WALDATA : Wavelet transform with Adversarial Learning to Detect Abnormal Trading Activities

Khaled Safa, Ammar Belatreche, Salima Ouadfel, Richard Jiang

Abstract

The detection of abnormal stock market trading activities is a challenging task due to several factors. One of the primary challenges is the lack of labeled data. Anomalies in stock market data are rare and difficult to capture, making it challenging to create a labeled dataset for supervised learning approaches. Moreover, the stock market is characterized by intricate temporal correlations in non-stationary stock price data, making it challenging to identify patterns or anomalies using traditional statistical methods. To overcome these challenges, we introduce a novel approach, WALDATA, which is based on the combination of Generative Adversarial Networks (GANs) and Wavelet Transform (WT). Our method takes advantage of the power of WT to handle non-stationary time series and perform efficient feature extraction. In addition, we leverage the recent success of GANs in computer vision and use them to learn normal stock price behaviors. WALDATA encodes equity price data as scalogram images using the Continuous Wavelet Transform, which converts the time series data into a 2D image representation. We then train a Generator/Discriminator Adversarial Network to learn normal stock price behaviors. The trained discriminator serves as an anomaly detector, allowing us to detect abnormal trading activities in the stock market. Our method was tested on 1-level tick data obtained from the LOBSTER project and achieved an average AUC of 0.99 with low false alarm rates. This demonstrates the effectiveness of our method in detecting abnormal trading activities in the stock market.

Keywords: Price manipulation, anomaly detection, continuous wavelet transform (CWT), generative adversarial network (GAN), market abuse, abnormal trading activities

1. Introduction

Market manipulation involves artificially altering financial instrument prices in a way that deviates from normal market conditions, damaging investor confidence and reducing market transparency. This practice is prohibited by financial regulators globally, with a notable example being the complicity of groups of traders in raising stock prices on the New

Email addresses: khaled.safa@univ-constantine2.dz (Khaled Safa), ammar.belatreche@northumbria.ac.uk (Ammar Belatreche), salima.ouadfel@univ-constantine2.dz (Salima Ouadfel)

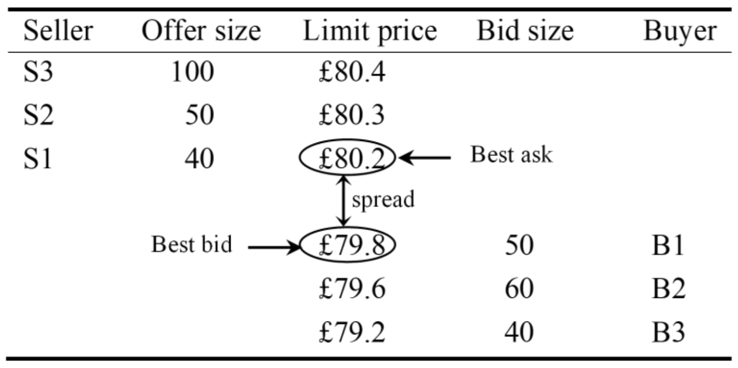
York market, which led to the establishment of the Securities of the Financial Markets act and the United States Securities and Exchange Commission (SEC) in 1934 Allen and Gale, 1992. Market manipulation can be divided into three categories: information-based, actionbased, and trade-based Allen and Gale, 1992. Information-based manipulation involves spreading false rumors or inside information to alter the market's perception of the stock's value. Action-based manipulation involves acquiring shares and announcing a takeover offer to drive up stock prices. Trade-based manipulation involves submitting manipulative trades to create artificial price movements. This form of manipulation is particularly challenging to detect and eradicate. The consequences of market manipulation can be far-reaching and harmful to investors, financial markets, and the economy as a whole. It is crucial for regulatory bodies to implement measures to prevent market manipulation and maintain the integrity of financial markets. Several academic papers have been published that use statistical and machine learning techniques to detect stock market manipulation. Early works in this area relied on statistical methods, such as time series analysis and statistical tests, to identify patterns and anomalies in trading data that may indicate manipulation. These methods are often based on the analysis of abnormal returns and trading volume and aim to identify instances of manipulation by detecting unusual patterns in these variables. However, despite their widespread use, these statistical methods have several limitations. They often rely on simple mathematical models and do not take into account complex relationships between variables, making them less accurate in detecting various instances of manipulation. Additionally, these methods are not well suited to handling large amounts of data, which is becoming increasingly important in the era of big data. This is where machine learning and deep learning techniques come in. Machine learning algorithms, such as decision trees, random forests, and support vector machines, are capable of handling large amounts of data and detecting complex relationships between variables, making them well suited to detecting stock market manipulation. Deep learning techniques, such as convolutional neural networks and recurrent neural networks, are even more advanced and are capable of handling even more complex data structures, making them even more effective at detecting stock market manipulation Cao et al., 2014; Sridhar et al., 2020; Uslu and Akal, 2022. Recently, wavelet analysis, as proposed by Mallat, 1989, has gained significant attention in the field of financial time series studies Gallegati, 2012; Lahmiri, 2014; Langi et al., 2012. This approach offers a unique method for analyzing time series data by decomposing it into different frequency bands. Wavelet analysis has proven to be particularly effective in the context of financial data, which is often non-stationary and exhibits complex patterns over time. One of the primary advantages of wavelet analysis is its ability to extract important features from financial time series data. By identifying different frequency components within the data, wavelet analysis enables researchers to isolate key patterns and trends that might otherwise be difficult to detect. This can be especially valuable in market manipulation detection, where accurate manipulation detection rely heavily on the ability to identify important patterns within the data. Moreover, wavelet analysis is also highly adaptable, making it a flexible tool for financial researchers. Its ability to handle non-stationary signals, for example, means that it can be used to analyze financial data that exhibits trends or seasonality. This makes it well-suited to a range of financial applications, from market analysis to risk management. In this paper, we introduce a new method, WALDATA, for detecting stock market manipulations. This approach combines wavelet analysis and deep Convolutional Generative Adversarial Network (DCGAN) to identify manipulation in ask/bid data based on normal market behavior. The process involves analyzing equity price data using Continuous Wavelet Transform (CWT), encoding it as RGB images, and training a DCGAN to learn the normal behavior of these images. The discriminator network is then used as a detector of anomalies. The proposed approach is validated on 1-level tick dataset of five stocks (Amazon, Apple, Google, Intel and Microsoft) obtained from the LOBSTER project, which covered five trading days, from the 11th to the 15th of June,2012. The obtained results show an average of 0.99 in terms of the area under the curve (AUC) and a very low false alarm rate. the key contributions of our work are as follows:

- we propose a new approach called WALDATA by combining the continuous wavelet transform with the power of generative adversarial network to solve anomaly detection problem. Our architecture is designed to detect unknown manipulation schemes so that we can improve their performance when some labelled dataset (normal and manipulative trade schemes) is available.
- To capture the temporal correlations of the normal behaviour of stock price data, we analyse the ask/bid price data on different frequency scales by using the Continuous Wavelet Transform.
- To take advantage of the deep convolutional neural network's ability to automatically learn relevant characteristics from images, the equity price data is transformed into RGB images.
- The discriminator network, consisting of a feature extractor and classifier, can be finetuned using labeled data to improve its performance in detecting manipulation schemes.

The reminder of this paper is organized as follow. Section 2 provides a background and a comprehensive overview of the related works in detecting market manipulation using machine learning techniques. Section 3 explains our proposed approach, WALDATA, in detail, including the data pre-processing and the steps of encoding equity price data as images and training the generative adversarial network. Section 4 presents the experimental results and evaluation of the performance of the proposed approach on the 1-level tick dataset obtained from the LOBSTER project. Section 5 concludes the paper and provides suggestions for future work in the field of market manipulation detection.

2. Background

In today's financial landscape, stock markets predominantly operate as order-driven markets, wherein participants contribute liquidity by submitting buy or sell orders through electronic trading systems. These systems utilize rules such as price priority and time precedence to match and execute these orders. A fundamental mechanism within electronic trading systems is the Limit Order Book (LOB), which displays all buy and sell orders and facilitates seamless trading. Market orders and limit orders are the most common order types, with limit orders executing at or below/above the specified limit price and market orders executing immediately at the current market price Figure1.



Seller	Offer size	Limit price	Bid size	Buyer
S3	100	£80.4		
S2	50	£80.3		
S1	40	£80.2		
		£79.8	50	B1
		£79.6	60	B2
		£79.2	40	B3

Best ask
spread
Best bid

Figure 1: Illustrative limit order book example. There are two sides, buyer side (also called bid side) and seller side (also called ask or offer side). Participants in the bid/ask side are providing liquidity with prices at which they are willing to buy/sell some quantities of the stock. The spread represents the differences between the best (lowest) ask price and the best (highest) bid price. A trade occurs when the price of the sell order and the buy order match.

The advent of technology has revolutionized financial instrument trading within the limit order book market, transitioning from manual human influence to algorithmic trading, including the rise of High-Frequency Trading (HFT). High-Frequency Trading is characterized by its remarkable speed, enabling trade orders to be placed and executed in mere nanoseconds, without long-term position holdings. However, this technological progress has also opened the door to various illegal trading strategies employed by manipulators. These manipulators employ deceptive tactics, disseminating false or misleading signals to market participants, distorting their perception of the true price of the traded instrument, all in pursuit of illicit profits. To effectively combat and detect such manipulative behaviors, it is essential to understand the specific strategies employed by manipulators. In the following subsection, we delve into the various manipulation strategies employed by these actors. By providing comprehensive definitions and insights into these strategies, we establish the groundwork for developing robust and accurate detection algorithms capable of identifying and mitigating instances of manipulation in the market.

2.1. Layering and spoofing strategies

In this illegal strategy, the deceptive trader sends a bona fide order at more profitable price on one side (ask/bid) and at the same time a single large non-bona fide order on the other side (bid/ask) without intention to be executed but to give a false impression of high buying/selling pressure at that price and drive the price up/down than there actually is. This non-bona fide order is cancelled either the order in question is executed or not. this strategy known as the terms “spoofing” and “layering” interchangeably in some regulators, while others use “layering” to describe placing multiple non-bona fide orders at progressively higher price levels, and “spoofing” to describe placing a single large or multiple non-bona fide orders at the same price levels , generally at or close to best (bid/ask) prices.

