ESG UQÀM

Centre d'intelligence en surveillance des marchés financiers

Les cahiers de recherche du CISMF
CISMF Research Paper Series

Deep Unsupervised Anomaly Detection in High-Frequency Markets

Cédric Poutré [a], Didier Chételat[b] & Manuel Morales[a]

[a] Department of Mathematics and Statistics, Université de Montréal, Canada
[b] Canada Excellence Research Chair in Data Science for Real Time Decision Making, Polytechnique Montréal, Canada

Juillet / July 2023

Deep Unsupervised Anomaly Detection in High-Frequency Markets

Cédric Poutré[1], Didier Chételat[2], and Manuel Morales[1]

[1] *Department of Mathematics and Statistics, Université de Montréal*

[2] *Canada Excellence Research Chair in Data Science for Real Time Decision Making, Polytechnique Montréal*

July 6, 2023

Abstract

Inspired by recent advances in the deep learning literature, this article introduces a novel hybrid anomaly detection framework specifically designed for limit order book (LOB) data. A modified Transformer autoencoder architecture is proposed to learn rich temporal LOB subsequence representations, which eases the separability of normal and fraudulent time series. A dissimilarity function is then learned in the representation space to characterize normal LOB behavior, enabling the detection of any anomalous subsequences out-of-sample. We also develop a complete trade–based manipulation simulation methodology able to generate a variety of scenarios derived from actual trade–based fraud cases. The complete framework is tested on LOB data of five NASDAQ stocks in which we randomly insert synthetic quote stuffing, layering, and pump-and-dump manipulations. We show that the proposed asset-independent approach achieves new state-of-the-art fraud detection performance, without requiring any prior knowledge of manipulation patterns.

Keywords: Limit order book; Time series anomaly detection; Deep learning; Trade–based manipulation; Dissimilarity model; Unsupervised learning

1 Introduction

Exchange regulators, who constantly monitor markets to unveil potential manipulations, traditionally perform their investigation manually. When a potentially fraudulent event, or sequence of events, is detected by the automated system, market analysts have the responsibility to carry on the necessary research and analysis to conclude whether or not there has been a violation to the exchange's rules and regulations. The enormous volume of orders means that this task is especially laborious, and investigations can take years.[1] A first solution to this problem is to implement rule–based systems that can automatically flag orders as suspicious. In fact, current market regulators' systems are based on deterministic rules inferred from a set of known delinquent patterns defined by experts (Golmohammadi et al. [21]), on which we shall return. However, such systems have seen limited success in practice, as it is difficult to completely formalize abnormal behaviors by a rule–based system, because defining all anomalies in a trading context is not realistically feasible.

A more promising avenue for this kind of problem is based in machine learning techniques, which have seen a lot of success in a variety of real-world applications (Pang et al. [40]). Moreover, the static nature of rule–based systems is fundamentally ill-matched with the dynamic nature of financial markets, where fraudulent patterns might be constantly evolving with the market (Lin [33]). In contrast, machine learning approaches can dynamically learn unusual order patterns over time, adapting to evolving market conditions. But the current literature on machine learning in financial market manipulation lacks generality, as only very limited sets of features and/or fraud types are studied simultaneously. A more generic framework able to detect several types of fraudulent patterns by utilizing a larger set of information would be valuable (Khodabandehlou and Golpayegani [26]). We aim to fill this gap by proposing a new model based on recent state-of-the-art methods in the deep learning literature capable of managing multivariate time series. Furthermore, previous papers rely on repeating the same limited sets of fraudulent orders to evaluate their methods, probably overestimating their capabilities. We also address that problem with a more exhaustive simulation approach generating further representative sets of trade–based manipulations. Hence, the detection results presented in this paper are more faithful to what could be achieved in practice.

The ultimate objective of any financial market fraud detection system, whether human, rule or

[1] SEC's Division of Enforcement (accessed March 18, 2023).

machine learning–based, is the detection of all *trade–based manipulations*. Trade–based manipulations are defined as "[...] a type of behavior [that] consists of effecting transactions or orders to trade which (a) give, or are likely to five a false or misleading impression as to the supply of, or demand for, or as to the price of one or more [qualifying investments] or (b) secure the price of one or more such investments at an abnormal or artificial level."[2] Multiple manipulation schemes fit that description: advancing the bid, reducing the ask, wash sales, marking the close, pump-and-dump, layering/spoofing, quote stuffing, and so on, all with their respective footprint.[3] In this context, anomalies are sequences of market events that are out of the ordinary, and that could potentially be associated with trade–based manipulations. Market events include new order submissions, cancellations or modifications of a past order, and executions. All of which are recorded sequentially in central limit order books (LOBs), forming a collection of outstanding limit orders. The LOB depicts the prices at which market participants are willing to buy and sell a given instrument, as well as the volume available at each price point, at any given time during trading hours. We note that, although frauds are anomalous in well regulated markets, not all anomalies are necessarily frauds. For example, unusual external events might trigger perfectly legal, but unusual, behavior on markets.

The ideal situation would be to develop a supervised machine learning model to classify orders as part of a manipulation tactic. Unfortunately, precisely because investigations often take years, and fraud methods evolve quickly, most markets have too few examples of such trading activities to reasonably think about using supervised machine learning methods. A more realistic alternative and useful approach would be to develop unsupervised anomaly detection techniques for financial markets. Such methods could flag specific subsequences of orders as suspicious, which could then be further investigated by market regulators, reducing the burden of their analysis. This challenge has two objectives. First and foremost, finding a suitable unsupervised anomaly detection method that can flag most, if not all, true cases of trade–based manipulation as anomalous, i.e., high recall. Second, the method should report as few non-fraudulent anomalies as possible, i.e., high precision, to limit the costs associated with the analysis non-fraudulent orders.

[2]Financial Conduct Authority, MAR 1.6.1 Market abuse (manipulation transactions) FCA Handbook (accessed March 18, 2023).

[3]See Siering et al. [53] for a complete taxonomy of financial market manipulations.

In this paper, we propose a novel approach specifically tailored to high-frequency financial markets that performs better than competing methods. The approach is twofold. First, an unsupervised autoencoder based on the Transformer architecture of Vaswani et al. [57] is trained on LOB–based features specifically built to capture a multitude of trade–based manipulations. We empirically show that the autoencoder learns temporal representation vectors useful in characterizing LOB subsequences, thus easing the separability of fraudulent patterns from regular subsequences. Second, a discriminative model estimates the boundary between normal and abnormal autoencoder representations, creating a dissimilarity function that enables the detection of fraudulent orders out-of-sample. This hybrid approach reduces the gap between state-of-the-art deep learning methods, and financial market fraud detection research. We show that our method, which does not make use of any fraud examples, tends to both capture all true frauds, and has lower false positive rate than competing unsupervised methods on the LOBSTER data set (Huang and Polak [23]). The ability of the framework to detect diverse types of manipulations is an important step in the advancement of stock market fraud detection literature (Khodabandehlou and Golpayegani [26]). Hence, we also propose a more exhaustive trade–based manipulation simulation methodology able to generate multiple fraud scenarios, setting us further apart from previous literature, and rendering our results more reliable. Finally, for the first time in the literature, we also quantitatively study the complexity of detecting certain types of trade–based manipulations, thus providing a new comparative standard other than the conventional anomaly detection metrics.

The paper is divided as follows. Section 2 introduces the literature on machine learning and deep anomaly detection models, with a focus on time series methods. It also provides a review of trade–based manipulations detection techniques put forward in the financial literature. Section 3 presents the unsupervised hybrid deep learning framework proposed to capture anomalous behavior in LOB time series. Section 4 details the financial data used, and describes three popular trade–based manipulation techniques that are then simulated to quantify the framework's performance. The section ends by presenting the proposed LOB features to capture fraudulent patterns. Section 5 carries out the numerical experiments and analyzes the effectiveness of the methodology on simulated frauds, and Section 6 concludes the paper.

2 Literature review

Anomaly detection is an active subfield of machine learning successful in a plethora of industrial applications (see Agrawal and Agrawal [3] for a comprehensive overview). Following the nomenclature of Blázquez-García et al. [5] and Chalapathy and Chawla [8], the problem addressed in this paper can be classified as multivariate collective outlier detection, a niche outlier type which limits the pool of potential detection techniques. Indeed, we are interested in identifying sets of orders that jointly behave unusually but are otherwise normal on an individual basis. Blázquez-García et al. [5] describe two types of subsequence outlier detection methods proposed in the literature: model–based, and dissimilarity–based. The first family of techniques tries to find subsequences that strongly deviate from a model's expected value, either by prediction or by estimation, whereas the second family aims to detect subsequences' representations that stray from a reference of normalcy. Predicting the stock market is a notoriously challenging task (Gandhmal and Kumar [19]), meaning that prediction–based models would be hazardous for anomaly detection on LOB data. This leaves only estimation and dissimilarity methods as suitable anomaly detectors in this context. The proposed framework falls in the second family, for which the literature is notably scarce.

Representing time series of LOBs necessitates a great ability in capturing temporal and spatial information, because of its complexity. Deep learning models have made great strides in that sense, especially for large data sets, which explains the growing interest in deep anomaly detection methods. Recent techniques include autoencoders (AEs): MSCRED (Zhang et al. [62]), OmniAnomaly (Su et al. [55]), USAD (Audibert et al. [4])). Generative adversarial networks (GANs): MAD-GAN (Li et al. [32]). Graph neural networks (GNNs): MTDAD-GAT (Zhao et al. [63]), GDN (Deng and Hooi [15]). Deep one-class networks: THOC (Shen et al. [52]). Most papers follow an unsupervised approach based on recurrent networks, e.g., long short-term memory networks (LSTMs, Hochreiter and Schmidhuber [22]). The unsupervised approach hinges on the assumption that anomalies in the data are either rare or absent, and that models are learning usual system behaviors well enough to distinguish outliers out-of-sample. The Transformer architecture (Vaswani et al. [57]) has also started to appear in the time series anomaly detection literature (Meng et al. [35], Xu et al. [61]), motivated by its success in natural language processing tasks, and its greater memory and parallelization

capabilities compared to recurrent models. Finally, hybrid approaches combining self-supervised representation learning and one-class classifiers, such as kernel density estimation (KDE, Parzen [41]), one-class support vector machine (OC-SVM, Schölkopf et al. [51]) or k-nearest neighbors (kNN, Cover and Hart [14]) recently achieved state-of-the-art unsupervised anomaly detection performance on the visual domain (Reiss et al. [43, 44]; Sohn et al. [54]). Inspired by these recent advances, we propose an unsupervised hydrid methodology combining the temporal context representation capabilities of Transformers, and the effectiveness of statistical discriminative models, to detect abnormal behavior in LOB time series.

