High-Frequency Jump Prediction Using Deep Learning Techniques

Author: Rune Langergaard

December 15, 2019

Abstract

This paper studies modern prediction techniques' ability to forecast stock price jumps in forthcoming minutes, using features extracted from the limit order book. Financial time-series require complex modeling techniques as they display both linear and non-linear dependencies over time. An advanced technique that can capture these dependencies is the long short term memory (LSTM) neural network model. LSTM is a very flexible technique which can capture linear and non-linear dependencies amongst features, while also capturing the time-dependencies in financial time-series data. High-frequency data is studied at half-minute, one-minute, and five-minute observations to predict whether a jump is going to happen in the next given interval. Jumps are detected using a non-parametric test with a microstructure noise bias correction. This test will be able to detect significant discontinuities in the price path. Recent studies have found limit order books to provide useful information when performing predictions, and this study finds similar results to these. The LSTM neural network is the highest performing model with an F1-score of 0.72 and a Cohen's kappa score of 0.7. Added complexity to the neural network is not found to increase performance. Adding attention to the model can increase the recall-score, but will cause the model to focus too much on certain features. Given this, the attention mechanism causes the model to make more false-positive predictions, and overall this will lower the F1-score. The LSTM model finds a better prediction balance, ultimately leading to it being the highest performing model at all sampling-frequencies. Using limit order book information and deep learning models it is possible to make viable predictions of jumps.

1 Introduction

Despite the efficient market hypothesis claiming that price movements cannot be predicted, limit order book (LOB) systems have been found to do so (Tsantekidis et al., 2017b; Mäkinen et al., 2019; Ntakaris, 2019; Kercheval and Zhang, 2015; Sirignano and Cont, 2019). LOB systems take in limit order submissions that specify the desired price and quantity to buy or sell securities at, which can provide useful information about the current state of the market. Orders are nowadays observed at extremely high frequencies where data is collected down to nanoseconds. This has brought new opportunities to study the behavior of the financial markets using much more data. In recent years high-frequency data have been widely used to forecast volatility (Ghysels et al., 2006; Andersen et al., 2004), but recently also to forecast price movements and jumps (Ntakaris, 2019; Tsantekidis et al., 2017b,a; Mäkinen, 2017; Mäkinen et al., 2019; Kercheval and Zhang, 2015; Dixon, 2018). The study of high-frequency data has attracted new tractable and data-driven approaches that rely less on the strict assumptions that might have been violated by the microstructure noise (Hansen and Lunde, 2006). Amongst these models span a wide array of machine learning techniques. These machine learning methods have found evidence that limit order book data can be used to predict price movements in stock markets (Cont et al., 2010; Cont, 2011; Kercheval and Zhang, 2015; Ntakaris et al., 2017; Tsantekidis et al., 2017b; Passalis et al., 2017; Mäkinen et al., 2019; Tran et al., 2018; Dixon, 2018; Sirignano and Cont, 2019; Zheng et al., 2012; Mäkinen, 2017; Ntakaris, 2019). Studying the posted limit orders will give an overview of the current spread, arrivals of orders, volumes on bid and ask prices, etc (see section 4.2). This information can be used to make strong prediction models.

Generally, it is deemed as very difficult to make strong predictions on financial time-series (Kara et al., 2011). Malkiel and Fama (1970) states with the efficient market hypothesis that it is not possible to predict future prices using past information. However, the learning property of machine learning models allow them to adjust their parameters depending on the current state of the market. Neural networks are one of the few algorithms that consistently have been able to give strong predictions on time-series data (Graves, 2012; Kara et al., 2011; Tsantekidis et al., 2017b; Mäkinen et al., 2019; Mäkinen, 2017; Tsantekidis et al., 2017b; Gers et al., 2002; Malhotra et al., 2015; Karim et al., 2017; Lipton et al., 2015b).

This paper studies jumps in high-frequency data at different time intervals and their predictability using the advanced neural network prediction technique. Stock price jumps are significant discontinuities in the price path, which means the realized return is much larger in absolute size than the usual continuous prices. Literature has found evidence of jumps being present in stock markets and reflecting the arrival of news (Eraker, 2004; Lee, 2011; Yang and Kanniainen, 2017; Kanniainen and Yue, 2019). Therefore, prediction of jumps can be used for both predicting arrival of new information and also for option price modeling (Tankov, 2003; Cont et al., 2007; Christensen et al., 2014; Piazzesi, 2005). Additionally, jumps also play a factor in risk management through the optimal hedging strategy. Multiple tests have been developed to detect jumps inside high-frequency time intervals (Barndorff-Nielsen and Shephard, 2006; Lee and Mykland, 2007, 2012; Xue et al., 2014; Jiang and Oomen, 2005, 2008). a widely used non-parametric jump test by Lee and Mykland (2007) can in principle detect jumps for each observation. This test uses bipower variation (BPV) that is robust to jumps, which in turn will make the test for each observation independent of the other tests. Lee and Mykland (2007) uses the distribution in the absence of jumps to calculate the rejection region to detect when there has been a jump. Jiang and Oomen (2008) propose a bias correction of the BPV to account for microstructure noise, while other tests use the pre-averaging estimator proposed by (Jacod et al., 2009) or utilize wavelet properties Xue et al. (2014).

A by-product of the Lee and Mykland (2007) test is also the ability to estimate the direction and size of the detected jumps, which allows portfolio managers to adjust their hedging strategies accordingly. Being able to predict when jumps are going to happen can make this dynamic hedging portfolio re-balancing even stronger (Noorian and Leong, 2014).

Given the ability to make strong predictions about non-negligible differences in the price (i.e. jumps), then you can make trading strategies based on this. Furthermore, Barndorff-Nielsen and Shephard (2006) also mention the advantage of being able to detect jumps as it has important implications for risk management and asset allocation. Considering the advantageous implications jump detection can have, it can pose similar advantages to also predict them prior to taking place (Mäkinen, 2017; Mäkinen et al., 2019; Xue et al., 2014; Zheng et al., 2012; Ntakaris, 2019; Kercheval and Zhang, 2015; Kanniainen and Yue, 2019; Kara et al., 2011; Tsantekidis et al., 2017b).

To analyze the deployed neural network models a dataset of high-frequency quote and trade data for twenty-three of the largest stocks on the Dow Jones Industrial Average (DJIA) as well as the exchange-traded fund of the S&P500 (SPY), totaling twenty-four different assets is used. Furthermore, the models are deployed for three different prediction-frequencies[1]. The used models are run over the different assets and frequencies to get a better study of their performance and the variability hereof.

[1] 30 seconds, 60 seconds and 300 seconds (5 minutes)

2 Limit order books

In limit order books, buy and sell orders are placed together with the desired quantity. You can either post a limit order or a market order. Bid orders are placed with a buying price below the current market price for a certain quantity. Ask orders are placed with an asking price above the current market price. Market orders are orders that are executed immediately meaning that they match a standing limit order. The quote data in the TAQ database consist of the current bid and ask prices throughout the whole trading day. After applying the data cleaning rules of Barndorff-Nielsen et al. (2009) it can be studied how the orders are spread throughout a day.

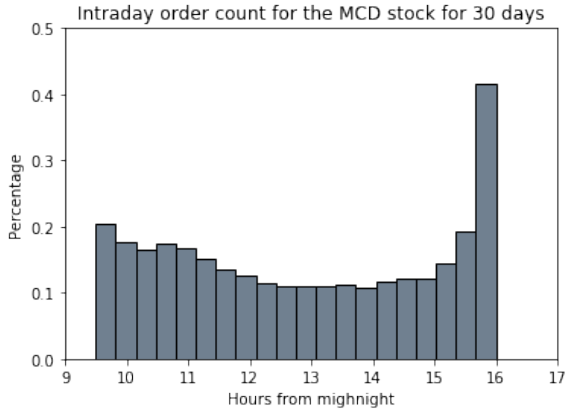

Figure 1: Density of intraday arrival of orders over thirty days for the MCD stock.

As can be seen in fig. 1 the quantity or orders at the beginning and end of the trading day is higher. Especially around closing time the number of orders is substantially different. This is the general pattern observed when studying the different stocks. Therefore, the time of the day becomes an important feature in the model. As observed by Cont et al. (2014), the volatility is changing throughout the day and this creates a different impact of trades on the price during the day. Likewise, the same behavior as seen in fig. 1 is observed if you plot intraday jumps. Jumps are more frequent during the start and end of the day.

2.1 Modelling stock prices with jumps

The stock price is assumed to follow an Itô process, which is an example of a semi-martingale (Mykland and Zhang, 2012; Aït-Sahalia and Jacod, 2014). The price formula without jumps is written as

$$X_t = X_0 + \int_0^t \mu_s ds + \int_0^t \sigma_s dW_s \tag{1}$$

μ_t is a stochastic drift term, σ_t is a stochastic volatility term and W_t is a standard Brownian motion. From Itô's lemma it is known that the log-price also follow an Itô process (Mykland and Zhang, 2012). The process in eq. (1) is often written in differential form where it becomes

$$dX_t = \mu_t dt + \sigma_t dW_t \tag{2}$$

Merton (1976) was the first to introduce jumps into financial time-series in order to explain the observed time discontinuities when sampling at discrete times. Following Lee and Mykland (2007); Aıt-Sahalia (2004); Christensen et al. (2014) adding jumps will change the model to

$$X_t = X_0 + \int_0^t \mu_s ds + \int_0^t \sigma_s dW_s + \sum_{i=1}^{N_t^J} J_i \tag{3}$$

and

$$dX_t = \mu_t dt + \sigma_t dW_t + J_t dN_t^J \tag{4}$$

N_J is a counting process that is independent of W_t. J_t is the jump size with mean $\mu_{J,t}$ and variance $\sigma_{J,t}$. Lee and Mykland (2007) assumes that the jump sizes are independent of each other and identically distributed.

