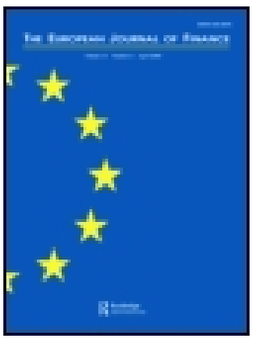

The European Journal of Finance

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/rejf20

Ascertaining price formation in cryptocurrency markets with machine learning

Fan Fang, Waichung Chung, Carmine Ventre, Michail Basios, Leslie Kanthan, Lingbo Li & Fan Wu

To cite this article: Fan Fang, Waichung Chung, Carmine Ventre, Michail Basios, Leslie Kanthan, Lingbo Li & Fan Wu (2021): Ascertaining price formation in cryptocurrency markets with machine learning, The European Journal of Finance, DOI: 10.1080/1351847X.2021.1908390

To link to this article: https://doi.org/10.1080/1351847X.2021.1908390

Published online: 05 Apr 2021.

THE EUROPEAN JOURNAL OF FINANCE
https://doi.org/10.1080/1351847X.2021.1908390

Routledge
Taylor & Francis Group

Ascertaining price formation in cryptocurrency markets with machine learning

Fan Fang[a], Waichung Chung[b], Carmine Ventre[a], Michail Basios[c,d], Leslie Kanthan[c,d], Lingbo Li[d] and Fan Wu[d]

[a]King's College London, London, UK; [b]University of Essex, Colchester, UK; [c]University College London, London, UK; [d]Turing Intelligence Technology Limited, London, UK

ABSTRACT

The cryptocurrency market is amongst the fastest-growing of all the financial markets in the world. Unlike traditional markets, such as equities, foreign exchange and commodities, cryptocurrency market is considered to have larger volatility and illiquidity. This paper is inspired by the recent success of using machine learning for stock market prediction. In this work, we analyze and present the characteristics of the cryptocurrency market in a high-frequency setting. In particular, we applied a machine learning approach to predict the direction of the mid-price changes on the upcoming tick. We show that there are universal features amongst cryptocurrencies which lead to models outperforming asset-specific ones. We also show that there is little point in feeding machine learning models with long sequences of data points; predictions do not improve. Furthermore, we solve the technical challenge to design a lean predictor, which performs well on live data downloaded from crypto exchanges. A novel retraining method is defined and adopted towards this end. Finally, the trade-off between model accuracy and frequency of training is analyzed in the context of multi-label prediction. Overall, we demonstrate that promising results are possible for cryptocurrencies on live data, by achieving a consistent 78% accuracy on the prediction of the mid-price movement on live exchange rate of Bitcoins vs. US dollars.

ARTICLE HISTORY
Received 19 November 2019
Accepted 11 March 2021

KEYWORDS
Cryptocurrency; machine learning; predictors ; model classification

1. Introduction

A powerful yet basic toolkit for algorithmic traders would efficiently predict the direction of price changes for financial assets; machine learning techniques, such as deep neural networks, are known as performant predictors for a variety of tasks and setups. This paper focuses on effectively applying neural networks on cryptocurrency market trading systems. Our objective is to predict the price changes; we consider both binary (up/down) and multi-class (e.g. degrees of increase/decrease) prediction of price changes.

The cryptocurrency market is a huge emerging market (Ahamad, Nair, and Varghese 2013). There were over 11, 641 exchanges available on the internet as of July 2018 (En.wikipedia.org 2018). Most of them are exchanges of small capitalization with low liquidity. Exchanges with the highest 24-hour volume are FCoin, BitMEX, and Binance. Bitcoin, as the pioneer and also the market leader, has a market capitalization of over 112 billion USD, and a 24-hour volume over 3.8 billion USD in early July 2018. The cryptocurrency market is one of the most rapidly growing markets in the world and is also considered one of the most volatile markets to trade in. For example, the price of a single Bitcoin increased significantly, from near zero in 2013 to nearly 19, 000 USD in 2017. For some alt-coins, the price can increase or fall over 50% within a day. Therefore, having a method to accurately predict these changes is a pervasive task, but one that could achieve a long-term profit for cryptocurrency traders.

CONTACT Fan Fang fan.fang@kcl.ac.uk

Table 1. Comparison between our research and Easley et al. #FDR refers to Feature Dimensionality Reduction.

	Our research	Easley et al.
Target Market	Cryptocurrency	Future contracts
Data Frequency	Tick-level	Tick-level
Core ML Model	LSTM	Random forest
Features	16	6
Methods of #FDR	PCA & Autoencoder	Correlation coefficient

There are a number of research papers that studied the structure of the limit order book (i.e. the bids at both sides of the market) and, more generally, the micro-structure of the market by using different methods ranging from stochastic to statistical and machine learning approaches (Biais, Hillion, and Spatt 1995; Huang, Nakamori, and Wang 2005; Altay and Satman 2005; Fletcher 2012; Nousi et al. 2019). The Limit-Order Books of cryptocurrency markets share many common characteristics with those of traditional markets, especially at the microstructure level. The main difference is due to the lower average depth of the book in cryptocurrency markets; this leads to other differences related to the way the order book absorbs order flows and trade flow imbalances (Silantyev 2019).

The objective of this paper is to understand the way prices at either side of the market move. Motivated by related literature, we focus on a particular measure, the *mid-price*, which intuitively captures the average difference between the best ask (the lowest price sellers are willing to accept) and best bid (the highest price buyers are willing to pay). Towards this aim we could, for example, use Markov chains to model the limit order book (Kelly and Yudovina 2017). We could view the limit order book as a queuing system with a random process and use birth-and-death chains to model its behavior. From this perspective, a natural way to explain the mid-price movement is to consider the value of the mid-price as the state of the chain. This value is controlled by the ratio between the probability of birth transitions p and the probability of death transitions q. A ratio p/q greater than 1 within a short interval of time indicates that there is a higher chance for birth transitions to happen (more buyers), and the value of the mid-price is expected to increase. Similarly, if the ratio is smaller than 1, the value of the mid-price is expected to decrease (Sundarapandian 2009). The problem with this approach is the way it models the order book, namely, is the limit order book a queuing system? If it were, how to correctly simulate the random process and how to accurately estimate p and q become vital questions for this approach.

Easley et al. (2019) researched similar topic in price formation. They investigated price dynamics in current complex markets using machine learning algorithms. We compare our research design with theirs along several dimensions (cf. Table 1). There are several differences. First, our research aims at emerging market – cryptocurrency market – which is characterized by high risk and high returns. Easley et al. focus on future contracts; dollar-volume bars are used in their analysis. The core machine learning model in our research is LSTM whilst they focus on Random Forests. Many researchers found that LSTM is more suitable than random forest in handling financial time series (Fischer and Krauss 2018). The memory cell in LSTM allows the model to remember relevant historical information more clearly. Second, our research uses 16 features including basic market features and order book features while Easley et al. focus on features related to volume. Third, there are differences in the methods used to detect features correlation; we use Principal Component Analysis (PCA) and Autoencoders while Easley et al. use Correlation Coefficient. Although Correlation Coefficient is a good method to find relationship among selected features, it is hard to reduce features' dimension when some features are strongly related. Combining PCA and Autoencoder could reduce the interference between similar features, as implied by our research. Finally, we design a new retraining method to refresh obsolete predictive machine learning models. Updating the model frequently makes sense in financial prediction, as from our back-testing experiment.

In this research, we propose to adopt a machine learning approach to reveal useful patterns from limit order books. We provide insights to a number of specific technical questions that arise from this approach. Specifically, we show that there are universal features amongst cryptocurrencies that can improve the predictive power of machine learning models, as there are in the case of equities (Sirignano and Cont 2019). The conceptual difference is that Sirignano and Cont focus on equities and our research focused on cryptocurrencies. We also show that feeding more data to train our deep neural network fails to improve the model performances; simpler

single-dimensional models are preferred. Third, we test the model on live data for different periods of varying length, which bears conceptual as well as technical challenges. Conceptually, we show that models developed with an ideal condition (carefully selected and split data) hardly perform well on real world cases, often because of sub-optimal accuracy and inefficient running time. This leads to the engineering challenge of designing a lean model which runs fast on live data, including retraining the model when necessary, whilst retaining accurate predictions. We show in this paper that, certain known architectures can meet both requirements, when using a novel training method that we call *Walkthrough Training*. Finally, we explore the problem of multi-label classification, by predicting 'small' or 'large' increase/decrease of the mid-price; we analyze the trade-off between performance and retrain frequency of Walkthrough Training in this context. Ultimately, our findings pave the way to the design of novel trading strategies and market estimators.

1.1. Related work

Since the birth of the market, traders have been trying to find accurate models to use to make a profit. Many studies and experiments have been conducted based on statistical modeling of the stock price data. Some studies attempted to model the limit order book by using statistical approaches, such as using Poisson Processes and Hawkes Processes to estimate the next coming order and to model the state of the limit order book (Toke and Pomponio 2012; Abergel and Jedidi 2015). Brooks et al. (2019) pointed out that financial data science and econometrics are highly complementary. The new research paradigm financial data science brings new opportunities for academic research in finance.

Others have used machine learning approaches to estimate the upcoming market condition by applying different machine learning models, such as support vector machine (SVM) (Kercheval and Zhang 2015), convolutional neural network (CNN) (Tsantekidis et al. 2017), Random Forest (Easley et al. 2019), and recurrent network such as Long-Short-Term-Memory (LSTM) (Dixon 2018). These studies show that it is possible to use a data-driven approach to discover hidden patterns within the market. In particular, Kercheval and Zhang (2015) modeled the high-frequency limit order book dynamics by using SVM. They discovered that some of the essential features of the order book lie on fundamental features, such as price and volume, and time-insensitive features like mid-price and bid-ask spread. Nousi et al. (2019) provided extensive study in high frequency limit order book information in predicting mid-price movements. Support vector machine (SVM), Single layer Feedforward Network (SLFN), and Multilayer Perceptron (MLP) are compared in examining whether the classifier learns the general trends and trends of the stock market by learning from some stocks and applying this knowledge to invisible stocks. The research evaluated these models in solving high speed, variance, quantity of limit order book data and showed that the feature extraction model can discover potential auxiliary knowledge (Nousi et al. 2019). As from above, Easley et al. (2019) investigated price dynamics in future contract markets using random forests. Mäkinen et al. (2019) proposed an approach with attention forecasting jump arrivals 1-minute ahead in stock prices. This mechanism, convolutional neural network, and Long Short-Term memory model are compared in their experiments. Tran et al. (2018) considered a neural network layer structure combining the idea of bilinear projection and enabling this layer to detect and focus on critical time information in financial time-series forecasting. Barbon (2019) proposed an encoder–decoder neural network augmented with an attention-based mechanism in predicting future transaction prices in NASDAQ. The results showed the model's behavior prefers liquidity provision rather than front-running strategies. Gu, Kelly, and Xiu (2020) compared different machine learning methods for the canonical problem of empirical asset pricing. Tree and neural networks are best-performing methods in measuring asset risk premiums from their research. Verstyuk (2020) took a range of small to large-scale Long Short-Term Memory recurrent neural networks and compared those to the VAR method. These methods are used to model multivariate time series such as GDP growth, inflation, commodity prices, and so on. The results showed that the neural networks used may also be a useful tool for policy simulation under actual relevant economic conditions, which can also discover different macroeconomic regimes. Finally, Sirignano and Cont (2019) suggested that there might be some universal features on the stock market's limit order book that have a non-linear relationship to the price change. They tried to predict the mid-price movement of the next tick by training a neural network using a significant amount of stock data. Their

findings suggest that instead of building a stock-specific model, a universal model for all kinds of stock could be built.

