1995). Recurrent excitation in neocortical circuits. *Science*, *269*(5226), 981-985.

Dozat, T. (2016). Incorporating nesterov momentum into adam. *Technical Report, Stanford University.*

Duch, W., & Jankowski, N. (1999). Survey of neural transfer functions. *Neural Computing Surveys*, *2*(1), 163-212.

Dugas, C., Bengio, Y., Bélisle, F., Nadeau, C., & Garcia, R. (2001). Incorporating second-order functional knowledge for better option pricing. In *Advances in neural information processing systems* (pp. 472-478).

Dumitrescu, E. I., & Hurlin, C. (2012). Testing for Granger non-causality in heterogeneous panels. *Economic Modelling*, *29*(4), 1450-1460.

Dunn, J. C. (1973). A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3, 32–57.

Easley, D., De Prado, M. M. L., & O'Hara, M. (2011). The microstructure of the" flash crash": Flow toxicity, liquidity crashes, and the probability of informed trading. *Journal of Portfolio Management*, *37*(2), 118.

Easley, D., Kiefer, N. M., O'hara, M., & Paperman, J. B. (1996). Liquidity, information, and infrequently traded stocks. *The Journal of Finance*, *51*(4), 1405-1436.

Easley, D., López de Prado, M. M., & O'Hara, M. (2012). Flow toxicity and liquidity in a high-frequency world. *The Review of Financial Studies*, *25*(5), 1457-1493.

Eigen, D., Rolfe, J., Fergus, R., & LeCun, Y. (2013). Understanding deep architectures using a recursive convolutional network. *arXiv preprint arXiv:1312.1847.*

Eisler, Z., Bouchaud, J. P., & Kockelkoren, J. (2011). Models for the impact of all order book events. *arXiv preprint arXiv:1107.3364.*

Eldan, R., & Shamir, O. (2016). The power of depth for feedforward neural networks. In *Conference on Learning Theory* (pp. 907-940).

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, *14*(2), 179-211.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P. A., Vincent, P., & Bengio, S. (2010). Why does unsupervised pre-training help deep learning?. *Journal of Machine Learning Research*, *11*, 625-660.

Ersan, O., & Alıcı, A. (2016). An unbiased computation methodology for estimating the probability of informed trading (PIN). *Journal of International Financial Markets, Institutions and Money*, *43*, 74-94.

ESMA/2014/1569, Final Report – ESMA's Technical Advice to the Commission on MiFID II and MiFIR (19 December 2014).

Estévez, P. A., Tesmer, M., Perez, C. A., & Zurada, J. M. (2009). Normalized mutual information feature selection. *IEEE Transactions on Neural Networks*, *20*(2), 189-201.

European Central Bank (ECB). (2017). Dark pools in European equity markets: emergence, competition and implications. *Occasional Paper Series. No. 193.*

Fama, E. F. (1998). Market efficiency, long-term returns, and behavioural finance1. *Journal of financial economics*, *49*(3), 283-306.

Fawcett, C., & Hoos, H. H. (2016). Analysing differences between algorithm configurations through ablation. *Journal of Heuristics*, *22*(4), 431-458.

Foucault, T. (1999). Order flow composition and trading costs in a dynamic limit order market1. *Journal of Financial markets*, *2*(2), 99-134.

Foucault, T., Kadan, O., & Kandel, E. (2005). Limit order book as a market for liquidity. *The review of financial studies*, *18*(4), 1171-1217.

Freitas, A. A. (2014). Comprehensible classification models: a position paper. *ACM SIGKDD explorations newsletter*, *15*(1), 1-10.

Friedlander, M. P., & Schmidt, M. (2012). Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, *34*(3), A1380-A1405.

Gal, Y., & Ghahramani, Z. (2016). A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems* (pp. 1019-1027).

Gao, X., & Deng, S. J. (2018). Hydrodynamic limit of order-book dynamics. *Probability in the Engineering and Informational Sciences*, *32*(1), 96-125.

Garman, M. B., & Klass, M. J. (1980). On the estimation of security price volatilities from historical data. *Journal of business*, 67-78.

Garson, G. D. (1991). Interpreting neural-network connection weights. *AI expert*, *6*(4), 46-51.

Gashler, M., Giraud-Carrier, C., & Martinez, T. (2008). Decision tree ensemble: Small heterogeneous is better than large homogeneous. In *Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on* (pp. 900-905). IEEE.

Gatheral, J. (2010). No-dynamic-arbitrage and market impact. *Quantitative finance*, *10*(7), 749-759.

Gatheral, J., Schied, A., & Slynko, A. (2012). Transient linear price impact and Fredholm integral equations. *Mathematical Finance*, *22*(3), 445-474.

Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural computation*, *4*(1), 1-58.

Gers, F. A., & Schmidhuber, J. (2000). Recurrent nets that time and count. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on* (Vol. 3, pp. 189-194). IEEE.