Market regulators have reported many real cases of layering such as the case described by the Financial Industry Regulatory Authority (FINRA), USA, and discussed by Nanex Nanex, 2022. In fact, at the beginning the national best bid and offer (NBBO) was 101.24−101.35, when a non-genuine trader wants to sell 1000 shares on NASDAQ OMX BX, at a price higher than the best buy price $101.24, he sends a bona-fide order to sell 1000 shares at a price of $101.32 , followed by sending 05 buy non-bona fide orders to NASDAQ totaling 2000 shares at progressively higher price levels from $101.27 to $101.31. The best buy price was elevated by seven basis points (bps) as a result. Trader’s non bona fide buy orders created a false appearance of demand at that price, which led an automatic buying Algorithm, through seven transactions, partially executed against the bona fide 1000 share sell order. It executed a total of 645 shares at a price of $101.32 (more than $101.24) against Trader A’s bona fide short sale order. The non-genuine trader sent six additional buy orders (totaling 600 shares) to Nasdaq at a price of $101.31, Buying Algo partially executed additional 200 shares at $101.32. The remaining 155 shares of the 1000 share bona fide sell order were canceled, and the manipulator canceled all eleven non-bona fide buy orders sent to NASDAQ. The NBBO returned to the pre-layering price levels of $101.24 - $101.35. Figure 2 above shows trader A’s trading activity from 11:08:33.063 to 11:08:33.883 in a total

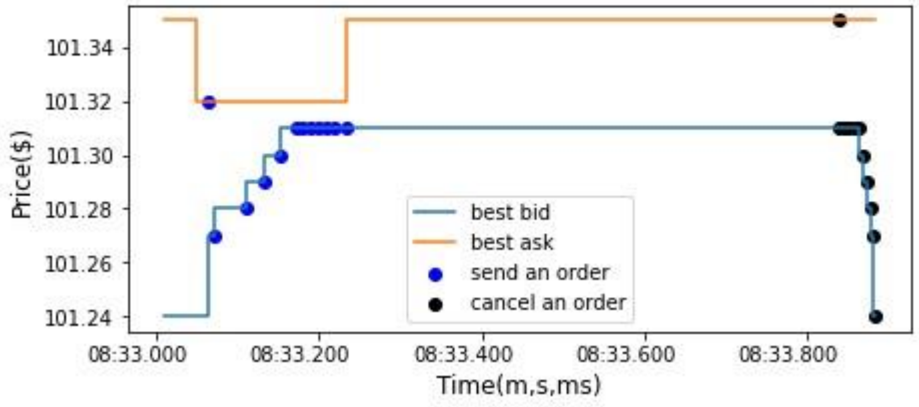


Figure 2: Layering takes a saw-tooth pattern on September 25, 2012. Disruptive trading activity from 11:08:33.063 to 11:08:33.883 in a total of 820 milliseconds. the hour is no shown on the Time axis of the graph.

of 820 milliseconds. This type of manipulation takes a form of saw tooth pattern that has been reported in Lin X, 2012.

In Figure 3 a single spoofing order is placed in the bid-ask. This allows small move up/down a percentage and reverts to its previous level. This manipulation takes the pulse pattern form Lin X, 2012.

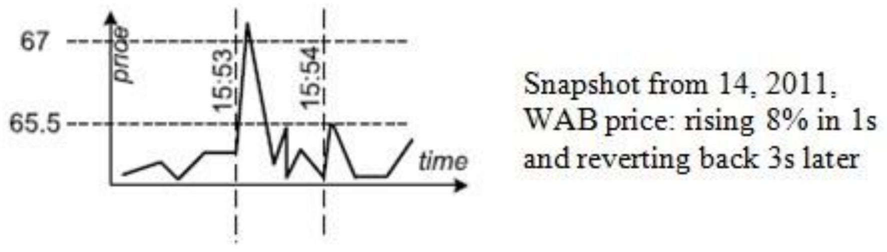


Figure 3: Pulse pattern : pump and dump manipulation activity from 14, 2011, Westinghouse Air Brake price: rising 8% and reverting back 3s later.

2.2. *Quote stuffing strategy*

Quote stuffing is a type of market manipulation that involves a high-frequency trader sending a large number of orders to a financial market in a short amount of time. The goal is to overload the market’s ability to process orders and cause temporary market disruptions. These disruptions can result in delayed order executions, missed trades, and increased volatility, which the trader can take advantage of to profit from price discrepancies. For example, a quote stuffer might send thousands of orders for a specific stock within a few milliseconds. This flood of orders creates a backlog in the market’s systems, causing delays in order processing and execution. The quote stuffer might then cancel the majority of these orders, leaving only a few that they hope to execute at an advantageous price. By taking advantage of the market inefficiencies created by the quote stuffing, the trader can buy low and sell high, making a profit. Figure 4 show a quote stuffing activity from July 13, 2012 of CAPC stock price. Lin X, 2012

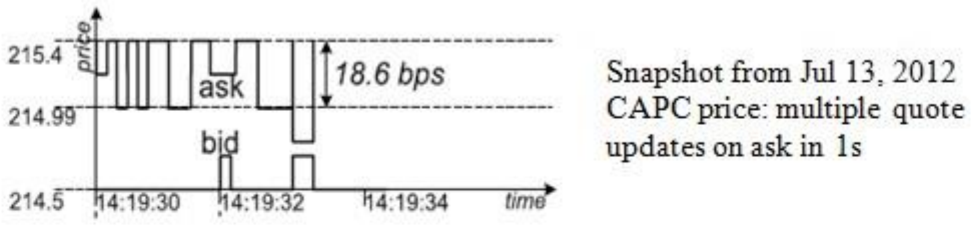


Figure 4: Quote stuffing activity from July 13, 2012 CAPC stock price: lot of quotes been sent for overloading the quotation systems.

In the following section, we review the existing literature on detection methods specifically tailored to identify the various manipulation strategies discussed. By examining these detection approaches, we aim to assess their effectiveness, identify potential limitations, and explore opportunities for further improvement in this critical area of market surveillance and integrity.

3. Literature review

In this section, we review previous related work on trade manipulation detection problem. We focus on data mining-based approaches which are related to our work.Stock market manipulation detection techniques can generally be divided into supervised and unsupervised techniques. Ögüt et al., 2009 applied a supervised learning technique to detect market manipulation of stocks traded on the Istanbul Stock Exchange by analysing statistical features of daily volatility, daily trading volume and daily return. They used support vector machines (SVM) and artificial neural networks (ANNs) which was reported to have outperformed multivariate statistical techniques such as discriminant analysis and logistic regression. As the manipulator uses new strategies to manipulate the stock, the proposed approach is only suitable for detecting known manipulation strategies and fails to discover unseen manipulation schemes. Diaz et al., 2011 combined three decision tree algorithms, namely QUEST, C5.0, and C&RT, to detect intraday price manipulation. The proposed model was applied to a dataset constructed from the SEC (US) manipulation cases in 2009. A set of 10 features was used which includes different technical indicators that reflect changes in trading liquidity, volatility, returns, and abnormal returns. these statistical features such as daily volatility, daily trading volume, daily return and liquidity were selected in the methods proposed above based only on empirical research' beliefs that changes in particular market variables imply manipulation. However, there is no evidence that these changes are necessary a condition for the manipulation. meaning that there are also others market events may cause a change in a market variable, such as news events. therefore, the proposed methods are ineffective in detecting real cases of manipulation. Golmohammadi et al., 2015 applied different classification methods (CART, conditional inference trees, C5.0, Random Forest, Naïve Bayes, Neural Networks, SVM, and KNN) to detect market manipulation using the percentage return price of stock as a feature instead of only the price of stock employed in the work by Diaz et al., 2011, their argument was that the price of securities did not reflect the size of a company nor the revenue, they used also the same manipulation cases dataset employed in the work Diaz et al., 2011. It was found that the Naïve Bayes classifier outperformed other counterpart classifiers achieving an F2 measure of 53%. However, the statistics features used in the proposed method make it suitable only in detecting single points as anomalies, it fails to detect collective anomalies like quotes stuffing strategy. Mongkolnavin and Tirapat, 2009 employed association rules to detect instances of "Mark the Close" misconduct in the Thai bond market. Their study involved analyzing historical trading data, including trading behaviors and price fluctuations, to monitor transactions in real-time and identify any unusual activity. The study examined a sample of 54,334 trading transactions reported to the Thai Bond Market Association (ThaiBMA) in 2005, and found that association rules can be effectively used to detect misconduct and alert regulators when enforcement action is required. Sridhar et al., 2020 proposed an approach to tackle the problem of market manipulation by utilizing an Ensemble Neural Network. The proposed system can identify three types of manipulation scenarios, namely, Price Manipulation, Volume Manipulation, and Trade Reversal. To construct the daily trading dataset, the affidavit information provided by the Securities and Exchange Board of India

(SEBI) was used, and data was collected from the Bombay Stock Exchange (BSE) website. The trading dataset was analyzed using the Ensemble Neural Network model with and without trainable sub-model layers. The results indicate that the model with trainable sub-model layers achieved an accuracy of 91%, while the model without trainable sub-model layers achieved an accuracy of 96%. Uslu and Akal, 2022 proposed a machine learning approach for detecting trade-based manipulations in the Borsa Istanbul (BIST) capital market. Their dataset comprised 22 cases of manipulation that occurred between 2010 and 2015, and they used daily data of manipulated stocks to train supervised machine learning classification models. The proposed model's performance was evaluated using the accuracy, sensitivity, and F1 score measurement methods. the proposed model outperformed LR (Logistic Regression), NB (Naive Bayes), and SVM (Support Vector Machine) in detecting trade-based manipulation in the stock market, Specifically, with an F1 score of 91% , sensitivity of 95%, and accuracy of 93%. This high score indicates that the model was able to accurately identify potential market manipulation with high precision and recall in the BIST capital market. The challenge with supervised methods in detecting stock price manipulation lies in obtaining labeled instances of normal and abnormal trades. This is difficult to achieve due to confidentiality and proprietary nature of trading data in most markets. As a result, there is a need for unsupervised learning algorithms to address this issue. Cao et al., 2014 used machine learning algorithms (OCSVM and KNN) to detect different forms of market manipulation, spoofing trading, and quote stuffing by transforming the equity price data into pseudo-stationary time series. They used order price, volume, and submission time as features. The proposed approach was tested on a dataset of four stocks (Apple, Google, Intel and Microsoft) taken from LOBSTER project, which covered five trading days, from the 11th to the 15th of June, 2012. The results of the performance evaluation showed better results in terms of AUC compared to the original market data. However, the one-class classification methods used do not take into account the temporal dependencies between data points. In another study, Cao et al., 2015 proposed an adaptive hidden Markov model with anomaly states (AHMMAS) to detect price manipulation activities. The method used wavelet transformation and gradient information as features. The reported results showed that the AHMMAS outperformed other benchmark models such as One-Class SVM, KNN, and Gaussian Mixture Models in terms of AUC and F-measure. But this method only focuses on detecting quote stuffing and may miss other forms of manipulation. The authors Abbas et al. have proposed multiple techniques for detecting price manipulation in stocks. These techniques were evaluated on level 1 tick dataset of five stocks (Apple, Amazon, Google, Microsoft and Intel Corp) obtained from the LOBSTER project. In Abbas et al., 2019, they used kernel density estimation and empirical mode decomposition for unsupervised price manipulation detection and achieved better results compared to other unsupervised methods such as PCA, k-Means and DPGMM. In Rizvi et al., 2019, they proposed a semi-supervised method using a modified dendritic cell immune system and kernel density estimation, and reported improved performance compared to K-Means, PCA, K-NN and OCSVM. In Rizvi et al., 2020, they used kernel principal component analysis and multidimensional kernel density estimation for unsupervised learning and showed significant improvement in performance and decrease in false alarm rate over other approaches. However, all these methods only focus on well-known manipulation schemes and do not consider the time dependence between data points. Leangarun et al., 2018 used Generative Adversarial Networks (GANs) that included Long Short-Term Memory (LSTM) networks in the generator and discriminator networks for detecting pump-and-dump manipulation. The discriminator network was utilized as an anomaly detector to identify manipulative behavior in the data. The approach was tested on data from the Thailand stock exchange and achieved a detection accuracy of 68.1% when evaluated on previously unseen market data. In a subsequent study, Leangarun et al., 2021 proposed an unsupervised learning approach using deep neural networks to detect stock price manipulation. The models were trained to recognize normal trading behaviors expressed in a limit order book and detect anomalous trading actions that did not follow the learned patterns as manipulative. The study assessed two model architectures, autoencoder (AE) and generative adversarial networks (GANs), on six prosecuted real manipulation cases from the Stock Exchange of Thailand (SET) during 2004–2016. Both models demonstrated a low false-positive rate and detected five out of six cases. Additionally, the study introduced "MinManiMax," a strategy to optimize the decision boundary for practical application of the models. However, this work had a low recall rate and struggled to detect new patterns. To address these limitations, Chullamonthon and Tangamchit, 2023 incorporated knowledge of the popular pump-and-dump pattern into a supervised long short-term memory (LSTM) network and combined it with the unsupervised LSTM-based autoencoder model in an ensemble approach. The resulting model achieved a higher detection rate with low false alarm rates on six real manipulation cases from the SET, demonstrating the potential of combining supervised and unsupervised learning methods to improve the accuracy of detecting stock price manipulation. Wang et al., 2019 used recurrent neural network-based ensemble learning for stock price manipulation detection, and considered company characteristics as features, reporting improved performance. However, this method is only suitable for supervised data and has limited memory to capture temporal dependence.