Most of the studies in the area focus on the traditional stock market like NYSE and NASDAQ (Güresen, Kayakutlu, and Daim 2011). Many researchers have studied these exchanges for many years. The quality of data and the market environment are more desirable than those of the cryptocurrency market. Although the traditional stock market may provide a less volatile and more regulated environment for traders, the high volatility of the cryptocurrency market may provide a higher potential return. Our research aims to apply the same philosophy to the cryptocurrency market and replicate the findings above. In other words, we try to model the cryptocurrency market by using a data-driven approach. In this paper, experiments are focusing on the engineering side of this approach.

Understanding price formation through mid-price for cryptocurrencies becomes even harder due to the fact that they are distinct from traditional fiat-based currencies. The latter are usually issued by banks or governments. The only way to create Bitcoins, the currently dominant cryptocurrency, is to run a computationally intensive algorithm to add new blocks to the blockchain. People who participate in this processing will verify transactions on the blockchain and try to earn Bitcoins as the reward of adding new blocks. These people are usually referred to as Bitcoins miners. The protocol of the Bitcoin fixed its total supply at 21 million (Nakamoto 2008). Every transaction on the blockchain is protected by a cryptographic hash algorithm called SHA-256. It is a computational intensive hash algorithm that is implemented to verify blocks on the blockchain. For instance, if a counterfeiter wants to forge a block on the blockchain, they will also need to redo all the hashing before that block. This property provides a trustless foundation for Bitcoin because neither an individual nor an institution can counterfeit the currency or the transaction unless it has a computational power in excess of the majority of the network (Nakamoto 2008).

Multi-label prediction is widely used in image processing, character recognition and forecasting of decisions or time series. Complex trading strategies might require more than binary classification. One may use the status box method to measure different stock statuses such as turning point, flat box, and up-down box (Zhang, Li, and Pan 2016) in order to reflect the relative position of the stock and classify whether the state coincides with the stock price trend. In this research, we propose to use the transaction fee as a threshold to decide whether the designated cryptocurrency market has a long or short signal. When prediction of price movement is under transaction fee, it means in the next trading cycle the market falls in a 'Buffer area' where the market is in a relatively stable position.

Marcus (2018) points critical aspects on the application of machine learning, especially deep learning, which provide insights for finance. First, machine learning models are data-hungry. Marcus gives an example in his research: one cannot rely on millions of training examples to represent abstract relationships between similar algebraic variables. Accordingly, in financial predictions, machine learning models might be misaligned due to limited training examples. Second, the knowledge gathered by deep learning systems is primarily concerned with correlations between features, rather than abstractions like quantified statements. For these reasons, machine learning is not yet able to reach human-level cognitive flexibility. Ultimately, a machine learning algorithm applied to finance needs to improve adaptability when facing a new market or order structure.

1.2. Roadmap

The paper is organized as follows. In Section 2, we provide a brief overview of the tools adopted, including machine learning, limit order books, and data sources we used. In Section 3, we design experiments to address our research questions. In Section 4, we give a brief discussion of the validity of our findings. Section 5 gives a conclusion of this paper.

2. Our tools

In this section, we first review the background of machine learning and limit order books, before introducing an overview of the trading system where our prediction model is trained on.

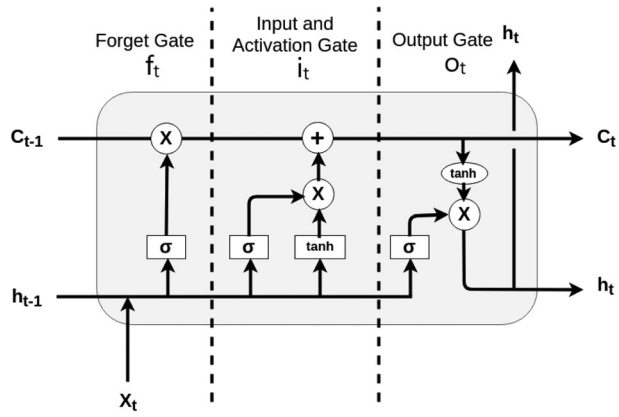

Figure 1. An overview of an LSTM cell

2.1. Machine learning

Artificial neural networks are computational algorithms mimicking biological neural systems, such as human brains. These algorithms are designed to recognize and generalize patterns from the input, and memorize them as weights in the neural network. The basic unit of a neural network is a neuron; a simple neural network, which is a conglomeration of neurons, is called Perceptron.

The neural network used in this paper is a type of recurrent neural network called Long-Short-Term-Memory (LSTM) (Hochreiter and Schmidhuber 1997). This is distinct from the feed-forward neural network such as Perceptrons, since the output of the neural network sends feedback to the input and affects the subsequent output. Therefore, LSTM is better suited for handling sequential data where the previous data can have an impact on subsequent data; this, in principle, works well for time series data for price prediction and forecasting (Figure 1).

An LSTM cell contains a few gates and a cell status to help the LSTM cell decide what information should be kept and what information should be forgotten. As a result, the LSTM cell can recall important features from the previous prediction by having a cell state. An LSTM cell can also be viewed as a combination of a few simple neural networks, each of them serving a different purpose. The first one is the forget gate (Hochreiter and Schmidhuber 1997). The previous output is concatenated with the new input and passed through a sigmoid function. After that, the output of the forget gate, f_t, will perform a Hadamard product (element-wise product) with the previous cell's state. Note that f_t is a vector containing elements that have a range from 0 to 1. A number closer to 0 means the LSTM should not recall it, whilst a number closer to 1 means the LSTM should recall and carry on to the next operation. This process helps the LSTM select which elements are to forget and remember, respectively. The second one is the input and activation gates (Hochreiter and Schmidhuber 1997). This process concatenates the previous output with the new input, determines which element should be ignored, and updates the internal cell state. The cell state is then updated by a combination of the output and a transformation of the input. The third one is the output gate (Hochreiter and Schmidhuber 1997). This process helps determine the output of the cell. Finally, the output of the LSTM cell is the Hadamard product of the current internal cell state and the output of the output gate (Christopher 2015; Adam 2015).

We use Root Mean Square Propagation (RMSprop) (Tieleman and Hinton 2012) – a stochastic gradient descent optimizer – to train the neural network, with the learning rate divided by the exponentially weighted average. Optimizer, learning rate, and loss function are core concepts in machine learning models. Optimizer ties together the loss function and model parameters by updating the model in response to the output of the loss function. Loss function is a method of evaluating how well the algorithm models a given dataset, it tells the optimizer whether it is moving in the right or wrong direction. The learning rate is a hyperparameter that controls how much the weight values should change in response to the estimated error each time the model's weights are updated.

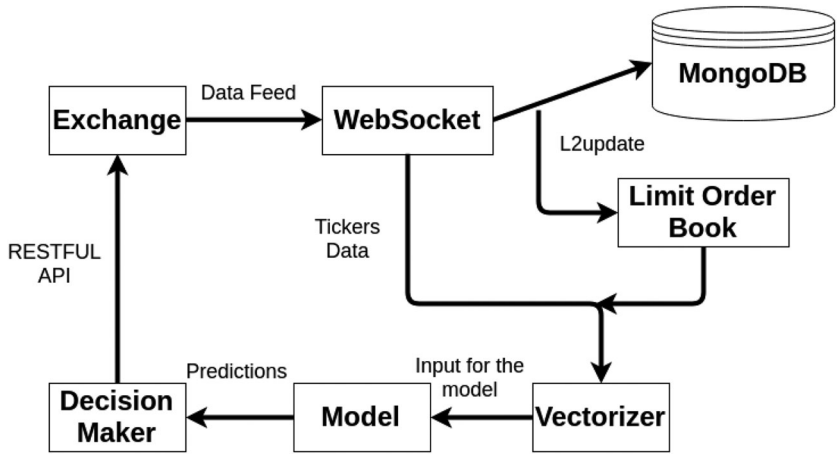

Figure 2. An overview of a simple trading system.

In our experiments, we also tested the use of an adaptive moment estimation, Adam in short, as the optimizer. While we observed that Adam helps the neural network to converge faster, we noted a tendency to overfit the data: the validation set has an increasing loss while the training set has a decreasing loss. This motivates our choice of RMSprop as optimizer.

2.2. Limit order books

The limit order book is technically a log file in the exchange showing the queue of the outstanding orders based on their price and arrival time. Let p_b be the highest price at the buy side, which is called the best bid. The best bid is the highest price that a trader is willing to pay to buy the asset. Let p_a be the lowest price at the sell side, which is called the best ask. The best ask is the lowest price a trader is willing to accept for selling the asset.

The mid-price of an asset is the average of the best bid and the best ask of the asset in the market.

$$M_p = \frac{(p_b + p_a)}{2}.$$

There are other metrics that are also useful for describing the state of the limit order book: Spread, Depth, and Slope.

2.3. Data source and overview of the envisioned trading system

Numerous exchanges provide Application Programming Interface (API) for systematic traders or algorithmic traders to connect to the exchange via software. Usually, an exchange provides two types of API, a RESTful API, and WebSocket API. Some exchanges also provide a Financial Information eXchange (FIX) protocol. In this study, a WebSocket API from an exchange called GDAX (Global Digital Asset Exchange) is used to retrieve the level-2 limit order book live data (GDAX 2018). The level-2 data provides prices and aggregated depths for top 50 bids and asks. GDAX is one of the largest exchanges in the world owned by the Coinbase company.