Gheyas, I. A., & Smith, L. S. (2010). Feature subset selection in large dimensionality domains. *Pattern recognition*, *43*(1), 5-13.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249-256).

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*(pp. 315-323).

Glosten, L. R., & Harris, L. E. (1988). Estimating the components of the bid/ask spread. *Journal of financial Economics*, *21*(1), 123-142.

Glosten, L. R., & Milgrom, P. R. (1985). Bid, ask and transaction prices in a specialist market with heterogeneously informed traders. *Journal of financial economics*, *14*(1), 71-100.

Goettler, R. L., Parlour, C. A., & Rajan, U. (2005). Equilibrium in a dynamic limit order market. *The Journal of Finance*, *60*(5), 2149-2192.

Goh, A. T. C. (1995). Back-propagation neural networks for modelling complex systems. *Artificial Intelligence in Engineering*, *9*(3), 143-151.

Golik, P., Doetsch, P., & Ney, H. (2013). Cross-entropy vs. squared error training: a theoretical and experimental comparison. In *Interspeech* (Vol. 13, pp. 1756-1760).

Gomber, P., Arndt, B., Lutat, M., & Uhle, T. (2011). High-frequency trading. *SSRN: 1858626.*

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., & Bengio, Y. (2013). Maxout networks. *arXiv preprint arXiv:1302.4389.*

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Vol. 1). Cambridge: MIT press.

Goodfellow, I., Lee, H., Le, Q. V., Saxe, A., & Ng, A. Y. (2009). Measuring invariances in deep networks. In *Advances in neural information processing systems* (pp. 646-654).

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Bengio, Y. et al. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672-2680).

Gorgoraptis, N., Catalao, R. F., Bays, P. M., & Husain, M. (2011). Dynamic updating of working memory resources for visual objects. *Journal of Neuroscience*, *31*(23), 8502-8511.

Gould, M. D., Porter, M. A., Williams, S., McDonald, M., Fenn, D. J., & Howison, S. D. (2013). Limit order books. *Quantitative Finance*, *13*(11), 1709-1742.

Goutte, C., & Larsen, J. (1998). Adaptive regularization of neural networks using conjugate gradient. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on* (Vol. 2, pp. 1201-1204). IEEE.

Granger, C. W. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society*, 424-438.

Graves, A., & Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning* (pp. 1764-1772).

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, *18*(5-6), 602-610.

Graves, A., Liwicki, M., Bunke, H., Schmidhuber, J., & Fernández, S. (2008). Unconstrained on-line handwriting recognition with recurrent neural networks. In *Advances in neural information processing systems* (pp. 577-584).

Graves, A., Mohamed, A. R., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on* (pp. 6645-6649). IEEE.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*, *28*(10), 2222-2232.

Gregoriou, A., Ioannidis, C., & Skerratt, L. (2005). Information Asymmetry and the Bid-Ask Spread: Evidence From the UK. *Journal of Business Finance & Accounting*, *32*(9-10), 1801-1826.

Gresse, C. (2013, June). Effects of lit and dark trading venue competition on liquidity: the MiFID experience. In *2013 FMA European Conference* (p. 49).

Groth, S. S. (2011). Does Algorithmic Trading Increase Volatility? Empirical Evidence from the Fully-Electronic Trading Platform Xetra. In *Wirtschaftsinformatik* (p. 112).

Guo, X., De Larrard, A., & Ruan, Z. (2017). Optimal placement in a limit order book: an analytical approach. *Mathematics and Financial Economics*, *11*(2), 189-213.

Guo, X., Ruan, Z., & Zhu, L. (2015). Dynamics of order positions and related queues in a limit order book. *arXiv preprint arXiv:1505.04810.*

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, *3*, 1157-1182.

Hadgu, A. T., Nigam, A., & Diaz-Aviles, E. (2015). Large-scale learning with AdaGrad on Spark. In *2015 IEEE International Conference on Big Data* (pp. 2828-2830). IEEE.

Hagströmer, B., & Norden, L. (2013). The diversity of high-frequency traders. *Journal of Financial Markets*, *16*(4), 741-770.

Halkidi, M., Batistakis, Y., & Vazirgiannis, M. (2001). On clustering validation techniques. *Journal of intelligent information systems*, *17*(2-3), 107-145.

Harris, L. (2003). *Trading and exchanges: Market microstructure for practitioners*. Mass: Oxford University Press.

Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *28*(1), 100-108.

Hasbrouck, J. (1991). Measuring the information content of stock trades. *The Journal of Finance*, *46*(1), 179-207.

Hasbrouck, J. (1995). One security, many markets: Determining the contributions to price discovery. *The journal of Finance*, *50*(4), 1175-1199.

Hasbrouck, J., & Saar, G. (2013). Low-latency trading. *Journal of Financial Markets*, *16*(4), 646-679.

Hasbrouck, J., & Sofianos, G. (1993). The trades of market makers: An empirical analysis of NYSE specialists. *The Journal of Finance*, *48*(5), 1565-1593.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning*. New York: Springer.