2.2 Detecting jumps

Several tests have been developed to detect jumps in financial time-series (Barndorff-Nielsen and Shephard, 2006; Lee and Mykland, 2007; Andersen et al., 2002; Chernov et al., 2003; Eraker et al., 2003; Aït-Sahalia, 2002). Few of these tests use high-frequency data. Barndorff-Nielsen and Shephard (2006) propose a test to determine if a jump occurred inside a given period. They utilize the properties of the bipower variation (BPV) proposed by (Barndorff-Nielsen and Shephard, 2004). BPV can in theory split up the individual components of the quadratic variation into the part that stems from the continuous part of the price and the part that stems from the jump, thereby making it robust to jumps. The null hypothesis is that the price follows a Brownian semi-martingale with no jumps. To ensure that the test also has power under the alternative an estimator consistent in the presence of jumps is used. The realized quadpower variation $\{Y_\delta\}_t^{[1,1,1,1]} = \delta^{-1} \sum_{j=4}^{\left[\frac{t}{\delta}\right]} |y_{j-3}| \, |y_{j-2}| \, |y_{j-1}| \, |y_j| \xrightarrow{p} \mu_1^4 \int_0^t \sigma_s^4 ds$ fulfills this requirement. This ultimately leads to a normally distributed feasible linear jump-ratio test statistic, using realized values

$$\hat{J} = \frac{\delta^{-\frac{1}{2}}}{\sqrt{\vartheta \max\left(1, \frac{\tilde{q}_i}{\{\tilde{v}_i\}^2}\right)}} \left(\frac{\mu_1^{-2} \tilde{v}_i}{\hat{v}_i} - 1 \right)$$

Where $\hat{v}_i$ is the increments of the realized quadratic variation, $\tilde{v}_i$ is the realized BPV, and $\tilde{q}_i$ realized quadpower variation. The test is asymptotically normally distributed and the null hypothesis is rejected if the test value is significantly negative. The test proposed by Barndorff-Nielsen and Shephard (2006) gives an indicator whether the given

series can be regarded as a continuous path or if there was significant evidence of some discontinuity in the price at some point in the series. Jiang and Oomen (2008) propose using higher order moments to get faster convergence of the test. Lee and Mykland (2007) later proposed a non-parametric jump-test to determine if each given sampling interval had a significant discontinuity. A test statistic $\mathcal{L}$ is developed to test at each sampling time t_i[2]. The test statistic does not make any assumption about whether or not there were a jump before or after t_i. The defined test statistic at time t_i is given as

$$\mathcal{L}_i \equiv \frac{\log\left(\frac{S(t_i)}{S(t_{i-1})}\right)}{\widehat{\sigma\left(t_i\right)}} \tag{5}$$

Where $\widehat{\sigma\left(t_i\right)}$ in eq. (5) is the square root of the realized bipower variation,

$$\widehat{\sigma\left(t_i\right)}^2 = \frac{1}{K-2}\sum_{j=i-K+2}^{i-1}\left|\log\left(\frac{S(t_j)}{S(t_{j-1})}\right)\right|\left|\log\left(\frac{S(t_{j-1})}{S(t_{j-2})}\right)\right| \tag{6}$$

The test uses bipower variation as it has been showed to be a consistent estimator for the integrated volatility in the presence of jumps (Aıt-Sahalia, 2004; Barndorff-Nielsen and Shephard, 2004). When using the realized bipower variation the detection test becomes independent of the presence of jumps over time. The window size K must be specified in a way such that the effect of jumps on the volatility estimation will disappear. The minimum K that satisfies this condition is given as $\sqrt{252 \cdot nobs}$ with $nobs$ being the number of observations per day. Lee and Mykland (2007) find that the test statistic $\hat{\mathcal{L}}_i$ under the absence of jumps asymptotically follows a normal distribution with mean 0 and variance $\frac{1}{c^2}$ with $c = E\left(|U_i|\right) = \sqrt{\frac{2}{\pi}}$. It follows this distribution as the Brownian increments are standard normally distributed. Each $\mathcal{L}_i$ then becomes independent of each other. In order to find acceptable rejection regions for the test, Lee and Mykland (2007) use the asymptotic distribution of maximums under the absence of jumps. This can give a guide to distinguish when a jump is present, which will cause the test-statistic value to be higher in magnitude. When the observed test statistic fall outside the region of maximums in the absence of jumps, then it is unlikely that the realized return comes from the continuous part of the jump-diffusion model. The magnitude of the test statistics without jumps follows

$$\frac{\max\left(|\mathcal{L}_i|\right) - C_n}{S_n} \to \xi, \tag{7}$$

where ξ has a cumulative distribution function $P\left(\xi < x\right) = e^{-e^{-x}}$. Lee and Mykland (2007) defines C_n and S_n as

$$C_n = \frac{\left(2\log\left(n\right)\right)^{\frac{1}{2}}}{c} - \frac{\log\left(\pi\right) + \log\left(\log\left(n\right)\right)}{2c\left(2\log\left(n\right)\right)^{\frac{1}{2}}}$$

$$S_n = \frac{1}{c\left(2\log\left(n\right)\right)^{\frac{1}{2}}}$$

The critical value of the test following this analogy is equal to $\beta^* = -\log\left(-\log\left(1-\alpha\right)\right)$ where α is the significance level. The null hypothesis is then rejected if

$$\frac{|\mathcal{L}_i| - C_n}{S_n} > \beta^*. \tag{8}$$

Lee and Mykland (2007) also proves that the probability of misclassification becomes negligible at higher frequencies of observations. The main difference between the test presented in Lee and Mykland (2007) and Barndorff-Nielsen and Shephard (2006) is that the test from Lee and Mykland (2007) also have the ability to tell something about when the jump occurred, the sign of the jump and the magnitude of the jump. Lee and Mykland (2007) also find that their test outperforms both Barndorff-Nielsen and Shephard (2006) and Jiang and Oomen (2005).

In this paper a microstructure noise version of the Lee and Mykland (2007) is applied. Jiang and Oomen (2008) propose a microstructure noise correction to the variance in order to make their swap variance test robust to microstructure noise. Jiang and Oomen (2008) claim that this bias correction method can be applied to the Lee and Mykland (2007) test, and such will be done in this study. To correct for the bias Jiang and Oomen (2008) suggest using an autocovariance-based noise variance estimator,

$$\hat{\omega}^2 = -\frac{1}{N-1}\sum_{i=1}^{N-1} S_i S_{i+1} \tag{9}$$

Jiang and Oomen (2008) assume that the return variance is constant $\left(\bar{V}\right)$ and can be estimated with the BPV, which leads to

$$E\left[BPV_N^*\right] = \left(1 + c_b\left(\gamma\right)\right) E\left[BPV_N\right], \tag{10}$$

where BPV_N^* and BPV_N denote the bipower variation computed from noise contaminated and clean return data respectively. The bias correction parameter is given as

$$\begin{aligned} c_b\left(\gamma\right) = {} & \left(1+\gamma\right)\sqrt{\frac{1+\gamma}{1+3\gamma}} + \gamma\frac{\pi}{2} - 1 \\ & + 2\frac{\gamma}{\left(1+\lambda\right)\sqrt{2\lambda+1}} + 2\gamma\pi\kappa\left(\lambda\right), \end{aligned} \tag{11}$$

with $\gamma = \frac{N\omega^2}{\bar{V}}$, $\lambda = \frac{\gamma}{1+\gamma}$, and
$\kappa\left(\lambda\right) = \int_{-\infty}^{\infty} x^2 \Phi\left(x\sqrt{\lambda}\right)\left(\Phi\left(x\sqrt{\lambda}\right) - 1\right)\phi\left(x\right)dx.$

Using the bias correction term from eq. (11) the bias corrected bipower variation can be found using the relation from eq. (10). Alternatively one can also use the non-overlapping pre-averaging variance estimator suggested in Christensen et al. (2014) to make the Lee and Mykland

[2] Lee and Mykland (2007) also develop a global test, but the main focus lies upon the individual tests.

(2007) test robust to microstructure noise. Here the variance is calculated as

$$\hat{\sigma}^{*2}_{i,M} = \frac{\pi}{2}\frac{1}{K-2}\sum_{j=i-K+2}^{i-1}\left|r^*_{j,M}\right|\left|r^*_{j-M,M}\right|, \quad (12)$$

where the window size K is set to the same value as recommended above. The bias correction from Jiang and Oomen (2008) relies on the assumption that the microstructure noise is i.i.d (Jiang and Oomen, 2008). If one does not want to impose this i.i.d assumption then other new microstructure noise robust tests have been proposed.