Our focus is to design a model that can successfully predict the mid-price movement in the context of cryptocurrencies. Such a model is a component of a trading system, as shown in Figure 2. There are a few essential components for the trading system. First of all, the WebSocket is used to subscribe to the exchange and receive live data including tickers, order flows, and the limit order book's update. Tickers data usually appears when two orders of the opposite side are matched and the opening of a candle on a candlestick chart. Tickers contain the best bid, best ask, and the price, thus reflecting the change in price in real time.

The ways the updates to the limit order book are communicated differ. Some exchanges provide a real-time snapshot of the order book. Some exchanges, including GDAX, only provide the update, i.e. updated data of a specific price and volume on the limit order book. Therefore, a local real-time limit order book is required to synchronize with the exchange limit order book. Additionally, we need to store all the data in a database. In this study, a non-relational database called MongoDB has been used to this purpose. Unlike a traditional relational database, MongoDB stores unstructured data in a JSON-like format as a collection of documents. The advantage of using a non-relational database is that data can be stored in a more flexible way. The local copy of the limit order book is reconstructed by using level-2 limit order book updates. The reconstructed limit order book can provide information on the shape and status of the actual exchange limit order book. This limit order book can be used for calculating order imbalance and can provide quantified features of the limit order book. The input to the model is then finalized by a vectorizer, used as a data parser, combining information and extracting features from the ticker data and the local limit order book. Features are then reshaped into the format that can fit into an LSTM model.

We leave to future research the design and experimentation of a decision maker, which should make use of the prediction given by the trained model and help manage the inventory. If the inventory and certain thresholds are met, the decision-maker would place an order to the exchange based on the prediction from the trained LSTM model through RESTful API.

3. Experimental study

3.1. Objective

The purpose of this research is to process real-time tick data using machine learning neural network approach on cryptocurrency trading system. As a machine learning model based on high-frequency trading, *accuracy* of prediction and computational efficiency are both important factors to consider in this research; accuracy here refers to the percentage of correct predictions made by the model.

3.2. Dataset

The data used in this study is live data recorded via a WebSocket through the GDAX exchange WebSocket API. The data contain the ticker data, level-2 order book updates, and the order submitted to the exchange. The time range of the collected data is from the time of 2018-07-02 17:22:14 to 2018-07-03 23:32:53. BTC-USD data from 2018-08-08 14:31:54 to 2018-08-09 09:01:13, BTC-USD data from 2018-08-11 12:09 to 2018-08-16 23:59, and from 2018-08-24 12:07 to 2018-08-29 23:59 are collected for live back-testing. The order flow data contain 61, 909, 286 records, the tickers data include 128, 593 ticker data points, and the level-2 data contain 40, 951, 846 records. Table 2 lists the available assets on the GDAX exchange and the corresponding number of records.

Following Brandvold et al. (2015), we statistically analyze our datasets in Table 3; 'Dataset1 – Dataset3' refer to data collected from three time periods as discussed in the previous paragraph. We remark that there is no bias of statistical significance in the collected data. Moreover, there are no outliers or extreme trades are present.

Table 2. Amount of data collected.

Product id \data type	Ticker	Level-2	Order Flow
BCH-USD	15,213	1,600,474	2,442,323
BTC-EUR	9769	4,656,627	7,002,588
BTC-GBP	3726	8,849,556	13,280,280
BTC-USD	25,904	4,110,818	6,282,022
ETH-BTC	4016	1,250,202	1,893,851
ETH-EUR	3180	4,876,886	7,323,178
ETH-USD	27,089	6,087,574	9,276,806
LTC-BTC	2167	611,682	923,070
LTC-EUR	4243	1,260,024	1,897,731
LTC-USD	32,203	2,391,377	3,700,271
BCH-EUR	4243	5,822,103	7,934,653

Table 3. Data statistics.

	Mean	Median	Max	Min	St. Dev	Skew	Kurtosis
Dataset1	7,362.338	7,448.725	7,747.219	6,687.066	244.084	−0.578	−1.192
Dataset2	6,260.872	6,298.001	6,596.083	5,904.602	144.310	−0.459	−0.585
Dataset3	6,408.662	6,322.626	7,132.121	5,904.602	335.889	1.045	−0.06

Table 4. Feature set.

Basic features	Description (*i* denotes time step)
f1 $= \{P_i\}$	price
f2 $= \{V_i\}$	last size
f3 $= \{\ln(P_i/P_{i-1})\}$	log return
f4 $= \{P_i - P_{i-1}\}$	price difference
f5 $= \{ema_t = \beta P_t + (1-\beta)ema_{t-1}, \beta = (2/5)\}$	EMA 4 periods
f6 $= \{ema_t = \beta P_t + (1-\beta)ema_{t-1}, \beta = (1/5)\}$	EMA 9 periods
f7 $= \{ema_t = \beta P_t + (1-\beta)ema_{t-1}, \beta = (2/19)\}$	EMA 18 periods
f8 $= \{rsi = 100 - 100/(1+RS)$, $RS = AvgGain/AvgLoss$ *in* 3 *periods*$\}$	RSI 3 period
Order Book Features	Description ($n = 7$)
f9 $= \{[(P_i^{ask} - P_i^{bid})/2]_{i=1}^{n}\}$	bid-ask spread
f10 $= \{[(P_i^{ask} + P_i^{bid})/2]_{i=1}^{n}\}$	mid-price
f11 $= \{[(P_i^{ask} + P_i^{bid})/2]_{i=1}^{n} - [(P_{i-1}^{ask} + P_{i-1}^{bid})/2]_{i=2}^{n}\}$	mid-price difference
f12 $= \{[P_i^{ask}, P_i^{bid}]_{i=1}^{n}\}$	bids and asks
f13 $= \{[D_i^{ask}, D_i^{bid}]_{i=1}^{n}\}$	depths of bids and asks
f14 $= \{\sum_{i=1}^{n}[D_i^{ask} + D_i^{bid}]\}$	cumulative sum of the depths
f15 $= \{[P_i^{bid}/P_i^{ask}]_{i=1}^{n}\}$	slope of bids and asks
f16 $= \{cumsum[(P_i^{ask} - P_i^{bid})_{i=1}^{n}]\}$	Cumulative sum of bid-ask diff

In Table 4, we have identified the set of features that we use as out input data; these are divided into two categories: basic and order book features. All of these can be directly computed from the aforementioned data.

3.3. Methodology

3.3.1. Model architecture

The simple architecture in Figure 3 served as the predictive model in this study. This neural network contains two layers of LSTM cells, one layer of fully connected neurons and one layer of softmax as the output layer which outputs the probability of price movements. The two layers of LSTM cells can be viewed as a filter for capturing non-linear features from the data, and the fully connected layer can be viewed as the decision layer based on the features provided by the last LSTM layer. This neural network is designed as simple as possible because in the tick data environment, every millisecond matters. Reducing the number of layers and neurons can significantly reduce the computational complexity, thus the time required for the data processing.

3.3.2. Multi-label prediction

Binary classification can be scarcely informative to a trader, as 'small' variations are not differentiated from 'big' ones. One might want to hold one's position in the former case and transact only in the latter.

We use 1-min and 5-min data to demonstrate the rate of price change, defined as the ratio between the price change and the transaction (close) price. In both cases, most relative price changes fall in −0.25% and 0.25%. Often these percentages are less than the transaction fees and traders should be able to know when this is the case to develop a successful trading strategy. Therefore, we also investigate multi-label prediction based on trading strategy needs. In this multi-label prediction, we replace binary target prediction with four-target prediction. At the structure level, we have four softmax units as output layer instead of two units. By effectively set the boundaries of four units, we can transform the original two-class classifier into a four-class classifier.

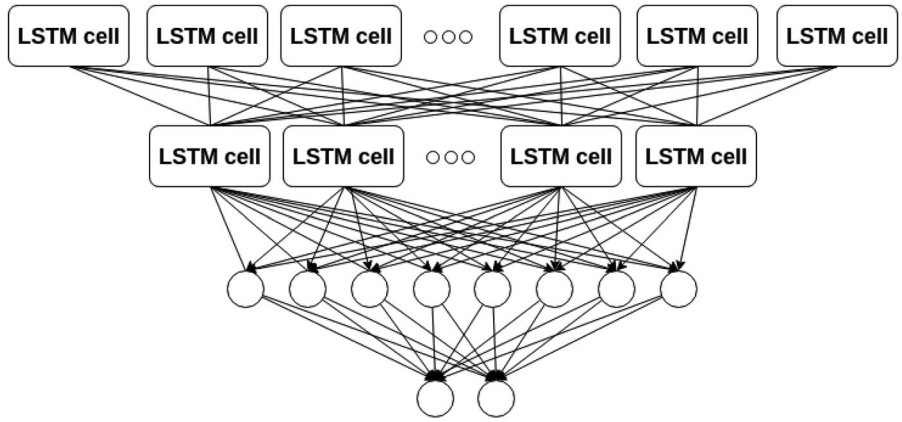

Figure 3. LSTM model architecture.

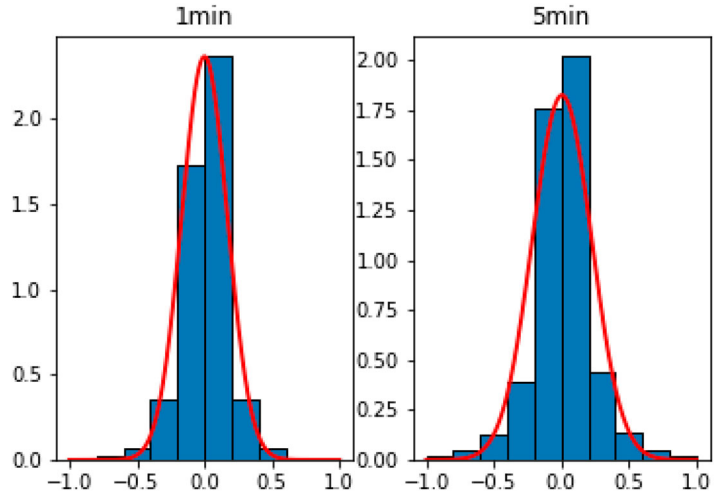

Figure 4. Distribution of historical price changes.

Table 5. Multi-label prediction.