Hautsch, N., & Huang, R. (2012). The market impact of a limit order. *Journal of Economic Dynamics and Control*, *36*(4), 501-522.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (pp. 1026-1034).

Heaton, J. B., Polson, N. G., & Witte, J. H. (2017a). Deep learning for finance: deep portfolios. *Applied Stochastic Models in Business and Industry*, *33*(1), 3-12.

Heaton, J., McElwee, S., Fraley, J., & Cannady, J. (2017b). Early stabilizing feature importance for TensorFlow deep neural networks. In *Neural Networks (IJCNN), 2017 International Joint Conference on* (pp. 4618-4624). IEEE.

Hendershott, T., & Riordan, R. (2013). Algorithmic trading and the market for liquidity. *Journal of Financial and Quantitative Analysis*, *48*(4), 1001-1024.

Hendershott, T., Jones, C. M., & Menkveld, A. J. (2011). Does algorithmic trading improve liquidity?. *The Journal of Finance*, *66*(1), 1-33.

Hermans, M., & Schrauwen, B. (2013). Training and analysing deep recurrent neural networks. In *Advances in neural information processing systems* (pp. 190-198).

Heskes, T. M., & Kappen, B. (1993). On-line learning processes in artificial neural networks. In *North-Holland Mathematical Library* (Vol. 51, pp. 199-233). Elsevier.

Hinton, G. E. (2012c). A practical guide to training restricted Boltzmann machines. In *Neural networks: Tricks of the trade* (pp. 599-619). Springer, Berlin, Heidelberg.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, *313*(5786), 504-507.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012b). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580.*

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N., Kingsbury, B. et al. (2012a). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, *29*(6), 82-97.

Ho, T. K. (1995). Random decision forests. In *Document analysis and recognition, 1995., proceedings of the third international conference on* (Vol. 1, pp. 278-282). IEEE.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, *20*(8), 832-844.

Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen Netzen. *Diploma, Technische Universität München*, *91*, 1.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *6*(02), 107-116.

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

Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural networks*, *4*(2), 251-257.

Howard, R. (1960). *Dynamic Programming and Markov Processes.* MIT Press, Cambridge, MA.

Huh, Y. (2014). Machines vs. machines: High frequency trading and hard information. *Finance and Economics Discussion Series No 2014-33.* Board of Governors of the Federal Reserve System (U.S.).

Hyvärinen, A., & Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural networks*, *13*(4-5), 411-430.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167.*

Jain, A., & Zongker, D. (1997). Feature selection: Evaluation, application, and small sample performance. *IEEE transactions on pattern analysis and machine intelligence*, *19*(2), 153-158.

Janocha, K., & Czarnecki, W. M. (2017). On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659.*

Jaques, N., Gu, S., Turner, R. E., & Eck, D. (2017). Tuning recurrent neural networks with reinforcement learning. *arXiv preprint arXiv:1611.02796v2.*

Jarnecic, E., & Snape, M. (2010). An analysis of trades by high frequency participants on the London Stock Exchange. In *17th Annual Conference of the Multinational Finance Society MFS* (Vol. 2010).

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., & Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia* (pp. 675-678). ACM.

Johnson, B. (2010). *Algorithmic trading and DMA.* London: 4Myeloma Press.

Jovanovic, B., & Menkveld, A. J. (2016). Middlemen in limit order markets. *SSRN: 1624329.*

Jozefowicz, R., Zaremba, W., & Sutskever, I. (2015). An empirical exploration of recurrent network architectures. In *International Conference on Machine Learning* (pp. 2342-2350).

Kempf, A., Mayston, D., Gehde-Trapp, M., & Yadav, P. (2015). Resiliency: A dynamic view of liquidity. *CFR Working Papers 15-04.* University of Cologne, Centre for Financial Research.

Kercheval, A. N., & Zhang, Y. (2015). Modelling high-frequency limit order book dynamics with support vector machines. *Quantitative Finance*, *15*(8), 1315-1329.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2017). On large-batch training for deep learning: Generalization gap and sharp minima. *In ICLR'2017. arXiv preprint arXiv:1609.04836.*

Khaidem, L., Saha, S., & Dey, S. R. (2016). Predicting the direction of stock market prices using random forest. *arXiv preprint arXiv:1605.00003.*

Kim, O., & Verrecchia, R. E. (1994). Market liquidity and volume around earnings announcements. *Journal of accounting and economics*, *17*(1-2), 41-67.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *International Conference on Learning Representations. arXiv preprint arXiv:1412.6980.*

Kiran, B. R., & Serra, J. (2017). Cost-complexity pruning of random forests. In *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing* (pp. 222-232). Springer, Cham.

Kirilenko, A., Kyle, A. S., Samadi, M., & Tuzun, T. (2017). The Flash Crash: High-frequency trading in an electronic market. *The Journal of Finance*, *72*(3), 967-998.

Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-normalizing neural networks. In *Advances in Neural Information Processing Systems* (pp. 972-981).

Klöck, F., Schied, A., & Sun, Y. (2011). Price manipulation in a market impact model with dark pool. *SSRN: 1785409.*

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial intelligence*, *97*(1-2), 273-324.

Kontschieder, P., Fiterau, M., Criminisi, A., & Bulo, S. R. (2015). Deep neural decision forests. In *Computer Vision (ICCV), 2015 IEEE International Conference on* (pp. 1467-1475). IEEE.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Krogh, A., & Hertz, J. A. (1992). A simple weight decay can improve generalization. In *Advances in neural information processing systems* (pp. 950-957).

Kumar, S. K. (2017). On weight initialization in deep neural networks. *arXiv preprint arXiv:1704.08863.*

Kuncheva, L. I., & Whitaker, C. J. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, *51*(2), 181-207.

Kyle, A. S. (1985). Continuous auctions and insider trading. *Econometrica: Journal of the Econometric Society*, 1315-1335.

Large, J. (2007). Measuring the resiliency of an electronic limit order book. *Journal of Financial Markets*, *10*(1), 1-25.

Lawson, R. G., & Jurs, P. C. (1990). New index for clustering tendency and its application to chemical problems. *Journal of chemical information and computer sciences*, *30*(1), 36-41.

Le, Q. V., Jaitly, N., & Hinton, G. E. (2015). A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941.*

LeBaron, B., & Yamamoto, R. (2008). The impact of imitation on long memory in an order-driven market. *Eastern Economic Journal*, *34*(4), 504-517.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436.

LeCun, Y., Bottou, L., Orr, G. B., & Müller, K. R. (1998). Efficient backprop. In *Neural networks: Tricks of the trade* (pp. 9-50). Berlin: Springer.

Lee, C. Y., Xie, S., Gallagher, P., Zhang, Z., & Tu, Z. (2015). Deeply-supervised nets. In *Artificial Intelligence and Statistics* (pp. 562-570).

Lennie, P. (2003). The cost of cortical computation. *Current biology*, *13*(6), 493-497.

Liang, M., & Hu, X. (2015). Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3367-3375).

Liew, S. S., Khalil-Hani, M., & Bakhteri, R. (2016). Bounded activation functions for enhanced training stability of deep neural networks on visual pattern recognition problems. *Neurocomputing*, *216*, 718-734.

Lillo, F., & Farmer, J. D. (2004). The long memory of the efficient market. *Studies in nonlinear dynamics & econometrics*, *8*(3).

Lillo, F., Farmer, J. D., & Mantegna, R. N. (2003). Econophysics: Master curve for price-impact function. *Nature*, *421*(6919), 129.

Lillo, F., Mike, S., & Farmer, J. D. (2005). Theory for long memory in supply and demand. *Physical review*, *71*(6), 066122.

Lin, H. W. W., & Ke, W. C. (2011). A computing bias in estimating the probability of informed trading. *Journal of Financial Markets*, *14*(4), 625-640.

Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019.*

Liu, H., & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on knowledge and data engineering*, *17*(4), 491-502.

Liu, H., Motoda, H., Setiono, R., & Zhao, Z. (2010). Feature selection: An ever evolving frontier in data mining. In *Feature Selection in Data Mining* (pp. 4-13).

Lo, D. K., & Hall, A. D. (2015). Resiliency of the limit order book. *Journal of Economic Dynamics and Control*, *61*, 222-244.

London Stock Exchange Group (LSEG). (2017). *Millenium Exchange Business Paramaters.* Retrieved from: http://www.londonstockexchange.com/products-and-services/technical-library/millennium-exchange-technical-specifications/millennium-exchange-technical-specifications.htm.

London Stock Exchange Group (LSEG). (2018). *Rules of the London Stock Exchange.* Retrieved from: https://www.londonstockexchange.com/traders-and-brokers/rules-regulations/rules-lse.pdf.

Lu, Z., Pu, H., Wang, F., Hu, Z., & Wang, L. (2017). The Expressive Power of Neural Networks: A View from the Width. In *Advances in Neural Information Processing Systems* (pp. 6232-6240).

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems* (pp. 4768-4777).

Ma, J., Sheridan, R. P., Liaw, A., Dahl, G. E., & Svetnik, V. (2015). Deep neural nets as a method for quantitative structure–activity relationships. *Journal of chemical information and modeling*, *55*(2), 263-274.

Ma, W. J., Husain, M., & Bays, P. M. (2014). Changing concepts of working memory. *Nature neuroscience*, *17*(3), 347.

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml* (Vol. 30, No. 1, p. 3).

Maass, W., Joshi, P., & Sontag, E. D. (2007). Computational aspects of feedback in neural circuits. *PLoS computational biology*, *3*(1), e165.

Maclaurin, D., Duvenaud, D., & Adams, R. (2015). Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning* (pp. 2113-2122).

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (Vol. 1, No. 14, pp. 281-297).