Lee and Mykland (2012) have also been developed to adjust Lee and Mykland (2007) for microstructure noise. This test uses a pre-average estimator for the price in each time interval. The price is averaged over non-overlapping intervals of block-size M to be

$$\hat{P}\left(t_j\right) = \frac{1}{M}\sum_{i=\lfloor j/k\rfloor}^{\lfloor j/k\rfloor+M-1}\tilde{P}\left(t_i k\right) \quad (13)$$

Where $\tilde{P}\left(t_i k\right)$ is the log-price over the block M. The test statistic becomes $\mathcal{L}\left(t_j\right) \equiv \hat{P}\left(t_{j+kM}\right) - \hat{P}\left(t_j\right)$. The test for the null hypothesis of no jumps is standardized and then becomes

$$\mathcal{X}\left(t_j\right) = \frac{\sqrt{M}}{\sqrt{V_n}}\mathcal{L}\left(t_j\right) \quad (14)$$

Lee and Mykland (2012) find that it is possible to use the maximum of the absolute differences in averaged log-prices and use the Gumbel distribution to select the rejection region for the maximum. To consistently estimate V_n in the standardized test statistic $\mathcal{X}\left(t_j\right)$, then $\hat{V}_n = \frac{2}{3}\hat{\sigma}^2 C^2 T + 2\hat{q}^2$ can be used. $\hat{\sigma}$ and $\hat{q}$ are consistent estimators for volatility and noise variance respectively. The estimator from Podolskij et al. (2009) is used and $\hat{q}$ is estimated by $\hat{q} = \frac{\sqrt{\frac{1}{n'}\sum_{m=1}^{n'}\left(\tilde{P}(t_m)-\tilde{P}(t_{m+k})\right)^2}}{\sqrt{2}}$. This averaging of the prices removes noise from the data. One must be careful how to choose the block-size M of the averaging estimator. You must also in this test specify the noise dependence (q). If set wrong then this can introduce bias into the test. Optimizing these extra parameters is avoided by using the bias correction approach. Both Lee and Mykland (2007) and Barndorff-Nielsen and Shephard (2006) are very recognized, while Lee and Mykland (2012) is not yet widely used. Lee and Mykland (2012) finds that there is little difference between their test results and Lee and Mykland (2007) when the noise dependence is very small. In this paper the bias correction for microstructure noise is also found to be small. Xue et al. (2014) utilizes properties of wavelets to overcome having to choose the window size K in the Lee and Mykland (2007) test, which they show is a special case of a wavelet-type test.

Both Lee and Mykland (2007) and Barndorff-Nielsen and Shephard (2006) is applied in this paper, but the jumps detected by Lee and Mykland (2007) with the microstructure noise correction is used as the outcome variable in the neural network predictor. The jump detection test of Lee and Mykland (2007) is also used in Mäkinen (2017); Mäkinen et al. (2019), but without the bias correction applied in this study. It has the useful additional features of also being able to report jump sign and jump magnitude.

When studying the jumps that were found in the data (an example is seen in fig. 2) it seems evident that a jump did occur, or at least a significant change in the price.

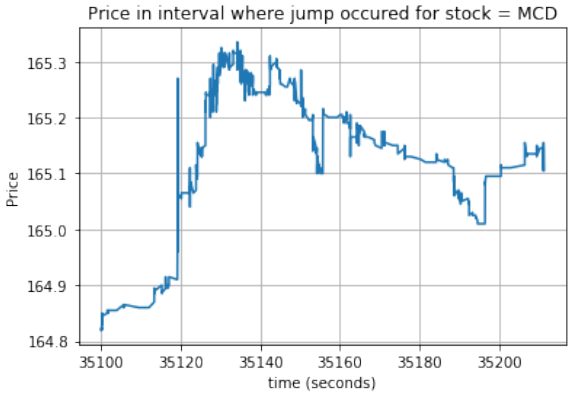

Figure 2: Plot of interval where a jump was classified in. The time is seconds from midnight.

The ability to detect jumps or drifts can provide valuable information for both risk management and option pricing. To go a step further, neural networks are later used to try and predict whether a jump will occur in the next interval, following the procedure of Mäkinen et al. (2019); Mäkinen (2017). This method has also been successful in mid-price prediction Kercheval and Zhang (2015); Ntakaris (2019); Tsantekidis et al. (2017b); Passalis et al. (2017); Tran et al. (2018); Sirignano and Cont (2019); Tsantekidis et al. (2017a) which poses a similar problem as the jump prediction problem. In the next section neural networks will be explained, to understand why they fit the proposed problem.

3 Neural Network

Neural networks are learning systems that are based on the neural structure of the human brain. The neurons in the network process the information and feed it forward through the network. Recurrent neural networks (RNN) are a special kind of neural networks that are made to handle time-series data. Especially the popular long short term memory (LSTM) variation is popular when handling time-series data.

3.1 Feed-forward neural network architecture

Neural networks are comprised of hidden layers that consist of artificial neurons that are referred to here as units or nodes. Between each of the nodes is a weight $w_{jj'}$ which determines the dependency structure. The inputs are transformed to a single value $a_j = \sum_{j'} w_{jj'} x_{j'}$ and thereafter passed through an activation function $v_j = l_j(a_j)$. The activation function allows the dependencies to be non-linear. Common choices for the activation function is the sigmoid function $\sigma(z) = \frac{1}{(1+e^{-z})}$, the $tanh$ function $\phi(z) = \frac{(e^z - e^{-z})}{(e^z + e^{-z})}$ and the rectified linear unit function (ReLU) $rl_j(z) = \max(0, z)$. It is possible to specify different activation functions for each of the layers in your neural network[3]. For multi-class classification you would normally apply a softmax function as the activation function for the output nodes $\hat{y}_k = \frac{e^{a_k}}{\sum_{k'=1}^{K} e^{a_{k'}}}$, which is a pointwise sigmoid function. In the task of the binary prediction whether there is a jump or not either sigmoid or softmax can be used interchangeably. To optimize the weights, the loss is minimized iteratively. The most applied training algorithm is the back-propagation algorithm that was proposed for neural networks by Rumelhart et al. (1985). This method uses chain rule derivatives to find the optimal direction to minimize loss for each parameter in the model starting backward from the output layer. The weights are then adjusted by gradient descent. There is no guarantee that the optimization algorithm will reach a global-minimum as the loss is not convex (Lipton et al., 2015a). Using batches ensures the network runs faster rather than using the full training data-set in each weight-update. Additionally, the number of epochs is also specified, which will determine how many times the full data is seen during training.

3.2 Recurrent neural network (RNN)

RNNs are neural networks that use sequence cycles in the network to capture the sequential structure of time-series. RNNs retain information from an arbitrary long context window. The feed-forward neural networks rely on the assumption that each observation is independent, which does not hold for financial time-series. RNNs can capture time dependencies that are both linear and non-linear (Lipton et al., 2015a). RNNs take a sequence of real valued vectors as inputs $\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(T)}\right)$ and outputs $\left(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, ..., \mathbf{y}^{(T)}\right)$. The previous input $\mathbf{x}^{(t-1)}$ is allowed to influence the current output $\hat{\mathbf{y}}^{(t)}$. The window of dependency can be set beforehand or it can be learned in training. If only the previous time-step is set to have an effect on the current time period, then the hidden layer node value becomes

$$\mathbf{h}^{(t)} = \sigma\left(W^{hx}\mathbf{x}^{(t)} + W^{hh}\mathbf{h}^{(t-1)} + \mathbf{b}_h\right) \tag{15}$$

As can be seen in eq. (15) the hidden layer node at time t depend on time $t-1$. This will capture the time dependency between each of the observations. The sequential dependency can go both ways, but only the backward dependency is present for financial time-series data[4]. The intuitive way of understanding the recurrent neural network is to see it as a normal neural network with one hidden layer per time-step that studies both the inputs of that time step and the previous k layers of the network (Lipton et al., 2015a). Werbos et al. (1990) introduced a version of the back-propagation algorithm that can train the neural network while still accounting for the time dependency that is present in the data.

Recurrent neural networks can experience problems during training as the gradient can explode or vanish when trying to capture the long-term dependencies. These problems can occur under different conditions, which are further explained in Pascanu et al. (2013). One way to overcome the problem is to use the truncated back-propagation through time method, that overcomes the problem at the sacrifice of capturing all the long-term dependencies. Another solution is to use the LSTM networks that assign fixed unit weights to the weights between time-steps. This will ensure that the gradient will not vanish or explode (Lipton et al., 2015a).

3.3 Long short term memory (LSTM)

Hochreiter and Schmidhuber (1997) introduced the LSTM hidden-layer where each node is replaced with a memory cell. Recurrent neural networks have both long and short-term memory. The long-term memory is achieved by allowing previously processed data (e.g. $\mathbf{h}^{(t-1)}$) as input in the model. The network also has a short-term memory where the current information passes through the network at each time step. A LSTM network creates a memory cell that has an intermediate level of memory between long and short-term memory. The memory cell consists of an input node, input gate, internal state, forget gate and an output gate. Input nodes $\mathbf{g}^{(t)}$ take the input $\mathbf{x}^{(t)}$ and the output of the hidden layer from the previous time-step $\mathbf{h}^{(t-1)}$ and transforms it via an activation function.

$$\mathbf{g}^{(\mathbf{t})} = \phi\left(W^{gx}\mathbf{x}^{(t)} + W^{gh}\mathbf{h}^{(t-1)} + \mathbf{b}_g\right) \tag{16}$$

Input gates $\mathbf{i}^{(t)}$ takes the same inputs as the input node, to find which of these inputs provides value. It uses a sigmoid activation function, which gives an output between zero and one. Effectively, this gives a weight to each of the

[3] The full network structure of a standard feed-forward neural network can be found in appendix A.1

[4] Graphical representation of the recurrent neural network can be found in appendix A.2

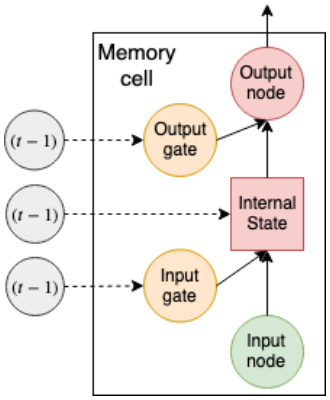

Figure 3: Structure of the LSTM memory cell. These memory cells replace the hidden layer nodes of the recurrent neural network that can be seen in appendix A.2

inputs. It is called a gate because if the value of the gate for an input is zero, then the connection to that input is cut off.