Label	Relative price change	Type
Significant increase	$(+0.2\%, +\infty)$	Sensitive Interval
Significant decrease	$(-0.2\%, -\infty)$	
Insignificant increase	$(0, +0.2\%]$	Insensitive Interval
Insignificant decrease	$[-0.2\%, 0)$	

Using the fees used by Coinbase Pro (Pro 2018), we use $\pm 0.2\%$ of the transaction price as a reasonable threshold to differentiate large and small changes (see Table 5 where we also name the intervals for future references) (Figure 4).

3.3.3. Walkthrough training

Prediction model in financial market has timeliness; this is especially true for the high-frequency financial market. For example, should we use historical financial data from 2015 to train a model and test it on 2017 data for predictions, this model might not have a good performance. The old model might not adapt well to the new

market environment as it has been trained and tailored on old market conditions. Although a machine learning approach can largely increase prediction accuracy of stock market, such models need to adapt themselves because the stock market is constantly changing. Wan and Banta (2006) propose the parameter incremental learning (PIL) method for neural networks; the main idea is that the learning algorithm should not only adapt to the newly presented input-output training pattern by adjusting parameters but also preserve the prior results. Inspired from this, we propose a method called *Walkthrough Training* in machine learning for our task. This approach is designed to retrain the original machine learning model itself when it 'appears' to no longer be valid. We consider two different Walkthrough training methods.

(i). *Walkthrough with stable retrain frequency.* Considering different trading cycles based on the data obtained from the API, we retrain our model at fixed time intervals. The length of the interval depends on our trading strategy and accuracy from data we obtained. This way of retraining helps the model to adjust to the newly acquired features and retain the knowledge gained from the original training.

(ii). *Walkthrough with dynamic retrain frequency.* We use Maximum Accuracy Drawdown (MAD), which is the maximum observed accuracy loss from a peak to a trough before a new peak is attained, as a condition of dynamic retraining. The idea is that stable retraining is not suitable for every condition in retraining model. More specifically, if the old model is aimed for long-term prediction, stable retraining will lead to waste of computing resources and overfitting problem (the model fits the data too well and leads to low prediction accuracy on unseen data).

During the process of prediction based on this method, we monitor accuracy of prediction over time. In the following formula, 'Min Accuracy Value' and 'Max Accuracy Value' identify the highest and lowest prediction accuracy, respectively. All parameters in the formula are in interval between last retraining time and current calculation time. After calculation, 'Modified MAD' is considered as hyper-parameter in the whole prediction model to optimize the retraining time.

$$\text{Modified MAD} = \frac{\text{Max Accuracy Value} - \text{Min Accuracy Value}}{\text{Min accuracy Value}}.$$

The modified MAD is a measure of accuracy loss that looks for greatest effective period of model. When modified MAD is over 15%, we consider the original machine learning model to be no longer applicable for latest market data. In such a case, we use historical data up to the point when the MAD is measured as training data to retrain original machine learning model. This process will be used throughout the whole time series prediction.

3.4. Research questions

We investigate four specific research questions (RQs, for short) in our general context of interest, price predictions through a machine learning model within the cryptocurrency markets.

RQ1: *How well does a universal machine learning model perform?*

Sirignano and Cont (2019) found that a universal machine learning model would predict well the price formation in relation to stock market. We ask this question to understand if a similar conclusion can be drawn for more emergent, less mature and more volatile cryptocurrency market.

RQ2: *How many successive data points should we use to train machine learning models?*

The sequential nature of time series naturally puts forward the question of optimizing the number of subsequent data points (i.e. time steps) used to train the deep network. Does it make sense to use more than one data point at a time? If so, how many time steps should be used?

RQ3: *How well do machine learning models work on live data?*

A good offline prediction based on machine learning may fail to perform well on live data, due to evolving patterns in a highly volatile environment like ours. Is there an accuracy decay on live data? If yes, would Walkthrough training methods help address the issue? Moreover, we want to understand if lean and fast architectures can perform well with tick online data.

RQ4: *What is the best Walkthrough method in the context of multi-label prediction?*

Making profit on tick data predictions might be too hard for a number of reasons. First, the execution time of the order might make the prediction on the next tick obsolete. Second, in the context of multi-label predictions, there might be very few data points in the sensitive intervals which would make transactions potentially more profitable than transaction costs. We therefore wish to determine the best Walkthrough method when we use minute-level data for the task of multi-label classification.

Our research questions are novel for a number of reasons. In RQ1, we analyze the effects of universality in cryptocurrency markets, which is an extension of Sirignano and Cont (2019). Given that the asset classes considered are rather different, it is interesting to study whether a sort of transfer learning translates across different markets. Similarly, whilst RQ2 has been studied by others, few researchers considered the problem for cryptocurrency prediction models. As for RQ3, we are not aware of any study in which the proposed models are tested on live data; this requires a balance between model complexity and performance. In RQ4, we test the performances of a brand new method in re-training the machine learning model. Ultimately, the findings from the questions above will help a cryptocurrency trader to design a better model and ultimately devise a more profitable trading strategy (i.e. the decision maker in the system of Figure 2).

3.5. Results and analysis

We organize the discussion of our results according to the research questions of interest. The answer to each question informs the design used to address the challenges of the subsequent questions. In this sense, we use an incremental approach to find our results.

3.5.1. How well does a universal machine learning model perform?

We begin by examining RQ1, through training product specific networks of Figure 3 in order to establish the baseline for comparison. For each product (i.e. currency pair), five neural networks with the same architecture are initialized. Five training sets are then created by extracting the first 10%, 20%, 50%, 70%, and 85% from the total data of the product. The neural networks are trained and tested with each data split. For example, a product-specific, such as BCH-USD, neural network is trained with the first 10% of the total data using only one time step; the rest of the data are then used to evaluate the performance of the neural network. Subsequently, another neural network is trained and tested with a different amount of data and so on.

The purpose of using this training approach is to evaluate the importance of the amount of data used. The high-frequency markets are often considered extremely noisy and full of unpredictability. If neural networks for the same product showed no performance gain with increasing amount of training data, then it may actually be the case that the majority of the data is noise. In these circumstances, a stochastic model might be a better option than a data-driven model, because a simpler model generally tends to be less overfitting compared to a complex model under noisy environment.

From the result in Table 6, the currency pairs with very little samples, such as BCH-EUR, BTC-GBP, ETH-EUR, and BTC-EUR, show a decreasing performance after using training data with a size greater than 50% (shown as Figure 5). The decrease in the performance could be a direct result of the lack of testing cases. For other currency pairs, the currency-pair-specified neural network models show a general rise in accuracy when increasing the size of the training data (Figure 6), which suggests that there might be some recognizable patterns in the data. The box plots (Figure 7) show the comparison of currency pairs with and without improvement. The result above suggests that, at least for our architecture, the neural network is able to learn the hidden pattern from within a dataset when given a sufficient amount of data for most of the currency pairs.

We are now ready to test the findings of Sirignano and Cont (2019) about the existence of a universal predictive model in the context of cryptocurrencies. We are interested to see whether a universal predictive model for all available currency pairs can outperform the product-specific ones introduced above. Table 7 displays the performance of different models using F1-score as performance measure. We selected F1 score as an indicator of measuring accuracy. F1 score is defined as

$$2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}},$$

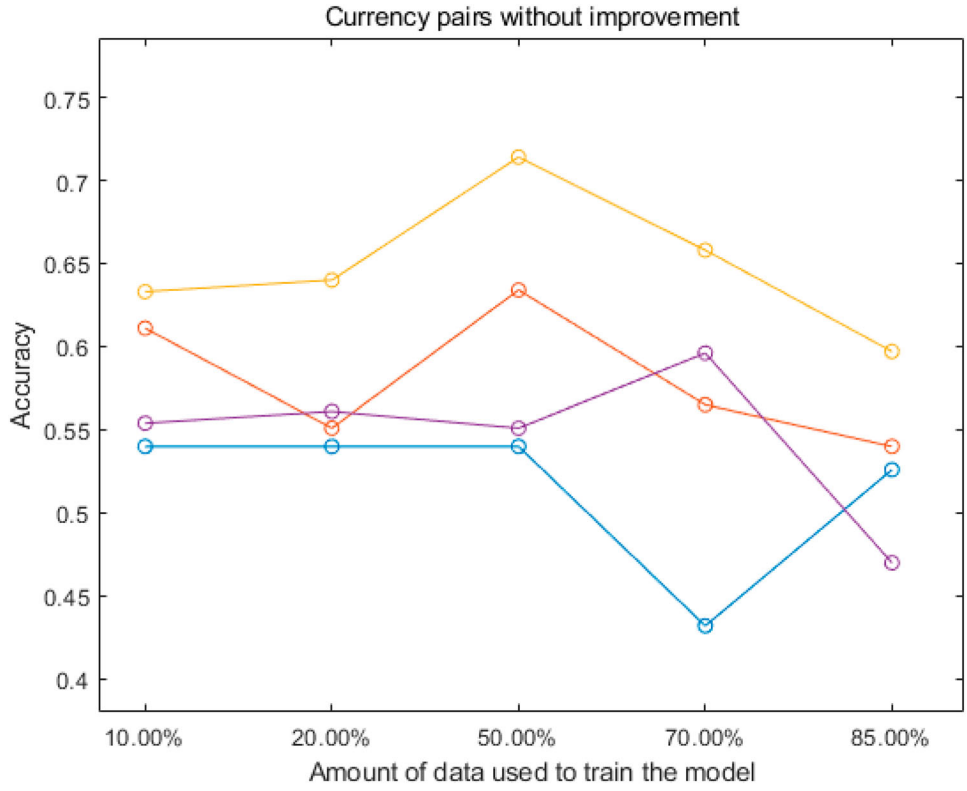

Figure 5. Currency pairs without improvement.

Table 6. Out-of-sample accuracy with respect to training sample sizes.