This paper first and foremost fits in the financial market anomalies literature. Although research on machine learning–based anomaly detection is plentiful, literature on machine learning for financial market anomalies is relatively scarce, and most deep learning methods presented above have not been investigated in this context. This paper aims to reduce the gap between deep learning research and stock price anomalies research. Ögüt et al. [64] were the first to investigate data mining algorithms (support vector machines and multilayer perceptron (MLP)) in the context of daily stock price manipulation detection. Using a labeled data set from January 1995 to March 2004, on an index traded on the Istanbul Stock Exchange, they find that data mining algorithms perform better in terms of total classification accuracy compared to multivariate statistical techniques (discriminant analysis and logistic regression). Their study utilizes daily data, which does not allow precise detection of the manipulations, meaning that regulators would have to analyze the complete daily trading data to find the fraudulent trades. Lastly, because of supervised learning algorithms, their methodology relies on labeled data. This is not desirable in practice, since it would require a substantial effort to generate a usable data set, and available fraud cases are very limited. This is the main factor driving the use of unsupervised models since they do not require any label. Furthermore, supervised models are only capable of detecting known patterns in the data, rendering them useless in unveiling emerging trade–based manipulations, or when market data's distribution inevitably drifts. These limitations support the adoption of unsupervised learning methods (Khodabandehlou and Golpayegani [26]).

Diaz et al. [16] were the first to introduce intraday trading data and an "open box" approach, in the form of decision trees, interpretable by market regulators to help detect trade–based manipulations.

They find that the average traded volume is lower during manipulation periods, whereas liquidity, returns, and volatility are higher than usual. This corroborates with the empirical findings of Aggarwal and Wu [2]. Both studies relied on a limited set of trade–based manipulations enforced by the SEC that are unclear on the manipulation type, and their exact time of occurrence.

Cao et al. [6] propose a hidden markov model to detect certain trade–based manipulations: spoofing, pump-and-dump, and "others". From a set of four mid-price features, the proposed Adaptive Hidden Markov Model with Anomaly States (AHMMAS) can outperform standard machine learning (OC-SVM, kNN), and other statistical methods, on simulated anomalies injected in a LOB data set from the LOBSTER project (Huang and Polak [23]). This approach has some important drawbacks. The biggest one being the curse of dimensionality faced by the model. Indeed, two hidden states, one normal and one anomalous, are created for each feature, bringing the total number of hidden states to 2^n, for n the number of features. Given that the Viterbi algorithm (Forney [18]) is employed to determine the most probable series of hidden states in a time series of length T, the anomaly detection complexity is $\mathcal{O}(T \times 2^{2^n})$. We argue that more features, such as: market event interarrival time, cancellation volume, trade volume, LOB volume, LOB prices, etc., are necessary to capture a larger range of fraudulent patterns, which is problematic for AHMMAS. On the other hand, deep learning methods have shown great performance in high-dimensionality problems (Wang et al. [60]) and are thus better suited to generalize trade-fraud detection to multiple patterns. In a second paper, Cao et al. [7], the authors work directly with the orders, rather than the LOB, and use the OC-SVM algorithm to detect spoofing and quote stuffing orders from their volume, price, and duration, achieving state-of-the-art detection capabilities at the time, in terms of area under the receiver operating characteristic curve. But the temporal context of the orders is ignored, so that significantly large orders, rapid orders sent by high-frequency traders, or trades walking the book, may be wrongly classified as anomalous. We propose to include such context so that those types of orders can be safely ignored when pertinent.

A series of studies are applying unsupervised hybrid models to first learn a representation of price time series features, and then cluster these representations to detect abnormal orders. Abbas et al. [1] use empirical mode decomposition (EMD), Close and Kashef [12]; Rizvi et al. [45] employ the dendritic cell algorithm, and Rizvi et al. [47] apply kernel based principal component analysis. All

of which then utilize KDE–based clustering techniques on the time series' representations, and all outperform the hidden markov model of Cao et al. [6]. The simulated fraudulent patterns in Abbas et al. [1]; Cao et al. [6, 7]; Close and Kashef [12]; Rizvi et al. [45, 47] are all fixed and repeated, thus probably leading to an overestimation of their methodologies' performances. Since we aim to build a more versatile and robust anomaly detection model, our set of simulated anomalies used out-of-sample includes more families of trade–based fraud, and they are also stochastically generated, leading to a greater range of fraudulent scenarios. In all these studies, only price features are used to detect potential frauds, which lacks generality to detect non-price related tactics, e.g., quote stuffing (Khodabandehlou and Golpayegani [26]). Furthermore, the chronological order of market events is important to characterize trade–based manipulations. For example, in pump-and-dump, there needs to be price ramping before observing large quantities of executions and cancellations. If that sequence of events is not respected, it is not fraudulent. It is unclear if the methods of Abbas et al. [1]; Close and Kashef [12]; Rizvi et al. [45, 47] are able to consider that chronological aspect. Alternatively, deep recurrent and attention models do consider the order in their time series representation.

Newer studies in financial market anomaly detection have started to focus on deep learning approaches. Leangarun et al. [28] use a supervised MLP to detect stock price manipulation using level 1 data. They can detect synthetic pump-and-dump cases with an accuracy above 88%, but without much success on synthetic spoofing cases. In a subsequent study, Leangarun et al. [29], employ a deep unsupervised framework based on GANs with LSTM network generators and discriminators utilizing uniformly sampled LOBs as input to achieve close to 70% accuracy out-of-sample on synthetic pump-and-dump activities. In a third paper, Leangarun et al. [30], the authors compare their LSTM-GAN to a LSTM-AE. They find that the AE model outperforms the GAN on synthetic pump-and-dumps, and both unsupervised models can detect five out of six real manipulation cases from the Stock Exchange of Thailand. Chullamonthon and Tangamchit [10] apply a supervised Transformer model to detect both synthetic pump-and-dumps, and the same real fraud cases of Leangarun et al. [30]. The model achieves higher accuracy than the MLP of Leangarun et al. [28]. All these studies only focus on pump-and-pumps. Instead, we are proposing a generic framework able to detect different types of frauds from LOB data, a natural step in the next generation of market manipulation

detectors (Khodabandehlou and Golpayegani [26]). Rizvi et al. [46] employ a MLP-AE trained on stock prices affinity matrices. The learned representations are then clustered with the kernel density estimation–based method proposed in Rizvi et al. [45]. This hybrid approach outperforms the works of Cao et al. [6] and Abbas et al. [1], cementing the idea that deep unsupervised models can learn better representations compared to previous methods. Again, it is unclear how the sequential aspect of trade–based manipulations is considered in this model, and the lack of non-price features is problematic for the detection of various trade–based manipulation types.

3 Methodology

3.1 Problem

We can formalize the problem as follows. Given an out-of-sample time series of length T and dimensionality m, $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\},\ \mathbf{x}_t \in \mathbb{R}^m\ \forall t$, we are interested in finding subsequences of fixed length $k \ll T$, $S = \{S_t = \{\mathbf{x}_{t-k+1}, \mathbf{x}_{t-k+2}, \ldots, \mathbf{x}_t\} \in \mathbb{R}^{k \times m} \mid t = k, \ldots, T\}$, that strongly deviate from normalcy. Hence, we need to predict the series $Y = \{y_k, y_{k+1}, \ldots, y_T\}$, where $y_t \in \{0 : \text{normal},\ 1 : \text{abnormal}\}$ is associated to S_t. In our context, the time series X contains LOB features useful in characterizing and detecting trade–based manipulation patterns, which will be presented in Section 4.

3.2 Approach

Lately, the literature on anomaly detection is seeing a resurgence of interest in one-class classifiers (e.g., OC-SVM, Schölkopf et al. [51], and SVDD, Tax and Duin [56]), applied on representations learned by a higher-level deep neural network, called an encoder. The models are trained end-to-end and achieve state of the art results on many benchmarks (Ruff et al. [48]). Unfortunately, this approach also suffers from collapses into trivial solutions, which makes training difficult. Recently, Sohn et al. [54] have proposed a simple fix, where they suggest to first train an encoder network on some auxiliary task, and separately train the one-class classifier on these representations, as two separate learning steps, achieving impressive results on image benchmarks.

We propose to use the same approach, with some modifications for the financial domain. In

their paper, Sohn et al. [54] focus on contrastive learning on augmentations (such as rotations, or flips) of the original images. Unfortunately, it is difficult to extend their approach to financial time series since possible notions of augmentations are less clear. Indeed, because the semantics of LOB time series are very intricate, blindly applying previously proposed data augmentations breaks their nature, which makes contrastive learning methods difficult. Instead, we propose to train the encoder to solve the typical autoencoding task jointly with some decoder network on selected LOB features.[4] Note that other self-supervised or unsupervised tasks could be employed to train the encoder, such as masked language modeling, or some other prediction task. But masked-autoencoders have been found to have poor anomaly detection capabilities (Reiss et al. [44]), and predicting the stock market is arduous (Gandhmal and Kumar [19]), which drastically increases the complexity of learning efficient representations of LOB time series. For this reason, we prefer to adopt the autoencoding task in this context. After training, the decoder network is discarded, and a separate one-class classifier is trained, keeping the encoder fixed. We now detail each step of the proposed methodology in turn.

3.3 Step 1: Autoencoding

In the first step, we train the deep neural network encoder $\phi_E : \mathbb{R}^{k \times m} \mapsto \mathbb{R}^d$ with parameters θ_E, jointly with another deep neural network $\phi_D : \mathbb{R}^d \mapsto \mathbb{R}^{k \times m}$ with parameters θ_D, called the decoder, where $d \ll k \times m$. The composition of the encoder and decoder networks forms the autoencoder, $\phi_D \circ \phi_E = \phi_{AE} : \mathbb{R}^{k \times m} \mapsto \mathbb{R}^{k \times m}$ with parameters $\Theta = \{\theta_E, \theta_D\}$. The goal of the encoder is to generate a representation vector, $\mathbf{z}_t \in \mathbb{R}^d$, semantically rich enough to summarize the input sequence $S_t \in \mathbb{R}^{k \times m}$, so the decoder is able to reconstruct that same sequence only from $\mathbf{z}_t$, i.e., $\phi_D\big(\mathbf{z}_t = \phi_E(S_t \mid \theta_E) \;\big|\; \theta_D\big) = \phi_{AE}(S_t \mid \Theta) \approx S_t$. The proposed autoencoder is defined in this next part.