$$\mathbf{i}^{(t)} = \sigma\left(W^{ix}\mathbf{x}^{(t)} + W^{ih}\mathbf{h}^{(t-1)} + \mathbf{b}_i\right) \tag{17}$$

The internal state $\mathbf{s}^{(t)}$ is connected to the previous periods of internal states with a unit weight. This will ensure that the gradient does not explode or vanish. It will additionally take the input from the input node and the input gate. It can also use a forget gate $\mathbf{f}^{(t)}$ to determine how much of the previous information to use. Forget gates can help to forget previous information in the internal state when it is no longer providing any value.

$$\mathbf{s}^{(t)} = \mathbf{g}^{(t)} \odot \mathbf{i}^{(t)} + \mathbf{s}^{(t-1)} \odot \mathbf{f}^{(t)} \tag{18}$$

Output gates $\mathbf{o}^{(t)}$ work much like the input gates. They are connected to the hidden layer of previous time steps $\mathbf{h}^{(t-1)}$ and the input. The final output of the memory cell $\mathbf{h}^{(t)}$ will therefore also be a combination between the internal state and the output gate (Lipton et al., 2015a).

$$\mathbf{h}^{(t)} = \phi\left(\mathbf{s}^{(t)}\right) \odot \mathbf{o}^{(t)} \tag{19}$$

Different extra hidden layer variations are proposed to enhance the model performance. These could potentially benefit the neural network. In this paper, the effect of adding convolutional and attention layers is studied. Additionally, methods to combat overfitting are also applied to the different models (see section 3.5). Overfitting is when too much information from the training data is learned, which makes the model unable to generalize to new unseen data.

3.4 Hidden layer variations

3.4.1 Convolutional layer

Convolutions are used to capture patterns in the data. In this context, it can be used to find patterns in the features, and reduce the dimensional space. Convolutions look at a specified grid of the data and aggregate information in it. There are different aggregation methods and amongst the most popular is averaging or maximizing over the grid. It looks through each possible grid of the data. In the application done in this paper a maximum aggregation will be done on each convolutional grid, which in effect will draw select highest feature in the grid[5].

3.4.2 Attention layer

As mentioned, convolutions are used to find patterns in the data. Another possibility is to use an attention layer that will provide a weight to each input, and thereby focus on the more important inputs by assigning them a higher weight (Vaswani et al., 2017; Mäkinen et al., 2019). By adding a layer to the neural network then you can train the network to find the most important features and give them extra attention. The weights here are then multiplied onto the inputs. There are several ways you can compute the attention layer, which are summarized in Weng (2018). In this paper, weights will be calculated for each input using much the same technique as the input gate described in section 3.3. Each input is given a weight using a sigmoid activation function, which gives a weight between zero and one. Attention weights will be updated during back-propagation training.

3.5 Overfitting

It is also important to keep overfitting in mind when designing the neural network model. Overfitting arises when the model is fitted excessively to the data used to estimate the model. This makes the model unable to generalize to new unseen data where it essentially have to be performing well. Therefore, techniques to overcome overfitting are added to the models.

3.5.1 Dropout

When training the neural network model, one possibility is to randomly omit nodes in each layer. This forces the network to learn from different information in the data. This can prevent the network from overfitting to the same information in each update (Hinton et al., 2012; Mäkinen et al., 2019).

3.5.2 Batch normalization

Batch normalization is a technique that standardizes the batch inputs that is put into the neural network. This

[5] A graphical explanation of the convolutional layer can be found in appendix A.3

can accelerate the training process for the neural network. It will transform the inputs to have zero mean and unit variance. When the inputs are stabilized during training then the model is less likely to be stuck in a local optima solution or have the gradient vanish/explode (Ioffe and Szegedy, 2015). Thus, eliminating the interval covariate shifts by stabilizing the inputs will give a faster convergence.

4 Data

It is very important to do proper data handling when using high-frequency data. As is reported in Falkenberry (2002), errors occur also in automated trading systems such as the data from the TAQ database. When interpreting model results it is important to have precise results. Likewise, when you are using a model for forecasting, it also becomes important to input accurate data that is not filled with errors (Barndorff-Nielsen et al., 2009). Inclusion of the poor quality observations can cause more harm than good for the chosen estimators and therefore they are removed. This paper uses the cleaning rules proposed in Barndorff-Nielsen et al. (2009).

4.1 Data cleaning

A set of general rules are applied to both trade and quote data:

- P1: Entries that fall outside the normal opening hours between 9.30am and 4pm are deleted.
- P2: Delete entries with either a bid, ask or transaction price of zero. This is an indicator of an error.
- P3: Here a modified version of P3 from Barndorff-Nielsen et al. (2009) is applied. Keep only data from the two most frequent exchanges for the given stock on the given day.

After applying the general rules, then special rules can be applied to the trade data.

- T1: Delete corrected trades. This is trades where the variable $CORR \neq 0$
- T2: Delete entries with an abnormal sales condition. This means keeping conditions E, F and blank.
- T3: If multiple observations have the same timestamp (the variable *"utcsec"*) then the median price of these is used. As the timestamp is reported in nanoseconds, this almost always results in no observations having the same timestamp.
- T4: Two different versions of the T4 rule can be applied.
  - T4.1: Delete entries where the price is above the current ask price plus the bid-ask spread. Also delete entries where the price is below the bid price minus the bid-ask spread.
  - T4.2: Delete entries where the current price is more than five times the mean absolute deviations (MAD) from a rolling centred median (excluding the observation under consideration) of fifty observations (twenty-five observations before and twenty-five after)[6].

This paper applies T4.2. Trade data is later matched to quotes but is kept separate during the data-cleaning process. There are also special rules for quote data that can be applied after the general rules.

- Q1: If any of the observations have the same timestamp then these are merged and the bid and ask price is replaced with the median. As mentioned with the trade data this does not occur often when times are reported in nanoseconds.
- Q2: Delete entries where the bid-ask spread is negative.
- Q3: Delete entries where the bid-ask spread is more than fifty times the median bid-ask spread on that day.
- Q4: Delete entries where the mid-quote deviated by more than five MADs from a rolling centred median (excluding the observation under consideration) using fifty observations (twenty five before and twenty five after).

Applying these rules around fifty percent of the observations are removed on each day. Most observations are removed by the P3 rule where only observations from the two most frequent exchanges are kept. The remaining rules does not delete many entries, but are crucial for removing faulty data that can distort the results. Q4 and T4.2 are the most computational rules, as they require a rolling window calculation for each observation.

Quote and trade data is merged by matching the trade data to the quote data-point with a timestamp just before the trade observation's. Further merging algorithms could have been applied, such as the matching scheme mentioned in Christensen et al. (2014). This will ensure that there is no error in matching if either quotes or trades are reported with a delay. This was unfortunately out of the scope of this paper. As mentioned in Brownlees and Gallo

[6] In the paper they use ten mean absolute deviations, but five is found to be sufficient here.

(2006); Liu et al. (2015) one can also use mid-quote data to estimate returns to reduce some of the effects of bid-ask bounces. It would also mitigate the need to merge the quote and trade data.

4.2 Feature creation and data aggregation

After cleaning both the trade and quote data and matching them together, then the data is aggregated to three different sampling levels. It is aggregated to thirty-second sampling, minute sampling and, five-minute sampling. Moreover, a set of features used as explanatory variables is created from the limit order data. Earlier studies found the features of Kercheval and Zhang (2015) to provide useful information when making predictions on jumps and price movements (Mäkinen, 2017; Mäkinen et al., 2019; Ntakaris, 2019; Tsantekidis et al., 2017a,b; Kercheval and Zhang, 2015). The set of features is reported in table 1.

Vector	Variables in vector	N_{var}
Basic set of features		
v_1	$\{P_i^{ask}, V_i^{ask}, P_i^{bid}, V_i^{bid}\}_{i=1}^{10}$	40
Time-insensitive feature vectors		
v_2	$\{\left(P_i^{ask} - P_i^{bid}\right), \frac{\left(P_i^{ask}+P_i^{bid}\right)}{2}\}_{i=1}^{10}$	20
v_3	$\{ P_{10}^{ask} - P_1^{ask}, P_1^{bid} - P_{10}^{bid}, \lvert P_{i+1}^{ask} - P_i^{ask}\rvert, \lvert P_{i+1}^{bid} - P_i^{bid}\rvert\}_{i=1}^{9}$	20
v_4	$\{\frac{1}{10}\sum_{i=1}^{10} P_i^{ask}, \frac{1}{10}\sum_{i=1}^{10} P_i^{bid}, \frac{1}{10}\sum_{i=1}^{10} V_i^{ask}, \frac{1}{10}\sum_{i=1}^{10} V_i^{bid}\}$	4
v_5	$\{\sum_{i=1}^{10}\left(P_i^{ask} - P_i^{bid}\right), \sum_{i=1}^{10}\left(V_i^{ask} - V_i^{bid}\right)\}$	2
Time-sensitive feature vectors		
v_6	$\{\frac{dP_i^{ask}}{dt}, \frac{dP_i^{bid}}{dt}, \frac{dV_i^{ask}}{dt}, \frac{dV_i^{bid}}{dt}\}$	40
v_7	$\{\lambda_{\Delta t}\}$	1
v_8	$\{1_{\{\lambda_{\Delta t} > \lambda_{\Delta T}\}}\}$	1
v_9	$\{\frac{d\lambda}{dt}\}$	1
Total number of features		129

Table 1: Accommodated version of the feature set proposed by Kercheval and Zhang (2015).