	Sample size used in training				
Currency pair	10%	20%	50%	70%	85%
BCH-USD	0.619	0.664	0.674	0.699	0.662
BTC-EUR	0.554	0.561	0.551	0.596	0.470
BTC-GBP	0.611	0.551	0.634	0.565	0.540
BTC-USD	0.702	0.789	0.797	0.825	0.814
ETH-BTC	0.788	0.825	0.839	0.775	0.743
ETH-EUR	0.633	0.640	0.714	0.658	0.597
ETH-USD	0.599	0.608	0.555	0.703	0.736
LTC-BTC	0.579	0.687	0.738	0.751	0.730
LTC-EUR	0.505	0.503	0.586	0.602	0.672
LTC-USD	0.574	0.620	0.767	0.787	0.814
BCH-EUR	0.540	0.540	0.540	0.432	0.526

where Precision is the fraction of relevant instances among the retrieved instances (i.e. the ratio between True Positives and the sum of true and false positives) and Recall is the fraction of the total amount of relevant instances that were actually retrieved (that is, the ratio between true Positives and the sum of true positives and false negatives). F1 score is an important evaluation measure when we are not familiar with the target class distribution. The label 'AVG' represents the mean performance of all the individual currency models. The label 'Universal' represents using joint data (merging all models as a new model). We know from the analysis above (Figures 5 and 6) that for some currency pairs, the current neural network architecture is not performing very well. Therefore, for more precise and targeted analysis, those currency pairs are excluded from the original dataset, and a new dataset is generated without them. The label of 'Selected' represents the mean performance

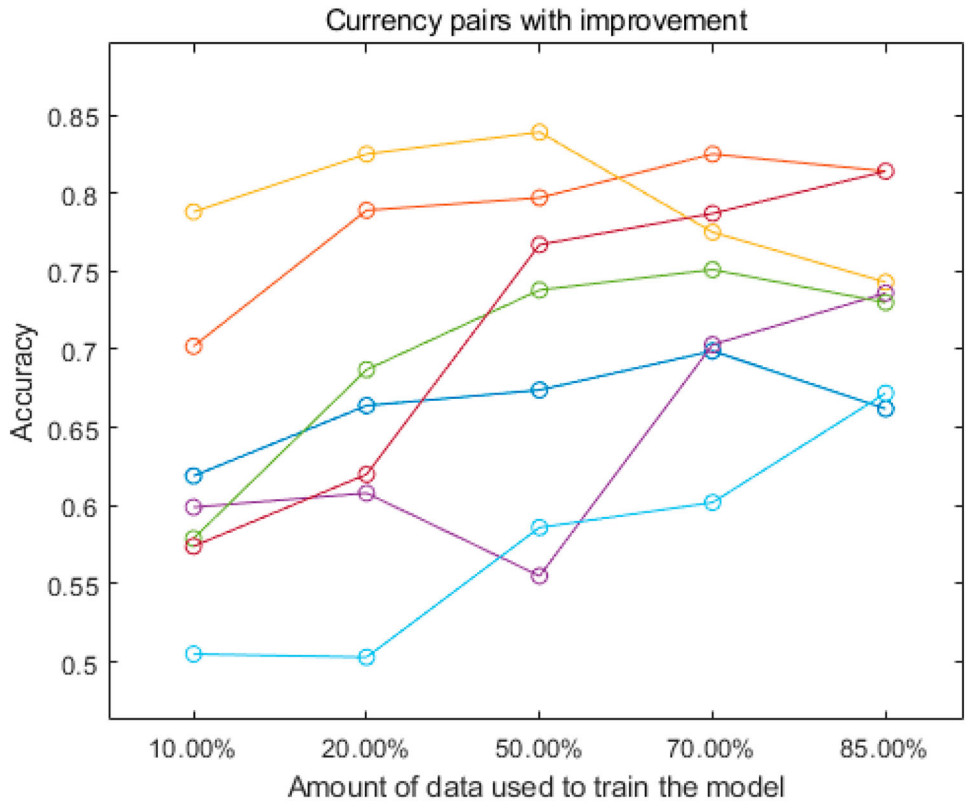

Figure 6. Currency pairs with improvement.

Table 7. Models' performance with different sample sizes used in training.

	Model F1-score on testing set			
Sample size	AVG	Selected	Universal	Universal selected
10%	0.58626	0.59067	0.64580	0.66039
20%	0.60633	0.64687	0.66634	0.68183
50%	0.66004	0.69423	0.68740	0.69205
70%	0.65757	0.73317	0.70973	0.72945
85%	0.61961	0.73457	0.71476	0.74087

of all models excluding those pairs, namely, BCH-EUR, BTC-GBP, ETH-EUR, and BTC-EUR. The 'Universal selected' neural network is trained with the 'selected' approach but with joined data across all available products.

We can see that the universal model slightly outperforms the mean of product-specific models, for each size of the training set, by an average of 5.88% in terms of F1-score. Similarly, the universal with selected currency pairs outperforms the selected product-specific model by an average of 7.50%. In general, both of the universal models achieved higher F1-score than the product-specific ones. Therefore, we can conclude that the universal model has better performance than the currency-pair specific model. The performance gain in the universal model and the universal model with selected currency pairs may be explained with the following rationale. First, there are some universal features on the limit order book which could be observed by the LSTM neural network for most of the currency pairs on the exchange. Second, the increased amount of the training data helps the network to generalize better, since 10% of joined data is much larger than 10% of one currency pair data. It also means that the LSTM model can learn the pattern from the data of multiple currency pairs having the same time horizon, then apply the pattern to another currency pair. To test whether this difference is statistically significant, we ran

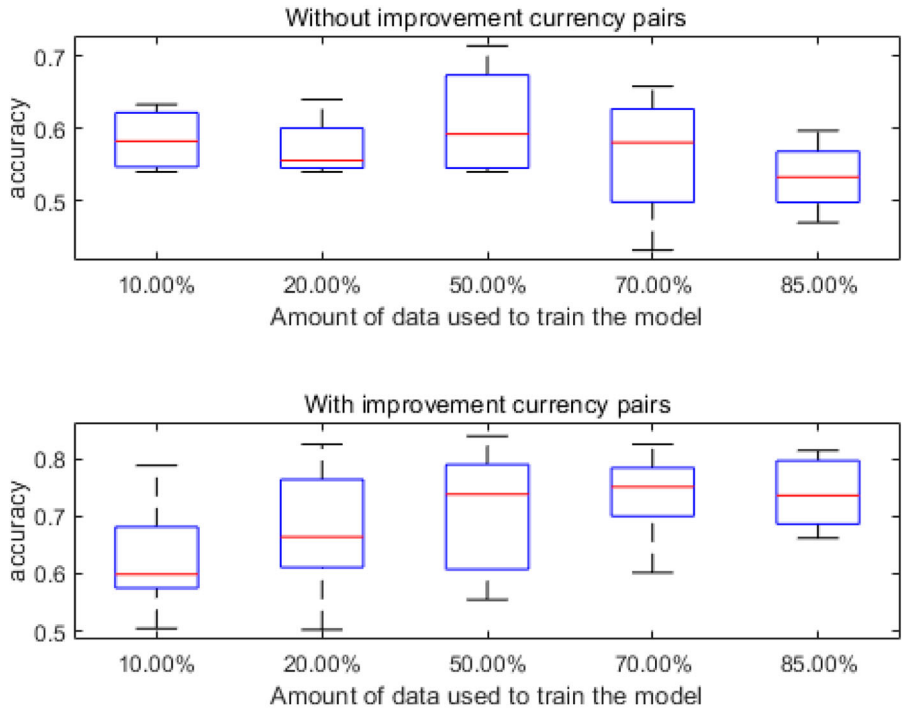

Figure 7. Box plots of currency pair with and without improvement.

the t-test and Wilcoxon test between the performance of product-specific and universal models (using accuracy as criteria). The t-test has the result that statistic is -53.885 and p-value is 3.0302e-22 ($\lll 0.0000$). The Wilcoxon test shows a p-value of 8.5745e-05 ($\lll 0.0000$). From both tests, we can conclude that the product-specific and universal models are statistically different.

We reach the following conclusion from this section. *The answer to RQ1* is that the universal model has better performance than the currency-pair specific model for all the available currency pairs in (the chosen) cryptocurrency market.

3.5.2. How many successive data points should we use to train machine learning models?

In this section, we examine RQ2. Informed by our findings in relation to RQ1, we next fix the training set size to 70% of the total sample size and focus our attention to the universal and universal selected neural networks. To investigate RQ2, we train both networks with 70% of the total data using increasing time steps of 1, 3, 5, 7, 10, 20, 40. For example, the 3-time-steps input contains the feature vector of the current tick F_t, feature vector of the previous tick F_{t-1}, and feature vector of two ticks prior F_{t-2}. This approach aims to discover whether there are any observable patterns related to the sequence of data and how persistent it is.

The results are shown in Table 8. To make sense of them, we fit the data points in a linear regression model, with ordinary least squares, and obtain the coefficients in Table 9, where Const represents the intercept and X the slope.

As depicted in Figure 8, the slopes of the linear equations are very close to zero, and are negative. This result suggests that increasing the time steps of the training data does not have a significant effect on the performance of the model. On the contrary, increasing the time steps too much may also have a negative impact to the model's performance. We here stress that Tran et al. (2018) studied this problem by applying a temporal attention-augmented bilinear network and testing using three types of movement (decrease, stationary and increase). The prediction horizon considered is 10, 20, and 50. Their method gives a comparison among different machine learning methods, whilst we here focus on our particular universal model.

Table 8. Performance of the models for different time steps using F1-score.

Time steps	Universal	Universal selected
1	0.7097	0.7294
3	0.7260	0.7442
5	0.7200	0.7419
7	0.7086	0.7369
10	0.7146	0.7389
20	0.7114	0.7421
40	0.7131	0.7273

Table 9. Coefficients of model F1-score in OLS regression.

	Universal	Universal selected
Const	0.7162	0.7400
X	−0.0001	−0.0002

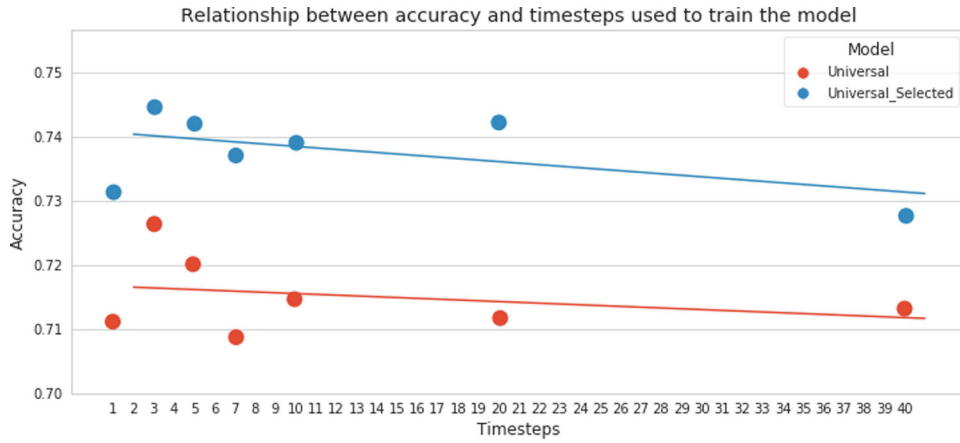

Figure 8. Relationship between accuracy and number of time steps used in training.