Madhavan, A. (2000). Market microstructure: A survey. *Journal of financial markets*, *3*(3), 205-258.

Maglaras, C., Moallemi, C. C., & Zheng, H. (2015). Optimal execution in a limit order book and an associated microstructure market impact model. *Columbia Business School Research Paper No. 15-60.*

Malinova, K., Park, A., & Riordan, R. (2013). Do retail traders suffer from high frequency traders. *SSRN: 2183806.*

Mandal, P., Senjyu, T., Urasaki, N., & Funabashi, T. (2006). A neural network based several-hour-ahead electric load forecasting using similar days approach. *International Journal of Electrical Power & Energy Systems*, *28*(6), 367-373.

Martens, J. (2010). Deep learning via Hessian-free optimization. In *ICML* (Vol. 27, pp. 735-742).

Maslov, S. (2000). Simple model of a limit order-driven market. *Physica A: Statistical Mechanics and its Applications*, *278*(3-4), 571-578.

Masters, T. (1993). *Practical neural network recipes in C++*. Morgan Kaufmann. Academic Press: San Diego.

McCullagh, P. (1984). Generalized linear models. *European Journal of Operational Research*, *16*(3), 285-292.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115-133.

Menkveld, A. J. (2013). High frequency trading and the new market makers. *Journal of Financial Markets*, *16*(4), 712-740.

Mikolov, T. (2012). Statistical language models based on neural networks. *Brno University of Technology.*

Mikolov, T., Joulin, A., & Baroni, M. (2016). A roadmap towards machine intelligence. In *International Conference on Intelligent Text Processing and Computational Linguistics* (pp. 29-61). Cham: Springer.

Miller, K., Hettinger, C., Humpherys, J., Jarvis, T., & Kartchner, D. (2017). Forward Thinking: Building Deep Random Forests. *arXiv preprint arXiv:1705.07366.*

Mishkin, D., & Matas, J. (2015). All you need is a good init. *arXiv preprint arXiv:1511.06422.*

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., & Petersen, S. et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529-533.

Mohamad, M. A., Nasien, D., Hassan, H., & Haron, H. (2015). A review on feature extraction and feature selection for handwritten character recognition. *International Journal of Advanced Computer Science and Applications*, *6*(2), 204-212.

Montavon, G., Lapuschkin, S., Binder, A., Samek, W., & Müller, K. R. (2017). Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, *65*, 211-222.

Montufar, G. F., Pascanu, R., Cho, K., & Bengio, Y. (2014). On the number of linear regions of deep neural networks. In *Advances in neural information processing systems* (pp. 2924-2932).

Moon, T., Choi, H., Lee, H., & Song, I. (2015). Rnndrop: A novel dropout for rnns in asr. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on* (pp. 65-70). IEEE.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (pp. 807-814).

Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate O (1/k2). In *Soviet Mathematics Doklady* (Vol. 27, No. 2, pp. 372-376).

Ng, A. Y. (2004). Feature selection, L 1 vs. L 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning* (p. 78). ACM.

Nister, D., & Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on* (Vol. 2, pp. 2161-2168). Ieee.

Ntakaris, A., Magris, M., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2017). Benchmark dataset for mid-price prediction of limit order book data. *arXiv preprint arXiv:1705.03233.*

O’Hara, M. (2015). High frequency market microstructure. *Journal of Financial Economics*, *116*(2), 257-270.

Obizhaeva, A. A., & Wang, J. (2013). Optimal trading strategy and supply/demand dynamics. *Journal of Financial Markets*, *16*(1), 1-32.

O'Hara, M. (1995). *Market microstructure theory* (Vol. 108). Cambridge, MA: Blackwell Publishers.

O'Hara, M. (2003). Presidential address: Liquidity and price discovery. *The Journal of Finance*, *58*(4), 1335-1354.

Olden, J. D., Joy, M. K., & Death, R. G. (2004). An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data. *Ecological Modelling*, *178*(3-4), 389-397.

Oshiro, T. M., Perez, P. S., & Baranauskas, J. A. (2012). How many trees in a random forest?. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition* (pp. 154-168). Springer, Berlin, Heidelberg.

Pachitariu, M., & Sahani, M. (2013). Regularization and nonlinearities for neural language models: when are they needed?. *arXiv preprint arXiv:1301.5650.*

Palmer, J. (1990). Attentional limits on the perception and memory of visual information. *Journal of Experimental Psychology: Human Perception and Performance*, *16*(2), 332.

Papernot, N., McDaniel, P., Wu, X., Jha, S., & Swami, A. (2016). Distillation as a defence to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on* (pp. 582-597). IEEE.

Parkinson, M. (1980). The extreme value method for estimating the variance of the rate of return. *Journal of Business*, 61-65.

Parlour, C. A. (1998). Price dynamics in limit order markets. *The Review of Financial Studies*, *11*(4), 789-816.