[4] The autoencoding task is an unsupervised learning technique which consists of encoding an efficient lower–dimensional representation of the data and then decoding, i.e., reconstructing, the initial data from that representation.

3.3.1 Bottlenecked-Transformer autoencoder

The architecture of the proposed autoencoder mostly follows the Transformer architecture of Vaswani et al. [57], along with some modifications. Figure 1 details the original architecture (left) and the proposed model (right).

Initially proposed in the field of natural language processing, Transformers are deep learning models adopting the encoder-decoder architecture of earlier sequence-to-sequence models. Transformers process entire sequences at once, instead of relying on recursion like deep recurrent models. They do so using self-attention in the "Multi-Head Attention" module (see Figure 1), allowing them to focus, i.e., put more weight, on relevant portions of the input data (timewise and feature-wise) depending on the sequence itself, to create more informative sequence representations. The Transformer encoder consists of N encoding layers, each successively transforming the sequence representation into a final matrix containing the sequence's contextual information called the encoding. The Transformer decoder is then tasked to autoregressively generate an output sequence based on this encoding, without attending the current or future observations of the expected output sequence. We refer the reader to Vaswani et al. [57] for the complete technical description of the Transformer model, as we want to focus on the alterations we make to their model.

As can be seen in Figure 1, the differences with the architecture of Vaswani et al. [57] are the addition of a bottleneck composed of the "flatten" and "linear" blocks, and the removal of the "softmax" layer at the end of the decoder. Similar models have been proposed for sentence embedding (Montero et al. [36]; Wang et al. [60]), and musical style encoding (Choi et al. [9]), but never in the context of anomaly detection. The bottleneck deserves some additional details. After N Transformer-encoder layers, the input matrix $S_t = (\mathbf{x}_{t-k+1}, \ldots, \mathbf{x}_t) \in \mathbb{R}^{k \times m}$ is mapped to a matrix $E_t = (\mathbf{e}_{t1}, \ldots, \mathbf{e}_{tk}) \in \mathbb{R}^{k \times d}$ following the standard Transformer architecture of Vaswani et al. [57]. After that, the bottleneck flattens, i.e., concatenates, this representation matrix into a kd-vector $\mathbf{e}_t$:

$$\mathbf{e}_t = \begin{bmatrix} e_{t1} & \ldots & e_{tk} \end{bmatrix} \in \mathbb{R}^{kd},$$

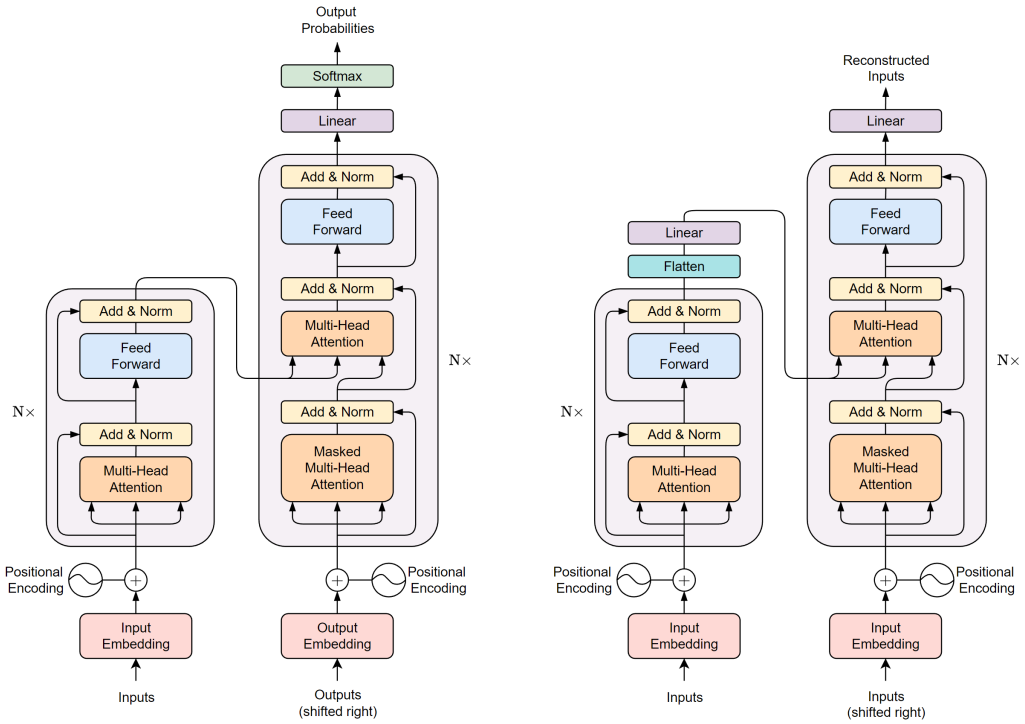

Figure 1: Original Transformer autoencoder of Vaswani et al. [57] (left) and the proposed bottlenecked-Transformer autoencoder (right). The encoder representation matrix in $\mathbb{R}^{k \times d}$ is flattened to a vector in $\mathbb{R}^{kd}$, then passed to a linear layer to reduce the representation further to $\mathbb{R}^d$, before being fed to the decoder.

which is then linearly projected into a d-vector, yielding the representation:

$$\mathbf{z}_t = W\mathbf{e}_t + \mathbf{b} = \phi_E(S_t \mid \theta_E) \in \mathbb{R}^d,$$

where $W \in \mathbb{R}^{d \times kd}$ and $\mathbf{b} \in \mathbb{R}^d$. Because $d \ll k \times m$, information is lost in this bottleneck. In turn, this means the decoder only attends to a limited subset of information summarizing the input sequence, and the encoder is forced to learn a semantically rich temporal context vector $\mathbf{z}_t$.

3.3.2 Autoencoder training

The autoencoder ϕ_{AE} is trained to minimize the L_2 reconstruction loss of any sequence S_t, which is the standard autoencoding task:

$$\mathcal{L}(\Theta) = \mathbb{E}\Big[\big(S_t - \phi_{AE}(S_t \mid \Theta)\big)^2\Big],$$
$$\Theta^* = \arg\min_\Theta \mathcal{L}(\Theta).$$

Supposing a fraud-free data set of N subsequences $D = \{s_1, \ldots, s_N\},\ s_i \in \mathbb{R}^{k \times m}$, the parameters of ϕ_{AE} are estimated by stochastic gradient descent on the empirical measure:

$$\widehat{\Theta} = \arg\min_\Theta \frac{1}{N} \sum_{i=1}^{N} \big|\big|s_i - \phi_{AE}(s_i \mid \Theta)\big|\big|_{\mathrm{F}},$$

for $||\cdot||_{\mathrm{F}}$ the Frobenius norm.

3.4 Step 2: Dissimilarity learning

In the second step, we discard the decoder ϕ_D, and train a OC-SVM on the representations $\mathbf{z}_i = \phi_E(s_i \mid \widehat{\theta}_E),\ i = 1, \ldots, N$, learned by the encoder on the fraud-free set D. The OC-SVM of Schölkopf et al. [51] is a novelty detection algorithm extending the support vector machine algorithm (SVM, Cortes and Vapnik [13]) to the unsupervised case. It finds a subset of the input space such that a new point drawn from the same distribution as the data will lie outside the subset with arbitrarily small probability, leaving abnormal data points outside, thus allowing us to detect any anomalous encoder representations. We introduce the algorithm here for completeness.

Defining the feature map $\Phi : \mathbb{R}^d \mapsto \mathcal{F}$ such that $\mathcal{F}$ is a space where the dot product in the image of Φ can be obtained by a kernel function, i.e.:

$$k(\mathbf{z}, \mathbf{z}') = \Phi(\mathbf{z}) \cdot \Phi(\mathbf{z}'),$$

the OC-SVM tries to separate the data from the origin in $\mathcal{F}$ with the hyperplane $\big\{\Phi(\mathbf{z}) \,\big|\, \mathbf{w} \cdot \Phi(\mathbf{z}) = \rho\big\}$ with maximum margin. If no such hyperplane exists, slack variables are added to some mapped representations (the support vectors), while also allowing points in the origin's half space (the outliers). That is, the OC-SVM solves the quadratic program:

$$\min_{\boldsymbol{w} \in \mathcal{F}, \boldsymbol{\xi} \in \mathbb{R}^N, \rho \in \mathbb{R}} \frac{1}{2}||\mathbf{w}||^2 + \frac{1}{\nu N} \sum_{i=1}^{N} \xi_i - \rho$$
$$\text{s.t. } \mathbf{w} \cdot \Phi(\mathbf{z}_i) \geq \rho - \xi_i,\ \xi_i \geq 0, \quad \forall i = 1, \ldots, N,$$

where the hyperparameter $\nu \in (0, 1)$ controls the upper bound on the fraction of outliers, and ξ_is are the slack variables. We use the radial basis function:

$$k_\gamma(\mathbf{z}, \mathbf{z}') = \exp(-\gamma\|\mathbf{z} - \mathbf{z}'\|^2)$$

as the kernel with hyperparameter $\gamma > 0$. The optimization problem can be solved efficiently with any standard quadratic programming solver. The OC-SVM decision function has a solution of the form:

$$f(\mathbf{z}) = \text{sign}\left(\sum_{i=1}^{N} \widehat{\alpha}_i k_\gamma(\mathbf{z}, \mathbf{z}_i) - \widehat{\rho}\right) \in \{-1,1\},$$

where outliers have a negative value. The $\widehat{\alpha}_i$ and $\widehat{\rho}$ estimated on the representations $\mathbf{z}_i$ are kept for the dissimilarity function detailed in the next subsection.