The answer to RQ2 is that one time step/data point is the best choice in our context. Choosing one time step carries some further advantages; one step, in fact, opens the possibility of using different machine learning algorithms since most of them are not designed to handle sequential input.

3.5.3. How well do machine learning models work on live data?

We here examine RQ3. In this section, the challenges and performances of using our predictive models on live data are discussed.

The horizontal line in Figure 9 is the baseline of the performance, and the black line is the performance of the predictive model on live data. This figure shows that the performance of the universal model slowly decays to almost random guessing over the period of interest. This behavior could be caused by some non-stationary features of the limit order book, which means that the hidden pattern captured by the universal model is no longer applicable to the new data.

To resolve this problem, an autoencoder is used (Figure 10). The characteristic of the autoencoder is that the input layer and the output layer usually have the same number of neurons, and the hidden layers of the autoencoder must have a lower number of neurons compared to the input and output layers. The reason for using such an architecture is that the reduced number of neurons in the hidden layers can form a bottleneck in the neural network. Thus the autoencoder cannot learn by simply remembering the input only. This architecture, in fact, forces the autoencoder to compress the input data and then decompress the data before outputting it.

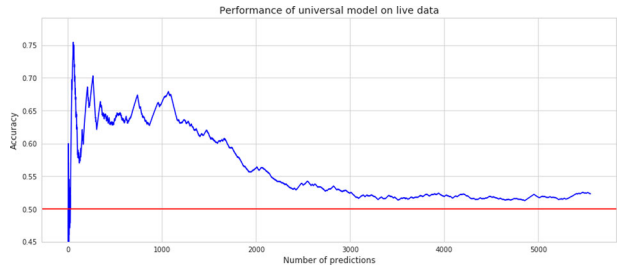

Figure 9. Performance decay on the live data.

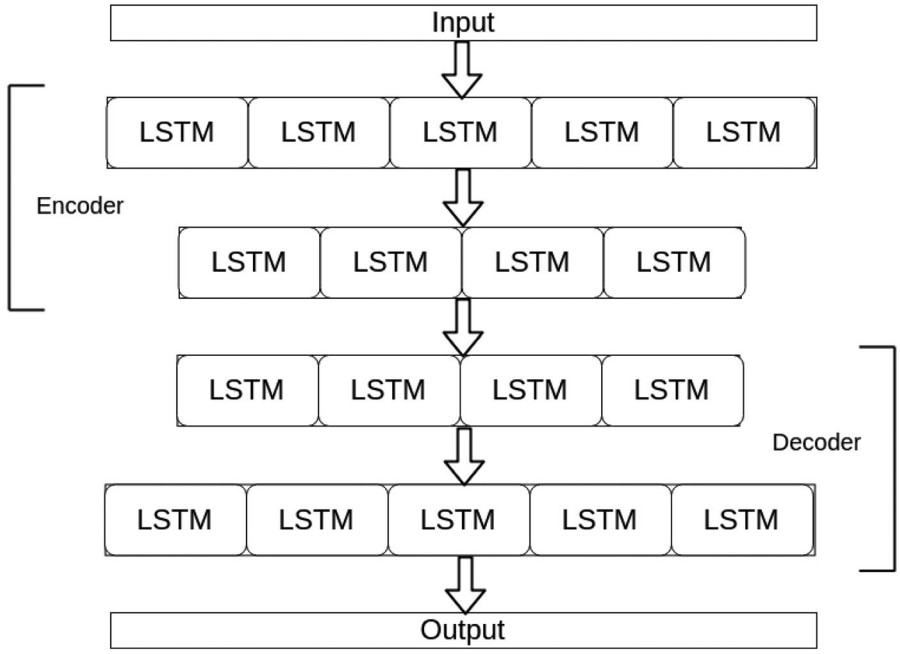

Figure 10. Architecture of the autoencoder.

Therefore, the autoencoder can learn from the input structure. The trained autoencoder performs two tasks. The first one is to remove noise; the trained autoencoder can suppress abnormal features by reconstructing the input data. This process usually removes abnormal spikes in a feature. The second one is to map the new data into a more familiar space for the LSTM model.

Figure 11 shows the prediction of the LSTM with an autoencoder by using live data of BTC-USD from 2018-08-01 15:10:43 to 2018-08-02 08:33:50. Bitcoin has a great dominance and the BTC-USD is also the most traded product on the market. The performance decays slower with the autoencoder than the original LSTM model. Figure 12 is the distribution of the predictions made by the universal model with autoencoder and the aggregated real-time target; each point of the aggregated real-time target is equal to the mean of upticks and downticks for every 20 samples. The darker line depicts the ratio of downticks given by the predictive model, and the lighter line is the ratio of downticks given by real-time target. From the distribution of prediction and real-time target, we can observe that the autoencoder is slightly biased to the downtrend market. This explains the gradual decrease in the accuracy under the uptrend market after the 3000 predictions mark (cf. Figure 11) because small errors accumulate over time and eventually affect the overall accuracy. In other words, the biased training data could

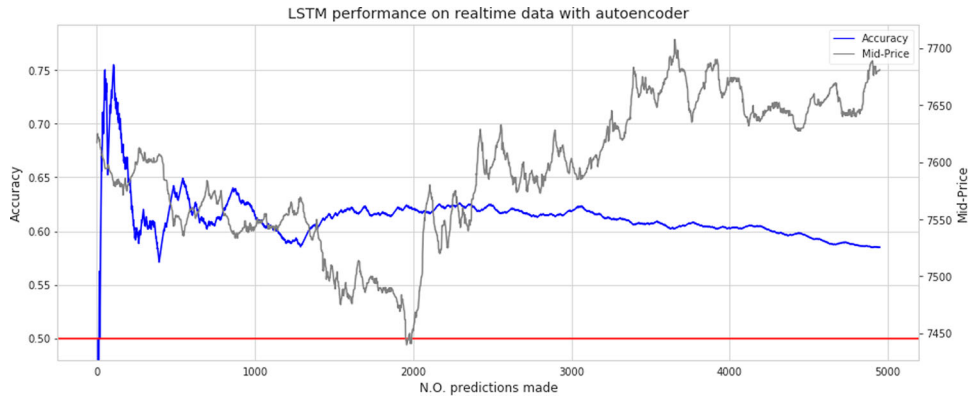

Figure 11. Performance of the universal model with autoencoder.

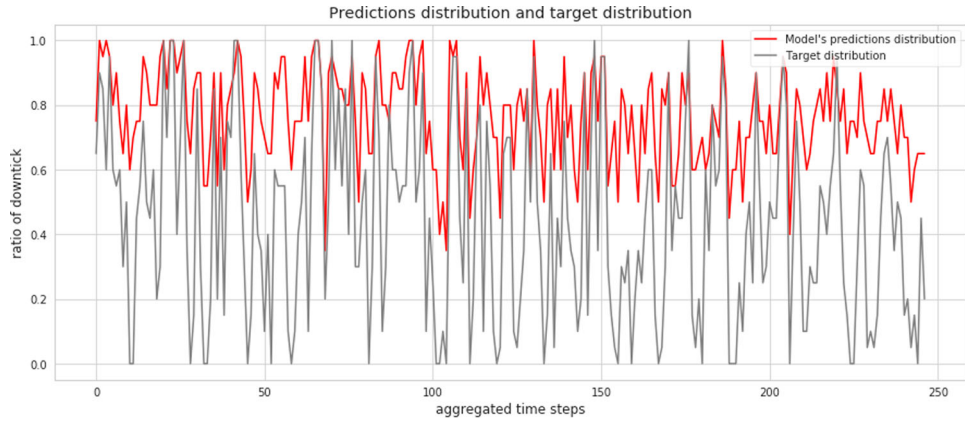

Figure 12. Predictions distribution and real-time target distribution.

cause a biased model. For example, the training data used to train the model could be experiencing a bearish market so that the model is more sensitive to the downtrends.

An intuitive way to adjust the bias of the model is walkthrough training, i.e. retraining the model with recent data. This way, the model can learn from the most recent data and integrate it with the knowledge learnt from the original data. We implement a walkthrough with stable retrain time as follows. First, a queue buffer is set up to collect features from the live data. After every 196 predictions made by the model, the model retrains by the newly collected features in the buffer.

To test the effectiveness of this modification, we use live data of BTC-USD from 2018-08-08 14:31:54 to 2018-08-09 09:01:13. The results are plotted in Figure 13. We observe that before the first retraining, the model lacks the predictive power on live data. It starts with an accuracy of less than 50%, which is worse than random guessing. After the first few instances of retrain, however, the model improves accuracy from 58% to 78%, to finally stabilize around 76%. Moreover, the distribution of the predictions of the model shows a similar shape to the real-time target distribution, and no apparent bias can be observed, cf. Figure 14.

We have ran an augmented Dickey–Fuller test (ADF) testing the null hypothesis that a unit root is present in the time series comprised of the live data samples. The results showed that the p-value of BTC-USD data from 2018-08-11 12:09 to 2018-08-16 23:59 is 0.781425 and the t-statistics value is -0.919673. We have a value of -3.431 when the confidence level is 1 %, value of -2.862 when the confidence level is 5%, value of -2.567 when the confidence level is 10%. We can see that the value is larger than the critical values in 1%, meaning that we can accept the null hypothesis and in turn conclude that the time series is non-stationary.

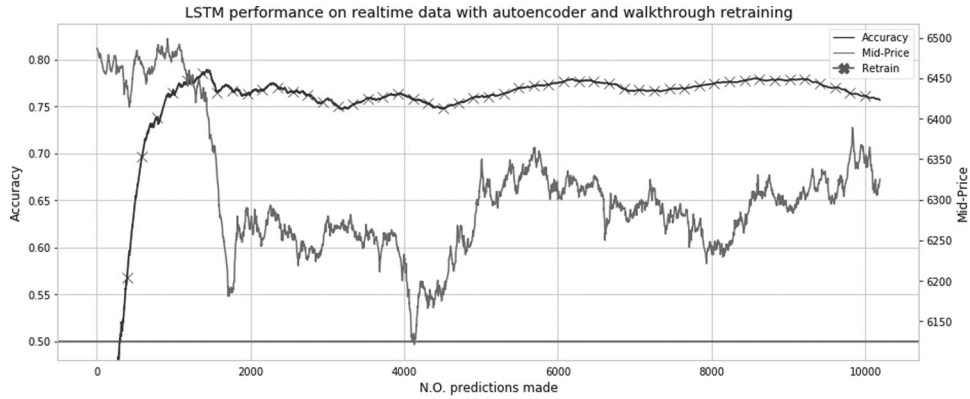

Figure 13. Performance of the universal model with autoencoder.