Most previous approaches solutions for identifying trade-based manipulation require a complex process of feature engineering, which involves the use of expert knowledge to create an effective representation of equity prices. These solutions also only focus on detecting specific manipulation types and require manual identification and description of relevant features. Additionally, these solutions treat trade-based manipulation as a single-point anomaly detection problem, neglecting the collective and temporal dependencies present in sequential trading activities. To overcome these limitations, this study proposes a method for detecting both known and unknown trade-based manipulation schemes by learning the normal behavior of historical bid/ask time series data using a deep generative model such as a generative adversarial network. The methodology of this proposed approach will be discussed in the following section.

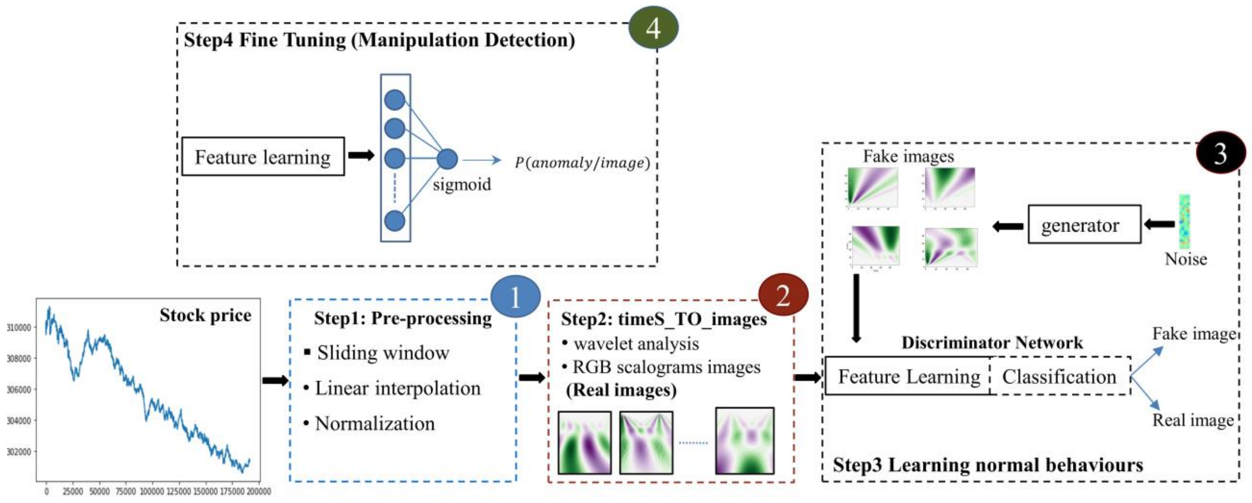


Figure 5: Overview of our Architecture , 1. and 2. Preprocessing and Encoding: after pre-processing, the equity price data is analyzed across multiple frequency scales using the Continuous Wavelet Transform (CWT). Once analyzed, the equity price data is then encoded into RGB images. 3. Training: A Deep Convolutional Generative Adversarial Network (DCGAN) is used to learn the normal behaviors of the encoded images. 4. Testing: The discriminator network is used as an anomaly detector to identify any abnormal behaviors in the stock market.

4. The proposed approach

The proposed approach aims to tackle the challenge of detecting manipulation schemes in the stock market by analyzing the changes in bid/ask time series data. Despite the fact that a raw time series data can provide insights into the behavior of the stock market, it can be difficult to extract hidden patterns due to its limited representation of the underlying dynamical system. To address this, the approach involves transforming the univariate bid/ask time series into a different space while preserving its temporal dependency structure. This process allows for the extraction of useful patterns that can aid in detecting trade-based manipulation. The overall approach consists of four main steps, as illustrated in Figure 5: data pre-processing, time series to image encoding, learning normal behaviors, and manipulation. To pre-process the data, sliding windows are used to partition the time series into different windows and normalize each window to the interval [0, 1]. The time series is then transformed into RGB scalogram images through the use of a continuous wavelet transformation (CWT). The normal behaviors of these scalograms are learned by training the Deep Convolutional Generative Adversarial Network (DC-GAN). Finally, the CNN is fine-tuned to distinguish normal scalograms from anomalous ones. The detailed process of each step will be described in more detail below.

4.1. Step1 : data pre-processing

The data used in this work consisted of the bid/ask price of five stocks: Amazon, Apple, Google, Intel, and Microsoft based on the official NASDAQ Historical TotalView-ITCH.

The ITCH protocol provides real-time updates on the status of orders from submission to execution or cancellation.

4.1.1. Sliding windows

The original bid/ask times series $x(t) = \{ x_1,x_2,...,x_L \}$ of length L is first divided into segments of a fixed size S ($S > 1$) using sliding windows method. The windows are shifted by $S/2$ and result in m time windows $x(t) = \{ x(t_1),x(t_2),...,x(t_m) \}$, where $m = (2*L-S)/S$. For instance, if a 10-point time series uses a time window of size $S = 4$, it would be divided into 4 segments, as shown in Figure 6. In this study, a time window size of 1000 milliseconds and a step size (shift) of 500 milliseconds was used. Thus, a 1-hour time series would be divided into 7199 time windows.

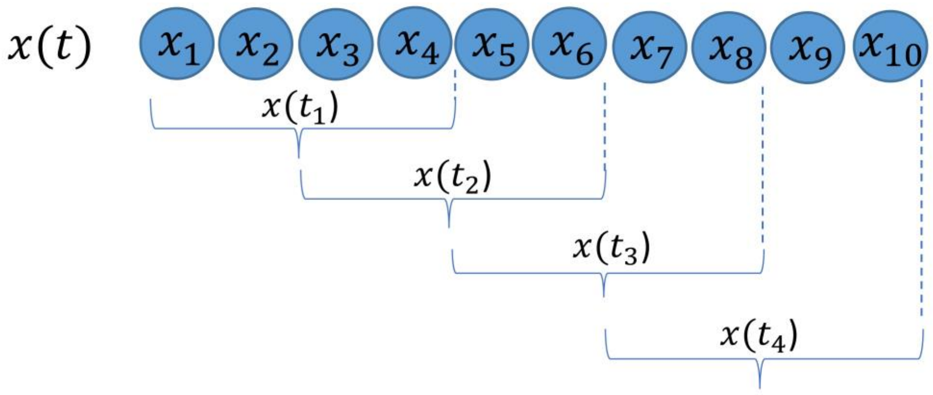


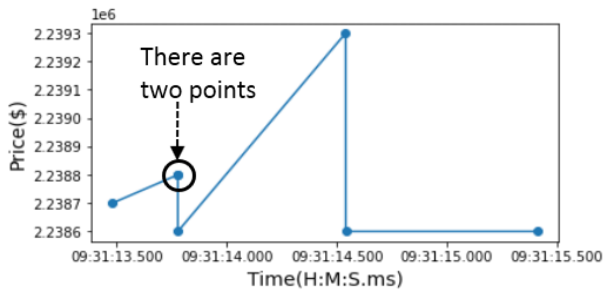
Figure 6: sliding windows mechanism: a time series $x(t)$ is windowed into four segments $x(t_1),x(t_2),x(t_3)$ and $x(t_4)$ each of them of size four.

4.1.2. Linear interpolation and normalization

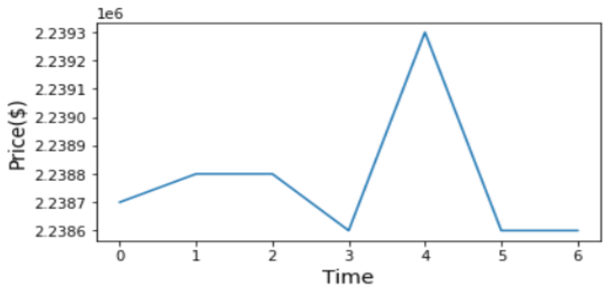
To handle the irregular spacing in each time window, we applied linear interpolation to convert it into an evenly spaced time series with a resolution of 10 milliseconds. As a result, each window becomes a row vector of 100 observations as shown in Figure 7. Furthermore, to ensure consistency across all windows, each window is normalized to the range of [0, 1] using the following equation.

$$X\_normalized = (X\_window - X\_min)/(X\_max - X\_min) \quad (1)$$

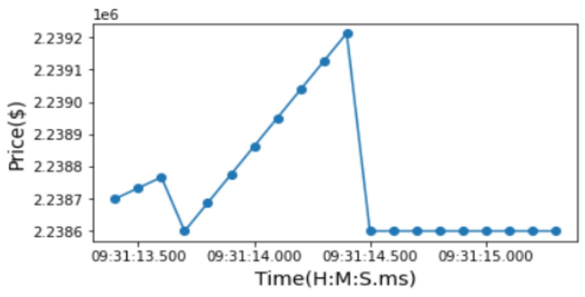
where $X\_normalized$ is the normalized value, X_window is the original value, X_min is the minimum value in the window, and X_max is the maximum value in the window. This normalization step is important as it helps to prevent the influence of outliers on the results. The normalized windows are then ready to be transformed into images using the Continuous Wavelet Transform (CWT).