3.5 Step 3: Predicting

In the third step, once both the encoder and the OC-SVM are trained, one makes a pass over the out-of-sample subsequences of set S, and record their dissimilarity value, which is a slight modification of the OC-SVM decision function:

$$\text{dissimilarity}(S_t) = \widehat{\rho} - \sum_{i=1}^{N} \widehat{\alpha}_i k_\gamma\Big(\phi_E\big(S_t \mid \widehat{\theta}_E\big), \phi_E\big(s_i \mid \widehat{\theta}_E\big)\Big) \in \mathbb{R},\ \forall S_t \in S.$$

This function quantifies how far the representation of the subsequence S_t is from the support learned by the OC-SVM. In other words, the greater the dissimilarity, the further the subsequence is from normal data, and the more anomalous it is. Finally, we classify a subsequence S_t as anomalous if

$$\text{dissimilarity}(S_t) > \tau,$$

for some threshold $\tau \in \mathbb{R}$ controlling the sensitivity of the algorithm, i.e.,

$$\widehat{y}_t = \mathbb{1}_{\{\text{dissimilarity}(S_t)>\tau\}},\ \forall S_t \in S,$$

for $\mathbb{1}_{\{\cdot\}}$ the indicator function. Previous financial anomaly detection papers (e.g., Leangarun et al. [30]) have used the L_2 reconstruction loss of autoencoders for this task, $\left\|S_t - \phi_{AE}(S_t \mid \widehat{\Theta})\right\|_{\mathrm{F}}$, but as we will show, this leads to suboptimal detection performance on our data set. Indeed, the reconstruction loss is highly volatile in time so true anomalies are difficult to detect, generating a high false positive ratio. Our approach instead utilizes the dissimilarity function, dissimilarity(S_t), to quantify the abnormality of S_t, yielding better results because normal data is more easily discernible in the representation space. These aspects will be discussed in detail in Section 5. Deep learning models have not been well explored for collective anomalies, unlike point anomalies (Pang et al. [40]), and therefore our paper also contributes to fill this gap in the time series literature with a direct application to LOB data. Figure 2 presents the overall proposed anomaly detection model combining the bottlenecked-Transformer encoder, and the OC-SVM algorithm.

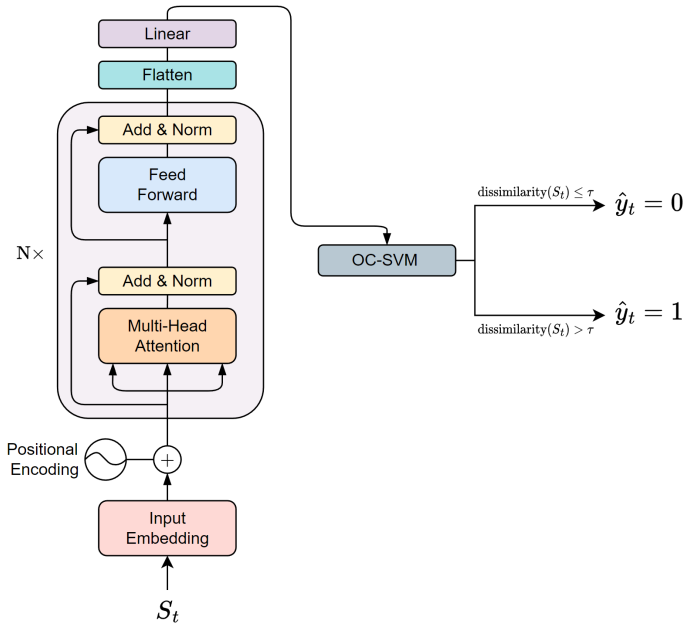

Figure 2: Proposed dissimilarity–based anomaly detection framework.

4 Data

As explained in Section 3, our approach is unsupervised, which means that we do not need explicit examples of fraud to train and run our model. However, it is useful to have such examples to evaluate our algorithm and compare it to alternative approaches. To do so, we will use a standard strategy of using a fraud-free data set of orders, and artificially add in examples of fraud in the data set. Note that although these examples of fraud will only be present in the test data, our approach will not be aware of their location, or amount, so that the methodology remains unsupervised. We now describe the data in further details.

4.1 Base data set

Our base data comes from the LOBSTER project (Huang and Polak [23]). It contains the LOB level 1 (L1) data of five NASDAQ stocks: Amazon (ticker: AMZN), Apple (ticker: AAPL), Google (ticker: GOOG), Intel (ticker: INTL), and Microsoft (ticker: MSFT) on June 21, 2012. The LOBSTER project data has also been used in most financial market fraud detection papers (Abbas et al. [1]; Cao et al. [6, 7]; Close and Kashef [12]; Leangarun et al. [28]; Rizvi et al. [45, 46, 47]), so it is a good point of comparison. The data is split into two files, one for market events, and one for rebuilt LOBs. Table 1 details the data, and Table 2 provides some descriptive stock statistics. It is from that raw data that LOB features are created to be used in the model of Section 3.

Table 1: Description of the information contained in LOBSTER project's messages and LOBs files.

Messages		**LOBs**	
Variable	**Description**	**Variable**	**Description**
Time	Nanoseconds past midnight	Best bid price	Best buying price
Type	1: Submission new limit order	Best bid size	Total number shares available at bid price
	2: Partial cancellation limit order	Best ask price	Best selling price
	3: Deletion limit order	Best ask size	Total number shares available at ask price
	4: Execution visible limit order		
	5: Execution hidden limit order		
OrderID	Unique order reference number		
Size	Number shares of order		
Price	Price of order		
Direction	-1: Sell limit order		
	1: Buy limit order		

Table 2: Descriptive statistics of LOBSTER stocks on June 21, 2012.

Statistic | Stock	AAPL	AMZN	GOOG	INTC	MSFT
Quotes	54,818	27,845	24,368	202,231	205,695
Trades	34,990	11,419	11,678	32,483	33,414
Cancels/Quotes	52.34%	65.54%	55.14%	84.20%	83.76%
Std. Mid-price	2.99	1.36	4.28	0.27	0.29
Avg. Bid-Ask Spread (bps)	2.66	6.10	5.45	4.92	4.40
Avg. Best Bid Size	227.20	249.34	172.11	17,194.81	14,965.07
Avg. Best Ask Size	147.19	145.36	134.99	14,360.18	15,796.58
Avg. Order Size	88.30	95.90	81.89	503.46	600.43
Avg. Trade Size	81.46	71.00	60.47	322.37	323.95

4.2 Synthetic manipulations

As mentioned earlier, reported trade–based manipulation cases are rare, hence most studies have instead simulated some, and inserted them back in their initial fraud-free data set. We also follow that methodology, but we significantly increase the complexity of fraudulent patterns, as explained in Section 2. Not only do we simulate more families of trade frauds, but we also stochastically generate them so that multiple scenarios are included in the study, while making sure not to denature the manipulations. Previous papers repeat the same fixed, limited, set of orders, which probably results in an overestimation of their model's true detection potential. Three distinct trade–based manipulation types are studied: pump-and-dump, layering, and quote stuffing. 50 trade–based manipulations of each type are randomly added per stock. 30 seconds of fraud-free data before and after each manipulation are kept to create the test set containing all frauds, so that the vast majority of the test data is also fraud-free, keeping a low anomaly ratio. The remainder of the LOB data is split to form the training set, representing 70% of the remainder, and the validation set, representing the last 30%. Given the large discrepancy in the data quantity between Amazon, Apple, Google and Intel, Microsoft (see Table 2), we oversample the first group of stocks in the training and validation data sets to get a more balanced representation. The training set is used to train the bottlenecked-Transformer autoencoder on the L_2 reconstruction loss detailed in Section 3.3.2, and the validation set is needed to select the optimal parameters $\widehat{\Theta}$ generating the lowest loss on that set. Then, the training and validation sets are concatenated to form a single fraud-free set on which the OC-SVM of Section 3.4 is trained to learn the dissimilarity function. The test set is only utilized to evaluate the methodology's out-of-sample performance. We now describe in detail

the simulated trade–based manipulations.

4.2.1 Pump-and-dump

The U.S. Securities and Exchange Commission (SEC) defines pump-and-dumps as: "[...] schemes [that] have two parts. In the first, promoters try to boost the price of a stock [...]. Once the stock price has been pumped up, fraudsters move on to the second part, where they seek to profit by selling their own holdings of the stock, dumping shares into the market."[5]

We closely follow the procedure of Chullamonthon and Tangamchit [11] to simulate pump-and-dump patterns. We replicate their "low degree" pump-and-dump scenario in which a price change of 3–4% occurs, with an increase in both bid and ask volumes of 25–100% during a pump period of one second. Following that price pump, a dumping stage of three seconds sees the same increased level of canceled bid volume and matched volume, bringing the LOB prices to their initial level. They also propose two other, more aggressive, scenarios, but we only replicate the lowest degree of fraud to be more conservative. The different scenarios of Chullamonthon and Tangamchit [11] are based on cryptocurrencies pump-and-dumps found in Kamps and Kleinberg [25], and on an event identified by Nanex Research, a Chicago–based firm specializing in high-frequency trading data; the Westinghouse Air Brake Technologies Corp stock's (NYSE:WAB) price jumped 8% in the span of one second, and then dropped back close to its initial value after three seconds on December 14, 2011 (Nanex Research [38]). Table 3 lists the pump-and-dump characteristics and the ranges used to randomly simulate the manipulations, and Figure 3 presents a simulated pump-and-dump pattern.

Table 3: Range of randomly simulated pump-and-dump scenarios.

Characteristic	**Total Price Increase**	**Non Bona Fide Order Size**	**Pump Duration**	**Dump Duration**
Range	[3, 4]%	[1.25, 2]× Average L1 Volume	[750, 1,250] ms	[2,250, 3,750] ms

4.2.2 Layering

The Investment Industry Regulatory Organization of Canada (IIROC) defines layering as: "[...] [the act of] placing a *bona fide* order on one side of the market while simultaneously "layering" orders in

[5]https://www.investor.gov/protect-your-investments/fraud/types-fraud/pump-and-dump-schemes (accessed May 23, 2023).