Pascanu, R., & Bengio, Y. (2014). Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584.*

Pascanu, R., Gulcehre, C., Cho, K., & Bengio, Y. (2013). How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026.*

Pasquariello, P., & Vega, C. (2007). Informed and strategic order flow in the bond markets. *The Review of Financial Studies*, *20*(6), 1975-2019.

Pástor, Ľ., & Stambaugh, R. F. (2003). Liquidity risk and expected stock returns. *Journal of Political economy*, *111*(3), 642-685.

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).

Perold, A. F. (1988). The implementation shortfall: Paper versus reality. *The Journal of Portfolio Management*, *14*(3), 4-9.

Peterson, A. D., Ghosh, A. P., & Maitra, R. (2018). Merging K-means with hierarchical clustering for identifying general-shaped groups. *Stat*, *7*(1), e172.

Phaisangittisagul, E. (2016). An analysis of the regularization between L2 and dropout in single hidden layer neural network. In *Intelligent Systems, Modelling and Simulation (ISMS), 2016 7th International Conference on* (pp. 174-179). IEEE.

Pham, V., Bluche, T., Kermorvant, C., & Louradour, J. (2014). Dropout improves recurrent neural networks for handwriting recognition. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on* (pp. 285-290). IEEE.

Povey, D., Zhang, X., & Khudanpur, S. (2015). Parallel training of DNNs with natural gradient and parameter averaging. *arXiv preprint arXiv:1410.7455.*

Prechelt, L. (2012). Early stopping—but when?. In *Neural networks: tricks of the trade* (pp. 53-67). Berlin: Springer.

Puheim, M., Nyulászi, L., Madarász, L., & Gašpar, V. (2014). On practical constraints of approximation using neural networks on current digital computers. In *Intelligent Engineering Systems (INES), 2014 18th International Conference on* (pp. 257-262). IEEE.

Raileanu, L. E., & Stoffel, K. (2004). Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, *41*(1), 77-93.

Ramachandran, P., Zoph, B., & Le, Q. V. (2018). Searching for activation functions. *arXiv preprint arXiv:1710.05941.*

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386.*

Riordan, R., & Storkenmaier, A. (2012). Latency, liquidity and price discovery. *Journal of Financial Markets*, *15*(4), 416-437.

Rish, I. (2001). An empirical study of the naive Bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence* (Vol. 3, No. 22, pp. 41-46). IBM.

Rokach, L., & Maimon, O. (2005). Decision trees. In *Data mining and knowledge discovery handbook* (pp. 165-192). Boston, MA: Springer.

Roll, R. (1984). A simple implicit measure of the effective bid-ask spread in an efficient market. *The Journal of finance*, *39*(4), 1127-1139.

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., & Bengio, Y. (2015). Fitnets: Hints for thin deep nets. *International Conference on Learning Representations.*

Rosasco, L., Vito, E. D., Caponnetto, A., Piana, M., & Verri, A. (2004). Are loss functions all the same?. *Neural Computation*, *16*(5), 1063-1076.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386.

Roşu, I. (2009). A dynamic model of the limit order book. *The Review of Financial Studies*, *22*(11), 4601-4641.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, *20*, 53-65.

Ruder, S. (2017). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747.*

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533.

Rummery, G. A., & Niranjan, M. (1994). *On-line Q-learning using connectionist systems* (Vol. 37). University of Cambridge, Department of Engineering.

Russell, S. J., & Norvig, P. (2016). *Artificial intelligence: a modern approach.* Malaysia: Pearson Education Limited.

Ruxton, G. D. (2006). The unequal variance t-test is an underused alternative to Student's t-test and the Mann–Whitney U test. *Behavioral Ecology*, *17*(4), 688-690.

Saerens, M., Latinne, P., & Decaestecker, C. (2002). Any reasonable cost function can be used for a posteriori probability approximation. *IEEE transactions on neural networks*, *13*(5), 1204-1210.

Sak, H., Senior, A., & Beaufays, F. (2014). Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128.*

Salehinejad, H., Baarbe, J., Sankar, S., Barfett, J., Colak, E., & Valaee, S. (2018). Recent Advances in Recurrent Neural Networks. *arXiv preprint arXiv:1801.01078.*

Sandås, P. (2001). Adverse selection and competitive market making: Empirical evidence from a limit order market. *The review of financial studies*, *14*(3), 705-734.

Sarkissian, J. (2016). Spread, volatility, and volume relationship in financial markets and market making profit optimization. *arXiv preprint arXiv:1606.07381.*

Sarle, W. (1995). Stopped training and other remedies for overfitting, SAS Institute Inc. In *Proceedings of the 27th Symposium on Interface.*

Saxe, A. M., McClelland, J. L., & Ganguli, S. (2014). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *International Conference on Learning Representations. arXiv preprint arXiv:1312.6120.*

Schäfer, A. M., & Zimmermann, H. G. (2007). Recurrent neural networks are universal approximators. *International journal of neural systems*, *17*(04), 253-263.