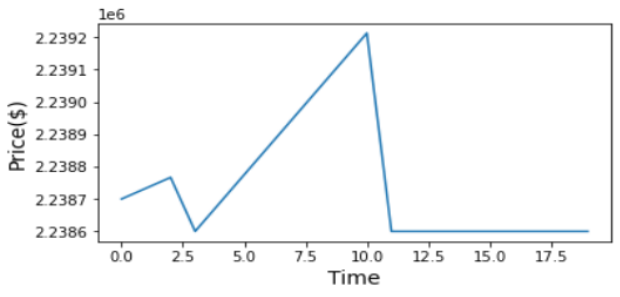
a- Original Bid order



b- Bid order as row vector



c- Bid order after interpolation



d- Bid order as row vector after interpolation

Figure 7: The Bid order as vector without time index (b) is very different to the original bid order with time index x-axis (a), the bid order as row vector after linear interpolation (d) is very similar to the original bid order with time index in x-axis (a).

4.2. Step2 : Continuous Wavelet Transformation (scalograms)

After normalizing the time series, a continuous wavelet transform (CWT) is applied to each window. The CWT provides a representation of a time series in the time-frequency domain, allowing the visualization of the evolution of frequency content over time. In this work, we have used the Morlet wavelet as the mother wavelet because it is a good choice for analyzing non-stationary signals like stock market data. The resulting representation is called a scalogram, which is a gray-scale image that shows the frequency content of the signal over time. In this work, we have transformed the scalogram into an RGB image, which allows for easier processing by deep learning algorithms. The RGB scalograms are used as input for the next step, learning the normal behaviors. The mathematical representation of the CWT of a signal $x(t)$ is given by Equation (2)

$$CWT_x(s,\tau) = \frac{1}{\sqrt{|s|}} \int_{-\infty}^{+\infty} x(t)\varphi^*(\frac{(t-\tau)}{s})dt \tag{2}$$

Where $x(t)$ is a time signal, $\varphi()$ is the mother wavelet function, s is the scale and τ is the translation parameter that determines the shifting position. In this study, we utilized a discretized version of the continuous Morlet wavelet transform Mallat, 1989 for its computational convenience. A demonstration of the translated and scaled Morlet wavelet is shown in Figure 8.

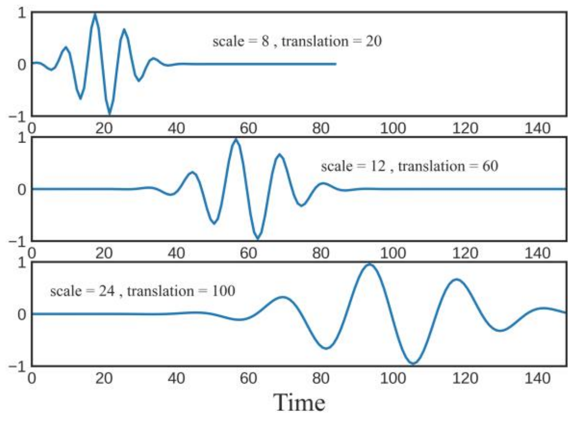


Figure 8: morlet wavelet at different scales and translation.

By using a smaller scale, we can capture sudden changes in the ask/bid time series, these suddenly changes are the most important features in detecting pump and dump manipulation schemes, On the other hand, a larger scale can capture lower frequencies (i.e slowly changes in ask/bid time series). the cwt coefficients $CWT_x(s,\tau)$ in equation (2) are obtained by continuously varying the values of the scale parameter s, and the translation parameter τ.

The CWT uses inner products to measure the similarity between the bid/ask time series and shifted and compressed/stretched versions of a wavelet function as shown in figure 9.

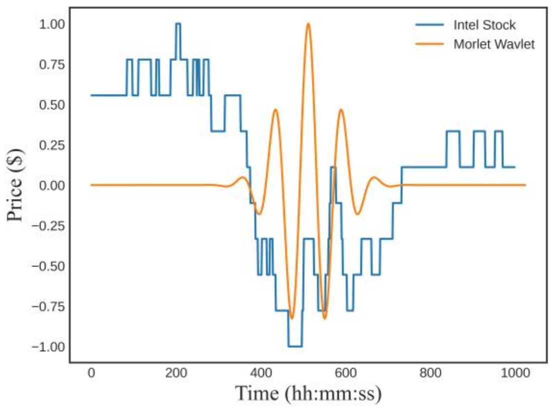


Figure 9: Intel stock with 1000 timesteps multiplied by a morlet wavelet (scale=16, translation=500).

By continuously varying the values of scale parameters, and the position parameter we obtain the cwt coefficients represented as scalogram image as shown in figure 10.

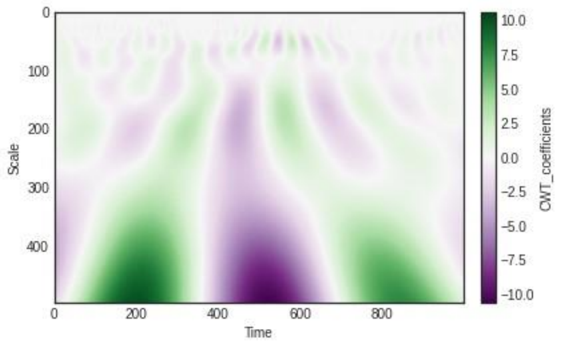


Figure 10: Scalogram image obtained by multiplying our bid time series with 1000 timesteps and 500 (in a range from 0 to 499) scales results in 500000 coefficients.

Figure 11 shows RGB images which are obtained through the application of CWT to sample non-stationary stock market data. These colour images illustrate both the normal and anomalous scalograms which correspond to normal and anomalous trading behaviours.

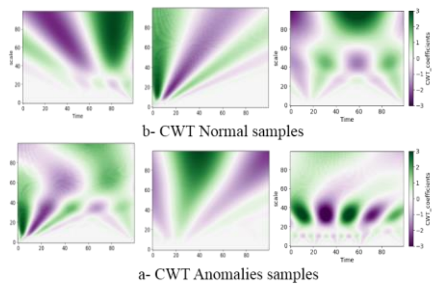


Figure 11: Scalogram images for normal and anomaly stock data. The obtained images for these samples show a clear difference between the normal and anomalous images.

4.3. Step3 : Learning normal behaviors

The scalograms obtained from the Continuous Wavelet Transformation step, are then fed into a Deep Convolutional Generative Adversarial Network (DC-GAN) to learn the normal behavior of these scalograms. The architecture and hyper-parameters of the DCGAN model are based on guidelines described in Radford et al., 2016 . It consists of two convolutional neural networks: the Generative network and the Discriminative network. The Generative network generates 100x100x3 fake images that appear real. It takes a random 100-dimensional vector from a normal distribution of mean 0 and variance 1 as input, and includes three deconvolution layers with decreasing number of filters. The output layer uses the Tanh activation function. The Discriminative network takes an image of dimension 100x100x3, passes it through three convolution layers with increasing number of filters, and ends with a linear activation in the output layer. The Discriminator network learns to distinguish normal behavior from anomalies as shown in Figure 12.

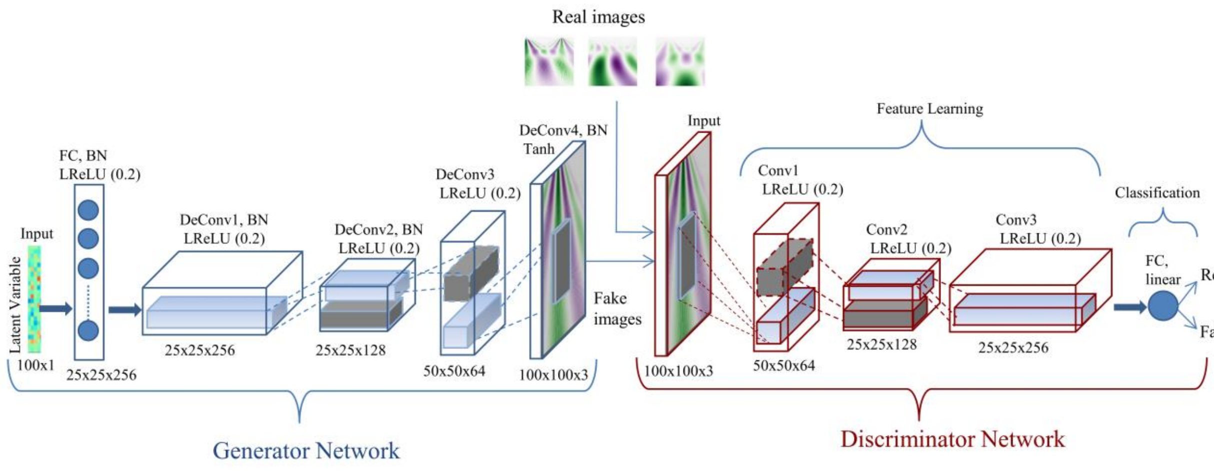


Figure 12: Learning normal behaviors of stock price.

4.4. *Step4 : Manipulation Detection using the discriminator*

Once the Generative network has learned the normal behavior, a selection of normal data samples with lower scores is used to evaluate the performance of the Discriminator network. The Discriminator network is then used as a feature learning method stacked with a fully connected layer to detect manipulation cases by using only a few normal data samples with a few injected anomalies. The process is visualized in figure 13

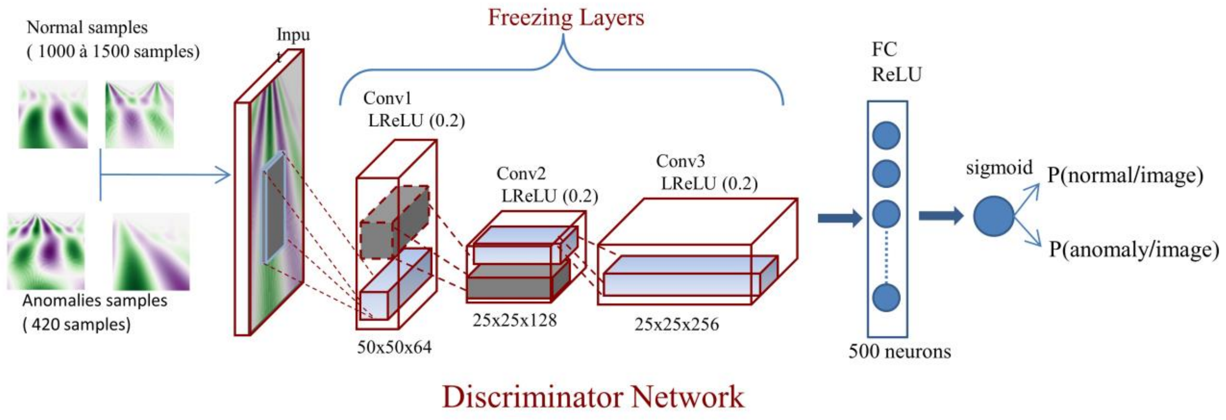


Figure 13: Manipulation Detection using the discriminator network.