In v_8 the arrival of orders is compared to the average arrivals over the last two sampling periods to see if there is a recent increase in the intensity of order arrivals. An indicator value of one means that the intensity has increased. As described by Cont (2011) one way to construct features is to study only the top levels at each given sampling window. The order book can be divided into levels at any given point in time. The best level is the lowest asking price and the highest bid price, as they are the most competitive prices. Cont (2011) argued that information reflected in deeper levels already is reflected on the higher levels. The procedure Kercheval and Zhang (2015); Mäkinen (2017); Mäkinen et al. (2019); Ntakaris (2019) is followed where the top ten highest level in each sampling window is used when data is aggregated. Thus, the variables indexed from $i \in \{1, ..., 10\}$ are the ten most competitive bid and ask offers in the sampling period under consideration. Mäkinen et al. (2019) also proposed using the hour-stamp of the trade as an additional feature v_{10} instead of the precise nanosecond timestamp. This is done to avoid the model placing too much weight on the exact timestamp when making a prediction.

In this paper a new set of features is proposed and used in addition to the features from table 1. These additional features are presented in table 2.

Vector	Variables in vector	N_{var}
$v_{11,t}$	$\{\sum_{j=1_t}^{k_t} \frac{V_j^{ask}\cdot P_j^{ask}}{V_j^{ask}}, \sum_{j=1_t}^{k_t} \frac{V_j^{bid}\cdot P_j^{bid}}{V_j^{bid}}\}$	2
$v_{12,t}$	$\{\sum_{j=1_t}^{k_t} \frac{V_j^{trade}\cdot P_j^{trade}}{V_j^{trade}}, \sum_{j=1_t}^{k_t} \frac{V_j^{trade}\cdot \log\left(P_j^{trade}\right)}{V_j^{trade}}\}$	2
$v_{13,t}$	$\{\sum_{j=1_t}^{k_t} \frac{\left(V_j^{ask}+V_j^{bid}\right)\cdot \frac{\left(P_j^{ask}+P_j^{bid}\right)}{2}}{\left(V_j^{ask}+V_j^{bid}\right)}, \sum_{j=1_t}^{k_t} \frac{\left(V_j^{ask}+V_j^{bid}\right)\cdot \log\left(\frac{\left(P_j^{ask}+P_j^{bid}\right)}{2}\right)}{\left(V_j^{ask}+V_j^{bid}\right)}\}$	2
$v_{14,t}$	$\{\sum_{j=1_t}^{k_t} \frac{\left(V_j^{ask}+V_j^{bid}\right)\cdot\left(P_j^{ask}-P_j^{bid}\right)}{\left(V_j^{ask}+V_j^{bid}\right)}, \sum_{j=1_t}^{k_t} \frac{\left(V_j^{ask}+V_j^{bid}\right)\cdot median_j^{Q4}}{\left(V_j^{ask}+V_j^{bid}\right)}, \sum_{j=1_t}^{k_t} \frac{\left(V_j^{ask}+V_j^{bid}\right)\cdot MAD_j^{Q4}}{\left(V_j^{ask}+V_j^{bid}\right)}\}$	3
$v_{15,t}$	$\{\sum_{j=1_t}^{k_t} V_j^{trade}, \sum_{j=1_t}^{k_t} V_j^{ask}, \sum_{j=1_t}^{k_t} V_j^{bid}\}$	3
Total		12

Table 2: Own set of features that is added to the feature-set from table 1. Here $j \in \{1_t, ..., k_t\}$ is each of the observations inside sampling interval t, not to be confused with the most competitive levels in each interval that is indexed with i. The sampling intervals are indexed as $t \in \{1, ..., T\}$ where $t = 1$ is the first sampling interval and $t = T$ is the last.

The additional features $v_{11,t}$, $v_{12,t}$, $v_{13,t}$ and $v_{14,t}$ consist of volume weighted averages of the ask, bid, trade, log-trade, midquote, log-midquote, spread, median and MAD respectively. They are based on the observations in the given sampling interval under consideration. $v_{15,t}$ contain the sum of trade, ask and bid volumes respectively. Median and MAD were the rolling window median and MAD calculated during the cleaning of the quote data with the Q4 cleaning rule. This gives a total of $129 + 1 + 12 = 142$ features for each observation.

5 Analysis

In the analysis data from the first ten months of 2018 are used for the thirty individual stocks on the dow jones industrial average (DJIA)[7] and the S&P500 exchange-traded fund (SPY). Seven of the DJIA stocks were too illiquid and dropped from the study[8]. The first eight months were used as training data and the last two months as test data. That gives a study of twenty-four models per sampling interval. Here three different sampling intervals are studied and four different models, giving a total of 288 different models trained and evaluated in this study. The study of multiple securities enables the ability to tell something about the variability of the performance in addition to the average performance of each model.

5.1 Models

Four models are analyzed in this paper. The basis for all the models will be the LSTM framework described in section 3.3 that is developed for analyzing time-series data. Furthermore, all four model use the base structure seen in fig. 4.

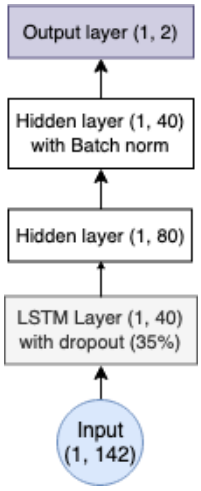

Figure 4: Structure of the LSTM model.

As seen in fig. 4 the model starts out with the LSTM layer to capture the time dependencies in the data. Further, two hidden layers are added to the model which will capture the dependencies amongst the features for each observation. These normal hidden layers treat each observation as independent. This structure works well in practice because you have the extra layers which allows the LSTM layer to focus more on the time dependencies while the current time-step feature dependencies can be captured in the latter normal layers.

[7] Refers to the current individual stocks on DJIA at the end of 2019, disregarding when they were added to the index.

[8] A list of the remaining included stocks can be found in appendix D

Furthermore, a model which adds convolutions prior to the LSTM layer is employed to capture patterns in the feature dimension. As proposed by Mäkinen et al. (2019) adding attention before the LSTM layer can also be beneficial to try and focus on the more important features by giving them a higher weight and lowering the focus on the less important features. Adding attention prior to the LSTM layer will be the third model in this study. Finally the fourth model is a model which combines attention, convolutions and LSTM. It starts by adding the attention layer and afterwards convolutional and LSTM layers.

All inputs are normalized using z-score scaling

$$x_i' = \frac{x_i - \bar{x}}{\sigma_x}. \tag{20}$$

Additionally, all the models use a synthetic minority oversampling technique (SMOTE) to help overcome the problem of data-imbalance Chawla et al. (2002). There are only a few observations with jumps in the total data set. Jumps are found to generally account for less than 1% of the data, as found in Christensen et al. (2014). Thus, the SMOTE algorithm is applied to the training data, such that the model has seen more examples of jump observations when it predicts for new unseen data. New synthetic examples are created instead of oversampling the minority class with replacement. The synthetic observations are created by using the k-nearest neighbors in the same class (i.e. jump observations) to compute the difference between them. Differences are used as a scale to create new observations Chawla et al. (2002).

5.2 Performance metrics

The main performance metric used in this study is the F1-score. The F1 score gives a harmonic-mean between the precision and recall score.

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{21}$$

Recall informs about how good the model is at correctly classifying jumps compared to how often it misses a jump.

$$Recall = \frac{tp}{tp + fn} \tag{22}$$

tp is true positives, which in this case is actual jumps that were correctly classified. False negatives fn are the actual jumps that were not classified as such. Precision inform about how precise the model is at making correct classifications in regard to how often it wrongly classifies negative samples as positive.

$$Precision = \frac{tp}{tp + fp} \tag{23}$$

Here fp is the false positives, which in this situation is the observations that were wrongly classified as having a significant jump. As seen from the above equations (eq. (21), eq. (22), eq. (23)) these measures only accounts for the models ability to predict positive samples. Moreover, F1 is a useful measure when the amount of positive samples is very small compared to the number of negative samples, making it a useful measure for the jump-prediction problem (Lipton et al., 2014). The performance of the models are also evaluated by Cohen's kappa which accounts for being correct by pure chance. Cohen's kappa is calculated as

$$\kappa = \frac{p_o - p_c}{1 - p_c} \tag{24}$$

p_o is the proportion of observations that are correctly classified and p_c is the proportion expected to be correct by just by chance (Cohen, 1960). Fleiss et al. (2013) suggest that κ values above 0.4 are acceptable and values above 0.75 are excellent.

5.3 Performance results

The average F1-score and standard-error of the models are reported in table 3[9].

Model	Half-minute sampling	One-minute sampling	Five-minute sampling
LSTM	**0.68** (0.18)	**0.72** (0.2)	**0.6** (0.26)
LSTM-CNN	0.54 (0.14)	0.52 (0.14)	0.29 (0.27)
LSTM-Attention	0.65 (0.15)	0.59 (0.15)	0.52 (0.25)
LSTM-CNN-Attention	0.49 (0.14)	0.51 (0.17)	0.29 (0.28)

Table 3: F1 score of the four models. Standard errors are reported in parenthesis. The best performing model for each sampling interval is reported in bold.