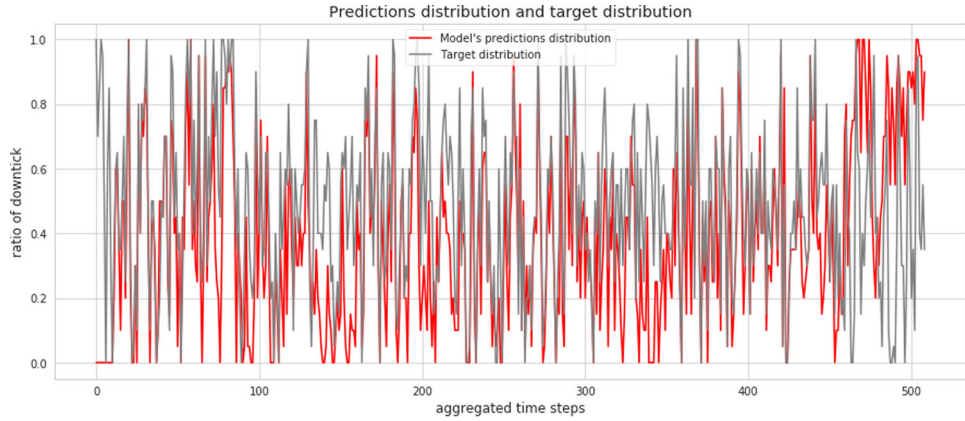

Figure 14. Predictions distribution and real-time target distribution.

A further improvement of the model to work on live data is needed to improve the execution speed and reduce the chance of overfitting. This is achieved by reducing the dimension of the input data. The intermediate output of the autoencoder, which is the output of the encoder part, is used instead of using the original data. Because of the architecture of the autoencoder, the hidden layer contains fewer neurons than the output layer. Although the hidden layer contains fewer neurons, it preserves all the essential information of the input data. By using this approach, the universal model can use fewer neurons to capture the information that is needed to make predictions. Therefore, the neural network has less freedom to be overfitted, and the reduction of the size of the neural network also improves the execution speed. Our architecture uses the intermediate encoder output as the input for the LSTM model, cf. Figure 15. The advantage of using the autoencoder instead of using Principal Component Analysis (PCA) directly is that autoencoder can map the 3D sequential data (sample size, time steps, features) into a vector. This process helps to capture the information from the sequence which could not be done by PCA only (PCA can only deal with 2D data).

The answer to RQ3 is summarized in Table 10, where we display the performance metrics of the predictive model with a reduced architecture on the same live data of BTC-USD from 2018-08-08 14:31:54 to 2018-08-09 09:01:13. In the table, 'autoencoder as denoiser' refers to architecture in Figure 10 (including encoder, decoder, PCA, universal model and output) whilst 'autoencoder as reducer' refers to the architecture in Figure 15 (including encoder, PCA, universal model and output). The difference of the precision score and the accuracy score is much lower than the original model, which suggests that there is less overfitting. Interestingly, the reduced model has a 2.43% increase in the accuracy score. Compared to research of Easley et al. (2019), our method gives an

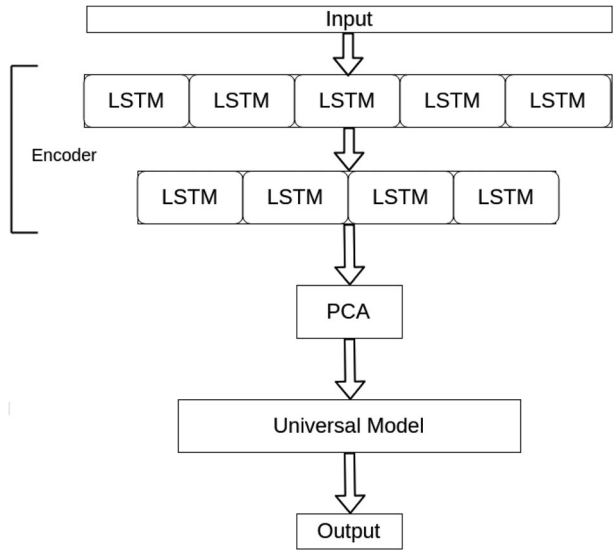

Figure 15. LSTM model with autoencoder.

Table 10. Classification report.

Classification report of autoencoder as denoiser			
Class	Precision	Accuracy	F1-score
↑	0.72	0.86	0.78
↓	0.81	0.65	0.72
avg	0.77	0.76	0.75
Classification report of autoencoder as reducer			
	Precision	Accuracy	F1-score
↑	0.79	0.78	0.79
↓	0.77	0.78	0.77
avg / total	0.78	0.78	0.78

optimization in LSTM structure; in particular, the machine learning model we designed is more suitable for live data prediction. Furthermore, we optimize the process of retraining, which can also be applied in multi-class prediction.

We also collected and displayed the performance metrics in the predictive model with the latest model on the live data of BTC-USD in OkEX. We separate the live dataset into to pieces: from 2018-08-11 12:09 to 2018-08-16 23:59 and from 2018-08-24 12:07 to 2018-08-29 23:59 (cf. Table 11). The results are slightly worse than the ones in Table 10 considering Precision and F1-score, especially in down classification. This might be due to a very unstable spread in those periods; this affects the hyper-parameters of our predictive model. A better and more stable live model against the cryptocurrency fluctuations is one question left open by our work.

3.5.4. What is the best walkthrough method in the context of multi-label prediction?

Lastly, we discuss RQ4. In this section, we concentrate on multi-label prediction using the four classes identified in Table 5. We use the last model in RQ3 but we use multi-label as target classification and walkthrough method as a research variable. We use 1-min live data in 2018 for 1 month-window and the time interval of data is randomly selected.

Figure 16 gives an overview of the comparison between the three different walkthrough methods we tested.

For the first method, we train the machine learning model statically without walkthrough, which means we will not retrain or update the model when the accuracy decays. We can observe how the accuracy drops

Table 11. Classification report 2.

Classification report of 2018/8/11-2018/8/16 (P1)			
Class	Precision	Accuracy	F1-score
↑	0.49	0.79	0.64
↓	0.55	0.65	0.58
avg	0.52	0.76	0.61
Classification report of 2018/8/24-2018/8/29 (P2)			
	Precision	Accuracy	F1-score
↑	0.51	0.81	0.67
↓	0.55	0.71	0.59
avg / total	0.53	0.76	0.63

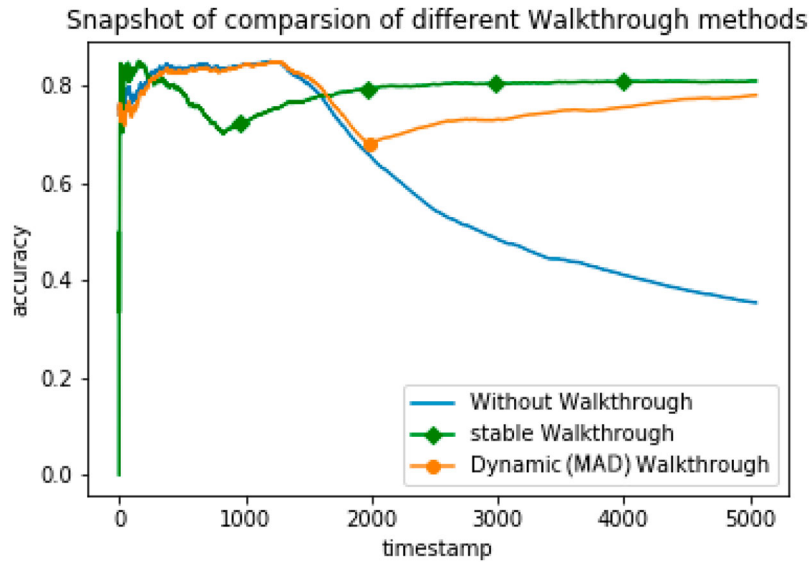

Figure 16. Snapshot of comparison between different walkthrough methods.

significantly after roughly 1500 predictions and reaches a value of less than 40%, almost as low as random-guessing (25%), in the end.

When we use stable walkthrough to train our model, we will retrain at regular time periods. Our tests are based on trading period or a financial trading cycle of 5 days. On our data, this leads to four retrains (identified by rectangular points on the accuracy line). The first retrain point has obvious effects in improving accuracy and it starts to go up slightly before retraining. After four instances of retrain, the accuracy stabilizes around 80%.

When we use MAD-Dynamic Walkthrough method, we retrain the original model when the accuracy drops by more than 15%. In our test, there is only one such instance (circle point at around 2000 mark). The accuracy has apparent growth after this model adjustment.

We also perform the experiment for 20 times to compare the different walkthrough methods in order to have a stronger statistical guarantee; results are shown in Figure 17. (Considering that repeated experiments cost significant computation power and time, we repeat the experiment 20 times to gather the results.) The results are in line with those discussed above, i.e. stable walkthrough is better than the other two methods. The results also show that, for dynamic (MAD) walkthrough method, only 2/3 retraining points occur in most experiments (90%) while 2 experiments require 5 retraining points. As a comparison, for stable walkthrough method, all the experiments need 4 retraining points. Therefore, when retrain time is a factor to consider, dynamic (MAD) walkthrough method is better because it needs less retraining in most cases.