5. Generative Adversarial Network (GAN)

The Generative Adversarial architecture (GAN) was described by Goodfellow et al., 2020 . The GAN can be considered as a min-max game between a Generator G and the Discriminator D. The Generator is trained to generate fake images and the Discriminator is trained to distinguish the fake examples (generated by G) from real examples (coming from the training dataset). Therefore, whenever the Discriminator is tricked into classifying a false image as real, the Generator realizes that he has accomplished something right. Alternately, each time the Discriminator effectively rejects an image created by the Generator as being false, the Generator gets the feedback that it needs to improve. This optimization process can be formulated using equation 3.

$min_G max_D = E_{x\sim p_{data}(x)}[log(D(x))] + E_{z\sim p_z(z)}[log(1 - D(G(z)))]$ (3) where p_{data} is the probability that the image comes from the real data, p_z is the probability that the image comes from fake data (generated), and z is the noise variable. The optimization process defined in (3) is known as Minimax GAN Loss. However, this loss function can suffers from saturation of the generator's loss, $log(1-D(G(z))$ if the generator is unable to learn as quickly as the discriminator. Thus, model cannot be trained effectively. The alternative solution is to train the Generator to maximize $logD(G(Z))$ i.e minimize $-log(D(G(Z))$

$$
\begin{aligned}
L_{NSGANG} &= -E_{z\sim p_z(z)}[log(D(G(z)))] \\
L_{NSGAND} &= E_{z\sim p_{data}(x)}[log(D(x))] - E_{z\sim p_z(z)}[log(1 - D(G(z)))]
\end{aligned}
\tag{4}
$$

The optimization process defined in (4) known as Non-Saturating GAN Loss. The NonSaturating Loss function minimizes the cross-entropy, but it also faces difficulties in training the generator when using the sigmoid activation function in the last layer with cross-entropy as the loss function Mao et al., 2017 . The vanishing gradient of the generator can result in an ineffective training process, as demonstrated in the figure 14.

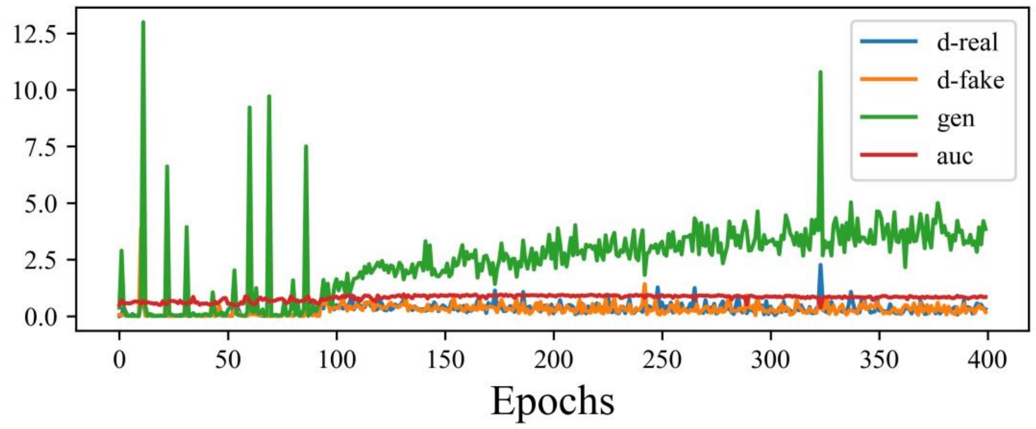


Figure 14: loss plot of the generator (green line) , discriminator loss of real and fake image (blue and orange line), area under cure red line(normal and anomalies image during training), use cross-entropy loss and sigmoid activation function in the last layer in the discriminator network.

October 5, 2023

Figure 14 shows that after 100 epochs the discriminator does not give enough feedback to the generator. Thus the generator network cannot be trained effectively we notice that the green line increases quickly from epoch 100 and takes on high value. We have used in this test the 1-level tick data of the Intel stock market for 21th June 2012 obtained from the lobster Data at "LOBSTER project, Atlanta, GA, USA. Limit order book", n.d. . To resolve this issue, we use in our approach the Least Squares GAN Loss function for the optimization process proposed by Mao et al., 2017 defined as:

$$L_D^{LSGAN} = \frac{1}{2}E_{x\sim p_{data}(x)}\left[(D(x)-1)^2\right] + \frac{1}{2}E_{z\sim p_z(z)}\left[(D(G(z)))^2\right]$$
$$L_G^{LSGAN} = \frac{1}{2}E_{z\sim p_z(z)}\left[(D(G(z))-1)^2\right] \tag{5}$$

Figure 15. shows the accuracy and the loss plot of the Discriminator and the Generator when using the linear activation function in last layer and the least square loss function for the Intel stock.

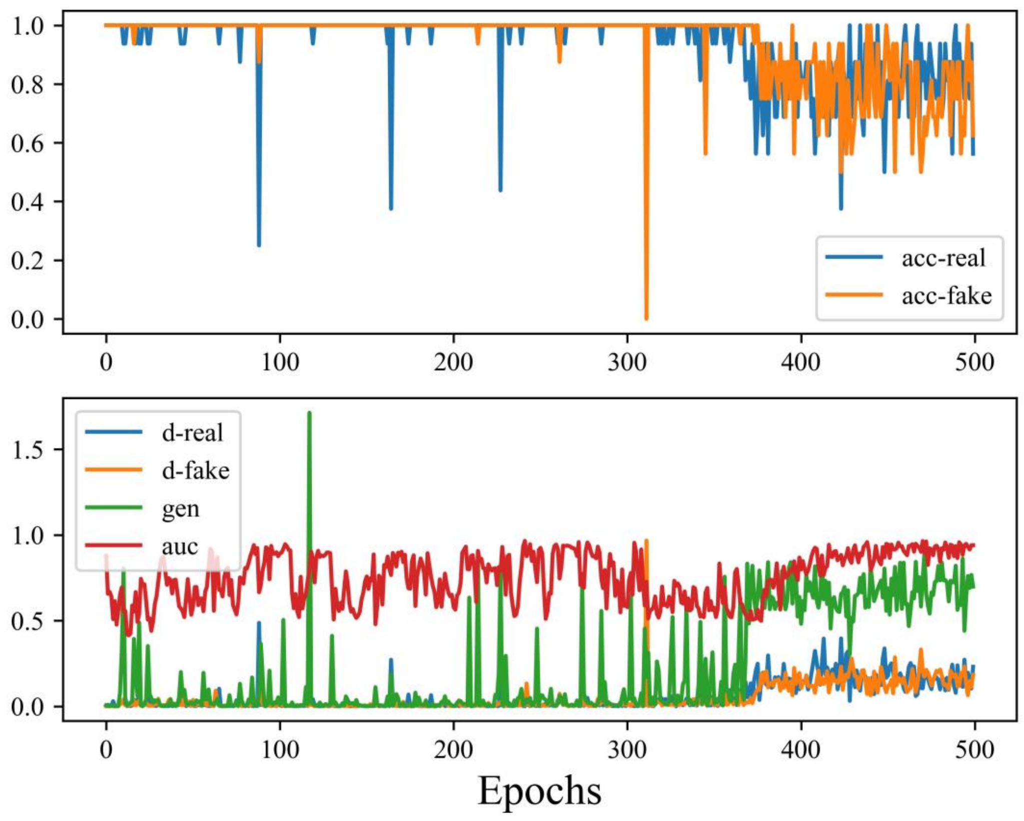


Figure 15: Top subplot represents accuracy for the discriminator for both real images and generated image, bottom subplot represents loss of the discriminator. Red line represents area under cure (normal and anomalies image during training), use least square loss and linear activation function in the last layer in the discriminator network.

In Figure 15, it is evident that the generator G struggled to produce realistic images at the outset of training (from 0 to 380 epochs). The discriminator D was able to confidently distinguish between fake images generated by the generator and real images, as the former differed significantly from the latter. The discriminator's classification accuracy (as shown in the top subplot) for identifying fake images from real ones remained consistently high throughout this period. The bottom subplot's red curve shows the area under the curve for this architecture's ability to identify normal scalogram images from manipulated ones (such as manipulated stock prices). During the first 380 epochs, this AUC was unstable, primarily due to the generator G's inability to deceive the discriminator D. However, starting from epoch 380, the generator G began to learn the normal behavior of real images and was able to generate more realistic images. As seen in the top subplot, the discriminator's classification accuracy decreased, making it increasingly difficult for the discriminator D to distinguish between real and fake images. In the bottom subplot, the discriminator loss for both real and fake samples hovered around 0.28, while the generator's loss was around 0.6. The most significant curve to note is the red curve, which represents the architecture's ability to identify real images from the three types of price manipulation injected into the normal bid/ask time series. The AUC curve stabilized over time, becoming smooth, with a mean value of around 0.92 between 400 and 500 epochs and a small standard deviation.

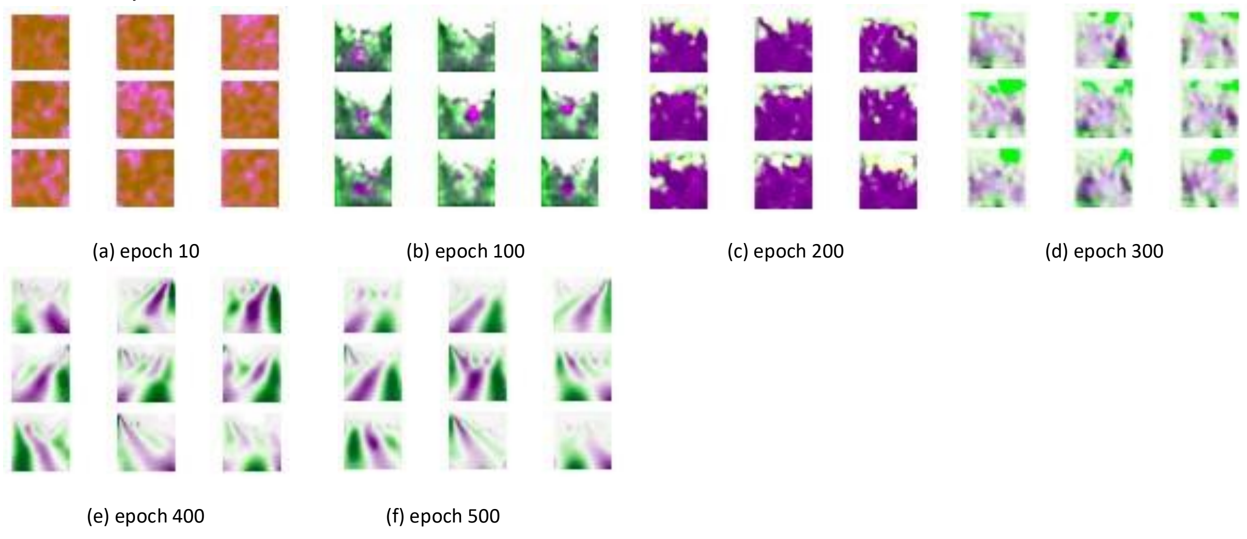


Figure 16: Generated images at different epoch (first run) for Intel stock

The figure depicted above indicates that the generator initially struggled to generate images that closely resembled real images. However, by epoch 400 and 500, the generator started learning the normal image behaviors and generating images that closely resembled the real ones. During the first run, the generator took a considerable amount of time (380 epochs) to start learning. This was primarily because of the network's hyper-parameter selection. In contrast, Figure 17 and Figure 18 show the results of a second run of the model, where the generator started generating realistic images much earlier, i.e., from epoch 180 instead of 380 epochs in the first run.