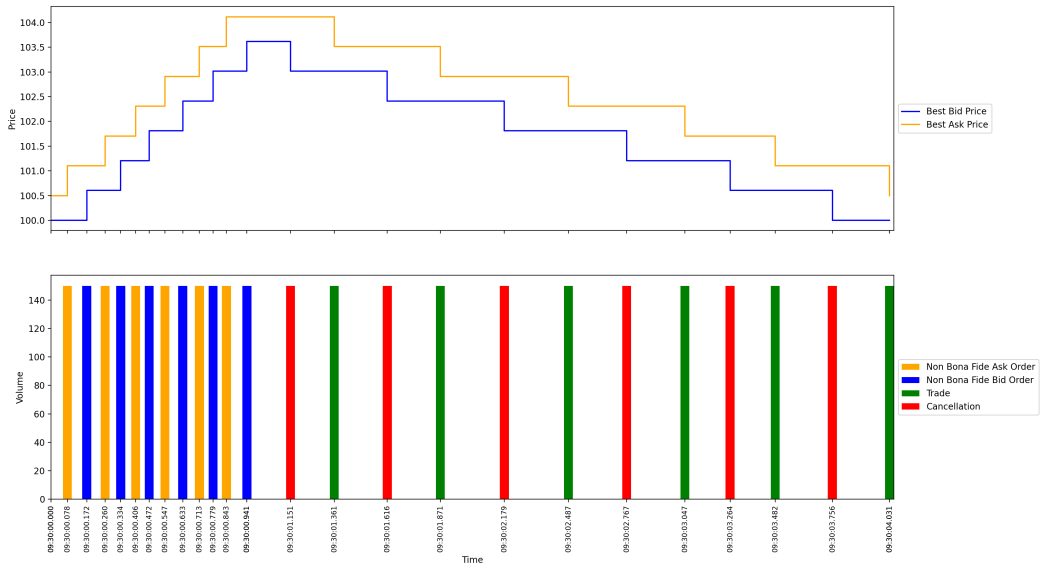

Figure 3: Toy example of "low degree" pump-and-dump manipulation.

the consolidated market display on the other side of the market without intention to trade [...] as [to induce] a false or misleading appearance of trading activity or artificial price. In this case, the purpose of the "layering" is to "bait" other market participants to react and trade with the *bona fide* order on the other side of the market at an artificial price." (IIROC [24])

On April 30, 2014, the SEC reported layering activities that occurred on the stock W.W. Grainger (NYSE: GWW) on June 4, 2010.[6] Figure 4 details the layering orders listed in the official document. At 11:08:55.152, the trader placed a *bona fide* order to sell 1,000 units of the stock at $101.34 per share when the best bid was at $101.27 and the best ask at $101.37. From 11:08:55.164 to 11:08:55.323, the trader sent 11 orders to buy GWW at increasing prices, totaling a volume of 2,600 shares, pushing the best bid price to $101.33. Another market participant, deceived by the layering orders, traded against the *bona fide* sell order of 1,000 shares at 11:08:55.333. At 11:08:55.932, the trader canceled all the *non bona fide* bid orders, and the L1 prices returned to their initial value.

From this case, multiple stylized facts of interest can be used to replicate layering patterns: the *bona fide* order is placed around 3bps inside the bid-ask spread, a sequence of 11 *non bona fide* orders are placed at a rate of one per 14ms, the *non bona fide* orders push the price around 6bps

[6]https://nj.gov/oag/newsreleases14/Hold-and-Tobias-Consent-Order-05-02-14.pdf (accessed March 13, 2023).

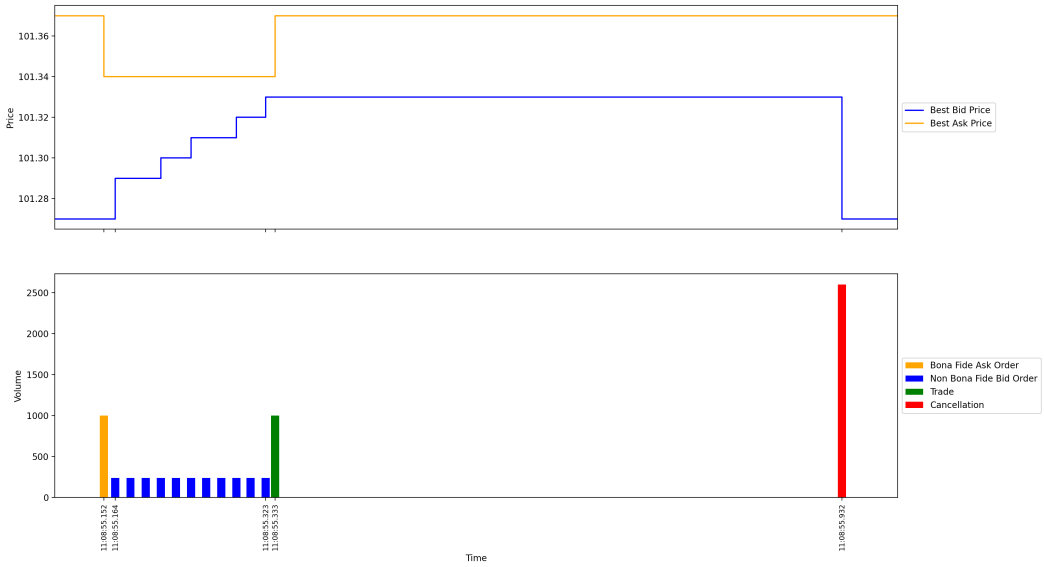

Figure 4: Recreation of the layering activities found by the SEC in NYSE:GWW on June 4, 2010.

away from its initial value, a trade occurs rapidly after a *non bona fide* order and their placement are stopped, all the *non bona fide* orders are cancelled less than 600ms after the trade, and finally, the *bona fide* order is 2.6 times smaller than the total volume of the *non bona fide* orders. Another important stylized fact is that spoofing orders are on average 5.6 times larger than typical orders (Lee et al. [31]).[7] Table 4 lists layering characteristics and the ranges used to randomly simulate the manipulations.

Table 4: Range of randomly simulated layering scenarios.

Characteristic	**Bona Fide Order Side**	**Bona Fide Order Price**	**Bona Fide Order Size**	**Nb. Non Bona Fide Orders**	**Non Bona Fide Interarrival Time**
Range	$\{Bid, Ask\}$	$Bidprice + (0, 3)$ *bps* if Bona Fide Side $= Bid$ $Askprice - (0, 3)$ *bps* if Bona Fide Side $= Ask$	$[2, 3]\times$ Non Bona Fide Total Size	$[10, 12]$	$[10, 20]$ *ms*
Characteristic	**Non Bona Fide Total Size**	**Non Bona Fide Price Movement**	**Trade Delay**	**Cancelation Delay**	
Range	$[5,\ 6]\times$ Average order size	$(0, 6]$ *bps*	$[5,\ 15]$ *ms*	$[100, 1100]$ *ms*	

[7] Multiple spoofing orders in a sequence generate a layering manipulation.

4.2.3 Quote stuffing

IIROC defines quote stuffing as: "[...] the input by a Participant or Access Person of excessive market data messages with the intent to "flood" systems and create "information arbitrage" opportunities for itself [...]." (IIROC [24]).

Egginton et al. [17] find that quote stuffing occurs often on only one side of the book at a time. They also notice a drastic increase in the new order rate and cancellation rates, a decrease in order size, and an augmentation of order updates slightly inside the spread during quote stuffing periods. Nanex also discovered multiple quote stuffing algorithm imprints. More specifically, they detail the orders sent by Citadel Securities for which a disciplinary action was taken by NASDAQ in 2014.[8] For example, from 13:32:53.029 to 13:33:00.998, Citadel placed 8 to 9 orders to buy 100 shares of NASDAQ:PENN, per millisecond, before immediately canceling them. This caused delays up to 16ms in the U.S. Securities Information Processor, creating arbitrage opportunities (Nanex Research [39]). Nanex has also found that order rates at 10 per millisecond and above will create delays in NYSE's consolidated quotation system (Nanex Research [37]). Although quote stuffing sequences can last thousands of events, we limit their length as to not have a disproportionate number of fraudulent orders in our data set. Considering these stylized facts, Table 5 lists quote stuffing characteristics and the ranges used to randomly create the manipulations. Figure 5 shows a toy example of quote stuffing activity where a high-frequency trader submits, then almost immediately cancels, limit orders inside the bid-ask spread. In the span of six milliseconds, 25 buy limit orders are sent to the market then rapidly canceled, thus sending a total of 50 messages to other market participants.

Table 5: Range of randomly simulated quote stuffing scenarios.

Characteristic	Fraud Side	Nb. Events	Order Rate	Order Size	Order Price
Range	$\{Bid, Ask\}$	$[50, 200]$	[8, 10] / ms	[1, 10]th percentile order volume distribution	$Bidprice + (0, midprice)$ if Fraud Side = Bid $Askprice - (0, midprice)$ if Fraud Side = Ask

[8] NASDAQ Disciplinary Actions against Citadel (accessed March 15, 2023)

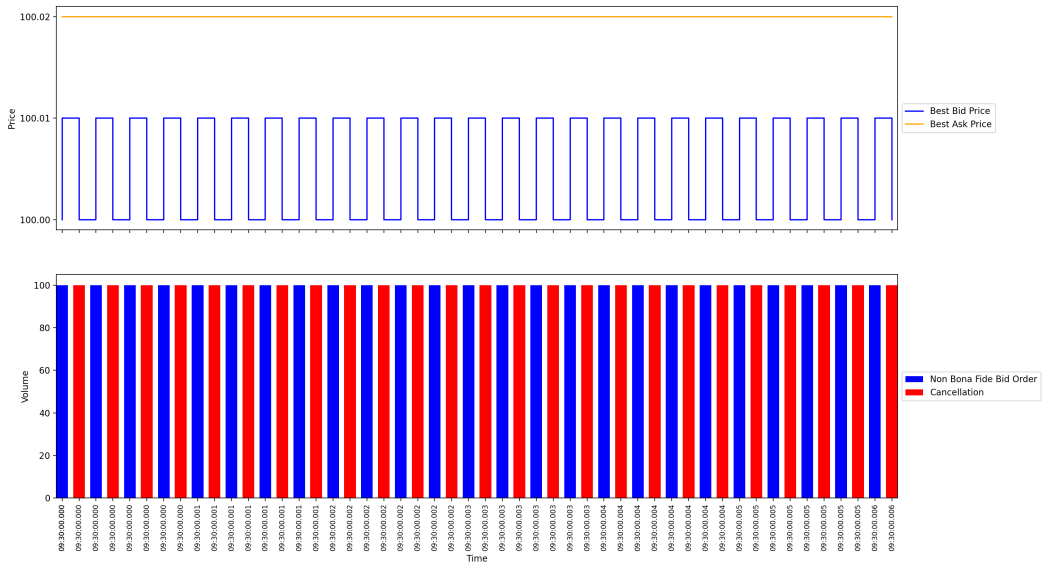

Figure 5: Toy example of quote stuffing.

4.3 Feature engineering

As mentioned previously, the proposed framework utilizes a set of features aiming to best describe the variations in LOB states so that multiple manipulation types can be detected from a single model. The trade–based manipulations described above are all very distinct, and an adequate set of features needs to be able to describe them all. Thus, we propose LOB–based features that cover LOB price returns, LOB volumes, cancellation volumes, and trade volumes, while also considering the rapidity at which the different market events occur. Table 6 details all the features used in the framework built from the original data presented at the beginning of this section. Each of the 14 features are standardized on a daily stock-per-stock basis. For any daily stock feature vector $\mathbf{x}$, its standardized version, $\mathbf{x}'$, is given by:

$$\mathbf{x}' = \frac{\mathbf{x} - \overline{x}}{\sigma},$$

where $\overline{x}$ and σ are its empirical mean and standard deviation, respectively. In Table 6, $P_t^{Bid/Ask}$ are the best bid/ask prices of the LOB at market event t, $Size_t^{Bid/Ask}$ the best bid/ask size of the LOB, $Size_{Cancel/Trade,t}^{Bid/Ask}$ the size of the cancellation/trade on the best bid/ask side, and Δt the time

Table 6: List of LOB features proposed to capture trade–based manipulations.