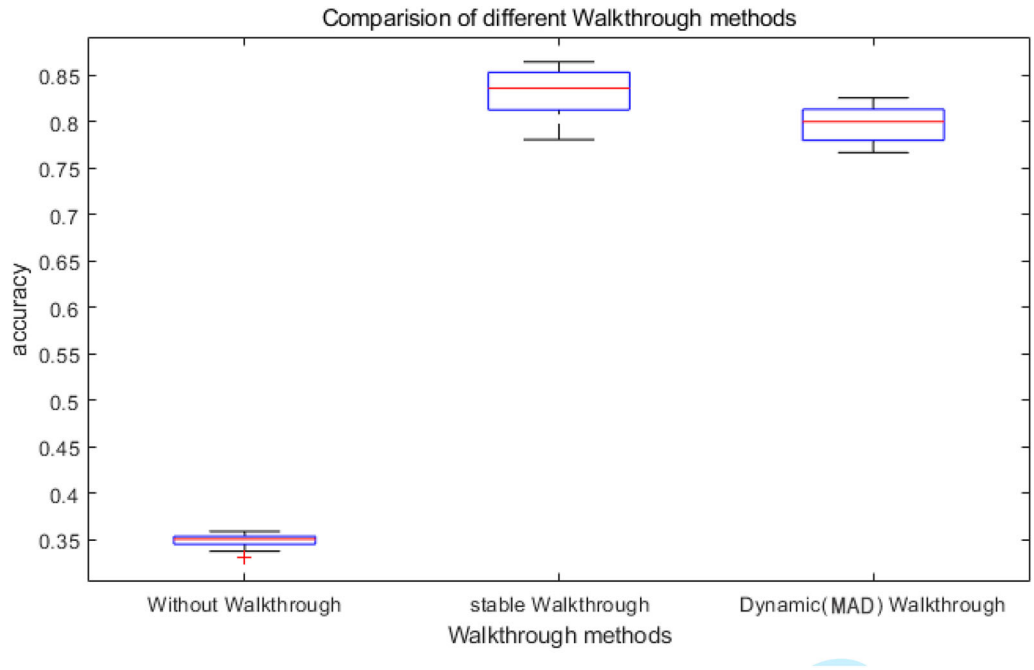

Figure 17. Comparison of different walkthrough methods.

The answer to RQ4 is multifold. First, in multi-label prediction with machine learning model, walkthrough training significantly improves the prediction accuracy. The reason is that, as discussed above, machine learning prediction models need to update itself. When the model is not fit for the new market conditions, then it must be updated to achieve accurate results. Second, stable walkthrough method is better than MAD-dynamic walkthrough method, unless retrain time is important.

4. Validity of findings

The model is based on trading system; we select historical data and collect live data for a long time span. The selection of data has no bias because historical trading contains all available transaction data and available currency pairs in cryptocurrency market. Moreover, the experiments are not affected by bull or bear market, policy impact and other factors. The experiments use an extensive data selection, including bull-market condition, bear-market condition, high-transaction-volume condition, low-transaction-volume condition, etc.

Quality of data is another important factor to discuss. As the data is collected live from Coinbase Pro, poor connection might affect the data (e.g. missing values). To mitigate this risk, we have compared the data collected from Coinbase Pro with other third party service providers to make sure the experiment have not been affected by inappropriate financial data.

5. Conclusion

This paper analyzes a data-driven approach to predict mid-price movements in cryptocurrency markets, and covered a number of research questions en route regarding parameter settings, design of neural networks and universality of the models. The main finding of our work is the successful combination of an autoencoder and a walkthrough retraining method to overcome the decay in predictive power on live data due to non-stationary features on the order book. Our results show that our model has achieved good performance, quantified in a consistent F1-score of around 78%. By comparing different retraining methods (we call that Walkthrough), we

found some tradeoffs between fixed and dynamic retraining. Prediction in high-frequency cryptocurrency markets is a challenging task because the environment contains noisy information and is highly unpredictable. We believe that our results can inform the design of higher level trading strategies and our networks architecture can be used as a feature to another estimator. One interesting direction for future research might be a more extensive treatment of how time persistent the performances of the model are, similarly to Sirignano and Cont (2019).

However, we must also realize that machine learning has obvious limitations, which must be overcome to reach artificial general intelligence (Marcus 2018). Marcus pointed out that machine learning models are data-hungry and the knowledge gathered by deep learning systems is primarily concerned with correlations between features, rather than abstractions like quantified statements. These characteristics have negative impacts on machine learning in financial prediction. Moreover, we know that when applying out-of-sample tests in non-stationary data, the prediction made are not entirely 'honest' (Inoue and Kilian 2005). The corresponding forecast error may underestimate the magnitude of the error that will arise when the model is used to forecast the future, as the data may overfit the squared error and the model and inadvertently fit some 'noise' during the estimation. To deal with these aspects, our model uses retraining and is tested on different live time series (and perform consistently well).

Disclosure statement

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

References

Abergel, F., and A. Jedidi. 2015. "Long-Time Behavior of A Hawkes Process–Based Limit Order Book." *SIAM Journal on Financial Mathematics* 6 (1): 1026–1043.

Adam, P. 2015. "Lstm Implementation Explained." https://apaszke.github.io/lstm-explained.html.

Ahamad, S., M. Nair, and B. Varghese. 2013. "A Survey on Crypto Currencies." In *Proceedings of the 4th International Conference on Advances in Computer Science, AETACS, NCR, India'*, 42–48. Citeseer.

Altay, E., and M. H. Satman. 2005. "Stock Market Forecasting: Artificial Neural Network and Linear Regression Comparison in An Emerging Market." *Journal of Financial Management & Analysis* 18 (2): 18.

Barbon, A. 2019. "Focusing at High Frequency: An Attention-Based Neural Network for Limit Order Books."

Biais, B., P. Hillion, and C. S. Spatt. 1995. "An Empirical Analysis of the Limit Order Book and the Order Flow in the Paris Bourse." *Journal of Finance* 50 (5): 1655–1689.

Brandvold, M., P. Molnár, K. Vagstad, and O. C. A. Valstad. 2015. "Price Discovery on Bitcoin Exchanges." *Journal of International Financial Markets, Institutions and Money* 36: 18–35.

Brooks, C., A. G. F. Hoepner, D. McMillan, A. Vivian, and C. W. Simen. 2019. "Financial Data Science: The Birth of a New Financial Research Paradigm Complementing Econometrics?" *The European Journal of Finance* 25 (17): 1627–1636. doi:10.1080/1351847X.2019.1662822.

Christopher, O. 2015. "Understanding Istm Networks – Colah's Blog." http://colah.github.io/posts/2015-08-Understanding-LSTMs.

Dixon, M. 2018. "Sequence Classification of the Limit Order Book Using Recurrent Neural Networks." *Journal of Computational Science* 24: 277–286.

Easley, D., M. L. de Prado, M. O'Hara, and Z. Zhang. 2019. "Microstructure in the Machine Age." *Available at SSRN 3345183*.

En.wikipedia.org. 2018. "List of Cryptocurrencies." https://en.wikipedia.org/wiki/List_of_cryptocurrencies.

Fischer, T., and C. Krauss. 2018. "Deep Learning with Long Short-Term Memory Networks for Financial Market Predictions." *European Journal of Operational Research* 270 (2): 654–669.

Fletcher, T. 2012. Machine learning for financial market prediction. PhD diss., University College London.

GDAX. 2018. "Gdax Api Reference." https://docs.gdax.com/#protocol-overview.

Gu, S., B. Kelly, and D. Xiu. 2020. "Empirical Asset Pricing Via Machine Learning." *The Review of Financial Studies* 33 (5): 2223–2273.

Güresen, E., G. Kayakutlu, and T. U. Daim. 2011. "Using Artificial Neural Network Models in Stock Market Index Prediction." *Expert Systems with Applications* 38 (8): 10389–10397.

Hochreiter, S., and J. Schmidhuber. 1997. "Long Short-Term Memory." *Neural Computation* 9 (8): 1735–1780.

Huang, W., Y. Nakamori, and S.-Y. Wang. 2005. "Forecasting Stock Market Movement Direction with Support Vector Machine." *Computers and Operations Research* 32 (10): 2513–2522.

Inoue, A., and L Kilian. 2005. "In-Sample Or Out-of-Sample Tests of Predictability: Which One Should We Use?" *Econometric Reviews* 23 (4): 371–402.

Kelly, F., and E. Yudovina. 2017. "A Markov Model of a Limit Order Book: Thresholds, Recurrence, and Trading Strategies." *Mathematics of Operations Research* 43 (1): 181–203.

Kercheval, A., and Y. Zhang. 2015. "Modelling High-Frequency Limit Order Book Dynamics with Support Vector Machines." *Quantitative Finance* 15 (8): 1315–1329.
Mäkinen, Y., J. Kanniainen, M. Gabbouj, and A. Iosifidis. 2019. "Forecasting Jump Arrivals in Stock Prices: New Attention-Based Network Architecture Using Limit Order Book Data." *Quantitative Finance* 19 (12): 2033–2050.
Marcus, G. 2018. "Deep Learning: A Critical Appraisal." *arXiv preprint arXiv:1801.00631*.
Nakamoto, S. 2008. *Bitcoin: A Peer-to-Peer Electronic Cash System*.
Nousi, P., A. Tsantekidis, N. Passalis, A. Ntakaris, J. Kanniainen, A. Tefas, M. Gabbouj, and A. Iosifidis. 2019. "Machine Learning for Forecasting Mid-Price Movements Using Limit Order Book Data." *Ieee Access* 7: 64722–64736.
Pro, C. 2018. "Global Charts — Coinmarketcap." https://support.pro.coinbase.com/customer/en/portal/articles/2945310-fees.
Silantyev, E. 2019. "Order Flow Analysis of Cryptocurrency Markets." *Digital Finance* 1 (1-4): 191–218.
Sirignano, J., and R. Cont. 2019. "Universal Features of Price Formation in Financial Markets: Perspectives From Deep Learning." *Quantitative Finance* 19 (9): 1449–1459.
Sundarapandian, V. 2009. *Probability, Statistics and Queuing Theory*. PHI Learning Pvt. Ltd, India.
Tieleman, T., and G. Hinton. 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.
Toke, I. M., and F. Pomponio. 2012. "Modelling Trades-through in a Limit Order Book Using Hawkes Processes." *Economics: The Open-Access, Open-Assessment E-Journal* 6 (2012-22): 1–23.
Tran, D. T., A. Iosifidis, J. Kanniainen, and M. Gabbouj. 2018. "Temporal Attention-Augmented Bilinear Network for Financial Time-Series Data Analysis." *IEEE Transactions on Neural Networks and Learning Systems* 30 (5): 1407–1418.
Tsantekidis, A., N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis. 2017. "Using Deep Learning to Detect Price Change Indications in Financial Markets." In *Proceedings of the 25th European Signal Processing Conference (EUSIPCO), IEEE, Nice, France,* Vol. 4.
Verstyuk, S. 2020. "Modeling Multivariate Time Series in Economics: From Auto-Regressions to Recurrent Neural Networks." *Available at SSRN 3589337*.
Wan, S., and L. E. Banta. 2006. "Parameter Incremental Learning Algorithm for Neural Networks." *IEEE Transactions on Neural Networks* 17 (6): 1424–1438.
Zhang, X.-D., A. Li, and R. Pan. 2016. "Stock Trend Prediction Based on a New Status Box Method and Adaboost Probabilistic Support Vector Machine." *Applied Soft Computing* 49: 385–398.