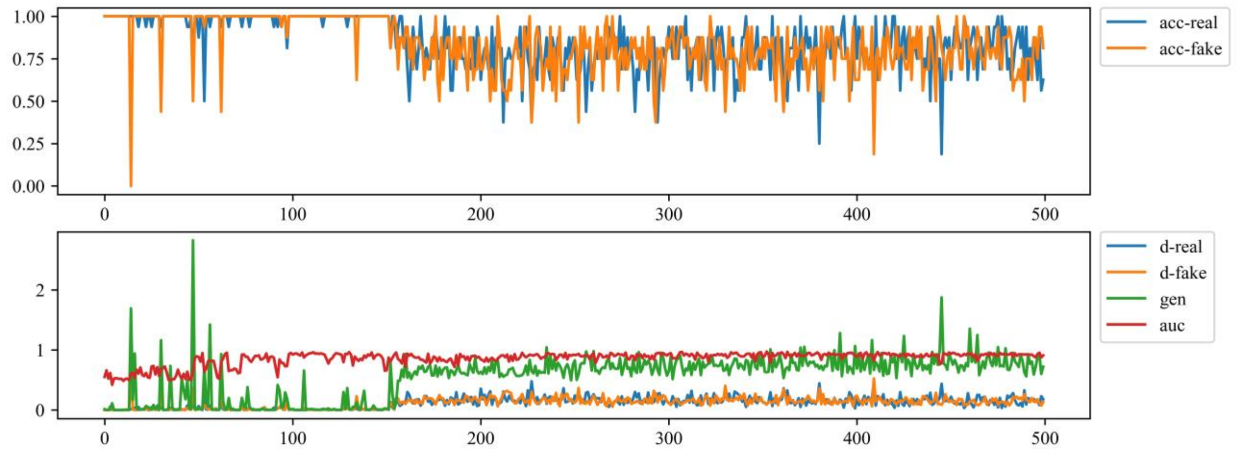
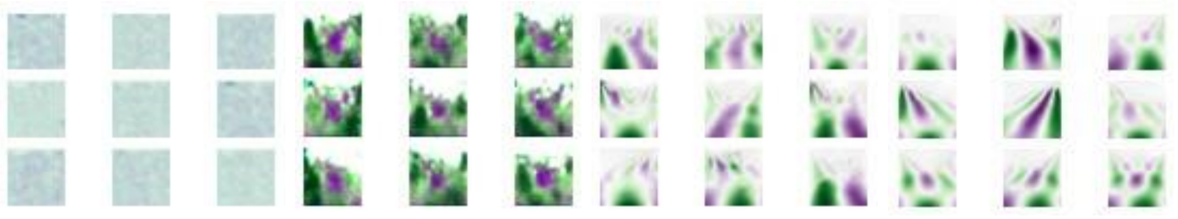


Figure 17: Top subplot represents accuracy for the discriminator for both real images and generated image, bottom subplot represents loss of the discriminator. Use least square loss and linear activation function in the last layer in the discriminator network



(a) epoch 10 (b) epoch 100 (c) epoch 200 (d) epoch 300

Figure 18: Generated images at different epoch (second run) for Intel stock

6. Results and discussion

The dataset utilized in this study comprised of 1-level tick data for June 21st, 2012, of five stocks, namely Amazon, Apple, Google, Intel, and Microsoft. The data was sourced from the official NASDAQ Historical TotalView-ITCH and was made available by Lobster Data “LOBSTER project, Atlanta, GA, USA. Limit order book”, n.d. . To assess the effectiveness of the proposed WALDATA approach , we introduced three distinct manipulation schemes randomly into the financial time series data, each designated as Type 1, Type 2, and Type 3. Type 1 involved the insertion of 6.9-basis-point (bps) saw-tooth patterns within an 819-millisecond window (Figure 2). Type 2 consisted of an 800-bps pulse occurring over a one-second interval (Figure 3), while Type 3 featured an 18.6-bps square pattern lasting a mere 1 second (Figure 4). For each of the five stocks, we injected a total of 800 instances of three different manipulation schemes, this led to a total of 4,000 instances across all five stocks. In the process of injecting anomalies into the dataset, meticulous attention is paid to maintaining the integrity of normal statistical features such as the mean and variance. This ensures that the anomalies, though introduced intentionally, do not disrupt the fundamental statistical characteristics of the data. So for amazon stocks we injected a 6.9bps Saw-tooth pattern figure 19, a 30 bps pulse pattern figure 20 and 8bps square pattern as shown in figure 21.

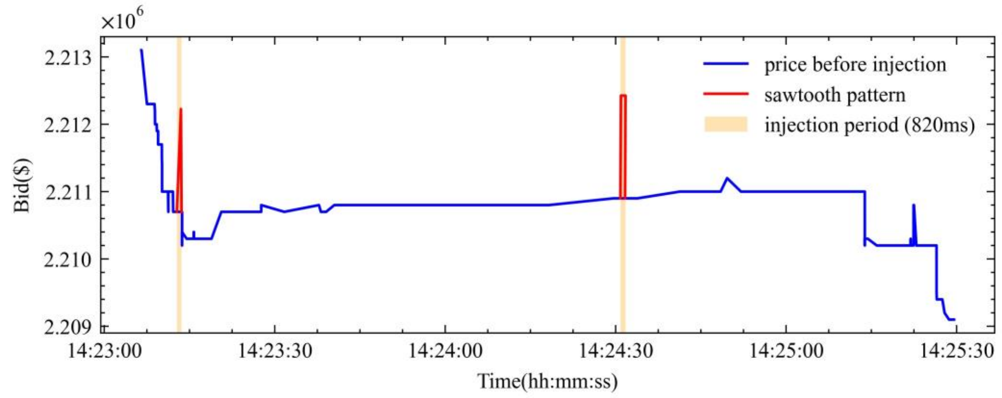


Figure 19: Sawtooth-injected manipulation of Amazon stock at random locations.

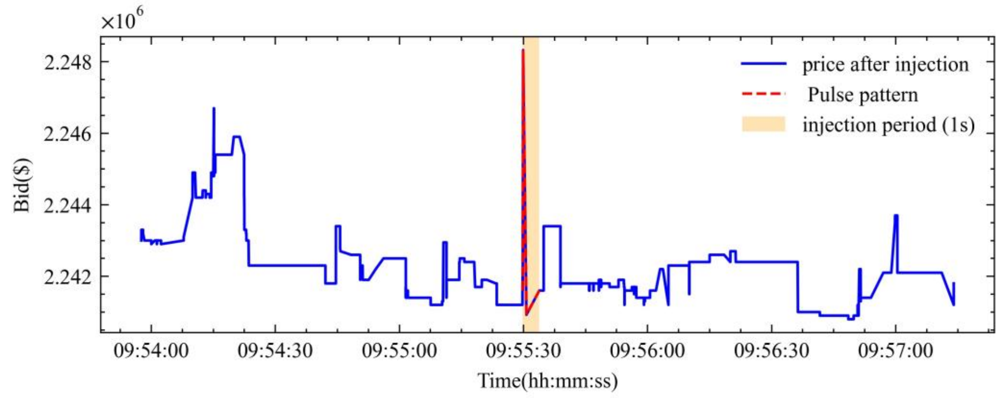


Figure 20: Pulse-injected manipulation of Amazon stock at random locations.

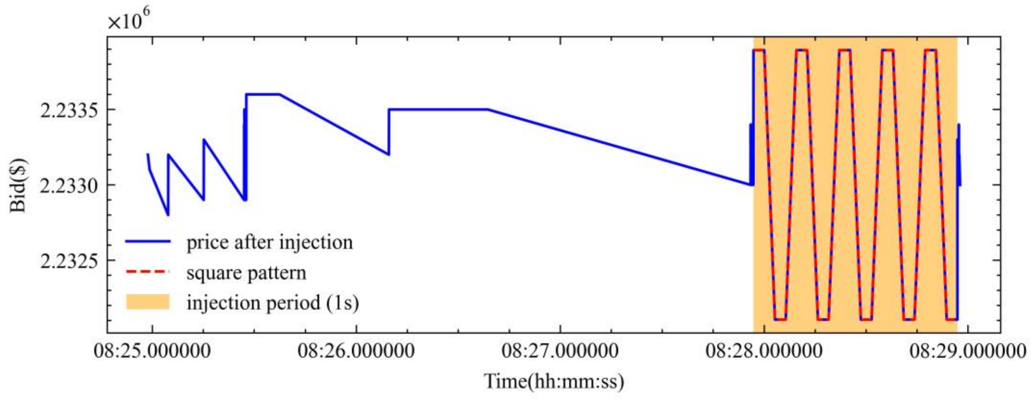


Figure 21: Square-injected manipulation of Amazon stock at random locations.

To prepare the data for training and testing, the normal time series were segmented into sliding windows of 1000 milliseconds. These windows were subsequently transformed into scalogram images using the continuous wavelet transform (CWT). Similarly, the anomalous time series were also encoded into scalogram images using CWT. The resulting scalogram images were then partitioned into separate sets for training and testing. The normal data set was divided into a 70% normal training set and a 30% normal testing set. This division ensured a significant portion of normal data for training the detection model while allowing for a robust evaluation on the testing set. Additionally, the anomalous data set was divided into a 60% anomalies training set and a 40% anomalies testing set. Table I provides further details on the dataset division.

Table 1: TRAINING AND TESTING EXAMPLES FOR 05 STOCKS

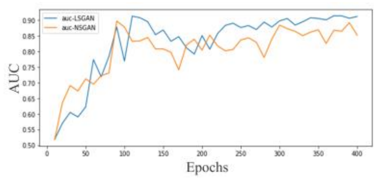
Stocks	training		testing	
	normal	anomalies	normal	anomalies
Amazon	4416	480	1893	320
Apple	9760	480	4183	320
Google	3028	480	1299	320
Intel	1053	480	452	320
Microsoft	1300	480	558	320

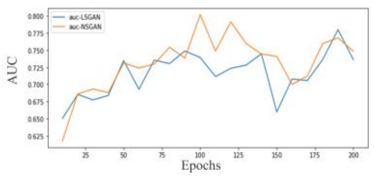
The evaluation of performance is based on the Confusion matrix presented in Table II. True Positive (TP) represents manipulation cases correctly identified as manipulation, False Positive (FP) indicates normal cases detected as manipulation , True Negative (TN) represents normal cases correctly identified as normal, and False Negative (FN) indicates manipulation cases detected as normal.