Feature	Description
r_t^{Bid}	Best bid-price return at event t: $r_t^{Bid} = \ln\left(P_t^{Bid}/P_{t-1}^{Bid}\right)$
r_t^{Ask}	Best ask-price return at event t: $r_t^{Ask} = \ln\left(P_t^{Ask}/P_{t-1}^{Ask}\right)$
$r_t^{Bid}/\Delta t$	Difference quotient of best bid-price return *w.r.t.* time, at event t
$r_t^{Ask}/\Delta t$	Difference quotient of best ask-price return *w.r.t.* time, at event t
$\overline{Size}_t^{Bid}$	Simple moving average of total size at best bid-price, at event t: $\overline{Size}_t^{Bid} = \sum_{k=0}^{9} Size_{t-k}^{Bid}/10$
$\overline{Size}_t^{Ask}$	Simple moving average of total size at best ask-price, at event t: $\overline{Size}_t^{Ask} = \sum_{k=0}^{9} Size_{t-k}^{Ask}/10$
$Size_{Trade,t}^{Bid}$	Size of trade consuming liquidity at best bid-price, at event t
$Size_{Trade,t}^{Ask}$	Size of trade consuming liquidity at best ask-price, at event t
$Size_{Cancel,t}^{Bid}$	Size of order cancellation/deletion located at best bid-price, at event t
$Size_{Cancel,t}^{Ask}$	Size of order cancellation/deletion located at best ask-price, at event t
$I_{Trade,t}^{Bid}$	Indicator of trade rapidity on best bid-price, at event t: $I_{Trade,t}^{Bid} = \mathbb{1}_{\{Size_{Trade,t}^{Bid} \neq 0\}}/\Delta t$
$I_{Trade,t}^{Ask}$	Indicator of trade rapidity on best ask-price, at event t: $I_{Trade,t}^{Ask} = \mathbb{1}_{\{Size_{Trade,t}^{Ask} \neq 0\}}/\Delta t$
$I_{Cancel,t}^{Bid}$	Indicator of cancellation rapidity on best bid-price, at event t: $I_{Cancel,t}^{Bid} = \mathbb{1}_{\{Size_{Cancel,t}^{Bid} \neq 0\}}/\Delta t$
$I_{Cancel,t}^{Ask}$	Indicator of cancellation rapidity on best ask-price, at event t: $I_{Cancel,t}^{Ask} = \mathbb{1}_{\{Size_{Cancel,t}^{Ask} \neq 0\}}/\Delta t$

delta between market events t and $t-1$.

5 Experiments

5.1 Setup

The models are fitted on the training data set containing the features computed from the LOBSTER data of Huang and Polak [23] presented in Section 4, for every stock. They are trained for 250 epochs, with a batch size of 512, and on subsequences of length $k = 25$. Autoencoders have a representation dimension of $d = 128$, which is enough to compress the subsequence dimensionality of $k \times m = 25 \times 14 = 350$, while ensuring they are able to reconstruct the time series input well enough. The rest of the bottlenecked-Transformer autoencoder's hyperparameters are set similarly to Vaswani et al. [57]: $h = 8$, $d_{ff} = 4d = 512$, and $N = 6$, and the model is trained following their proposed methodology based on the Adam optimizer of Kingma et al. [27]. As for the OC-SVM, γ is set like in Sohn et al. [54], i.e., $\gamma = 10/\Big(d \times \mathrm{Var}\left(\phi_E(s_i \mid \widehat{\theta}_E)\right)\Big)$, and the other hyperparameters

are set to their default values in scikit-learn (Pedregosa et al. [42]). No hyperparameter tuning is done.

All metrics used in this section rely on combinations of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Most studies in the trade–based manipulation detection literature do not formally define them, and given the problem's context, their exact definitions are not clear. We formally introduce them to eliminate any ambiguity:

- TP: market events part of a fraud, and part of at least one detected subsequence,
- TN: market events not part of a fraud, and not part of any detected subsequences,
- FP: market events not part of a fraud, and part of at least one detected subsequence,
- FN: market events part of a fraud, and not part of any detected subsequences,

all of which are function of the dissimilarity threshold τ. Also, whenever a market event part of a fraud is detected, then we consider that all market events in that fraud are also detected. This is reasonable, since market regulators would catch the fraudulent behavior by looking at the surrounding events of a flagged anomaly.

As mentioned in Golmohammadi and Zaine [20], and Khodabandehlou and Golpayegani [26], the misclassification cost of fraudulent orders is higher than the one of normal orders, meaning that an emphasis on recall, as opposed to precision, is necessary to correctly evaluate trade–based manipulation detection frameworks. Hence, a $\beta > 1$ in F_β-measures is more appropriate. We detail the different statistics for any given τ:

$$Precision(\tau) = \frac{TP(\tau)}{TP(\tau) + FP(\tau)} \in [0,1]$$

$$Recall(\tau) = TPR(\tau) = \frac{TP(\tau)}{TP(\tau) + FN(\tau)} \in [0,1]$$

$$F_\beta(\tau) = (1 + \beta^2) \cdot \frac{Precision(\tau) \cdot Recall(\tau)}{(\beta^2 \cdot Precision(\tau)) + Recall(\tau)} \in [0,1]$$

$$AUPRC = \int_{-\infty}^{\infty} Precision(\tau) Recall'(\tau) d\tau \in [0,1]$$

$$FPR(\tau) = \frac{FP(\tau)}{FP(\tau) + TN(\tau)} \in [0,1]$$

$$AUROC = \int_{-\infty}^{\infty} TPR(\tau)FPR'(\tau)d\tau \in [0,1]$$

5.2 Main results

5.2.1 General performance and comparative study

Like Golmohammadi and Zaine [20], we use the F_4-measure to quantitatively compare the different models, and we also provide the area under the precision-recall curve (AUPRC) to evaluate the models' general performance. The area under the receiver operating characteristic curve (AUROC) is also included as a third metric, since it is still used in recent literature, e.g., Close and Kashef [12]; Rizvi et al. [46]; Wang et al. [59], even though it can be misleading in imbalanced data sets, like in anomaly detection (Saito and Rehmsmeier [50]). Table 7 compares the performance of our proposed method to some others put forward in past studies. Only unsupervised frameworks working on tick data, i.e., orders or LOBs, are considered, and all are trained and tested on our data. The optimal threshold for each method is selected to optimize the F_4-measure, and we also detail the precision and recall at that point of the precision-recall curve.

Table 7: Comparative study of the proposed method with previous unsupervised papers, on our data. Best metric in bold, second best is underlined.

Paper	Model	AUROC	AUPRC	F_4	Precision	Recall
Ours	Transformer-AE + OC-SVM	<u>0.900</u>	**0.847**	**0.935**	0.628	0.965
Abbas et al. [1]	EMD + KDE Clustering	0.700	0.375	0.803	0.195	0.992
Cao et al. [7]	OC-SVM	**0.944**	<u>0.732</u>	<u>0.899</u>	0.397	0.976
Leangarun et al. [29]	LSTM-GAN	0.694	0.323	0.774	0.197	0.878
Leangarun et al. [30]	LSTM-AE	0.682	0.293	0.773	0.167	1.000
Rizvi et al. [46]	MLP-AE + MKDE Clustering	0.674	0.371	0.776	0.169	1.000

By working on uniform sampling of the LOB with one second intervals, Leangarun et al. [29, 30] miss the entire temporal context of fast trade–based manipulations, like quote stuffing and layering.

Furthermore, the impact of layering is minimal when looked at a second resolution, making it nearly impossible to detect in their methodology (see Section 4.2.2). This is represented in Table 7, where they achieve the lowest AUPRC and F_4-measure on our data set. The hybrid clustering techniques of Abbas et al. [1] and Rizvi et al. [46] fare a bit better by working directly on the LOB. But by only focusing on price features, they lack the LOB volume and cancellation features needed to detect various trade–based manipulations. Also, the temporal context is not considered in their methodology, which can be appropriate for point anomalies, but is not when trying to capture collective time series anomalies. This aspect is also a shortcoming of Cao et al. [7], where single orders are classified without any context. But still, they achieve a greater performance than newer methods because of better-crafted features able to encapsulate important orders' characteristics: size, LOB price effect, and duration. The dissimilarity objective of the OC-SVM algorithm also seems to be more apt than clustering (in Abbas et al. [1]; Rizvi et al. [46]), and estimation models (in Leangarun et al. [29, 30]), for trade–based fraud detection.

Overall, when tested on pragmatic simulations of sophisticated, and distinct, trade–based manipulations, all previously proposed unsupervised methods fall short when compared to our model in terms of AUPRC and F_4-measure, mostly because of their lower precision, i.e., higher false positive rate. For market regulators, this means that less time is wasted on false alarms, which is the main goal of this paper. The work of Cao et al. [7] is a close second, and also significantly outperforms the other, more recent, models evaluated in Table 7. Finally, our methodology integrates more relevant features of the LOB, and their temporal context. It also learns a more descriptive representation of the data, enabling an easier separation of normal and abnormal subsequences compared to previous methods, which is corroborated by Table 7.

5.2.2 Performance per stock

Table 8 details the proposed method's performance on a per-stock basis with the same F_4-optimal threshold τ^* found for Table 7. An important observation can be made from Table 8, which is that the model's performance is not drastically different for any stock in terms of F_4-measure, meaning that it was able to learn an asset–independent representation of normality. This is important for market regulators since a single model is enough to capture anomalies for all assets. This contrasts

Table 8: Per-stock performance of the proposed methodology with the general F_4-optimal τ^*.

Stock	$\mathbf{F}_4$	Precision	Recall
AAPL	0.955	0.710	0.977
AMZN	0.893	0.783	0.901
GOOG	0.940	0.820	0.949
INTC	0.949	0.530	0.998
MSFT	0.934	0.499	0.988

with rule–based systems where their rules have to be manually adjusted for each asset, depending on their stylized facts (see Table 2 for an overview of important stylized facts in stocks). Moreover, the F_4-measures are all approaching 1, demonstrating the great general performance of the methodology.