Securities and Exchange Commission (SEC). (2010). Findings regarding the market events of May 6, 2010. *Report of the Staffs of the CFTC and SEC to the Joint Advisory Committee on Emerging Regulatory Issues.*

Seide, F., Li, G., & Yu, D. (2011). Conversational speech transcription using context-dependent deep neural networks. In *Twelfth Annual Conference of the International Speech Communication Association.*

Senior, A., Heigold, G., & Yang, K. (2013). An empirical study of learning rates in deep neural networks for speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on* (pp. 6724-6728). IEEE.

Serre, T., Kreiman, G., Kouh, M., Cadieu, C., Knoblich, U., & Poggio, T. (2007). A quantitative theory of immediate visual recognition. *Progress in brain research*, *165*, 33-56.

Shah, A., Kadam, E., Shah, H., Shinde, S., & Shingade, S. (2016). Deep residual networks with exponential linear unit. In *Proceedings of the Third International Symposium on Computer Vision and the Internet* (pp. 59-65). ACM.

Shek, H. H. S. (2011). Modeling high frequency market order dynamics using self-excited point process. *SSRN: 1668160.*

Shrikumar, A., Greenside, P., & Kundaje, A. (2017). Learning important features through propagating activation differences. *arXiv preprint arXiv:1704.02685.*

Siegelmann, H. T., & Sontag, E. D. (1995). On the computational power of neural nets. *Journal of computer and system sciences*, *50*(1), 132-150.

Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034.*

Singh, S., Jaakkola, T., Littman, M. L., & Szepesvári, C. (2000). Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine learning*, *38*(3), 287-308.

Sirignano, J. (2016). Deep learning for limit order books. *arXiv preprint arXiv:1601.01987.*

Smith, R. (2010). Is high-frequency trading inducing changes in market microstructure and dynamics?. *SSRN: 1632077.*

Smith, T., & Whaley, R. E. (1994). Estimating the effective bid/ask spread from time and sales data. *Journal of Futures Markets*, *14*(4), 437-455.

Sodhi, S. S., & Chandra, P. (2014). Bi-modal derivative activation function for sigmoidal feedforward networks. *Neurocomputing*, *143*, 182-196.

Song, Q., Ni, J., & Wang, G. (2013). A fast clustering-based feature subset selection algorithm for high-dimensional data. *IEEE transactions on knowledge and data engineering*, *25*(1), 1-14.

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806.*

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929-1958.

Strehl, A., Ghosh, J., & Mooney, R. (2000). Impact of similarity measures on web-page clustering. In *Workshop on artificial intelligence for web search (AAAI 2000)* (Vol. 58, p. 64).

Strobl, C., Malley, J., & Tutz, G. (2009). An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests. *Psychological methods*, *14*(4), 323.

Subrahmanyam, A., & Zheng, H. (2016). Limit order placement by high-frequency traders. *Borsa Istanbul Review*, *16*(4), 185-209.

Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic attribution for deep networks. *arXiv preprint arXiv:1703.01365.*

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning* (pp. 1139-1147).

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104-3112).

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction* (Vol. 1, No. 1). Cambridge: MIT press.

Sutton, R. S., & Barto, A. G. (2012). *Reinforcement learning: An introduction* (Second Ed. Draft). Cambridge: MIT press.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks. *International Conference on Learning Representations. arXiv preprint arXiv:1312.6199.*

Székely, G. J., Rizzo, M. L., & Bakirov, N. K. (2007). Measuring and testing dependence by correlation of distances. *The annals of statistics*, 2769-2794.

TABB Group. (2011). Breaking down the UK equity market. *Research Report.*

Tang, Y. (2013). Deep learning using linear support vector machines. *arXiv preprint arXiv:1306.0239.*

Taranto, D. E., Bormetti, G., Bouchaud, J. P., Lillo, F., & Toth, B. (2016). Linear models for the impact of order flow on prices I. Propagators: Transient vs. History Dependent Impact. *arXiv preprint arXiv:1602.02735.*

Telgarsky, M. (2016). Benefits of depth in neural networks. *arXiv preprint arXiv:1602.04485.*

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 267-288.

Tibshirani, R., Walther, G., & Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *63*(2), 411-423.

Tieleman, T., & Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, *4*(2), 26-31.

Tikhonov, A. N. (1943). On the stability of inverse problems. In *Dokl. Akad. Nauk SSSR* (Vol. 39, pp. 195-198).

Toke, I. M., & Pomponio, F. (2012). Modelling trades-through in a limit order book using Hawkes processes. *Economics: The Open-Access, Open-Assessment E-Journal*, vol-6.

Toth, B., Palit, I., Lillo, F., & Farmer, J. D. (2015). Why is equity order flow so persistent?. *Journal of Economic Dynamics and Control*, *51*, 218-239.