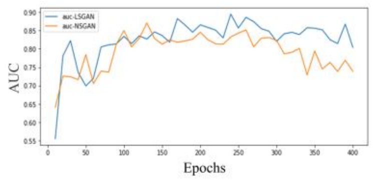
Table 2: CONFUSION MATRIX

	Manipulation cases	Normal cases
Test outcome manipulation	True positive (TP)	False negative (FN)
Test outcome normal	False positive (FP)	True negative (TN)

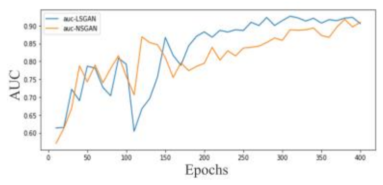
The ROC curve and AUC were also employed as performance measures to assess the performance of the proposed method on real stock datasets Fawcett, 2006 . The use of real stock datasets adds to the validity and applicability of the proposed method in realworld scenarios. The stocks of the five companies included in the study are well-known and widely traded in the financial market, which highlights the relevance of the proposed method in practical settings. The ROC curve is a graphical representation of the trade-off between the true positive rate (sensitivity) and the false positive rate (1-specificity) of a binary classifier system. The AUC represents the area under the ROC curve and is a widely used performance measure for evaluating classification systems. A higher AUC indicates better classification performance. To evaluate the proposed method, the average AUC for every 10 epochs of the Discriminator network on the testing dataset was computed using Non-saturating Generative Adversarial Networks (NS-GAN) and Least Square Generative Adversarial Networks (LS-GAN). The results are presented in figures 19. These figures demonstrate the effectiveness of the proposed method in detecting anomalies in financial time series data.

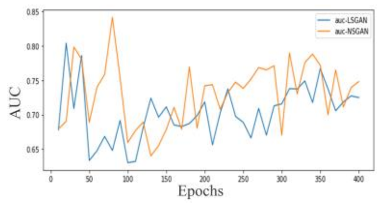






(a) Intel Stock: max average AUC- (b) Apple Stock: max average AUC- (c) Amazon Stock: max average AUC-
LSGAN: 0.9135 max average AUC- LSGAN: 0.78 max average AUC- LSGAN: 0.8950 max average AUC-
NSGAN : 0.8967 NSGAN : 0.80 NSGAN : 0.8707





(d) Microsoft Stock: max average (e) Google Stock : max average AUCAUC-LSGAN: 0.9260 max average
LSGAN: 0.8041 max average AUC-
AUC-NSGAN : 0.9174 NSGAN : 0.8417

Figure 22: Average of the area under curve (AUC) on Intel, Apple, Amazon, Microsoft and Google stocks for every 10 epoch of the discriminator network on the testing dataset, blue curve represent AUC of LS-GAN and orange represent AUC of NS-GAN.

Figure 19 illustrate the Discriminator network's initial inability to distinguish normal cases from manipulation cases during the training process, as the Generator initially produces fake samples that significantly differ from the real samples. As a result, the Discriminator, which acts as a binary classifier, identifies manipulation cases as being more similar to the real samples than the fake ones. However, as the Generator progresses in generating fake samples that more closely resemble real ones, the Discriminator becomes increasingly capable of detecting manipulation cases. Notably, the AUC achieves a maximum value of 0.8950 for Amazon, 0.9260 for Microsoft, and 0.9135 for Intel, with the exception of Google, whose AUC value was 0.8041. This disparity can be attributed to the high variability in data and the potential overlap of normal patterns. The maximum AUC value of 0.78 achieved for Apple Stock is relatively low due to the high number of normal samples in this dataset compared to the others, with 13943 cases in the training dataset and 9760 in the testing dataset (refer to Table I). As a result, the generator requires more time to learn and optimize parameters efficiently, such as the selection of the latent dimension. Given the difficulty in learning the generator to generate fake samples that cover the entire normal case distribution, the discriminator may predict these normal cases as manipulations cases, leading to an increase in the number of false positives and a corresponding decrease in the AUC value. To further enhance the detection capabilities, particularly in scenarios where some manipulation schemes is available, we conducted a fine-tuning process. This iterative step involved training the discriminator network stacked with a fully connected layer in a supervised manner, By incorporating anomalous samples alongside carefully selected normal samples that had initially proven challenging for the discriminator to classify as normal during the previous training phase. During the subsequent testing phase, the discriminator network's output was interpreted as a probability value ranging from 0 to 1. Initially, a threshold of 0.5 was employed to classify samples as either normal or anomalous. If the output probability exceeded 0.5, the sample was classified as normal, while a value below 0.5 resulted in the sample being classified as anomalous. The effectiveness of training the discriminator network with available manipulation schemes is demonstrated in Table 3 and Table 4. These tables present the results obtained from the experiments when using threshold of 0.5, showing the performance metrics such as accuracy, precision, recall, and F1-score. The outcomes highlight the discriminative power and efficacy of the fine-tuned discriminator network in accurately detecting and classifying instances of manipulation, underscoring the value of incorporating information about manipulation schemes into the training process.

	Manipulation	Normal
Manipulation	286	36
Normal	149	4034

(a) Apple Stock

	Manipulation	Normal
Manipulation	306	14
Normal	79	1814

(b) AMZN Stock

	Manipulation	Normal
Manipulation	312	8
Normal	31	527

(c) MSFT Stock

	Manipulation	Normal
Manipulation	291	29
Normal	7	445

(d) INTC Stock

	Manipulation	Normal
Manipulation	309	11
Normal	38	1261

(e) GOOG Stock

Table 3: Confusion matrix at threshold 0.5 using LS-GAN

The results validate the importance of incorporating manipulation schemes into the training phase, as it significantly enhances the discriminator network's ability to effectively identify and differentiate between normal market behavior and various manipulation schemes. However, given the imbalanced nature of the dataset, where the number of normal samples outweighed the number of anomalous samples, we encountered challenges in accurately detecting and classifying the minority class. To address this issue, we explored the impact of adjusting the threshold value. In our investigation, we found that lowering the threshold from 0.5 to 0.3 yielded improved results in terms of accuracy and performance metrics Table 5 and Table 6. By reducing the threshold, we increased the precision of the detection system towards detecting anomalous instances. This adjustment allowed for a more conservative classification approach, enabling the model to identify a greater number of true positives

Table 4: Summary of model performance using LS-GAN at threshold 0.5

	Manipulation	Normal
Manipulation	305	15
Normal	20	1873

(b) AMZN Stock

	Manipulation	Normal
Manipulation	290	30
Normal	4	448

	Accuracy	Precision	Recall	F1-score	AUC	FAR(%)
APPLE	0.9594	0.6575	0.8938	0.7576	0.9836	3.56
AMZN	0.9580	0.7948	0.9563	0.8681	0.9918	4.17
MSFT	0.9556	0.9096	0.9750	0.9412	0.9914	5.56
INTC	0.9534	0.9765	0.9094	0.9417	0.9953	1.55
GOOG	0.9697	0.8905	0.9656	0.9265	0.9947	2.93

and reducing the risk of false positives.

Table 5: Confusion matrix at threshold 0.3 using LS-GAN

	Manipulation	Normal
Manipulation	284	36
Normal	30	4153

(a) Apple Stock

	Manipulation	Normal
Manipulation	311	9
Normal	10	548

(c) MSFT Stock

(d) INTC Stock

	Manipulation	Normal
Manipulation	306	14
Normal	10	1289

(e) GOOG Stock

Table 6: Summary of model performance using LS-GAN at threshold 0.3

	Accuracy	Precision	Recall	F1-score	AUC	FAR(%)
APPLE	0.9853	0.9045	0.8875	0.8959	0.9836	0.72
AMZN	0.9842	0.9385	0.9531	0.9457	0.9918	1.06
MSFT	0.9784	0.9688	0.9719	0.9704	0.9914	1.80
INTC	0.9560	0.9864	0.9063	0.9446	0.9953	0.88
GOOG	0.9852	0.9684	0.9563	0.9623	0.9947	0.77

The results demonstrate that this approach yields stable and significant outcomes, with a low false alarm rate and high AUC for all five stocks. The success of this approach can be attributed to the ability of deep convolutional neural networks to automatically learn relevant features from images, which is employed in both the generator and discriminator networks. The generator produces fake images which represent manipulation trading (specifically, three manipulation schemes), and the discriminator learns to distinguish between these fake images and normal images which represent normal trading. The AUC for Intel stock achieved a score of 0.9953 after training only the last layer for one epoch, as the discriminator had already learned to distinguish between normal and fake images through its interaction with the generator. This approach was also successful in achieving a high AUC for the other stocks, as shown table 6.

Table 7: AUC COMPARISON OF WALDATA APPROACH (AFTER 800 EPOCH, THRESHOLD 0.30) WITH BENCHMARK TECHNIQUES FOR ANOMALY DETECTION.

	Amazon	Apple	Google	INTC	MSFT

WALDATA after fine-tuned	0.9918	0.9836	0.9947	0.9953	0.9914
WALDATA before fine-tuned	0.9014	0.8415	0.8717	0.9263	0.9370
KPCA-MKDE	0.9602	0.9206	0.8996	0.8732	0.9143
kNN	0.7982	0.7926	0.5612	0.5469	0.5509
PCA	0.9013	0.6902	0.7993	0.8680	0.8655
k-means	0.5799	0.5819	0.6328	0.5077	0.5047
OCSVM	0.8933	0.6603	0.5911	0.6770	0.6419
AHMMAS	0.5152	0.5344	0.5119	0.5169	0.6711

Table 8: FAR COMPARISON OF WALDATA APPROACH (AFTER 800 EPOCH, THRESHOLD 0.30) WITH BENCHMARK TECHNIQUES FOR ANOMALY DETECTION.

	Amazon	Apple	Google	INTC	MSFT
WALDATA after fine-tuned	1.03	0.72	0.76	0.88	1.80
WALDATA before fine-tuned	1.94	1.58	1.43	1.76	2.27
KPCA-MKDE	1.22	1.07	1.62	0.54	0.71
kNN	0.14	0.45	0.68	0.23	0.08
PCA	3.9	6.64	7.22	57.29	49.89

k-means	7.33	1.26	9.95	0.02	0.02
OCSVM	49.54	67.8	75.2	59.08	77.48
AHMMAS	9.22	7.83	0.5	1.15	0.52

Table 7 and table 8 demonstrate the evaluation of the performance of the WALDATA approach compared to commonly used anomaly detection techniques such as KPCA-KDE based Rizvi et al., 2020, K-nearest neighbor based Bishop, 2006 , PCA-based Shyu et al., 2003, k-means based Chawla and Gionis, 2013, OCSVM based Bishop, 2006, and AHMMAS based Cao et al., 2015 in both unsupervised and supervised learning.