5.2.3 Performance per manipulation type

For the first time in the financial market fraud detection literature, we quantitatively study the difficulty of detecting different trade–based manipulation types. Figure 6 uses the t-distributed stochastic neighbor embedding (t-SNE) of Maaten and Hinton [34] to visualize the representation space of dimension $d = 128$.[9] As can be observed, different clusters for each manipulation tactics lie outside the main cluster of normal data, meaning that the representation vectors are able to discriminate between normal and abnormal subsequences, even differentiating between fraud types. Figure 7 shows the empirical distribution of our trained dissimilarity measure on normal and fraudulent subsequences on all stocks, from the representation vectors shown in Figure 6. Visually, pump-and-dumps seem to be the easiest to detect given their smaller overlap with the distribution of normal data, whereas layering activities appear to be the hardest to detect. Quote stuffing resides in between the other two manipulation types.

Table 9 statistically confirms that ordering, based on first-order stochastic dominance of the dissimilarity distributions. It applies the two-sample Kolmogorov-Smirnov test where H_0: $F(x) \leq G(x)$, $\forall x$ and H_1: $F(x) > G(x)$ for at least one x, to evaluate the stochastic ordering between each

[9] t-SNE is a nonparametric, nonlinear dimensionality reduction technique frequently used to visualize high-dimensional data in two or three dimensions. The low dimensional representations are modeled such that nearby points, as defined by the Euclidean distance, are similar in the high dimensional space, while distant points are dissimilar with high probability, thus maintaining the relationship between surrounding points.

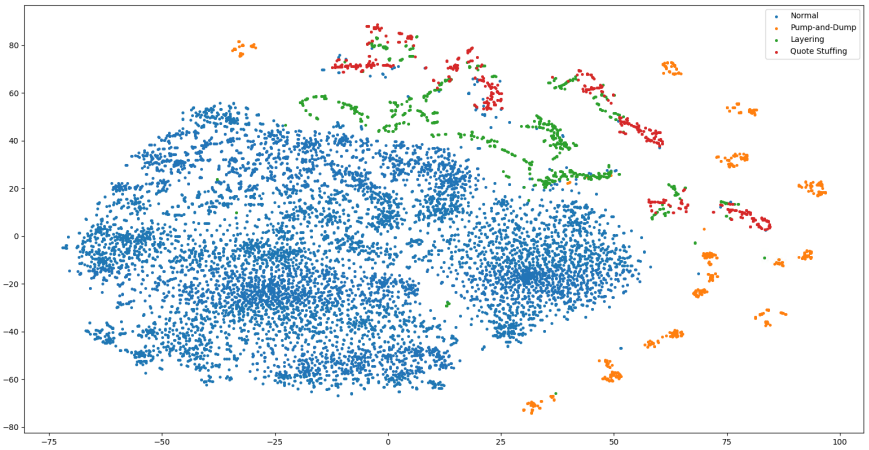

Figure 6: t-SNE visualization of the representation space generated by the bottlenecked-Transformer encoder.

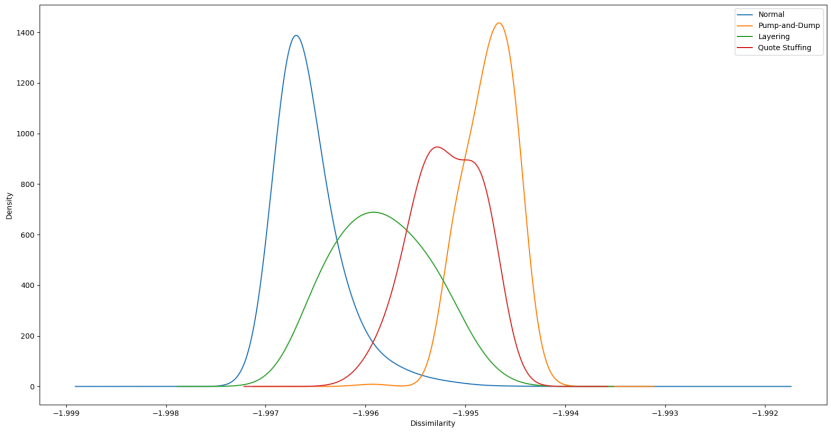

Figure 7: Empirical distributions of our trained dissimilarity measure on the temporal representations for normal, pump-and-dump, layering, and quote stuffing subsequences.

pair of dissimilarity distributions. Firstly, we can conclude that pump-and-dump sequences have first-order stochastic dominance over all the order distributions, and quote stuffing dominates layering. Secondly, all trade–based manipulation dissimilarity distributions have first-order stochastic dominance over the normal dissimilarity distribution, further demonstrating that the hybrid model is indeed able to differentiate between normal and abnormal LOB sequences.

Table 9: $p-values$ of two-sample Kolmogorov-Smirnov test for the first-order stochastic dominance of dissimilarity distributions found by the proposed hybrid model.

F(x) | G(x)	**Pump-and-Dump**	**Quote Stuffing**	**Layering**	**Normal**
Pump-and-Dump	-	1.000	1.000	1.000
Quote Stuffing	0.000	-	1.000	0.990
Layering	0.000	0.000	-	0.996
Normal	0.000	0.000	0.000	-

5.3 Ablation study

To better understand the importance of each aspect in the proposed methodology (Transformer vs. recurrent models, dissimilarity vs. estimation methods, temporal encoding vs. no encoding/non-sequential encoding), we contrast its performance against four similar model variants. The hybrid Transformer-AE/OC-SVM model (Dissimilarity Transformer) is compared to an equivalent LSTM-AE/OC-SVM model (Dissimilarity LSTM), and an equivalent MLP-AE/OC-SVM model (Dissimilarity MLP) that all have the same representation vector dimension $d = 128$.[10] The proposed framework is also evaluated against a simple OC-SVM (Dissimilarity only OC-SVM), which takes in the entire multivariate subsequence as a single vector, thereby ignoring the temporality of the data and any dimensionality reduction of encoders. Finally, the dissimilarity approach is compared to the usual reconstruction method (Reconstruction Transformer), using the complete bottlenecked-Transformer-AE trained for the hybrid approach, and ignoring the OC-SVM. Table 10 presents the performance of these methods on the test set containing all stocks, and all simulated frauds, for a general F_4-optimal τ^* per method.

As can be concluded from Table 10, the bottlenecked-Transformer architecture is able to generate more descriptive representation vectors than LSTMs, facilitating the discriminative objective of the OC-SVM in the representation space, which in turn results in greater AUROC, AUPRC, and F_4-measure. We can also notice the importance of integrating the temporality of the data in its representation, as the MLP generates a worse performance than the Transformer encoder's, even lower than the OC-SVM alone. This demonstrates that simple encoding is not enough, and that the performance of the bottleneck-Transformer model results mainly from its ability to learn rich

[10] In this context, the term "equivalent" refers to a similar L_2 reconstruction loss obtained by the autoencoders on the validation set.

Table 10: Overall performance of the proposed methodology on test set containing all stocks, compared to variant approaches. Best metric in bold, second best is underlined.

Method | Metric	AUROC	AUPRC	F_4	Precision	Recall
Dissimilarity Transformer	**0.900**	**0.847**	**0.935**	0.628	0.965
Dissimilarity LSTM	<u>0.877</u>	0.514	<u>0.869</u>	0.332	0.967
Dissimilarity MLP	0.722	0.628	0.767	0.162	1.000
Dissimilarity only OC-SVM	0.803	<u>0.652</u>	0.792	0.602	0.808
Reconstruction Transformer	0.467	0.431	0.782	0.174	1.000

temporal representations. Finally, the estimation method using the reconstruction error of the same bottlenecked-Transformer autoencoder results in poor detection accuracy compared to all dissimilarity methods. Overall, Table 10 demonstrates that, out of multiple similar variants, the combination of temporal encoding from a Transformer model, and a dissimilarity approach, generates the best performance in terms of AUROC, AUPRC, and F_4-measure, in the context of LOB time series anomaly detection.

6 Conclusion

In this article, we propose a novel time series anomaly detection model tailored to LOB time series, with an application to trade–based manipulation detection in five NASDAQ stocks. We introduce a new autoencoder, the bottlenecked-Transformer autoencoder, which can learn semantically rich temporal representations of LOB time series. The representation space of its encoder eases the separability of normal and abnormal LOB behavior, allowing a one-class classification algorithm to discriminate between the two categories with a dissimilarity function, thus detecting accurately anomalous LOB subsequences out-of-sample. The model utilizes a greater pool of LOB features to capture a greater range of fraud types compared to the previous literature, by integrating the price, volume, and time dynamics of the LOB, a necessary step in the evolution of trade–based

manipulation detection (Khodabandehlou and Golpayegani [26]). Finally, the framework achieves new state-of-the-art performance by adapting recent deep learning methods proposed for image anomaly detection, reducing the gap between this active field and the financial market anomaly detection literature.

We also present a complete trade–based manipulation scenarios simulator able to generate pump-and-dump, layering, and quote stuffing tactics. The random scenarios are used to quantify the performance of the anomaly detection model. This is an important departure from earlier literature in financial fraud detection, since it relied on repeating the same limited set of orders, hence overestimating the performance of previous methods. We show that the proposed deep unsupervised anomaly detection model captures these three types of fraud on all five stocks, meaning that it learns an asset-independent notion of normal LOB behavior, without needing any prior knowledge on fraudulent patterns. We also empirically show that the Transformer–based model learns better representations than the popular LSTM network, and that dissimilarity methods outperform more traditional estimation–based anomaly detection on LOB time series. Furthermore, we quantify the difficulty of detecting pump-and-dump, layering, and quote stuffing manipulations, a first in the literature. Providing this kind of analysis is also helpful in determining the comparative strengths and weaknesses of new trade–based manipulation detectors, in addition to traditional performance metrics.

The proposed framework is a strong alternative to the rule–based systems currently used by market regulators (Golmohammadi and Zaine [20]). First, it learns a general notion of normalcy, so a single model instance can be utilized for any asset, and to detect any type of anomalous behavior. Second, it can dynamically adapt to market regimes, whereas rule–based systems need to be manually adjusted. But market data drift is an important aspect to consider when deploying any data–based model (Žliobaitė et al. [58]), because the past learned notion of normality might slowly depart from future normal market behaviors. It is primordial to know when to retrain anomaly detection frameworks, and further research in that sense, in the context of financial markets, is important.

Also, semi-supervised learning techniques for anomaly detection have recently been proposed in the deep learning literature (e.g., Ruff et al. [49]), where small sets of known anomalies are also

used to train the models, hence boosting their detection performance over pure unsupervised methods. It would be worthwhile to explore and adapt these methods to the trade–based manipulation detection problem, as only limited collections of frauds are available to researchers. Additionally, semi-supervised approaches open the door to human-in-the-loop models where they could learn from an ever-growing pool of detected frauds confirmed by market regulators, constantly raising their detection capabilities. Our framework can act as a starting point on which the semi-supervised methods can be built upon.