It can be seen in table 3 that the LSTM model performs best for all three sampling intervals. The most complex model that uses both convolutions and attention has the lowest average F1-score in all three sampling-schemes but has similar performance to the LSTM-CNN model. It seems that the convolutions are not managing to capture patterns in the data to make better predictions. It might even be that the convolutions distort the signals in the data. The attention mechanism is more successful in weighting the most important features. It can be seen in appendix B that the attention mechanism induces the model to have a very high recall score, at the loss of lower precision. The attention mechanism causes the important features in regards to jumps to be highlighted and this causes the model to be able to detect jumps very well. But it also causes the model to think there are jumps more often than there truly is and therefore it also lowers the precision score.

From the standard errors of the models reported in table 3 it can also be noted that the models have varying performance, which can indicate that different mechanisms and dynamics are at play for different assets. Some of them seem to have systematic jumps while other are more random and harder to predict. It suggests that the path-dependence is a stock specific feature that relates to the individual asset (Mäkinen et al., 2019).

The results found in this study are very similar to those of Mäkinen et al. (2019) and it indicates promising results in this emerging area of jump predictions. Several papers have found good results when using limit order data to make predictions both on jumps and price-movements (Ntakaris, 2019; Tsantekidis et al., 2017b,a; Kercheval and Zhang, 2015; Kara et al., 2011; Mäkinen et al., 2019; Mäkinen, 2017). Furthermore, the model confidence set is applied to determine if there are significant differences in the performance of the models.

5.4 Performance test

The model confidence set (MCS) of Hansen et al. (2011) is applied to find the set of best models $\widehat{\mathcal{M}}^*$ in each sampling interval, within a given level of confidence. $\widehat{\mathcal{M}}^*$ is going to be data-dependent as the information from the data is used to compute the set $\widehat{\mathcal{M}}^*$. The advantage of using the approach of Hansen et al. (2011) is that it incorporates the limitations of the data. If the data does not contain much information then the MCS will be large and not exclude many models. On the other hand if data is very informative about the models performances then the MCS can accurately find a small set of models that are best. The MCS is computed from the full set of models $\mathcal{M}_0$, which in this study is the four models summarized in section 5.1. An equivalence test is first applied to the set $\mathcal{M}$, which tests if the models are equivalently well performing. If this test is rejected, then an elimination rule is applied to the set. This procedure is repeated until the equivalence test is not rejected. MCS also provides p-values, where small p-values indicate that the given model is unlikely to be one of the best alternatives in $\mathcal{M}_0$. MCS use the loss-sequence of the forecasted observations.

The sequence of losses will here be computed as the average F1 at each point in time over the twenty-four assets studied. Instead of the goal being to minimize loss, it will then be to maximize the average F1-score. A significance level of $\alpha = 5\%$ is used.

[9] Results for the other performance metrics can be found in appendix B.

5.4.1 Model confidence set results

The result of the model confidence set approach of Hansen et al. (2011) is reported in table 4.

	Half-minute sampling	One-minute sampling	Five-minute sampling
Models included (p)	M_1 (1), M_3 (0.488)	M_1 (1), M_2 (0.238), M_3 (0.238), M_4 (0.238)	M_1 (1)

Table 4: MCS results based on F1 losses over time. P-values are reported in parenthesis. M_1, M_2, M_3, M_4 refers to the LSTM, LSTM-CNN, LSTM-Attention LSTM-CNN-Attention model presented in section 5.1

It can be seen from table 4 that the strongest model from table 3 is always included in the MCS with a p-value of one, indicating that it is very strongly in the set. In one minute sampling we see that data is not very informative and all models are included into the final set $\widehat{\mathcal{M}}^*$. For half-minute sampling the attention model is also included. These results are in line with the performance results reported in table 3.

5.5 Microstructure noise correction study

In this paper a new version of Lee and Mykland (2007) is applied that uses the bias correction of Jiang and Oomen (2008) to handle microstructure noise. The magnitude of the bias correction was studied over the different assets and the magnitude was on average very small. It was not found to make a substantial difference from the original Lee and Mykland (2007) test results. Even though the bias correction was found to be small in this study, it is still recommended to use a bias correction or a similar test that accounts for microstructure noise (Jiang and Oomen, 2008). From studying the autocorrelation functions at each sampling interval the first order correlation is found to be significant and negative for all sampling schemes. As found in Jiang and Oomen (2008) the autocorrelation is high at ultra high-frequency and decreases with the sampling frequency[10]. This is also inline with the findings of Hansen and Lunde (2006), that find small microstructure noise on the liquid Dow Jones stocks. One could also have applied the method used in Christensen et al. (2014) to account for microstructure noise in the Lee and Mykland (2007) test, where a non-overlapping pre-averaging estimator is applied.

[10] Autocorrelation plots can be found in appendix C

6 Discussion

6.1 Jumps

One could as argued in Christensen et al. (2014) *"zoom"* further in to find that jumps observed at discrete times are large continuous drifts in prices and not actual jumps. As mentioned in this paper and in Christensen et al. (2014) identifying the discontinuous jump part of the price process becomes important for option pricing. But the identified jumps found here might in reality just be strong price drifts, not belonging to the discontinuous part.

6.2 Features

One could extend the feature set proposed in this paper with additional news-information features by scraping high-frequency news. This could be obtained from real-time financial news databases, but could also be obtained from reliable sources on social media pages such as Twitter. Lee and Mykland (2007) suggests Factiva which provides financial news.

6.3 Alternative models

As mentioned by Lipton et al. (2015a) another possible method that can effectively capture financial time-series are the hidden Markov models (HMM). These HMMs will assume that the observed sequence follows a sequence of unobserved probability states. However, these traditional HMM approaches a limited by the states that must be drawn from a discrete state space S. They tend to become infeasible when the set of states becomes very large. These models also tend to become computationally infeasible with long term dependencies. Recurrent neural networks on the other hand model the long term dependencies (Lipton et al., 2015a) with the computational complexity growing at most quadratically.

7 Conclusion

Recent studies Tsantekidis et al. (2017a,b); Mäkinen (2017); Mäkinen et al. (2019); Ntakaris (2019); Kercheval and Zhang (2015) are able to make useful predictions using information present in limit order book data. In this paper, similar results are found and good prediction ability is seen in regards to predicting jumps in forthcoming intervals over shorter periods.

This paper differentiates itself from recent studies that have used the jump detection test of Lee and Mykland (2007), which does not account for microstructure noise. In this study the bias correction of Jiang and Oomen (2008) is added to the Lee and Mykland (2007) test to make it robust to microstructure noise. This allows one to use the

additional information the test provides while accounting for microstructure noise.

Four different models are studied and their predictive ability of jumps is analyzed. Here the $F1$ score is used as a measure to compare the models. It measures how well the models predict the positive samples, which here are jumps in the stock price. The LSTM model is used as a really strong base model, that is tailored specifically to handle time-series data. Furthermore, it is also able to handle dependencies between feature-variables for each given observation. In addition, the neural network models do not impose assumptions about the structure of these dependencies but instead allow these to be estimated from the data. Three other models were also used in this study which added extra complexities to the LSTM model. Convolutions were used to detect patterns in the time and feature dimension. Also an attention model were used. Attention adds more (less) weight to the important (irrelevant) features. Finally a model combining both attention and convolutions were used on top of the LSTM network. Among the four models the LSTM model without extra added complexities were found to be the highest performing model. This is contrary to the results found in Mäkinen et al. (2019). Here no extra value was found when adding convolutions and attention (or a combination) to the high performing LSTM model. Adding attention was found to increase recall scores, improving the ability to correctly detect jumps. But it also causes the model to produce more false positives, and thus it lowered the the overall F1-score.

Furthermore, in this study a new set of features were created that can be used as additional limit order book derived features. These features consisted mainly of using volume weighted averages of the available information in the interval under consideration.

Thus, it is found that limit order books provides useful information together with deep learning models to predict jump arrivals. This can have important implications for both risk management, option pricing, and foreseeing arrival of new information.

References

Yacine Aït-Sahalia. Telling from discrete data whether the underlying continuous-time model is a diffusion. The Journal of Finance, 57(5):2075–2112, 2002.

Yacine Aıt-Sahalia. Disentangling diffusion from jumps. Journal of financial economics, 74(3):487–528, 2004.

Yacine Aït-Sahalia and Jean Jacod. High-frequency financial econometrics. Princeton University Press, 2014.

Torben G Andersen, Luca Benzoni, and Jesper Lund. An empirical investigation of continuous-time equity return models. The Journal of Finance, 57(3):1239–1284, 2002.

Torben G Andersen, Tim Bollerslev, and Nour Meddahi. Analytical evaluation of volatility forecasts. International Economic Review, 45(4):1079–1110, 2004.

Ole E Barndorff-Nielsen and Neil Shephard. Power and bipower variation with stochastic volatility and jumps. Journal of financial econometrics, 2(1):1–37, 2004.

Ole E Barndorff-Nielsen and Neil Shephard. Econometrics of testing for jumps in financial economics using bipower variation. Journal of financial Econometrics, 4 (1):1–30, 2006.

Ole E Barndorff-Nielsen, P Reinhard Hansen, Asger Lunde, and Neil Shephard. Realized kernels in practice: Trades and quotes, 2009.

Christian T Brownlees and Giampiero M Gallo. Financial econometric analysis at ultra-high frequency: Data handling concerns. Computational Statistics & Data Analysis, 51(4):2232–2245, 2006.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16:321–357, 2002.

Mikhail Chernov, A Ronald Gallant, Eric Ghysels, and George Tauchen. Alternative models for stock price dynamics. Journal of Econometrics, 116(1-2):225–257, 2003.