The experiments demonstrated that our WALDATA approach achieved a higher rate of detection of manipulation of three types (Saw tooth, Spike & Square pattern) in stock price compared to of the existing approaches for stock price manipulation detection such as KPCAMKDE and AHMMAS, as well as some existing benchmark techniques for anomaly detection like PCA, K-means, kNN, and OCSVM. Specifically, the approach achieved a detection rate of 99% for the tested manipulation schemes, indicating that it is highly effective at detecting manipulation in financial markets. Additionally, the false alarm rate (FAR) of the proposed approach was comparable to existing approaches, demonstrating that it is capable of detecting manipulation while minimizing false positives. The robustness of the proposed approach can be attributed to the use of wavelet analysis in decomposing the stock price time series into multiple scales. This allowed for the identification of complex patterns and features that may indicate manipulation, leading to a more comprehensive understanding of the data. By leveraging this additional information, the WALDATA approach achieved higher detection rates compared to existing approaches. Additionally, the use of a DCGAN enabled the automatic learning of relevant features from the wavelet coefficients of the stock price data, without the need for human experts to handcraft features. This approach allowed for the learning of complex, high-level features from the data, which is particularly useful in cases where labeled data for stock price manipulation is not available.

7. Conclusion

The proposed WALDATA approach demonstrated its efficacy in detecting unknown manipulation schemes in stock markets by combining the Continuous Wavelet Transform and the Deep Convolutional Generative Adversarial Network. The results showed high accuracy and low false alarm rates in identifying trade manipulations, even in the absence of labeled datasets. The combination of deep learning and signal processing techniques opens up new avenues for anomaly detection in financial time series. Our approach is in line with previous studies that utilize generative neural networks for anomaly detection, including GANs and Variational Autoencoders (VAEs). These models have been shown to be effective in detecting financial fraud, such as insider trading and financial statement fraud. The proposed approach also highlights the potential of using image-based analysis in the financial domain, as demonstrated by the conversion of time series data into 2D images using the Continuous Wavelet Transform. One of the strengths of our approach is the use of normal trading behavior when training the generator/discriminator network. This allows our approach to detect all types of manipulation schemes, not just those considered in this study. Furthermore, our approach provides significant and stable results in terms of area under the curve (AUC) and a very low rate of false alarms for the five stocks tested: Amazon, Apple, Google, Intel, and Microsoft. Another advantage of our approach is that it is able to adapt to changes in the market conditions and update its understanding of normal behavior. This is because the generative neural network is trained on a continuous stream of data, which allows it to adapt to changing market conditions. In addition to stock market manipulation detection, the proposed approach could have potential applications in other fields that involve time series data, such as detecting anomalies in medical signals, and financial statement fraud. Our approach is not limited to the stock market and could be adapted to different types of time series data with the appropriate modification to the generator/discriminator network architecture. However, some limitations of the proposed approach should be considered. One of them is the need for a large amount of computational power to train the model. In addition, the proposed approach requires extensive parameter tuning, especially for the generator network, to ensure that the generated fake samples resemble the normal patterns. Moreover, further work is required to enhance the generalizability of the model to new datasets. In summary, our WALDATA approach provides a promising solution to the problem of detecting unknown manipulation schemes in financial time series. The combination of deep learning and signal processing techniques creates new opportunities for anomaly detection in financial time series, and it is hoped that this research will encourage further exploration and development in this field.

Acknowledgements

References

Abbas, B., Belatreche, A., & Bouridane, A. (2019). Stock price manipulation detection using empirical mode decomposition based kernel density estimation clustering method. *Intelligent Systems and Applications: Proceedings of the 2018 Intelligent Systems Conference (IntelliSys) Volume 2*, 851– 866.

Allen, F., & Gale, D. (1992). Stock-price manipulation. *The Review of Financial Studies*, *5*(3), 503–529.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Cao, Y., Li, Y., Coleman, S., Belatreche, A., & McGinnity, T. M. (2014). Detecting price manipulation in the financial market. *2014 IEEE Conference on Computational Intelligence for Financial Engineering Economics (CIFEr)*, 77–84. https://doi.org/10.1109/CIFEr.2014.6924057

Cao, Y., Li, Y., Coleman, S., Belatreche, A., & McGinnity, T. M. (2015). Adaptive hidden markov model with anomaly states for price manipulation detection. *IEEE Transactions on Neural Networks and Learning Systems*, *26*, 318–330. https://doi.org/10.1109/TNNLS.2014.2315042

Chawla, S., & Gionis, A. (2013). K-means-: A unified approach to clustering and outlier detection. *SDM*.

Chullamonthon, P., & Tangamchit, P. (2023). Ensemble of supervised and unsupervised deep neural networks for stock price manipulation detection. *Expert Systems with Applications*, *220*, 119698. https://doi.org/https://doi.org/10.1016/j.eswa.2023.119698

Diaz, D., Theodoulidis, B., & Sampaio, P. R. F. (2011). Analysis of stock market manipulations using knowledge discovery techniques applied to intraday trade prices. *Expert Syst. Appl.*, *38*, 12757– 12771.

Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recogn. Lett.*, *27*, 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

Gallegati, M. (2012). A wavelet-based approach to test for financial market contagion [1st issue of the Annals of Computational and Financial Econometrics Sixth Special Issue on Computational Econometrics]. *Computational Statistics Data Analysis*, *56*, 3491–3497. https://doi.org/https://doi.org/10.1016/j.csda.2010.11.003

Golmohammadi, K., Zaïane, O., & Diaz, D. (2015). Detecting stock market manipulation using supervised learning algorithms. *DSAA 2014 - Proceedings of the 2014 IEEE International Conference on Data Science and Advanced Analytics*, 435–441. https://doi.org/10.1109/DSAA.2014.7058109

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., & Bengio, Y. (2020). Generative adversarial networks. *Commun. ACM*, *63*, 139–144. https://doi.org/ 10.1145/3422622

Lahmiri, S. (2014). Wavelet low- and high-frequency components as features for predicting stock prices with backpropagation neural networks. *Journal of King Saud University - Computer and Information Sciences*, *26*, 218–227. https://doi.org/https://doi.org/10.1016/j.jksuci.2013.12.001

Langi, A. Z. R., Pitara, S. W., & Kuspriyanto. (2012). Stock prices trends analysis using wavelet transform. *2012 International Conference on Cloud Computing and Social Networking (ICCCSN)*, 1–4.

Leangarun, T., Tangamchit, P., & Thajchayapong, S. (2018). Stock price manipulation detection using generative adversarial networks. *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2104–2111. https://doi.org/10.1109/SSCI.2018.8628777

Leangarun, T., Tangamchit, P., & Thajchayapong, S. (2021). Stock price manipulation detection using deep unsupervised learning: The case of thailand. *IEEE Access*, *9*, 106824–106838. https://doi.org/10.1109/ACCESS.2021.3100359

Lin X, V. D. T. J. (2012). *High frequency trading measurement, detection and response.* Credit Suisse.

Lobster project, atlanta, ga, usa. limit order book. (n.d.). https://lobsterdata.com/info/DataSamples.php

Mallat, S. G. (1989). A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *11*, 674–693. https://doi.org/10.1109/34.192463

Mao, X., Li, Q., Xie, H., Lau, R. Y. K., Wang, Z., & Smolley, S. P. (2017). Least squares generative adversarial networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2813– 2821. https://doi.org/10.1109/ICCV.2017.304

Mongkolnavin, J., & Tirapat, S. (2009). Marking the close analysis in thai bond market surveillance using association rules. *Expert Systems with Applications*, *36*, 8523–8527. https://doi.org/https://doi. org/10.1016/j.eswa.2008.10.073

Nanex. (2022). Whac-a-mole is manipulation. Retrieved%20from%20http://www.nanex.net/aqck2/3598.

Ögüt, H., Doganay, M. M., & Aktas, R. (2009). Detecting stock-price manipulation in an emerging market: The case of turkey. *Expert Syst. Appl.*, *36*, 11944–11949.

Radford, A., Metz, L., & Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, *abs/1511.06434*.

Rizvi, B., Belatreche, A., & Bouridane, A. (2019). A dendritic cell immune system inspired approach for stock market manipulation detection. *2019 IEEE Congress on Evolutionary Computation (CEC)*, 3325–3332. https://doi.org/10.1109/CEC.2019.8789938

Rizvi, B., Belatreche, A., Bouridane, A., & Watson, I. (2020). Detection of stock price manipulation using kernel based principal component analysis and multivariate density estimation. *IEEE Access*, *8*, 135989–136003. https://doi.org/10.1109/ACCESS.2020.3011590

Shyu, M.-L., Chen, S.-C., Sarinnapakorn, K., & Chang, L. (2003). A novel anomaly detection scheme based on principal component classifier.

Sridhar, S., Mootha, S., & Subramanian, S. (2020). Detection of market manipulation using ensemble neural networks. *2020 International Conference on Intelligent Systems and Computer Vision (ISCV)*, 1–8. https://doi.org/10.1109/ISCV49265.2020.9204330

Uslu, N. C., & Akal, F. (2022). A machine learning approach to detection of trade-based manipulations in borsa istanbul. *Computational Economics*, *60*, 25–45. https://doi.org/10.1007/s10614-021-10131-8

Wang, Q., Xu, W., Huang, X., & Yang, K. (2019). Enhancing intraday stock price manipulation detection by leveraging recurrent neural networks with ensemble learning. *Neurocomputing*, *347*, 46–58. https: //doi.org/https://doi.org/10.1016/j.neucom.2019.03.006

Author biography

Khaled Safa received the Engineer degree in computer Science from Ibn Khaldoun University, Tiaret, Algeria, in 2005, and the magister degree in Computer Science from the National Institute of Telecommunications and Information and Communication Technologies, Oran, Algeria, in 2009. He is currently pursuing the PhD degree in deep learning and bio-inspired models for automatic detection of stock markets manipulation at the Computer
Science department of Constantine2 University. His current interests include deep learning and artificial intelligence in stock market data analysis.

Ammar belatreche (Member, IEEE) received the Ph.D. degree in computer science from Ulster University, Derry, U.K. He joined Northumbria University, in May 2016. He was a Research Associate with the Intelligent Systems Research Centre (ISRC) and a Lecturer in computer science with the School of Computing and Intelligent Systems, Ulster University.He is currently an Associate Professor in computer science and a Program Leader with the M.Sc Advanced Computer

Science, Department of Computer and Information Sciences. He is also a member of the Computational Intelligence and Visual Computing (CIVC) Research Group. He has extensive experience across academic and research and development. He has led a number of research and consultancy projects. He has successfully supervised/co-supervised eight Ph.D. students to completion. His research interests include machine learning and AI, bio-inspired intelligent systems, structured and unstructured data analytics, capital markets engineering, and image processing and understanding. He is a Fellow of the Higher Education Academy. He was a program committee member of several international conferences and journals. He serves as an Associate Editor for Neurocomputing (Elsevier) He served as a reviewer for several international conferences and journals

Salima Ouadfel received her State Engineer degree, Magister degree in Computer Science from Mentouri University in Constantine and she received a PhD in Computer Science from the University of Batna. She is currently Associate Professor at Abdelhamid Mehri University, Constantine2. Her current research includes nature inspired metaheuristics, image processing, data mining, Bioinformatics and deep learning

October 5, 2023