Unterthiner, T., Mayr, A., Klambauer, G., & Hochreiter, S. (2015). Toxicity prediction using deep learning. *arXiv preprint arXiv:1503.01445.*

Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep Reinforcement Learning with Double Q-Learning. In *AAAI* (Vol. 16, pp. 2094-2100).

van Laarhoven, T. (2017). L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350.*

Vapnik, V. (1963). Pattern recognition using generalized portrait method. *Automation and remote control*, *24*, 774-780.

Velez, D. R., White, B. C., Motsinger, A. A., Bush, W. S., Ritchie, M. D., Williams, S. M., & Moore, J. H. (2007). A balanced accuracy function for epistasis modeling in imbalanced datasets using multifactor dimensionality reduction. *Genetic epidemiology*, *31*(4), 306-315.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P. A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, *11*(Dec), 3371-3408.

Vrieze, S. I. (2012). Model selection and psychological theory: a discussion of the differences between the Akaike information criterion (AIC) and the Bayesian information criterion (BIC). *Psychological methods*, *17*(2), 228.

Wang, C., Venkatesh, S. S., & Judd, J. S. (1994). Optimal stopping and effective machine complexity in learning. In *Advances in neural information processing systems* (pp. 303-310).

Wang, W., Yang, X., Ooi, B. C., Zhang, D., & Zhuang, Y. (2016). Effective deep learning-based multi-modal retrieval. *The VLDB Journal*, *25*(1), 79-101.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards* (Doctoral dissertation, King's College, Cambridge).

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning*, *8*(3-4), 279-292.

Werbos, P. (1974). Beyond regression: new fools for prediction and analysis in the behavioural sciences. *PhD thesis, Harvard University.*

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, *78*(10), 1550-1560.

Weston, J., Ratle, F., Mobahi, H., & Collobert, R. (2012). Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade* (pp. 639-655). Springer, Berlin, Heidelberg.

Willmore, B., & Tolhurst, D. J. (2000). Sparseness and kurtosis of computational models of simple-cell coding in primary visual cortex. *Journal of Physiology*, *526*, 158-158.

Wilson, D. R., & Martinez, T. R. (2003). The general inefficiency of batch training for gradient descent learning. *Neural Networks*, *16*(10), 1429-1451.

Witten, D. M., & Tibshirani, R. (2010). A framework for feature selection in clustering. *Journal of the American Statistical Association*, *105*(490), 713-726.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, *1*(1), 67-82.

Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853.*

Yang, J., & Honavar, V. (1998). Feature subset selection using a genetic algorithm. In *Feature extraction, construction and selection* (pp. 117-136). Boston, MA: Springer.

Yang, S. Y., Qiao, Q., Beling, P. A., Scherer, W. T., & Kirilenko, A. A. (2015). Gaussian process-based algorithmic trading strategy identification. *Quantitative Finance*, *15*(10), 1683-1703.

Yao, K., Cohn, T., Vylomova, K., Duh, K., & Dyer, C. (2015). Depth-gated recurrent neural networks. *arXiv preprint arXiv:1508.03790.*

Yao, Y., Rosasco, L., & Caponnetto, A. (2007). On early stopping in gradient descent learning. *Constructive Approximation*, *26*(2), 289-315.

Zagoruyko, S., & Komodakis, N. (2017). DiracNets: training very deep neural networks without skip-connections. *arXiv preprint arXiv:1706.00388.*

Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701.*

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818-833). Cham: Springer.

Zeiler, M. D., Ranzato, M., Monga, R., Mao, M., Yang, K., Le, Q. V., Hinton, G. E. et al. (2013). On rectified linear units for speech processing. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference* (pp. 3517-3521).

Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. In ICLR'2017. arXiv preprint arXiv:*1611.03530.*

Zhang, S., & Riordan, R. (2011). Technology and market quality: the case of high frequency trading. In *Proc ECIS.* Paper 95.

Zheng, H., Yang, Z., Liu, W., Liang, J., & Li, Y. (2015). Improving deep neural networks using softplus units. In *Neural Networks (IJCNN), 2015 International Joint Conference on*(pp. 1-4). IEEE.

Zheng, S., Song, Y., Leung, T., & Goodfellow, I. (2016). Improving the robustness of deep neural networks via stability training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4480-4488).

Zhou, Z. H., & Feng, J. (2017). Deep forest: Towards an alternative to deep neural networks. *arXiv preprint arXiv:1702.08835.*

Zhu, M., Atri, S., & Yegen, E. (2016). Are candlestick trading strategies effective in certain stocks with distinct features?. *Pacific-Basin Finance Journal*, *37*, 116-127.

Zhuangzhuang, T., Ronghui, Z., Jiemin, H., & Jun, Z. (2016). Adaptive learning rate CNN for SAR ATR. In *Radar (RADAR), 2016 CIE International Conference on* (pp. 1-5).

Zintgraf, L. M., Cohen, T. S., Adel, T., & Welling, M. (2017). Visualizing deep neural network decisions: Prediction difference analysis. *arXiv preprint arXiv:1702.04595.*