Kim Christensen, Roel CA Oomen, and Mark Podolskij. Fact or friction: Jumps at ultra high frequency. Journal of Financial Economics, 114(3):576–599, 2014.

Jacob Cohen. A coefficient of agreement for nominal scales. Educational and psychological measurement, 20 (1):37–46, 1960.

Rama Cont. Statistical modeling of high-frequency financial data. IEEE Signal Processing Magazine, 28(5):16–25, 2011.

Rama Cont, Peter Tankov, and Ekaterina Voltchkova. Hedging with options in models with jumps. In Stochastic analysis and applications, pages 197–217. Springer, 2007.

Rama Cont, Sasha Stoikov, and Rishi Talreja. A stochastic model for order book dynamics. Operations research, 58 (3):549–563, 2010.

Rama Cont, Arseniy Kukanov, and Sasha Stoikov. The price impact of order book events. Journal of financial econometrics, 12(1):47–88, 2014.

Matthew Dixon. A high-frequency trade execution model for supervised learning. High Frequency, 1(1):32–52, 2018.

Bjørn Eraker. Do stock prices and volatility jump? reconciling evidence from spot and option prices. The Journal of Finance, 59(3):1367–1403, 2004.

Bjørn Eraker, Michael Johannes, and Nicholas Polson. The impact of jumps in volatility and returns. The Journal of Finance, 58(3):1269–1300, 2003.

Thomas N Falkenberry. High frequency data filtering. White paper, Tick Data Inc, 2002.

Joseph L Fleiss, Bruce Levin, and Myunghee Cho Paik. Statistical methods for rates and proportions. John Wiley & Sons, 2013.

Felix A Gers, Douglas Eck, and Jürgen Schmidhuber. Applying lstm to time series predictable through time-window approaches. In Neural Nets WIRN Vietri-01, pages 193–200. Springer, 2002.

Eric Ghysels, Pedro Santa-Clara, and Rossen Valkanov. Predicting volatility: getting the most out of return data sampled at different frequencies. Journal of Econometrics, 131(1-2):59–95, 2006.

Alex Graves. Supervised sequence labelling. In Supervised sequence labelling with recurrent neural networks, pages 5–13. Springer, 2012.

Peter R Hansen and Asger Lunde. Realized variance and market microstructure noise. Journal of Business & Economic Statistics, 24(2):127–161, 2006.

Peter R Hansen, Asger Lunde, and James M Nason. The model confidence set. Econometrica, 79(2):453–497, 2011.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580, 2012.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.

Jean Jacod, Yingying Li, Per A Mykland, Mark Podolskij, and Mathias Vetter. Microstructure noise in the continuous case: the pre-averaging approach. Stochastic processes and their applications, 119(7):2249–2276, 2009.

George Jiang and Roel Oomen. A new test for jumps in asset prices. Preprint, 2005.

George J Jiang and Roel CA Oomen. Testing for jumps when asset prices are observed with noise–a "swap variance" approach. Journal of Econometrics, 144(2):352–370, 2008.

Juho Kanniainen and Ye Yue. The arrival of news and jumps in stock markets. Available at SSRN 2897503, 2019.

Yakup Kara, Melek Acar Boyacioglu, and Ömer Kaan Baykan. Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange. Expert systems with Applications, 38(5):5311–5319, 2011.

Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Shun Chen. Lstm fully convolutional networks for time series classification. IEEE Access, 6:1662–1669, 2017.

Alec N Kercheval and Yuan Zhang. Modelling high-frequency limit order book dynamics with support vector machines. Quantitative Finance, 15(8):1315–1329, 2015.

Suzanne S Lee. Jumps and information flow in financial markets. The Review of Financial Studies, 25(2):439–479, 2011.

Suzanne S Lee and Per A Mykland. Jumps in financial markets: A new nonparametric test and jump dynamics. The Review of Financial Studies, 21(6):2535–2563, 2007.

Suzanne S Lee and Per A Mykland. Jumps in equilibrium prices and market microstructure noise. Journal of Econometrics, 168(2):396–406, 2012.

Zachary C Lipton, Charles Elkan, and Balakrishnan Naryanaswamy. Optimal thresholding of classifiers to maximize f1 measure. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 225–239. Springer, 2014.

Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019, 2015a.

Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzel. Learning to diagnose with lstm recurrent neural networks. arXiv preprint arXiv:1511.03677, 2015b.

Lily Y Liu, Andrew J Patton, and Kevin Sheppard. Does anything beat 5-minute rv? a comparison of realized measures across multiple asset classes. Journal of Econometrics, 187(1):293–311, 2015.

Milla Mäkinen. Neural networks in stock price jump prediction. 2017.

Ymir Mäkinen, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Forecasting jump arrivals in stock prices: new attention-based network architecture using limit order book data. Quantitative Finance, pages 1–18, 2019.

Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long short term memory networks for anomaly detection in time series. In Proceedings, page 89. Presses universitaires de Louvain, 2015.

Burton G Malkiel and Eugene F Fama. Efficient capital markets: A review of theory and empirical work. The journal of Finance, 25(2):383–417, 1970.

Robert C Merton. Option pricing when underlying stock returns are discontinuous. Journal of financial economics, 3(1-2):125–144, 1976.

Per A Mykland and Lan Zhang. The econometrics of high frequency data. Statistical methods for stochastic differential equations, 124:109, 2012.

Farzad Noorian and Philip HW Leong. Dynamic hedging of foreign exchange risk using stochastic model predictive control. In 2014 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr), pages 441–448. IEEE, 2014.

Adamantios Ntakaris. Mid-price movement prediction in limit order books using feature engineering and machine learning. 2019.

Adamantios Ntakaris, Martin Magris, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Benchmark dataset for mid-price prediction of limit order book data. arXiv preprint arXiv:1705.03233, 2017.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In International conference on machine learning, pages 1310–1318, 2013.

Nikolaos Passalis, Avraam Tsantekidis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Time-series classification using neural bag-of-features. In 2017 25th European Signal Processing Conference (EUSIPCO), pages 301–305. IEEE, 2017.

Monika Piazzesi. Bond yields and the federal reserve. Journal of Political Economy, 113(2):311–344, 2005.

Mark Podolskij, Mathias Vetter, et al. Estimation of volatility functionals in the simultaneous presence of microstructure noise and jumps. Bernoulli, 15(3):634–658, 2009.

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

Justin Sirignano and Rama Cont. Universal features of price formation in financial markets: perspectives from deep learning. Quantitative Finance, 19(9):1449–1459, 2019.

Peter Tankov. Financial modelling with jump processes. Chapman and Hall/CRC, 2003.

Dat Thanh Tran, Alexandros Iosifidis, Juho Kanniainen, and Moncef Gabbouj. Temporal attention-augmented bilinear network for financial time-series data analysis. IEEE transactions on neural networks and learning systems, 30(5):1407–1418, 2018.

Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Forecasting stock prices from the limit order book using convolutional neural networks. In 2017 IEEE 19th Conference on Business Informatics (CBI), volume 1, pages 7–12. IEEE, 2017a.

Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Using deep learning to detect price change indications in financial markets. In 2017 25th European Signal Processing Conference (EUSIPCO), pages 2511–2515. IEEE, 2017b.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in neural information processing systems, pages 5998–6008, 2017.

Lilian Weng. Attention? attention! lilianweng.github.io/lil-log, 2018. URL http://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html.

Paul J Werbos et al. Backpropagation through time: what it does and how to do it. Proceedings of the IEEE, 78(10): 1550–1560, 1990.

Yi Xue, Ramazan Gencay, and Stephen Fagan. Jump detection with wavelets for high-frequency financial time series. Quantitative Finance, 14(8):1427–1444, 2014.

Hanxue Yang and Juho Kanniainen. Jump and volatility dynamics for the s&p 500: Evidence for infinite-activity jumps with non-affine volatility dynamics from stock and option markets. Review of Finance, 21(2):811–844, 2017.

Ban Zheng, Eric Moulines, and Frédéric Abergel. Price jump prediction in limit order book. arXiv preprint arXiv:1204.1381, 2012.

A Neural network architecture examples

A.1 Normal feed forward neural network

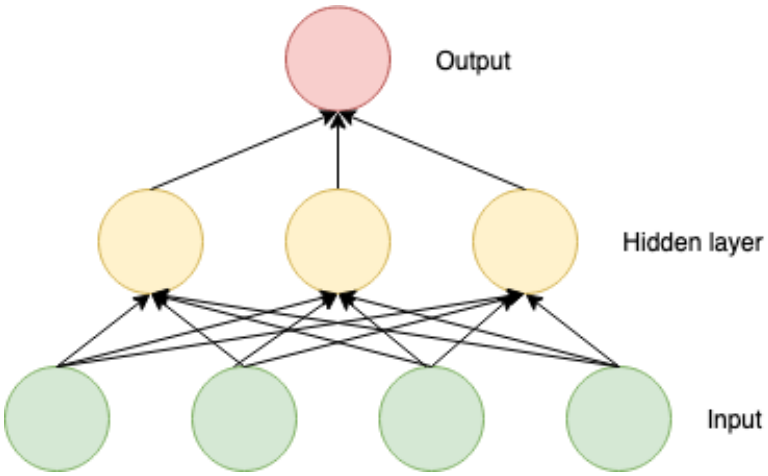

Figure 5: Standard feed forward neural network structure from input at the bottom to output at the top.

The architecture of the feed forward neural network starts from the inputs and are then summarized into a node using the weights and then transformed by the activation function. Then it is fed forward to the next layer, which in fig. 5 here is the output layer. The output layer can consist of multiple outputs and can be both regression or classification output.

A.2 Recurrent Neural Network

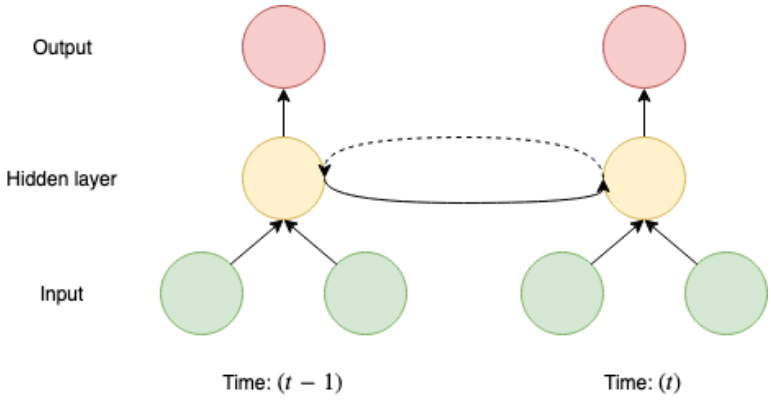

Figure 6: Simple recurrent neural network structure with time dependence.

Figure 6 depicts the simple recurrent neural network structure. The input is transformed into the hidden layer. The hidden layers have in different time steps have dependence between them and the correct weights are learned in training. The dependence can be bidirectional, i.e. both forward in time and backwards in time, or it can be just one of the ways.

A.3 Convolutional neural network layer

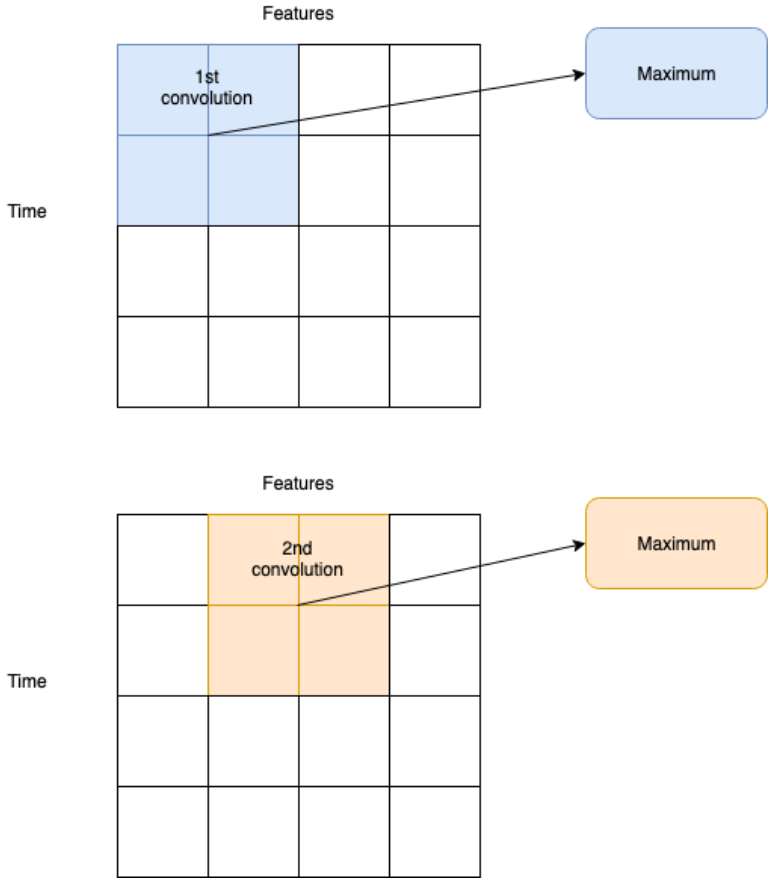

Figure 7: 1st and 2nd convolution in a convolutional layer that only jumps one position after each convolution.

B Model results

B.1 Recall

Model	Half-minute sampling	One-minute sampling	Five-minute sampling
LSTM	0.77 (0.2)	0.73 (0.22)	0.56 (0.28)
LSTM-CNN	0.65 (0.19)	0.67 (0.2)	0.52 (0.46)
LSTM-Attention	**0.96** (0.07)	**0.98** (0.04)	**0.87** (0.28)
LSTM-CNN-Attention	0.62 (0.2)	0.66 (0.26)	0.51 (0.45)

Table 5: Recall score of the four models. Standard errors are reported in parenthesis. The best performing model for each sampling interval is reported in bold.

B.2 Precision

Model	Half-minute sampling	One-minute sampling	Five-minute sampling
LSTM	**0.65** (0.21)	**0.75** (0.22)	**0.7** (0.27)
LSTM-CNN	0.48 (0.14)	0.46 (0.16)	0.21 (0.2)
LSTM-Attention	0.51 (0.17)	0.44 (0.16)	0.4 (0.26)
LSTM-CNN-Attention	0.42 (0.12)	0.45 (0.16)	0.21 (0.21)

Table 6: Precision score of the four models. Standard errors are reported in parenthesis. The best performing model for each sampling interval is reported in bold.

B.3 Cohen's kappa

Model	Half-minute sampling	One-minute sampling	Five-minute sampling
LSTM	**0.64** (0.2)	**0.7** (0.22)	**0.58** (0.27)
LSTM-CNN	0.49 (0.15)	0.47 (0.16)	0.21 (0.3)
LSTM-Attention	0.61 (0.17)	0.54 (0.17)	0.45 (0.28)
LSTM-CNN-Attention	0.43 (0.15)	0.46 (0.19)	0.21 (0.3)

Table 7: Cohen's kappa score of the four models. Standard errors are reported in parenthesis. The best performing model for each sampling interval is reported in bold.

C Autocorrelation function plots

Autocorrelation function for each sampling interval for the UNH stock.

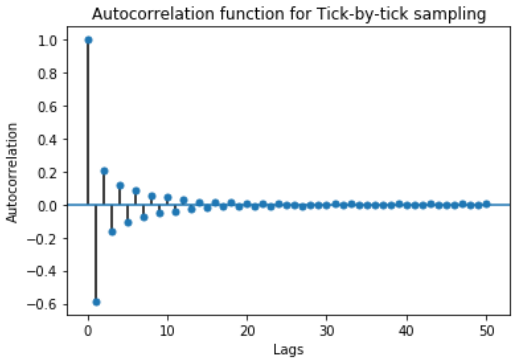

Figure 8: Autocorrelation function for tick level returns

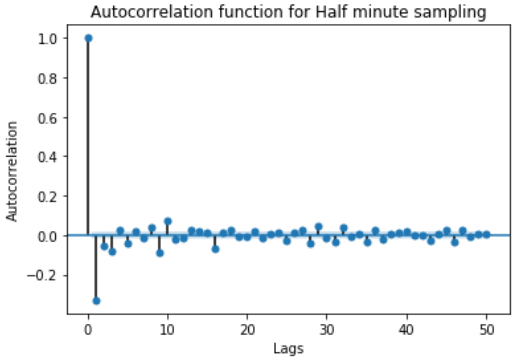

Figure 9: Autocorrelation function for half-minute level returns

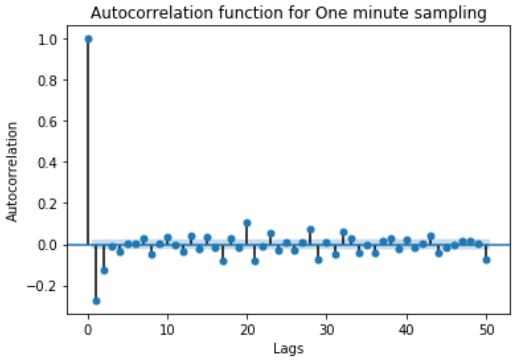

Figure 10: Autocorrelation function for one-minute level returns

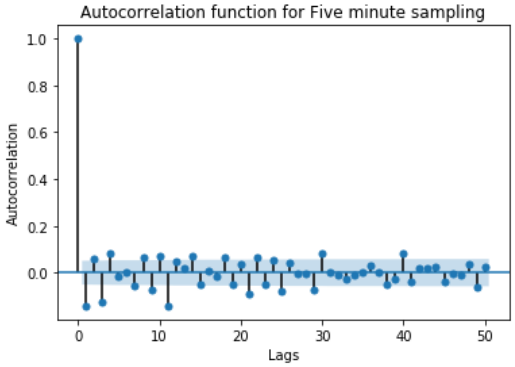

Figure 11: Autocorrelation function for five-minute level returns

D List of stocks included in analysis

Ticker/Symbol	Full company name
MCD	McDonald's Corporation
WBA	Walgreens Boots Alliance, Inc.
NKE	NIKE, Inc.
XOM	Exxon Mobil Corporation
PG	The Procter & Gamble Company
DIS	The Walt Disney Company
AAPL	Apple Inc.
IBM	International Business Machines Corporation
MSFT	Microsoft Corporation
CVX	Chevron Corporation
MMM	3M Company
CAT	Caterpillar Inc.
CSCO	Cisco Systems, Inc.
JNJ	Johnson & Johnson
UNH	UnitedHealth Group Incorporated
WMT	Walmart Inc.
UTX	United Technologies Corporation
INTC	Intel Corporation
HD	The Home Depot, Inc.
V	Visa Inc.
GS	The Goldman Sachs Group, Inc.
VZ	Verizon Communications Inc.
JPM	JPMorgan Chase & Co.
SPY	S&P 500 ETF Trust

Table 8: List of the company names and their symbols for the stocks used in the analysis of this paper.