Acknowledgments

Financial support from Fin-ML, the Montreal Exchange (TMX), Centre d'intelligence en surveillance des marchés financiers, and Mitacs is graciously acknowledged. We also thank the TMX regulatory division team and IVADO for their constant support in this project.

References

[1] B. Abbas, A. Belatreche, and A. Bouridane. 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*, pages 851–866, 2019.

[2] R.K. Aggarwal and G. Wu. Stock market manipulation. *The Journal of Business*, 79(4):1915–1953, 2006.

[3] S. Agrawal and J. Agrawal. Survey on anomaly detection using data mining techniques. *Procedia Computer Science*, 60:708–713, 2015.

[4] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga. Usad: Unsupervised anomaly detection on multivariate time series. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3395–3404, 2020.

[5] A. Blázquez-García, A. Conde, U. Mori, and J.A. Lozano. A review on outlier/anomaly detection in time series data. *ACM Computing Surveys (CSUR)*, 54(3):1–33, 2021.

[6] Y. Cao, Y. Li, S. Coleman, A. Belatreche, and T.M. McGinnity. A hidden markov model with abnormal states for detecting stock price manipulation. *IEEE transactions on neural networks and learning systems*, 26(2): 318–3019, 2013.

[7] Y. Cao, Y. Li, S. Coleman, A. Belatreche, and T.M. McGinnity. Detecting price manipulation in the financial market. In *IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*, pages 77–84. IEEE, 2014.

[8] R. Chalapathy and S. Chawla. Deep learning for anomaly detection. *arXiv preprint arXiv:1901.03407*, 2019.

[9] K. Choi, C. Hawthrone, I. Simon, M. Dinculescu, and J. Engel. Encoding musical style with transformer autoencoders. *International Conference on Machine Learning*, pages 1899–1908, 2020.

[10] P. Chullamonthon and P. Tangamchit. A transformer model for stock price manipulation detection in the stock exchange of thailand. *IEEE International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, 2022.

[11] P. Chullamonthon and P. Tangamchit. Ensemble of supervised and unsupervised deep neural networks for stock price manipulation detection. *Expert Systems with Applications*, 220:119698, 2023.

[12] L. Close and R. Kashef. Combining artificial immune system and clustering analysis: A stock market anomaly detection model. *Journal of Intelligent Learning Systems and Applications*, 12:83–108, 2020.

[13] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[14] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1): 21–27, 1967.

[15] A. Deng and B. Hooi. Graph neural network-based anomaly detection in multivariate time series. *Proceedings of the AAAI conference on artificial intelligence*, 35(5):4027–4035, 2021.

[16] D. Diaz, B. Theodoulidis, and P. Sampaio. Analysis of stock market manipulations using knowledge discovery techniques applied to intraday trade prices. *Expert Systems with Applications*, 38(10):12757–12771, 2011.

[17] J.F Egginton, B.F. Van Ness, and R.A. Van Ness. Quote stuffing. *Financial Management*, 45(3):583–608, 2016.

[18] G.D. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61:268–278, 1973.

[19] D.P. Gandhmal and K. Kumar. Systematic analysis and review of stock market prediction techniques. *Computer Science Review*, 34, 2019.

[20] K. Golmohammadi and O.R. Zaine. Time series contextual anomaly detection for detecting market manipulation in stock market. *International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10, 2015.

[21] K. Golmohammadi, O.R. Zaine, and D. Diaz. Detecting stock market manipulation using supervised learning algorithms. *International Conference on Data Science and Advanced Analytics (DSAA)*, pages 435–441, 2014.

[22] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[23] R. Huang and T. Polak. Lobster: Limit order book reconstruction system. Available at https://ssrn.com/abstract=1977207, 2011.

[24] IIROC. Iiroc guidance on certain manipulative and deceptive trading practices. https://www.iiroc.ca/news-and-publications/notices-and-guidance/guidance-certain-manipulative-and-deceptive-trading-practices, 2013. Accessed March 13, 2023.

[25] J. Kamps and B. Kleinberg. To the moon: defining and detecting cryptocurrency pump-and-dumps. *Crime Science*, 7(18), 2018.

[26] S. Khodabandehlou and S.A.H Golpayegani. Market manipulation detection: A systematic literature review. *Expert Systems with Applications*, page 118330, 2022.

[27] D. P. Kingma, J., and Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[28] T. Leangarun, P. Tangamchit, and S. Thajchayapong. Stock price manipulation detection based on mathematical models. *International Journal of Trade, Economics and Finance*, pages 81–88, 2016.

[29] T. Leangarun, P. Tangamchit, and S. Thajchayapong. Stock price manipulation detection using generative adversarial networks. *IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2104–2111, 2018.

[30] T. Leangarun, P. Tangamchit, and S. Thajchayapong. Stock price manipulation detection using deep unsupervised learning: The case of thailand. *IEEE Access*, 9:106824–106838, 2021.

[31] E.J. Lee, K.S. Eom, and K.S. Park. Microstructure-based manipulation: Strategic behavior and performance of spoofing traders. *Journal of Financial Markets*, 16:227–252, 2013.

[32] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.K. Ng. Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks. *Artificial Neural Networks and Machine Learning–ICANN 2019: Text and Time Series: 28th International Conference on Artificial Neural Networks*, pages 703–716, 2019.

[33] T.C.W Lin. The new market manipulation. *Emory Law Journal*, 66:1253, 2016.

[34] L. Van De Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11), 2008.

[35] H. Meng, Y. Zhang, Y. Li, and H. Zhao. Spacecraft anomaly detection via transformer reconstruction error. In *Proceedings of the International Conference on Aerospace System Science and Engineering 2019*, pages 351–362. Springer, 2020.

[36] I. Montero, N. Pappas, and N.A. Smith. Sentence bottleneck autoencoders from transformer language models. *arXiv preprint arXiv:2109.00055*, 2021.

[37] Nanex Research. Latency on demand? http://www.nanex.net/FlashCrash/FlashCrashAnalysis_LOD.html, 2010. Accessed March 15, 2023.

[38] Nanex Research. The wab event. http://www.nanex.net/StrangeDays/12142011.html, 2011. Accessed March 15 2023.

[39] Nanex Research. The quote stuffing trading strategy. http://www.nanex.net/aqck2/4670.html, 2014. Accessed March 15, 2023.

[40] G. Pang, C. Shen, L. Cao, and A.V.D. Hengel. Deep learning for anomaly detection: A review. *ACM computing surveys (CSUR)*, 54(2):1–38, 2021.

[41] E. Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33 (3):1065–1076, 1962.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, , and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[43] T. Reiss, N. Cohen, L. Bergman, and Y. Hoshen. Panda: Adapting pretrained features for anomaly detection and segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2806–2814, 2021.

[44] T. Reiss, N. Cohen, E. Horwitz, R. Abutbul, and Y. Hoshen. Anomaly detection requires better representations. *Computer Vision–ECCV 2022 Workshops: Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part IV*, pages 56–68, 2023.

[45] B. Rizvi, A. Belatreche, and A. Bouridane. A dendritic cell immune system inspired approach for stock market manipulation detection. *IEEE Congress on Evolutionary Computation (CEC)*, pages 3325–3332, 2019.

[46] B. Rizvi, A. Belatreche, A. Bouridane, and K. Mistry. Stock price manipulation detection based on autoencoder learning of stock trades affinity. *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2020.

[47] B. Rizvi, A. Belatreche, A. Bouridnae, and I. Watson. Detection of stock price manipulation using kernel based principal component analysis and multivariate density estimation. *IEEE Access 8*, pages 135989–136003, 2020.

[48] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S.A. Siddiqui, A. Binder, E. Müller, and M. Kloft. Deep one-class classification. In *International conference on machine learning*, pages 4393–4402. PMLR, 2018.

[49] L. Ruff, R.A. Vandemeulen, N. Görnitz, A. Binder, E. Müller, K.R. Müller, and M. Kloft. Deep semi-supervised anomaly detection. *Eight International Conference on Learning Representations (ICLR)*, 2020.

[50] T. Saito and M. Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PLOS one*, 10(3):e0118432, 2015.

[51] B. Schölkopf, R.C. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt. Support vector method for novelty detection. *Advances in neural information processing systems*, 12:582–588, 1999.

[52] L. Shen, Z. Li, and J. Kwok. Time series anomaly detection using temporal hierarchical one-class network. *Advances in Neural Information Processing Systems*, 33:13016–13026, 2020.

[53] M. Siering, B. Clapham, O. Engel, and P. Gomber. A taxonomy of financial market manipulations: establishing trust and market integrity in the financialized economy through automated fraud detection. *Journal of Information Technology*, 32:251–269, 2017.

[54] K. Sohn, C.L. Li, J. Y, M. Jin, and T. Pfister. Learning and evaluating representations for deep one-class classification. *International Conference on Learning Representations (ICLR)*, 2021.

[55] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2828–2837, 2019.

[56] D.M.J Tax and R.P.W. Duin. Support vector data description. *Machine learning*, 54:45–66, 2004.

[57] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[58] I. Žliobaitė, M. Pechenizkiy, and J. Gama. An overview of concept drift applications. In Nathalie N. Japkowicz and J. Stefanowski, editors, *Big Data Analysis: New Algorithms for a New Society*, pages 91–114. Springer International Publishing, Cham, 2016. ISBN 978-3-319-26989-4. doi: 10.1007/978-3-319-26989-4_4. URL https://doi.org/10.1007/978-3-319-26989-4_4.

[59] Q. Wang, W. Wu, X. Huang, and K. Yang. Enhancing intraday stock price manipulation detection by leveraging recurrent neural networks with ensemble learning. *Neurocomputing*, 347:46–58, 2019.

[60] X. Wang, Y. Zhao, and F. Pourpanah. Recent advances in deep learning. *International Journal of Machine Learning and Cybernetics*, 11:747–750, 2020.

[61] J. Xu, H. Wu, J. Wang, and M. Long. Anomaly transformer: Time series anomaly detection with association discrepancy. *International Conference on Learning Representations (ICLR)*, 2022.

[62] C. Zhang, D. Song, Y. Chen, X. Feng, C. Lumezanu, W. Cheng, J. Ni, B. Zong, H. Chen, , and N. V. Chawla. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:1409–1416, 2019.

[63] H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, and Q. Zhang. Multivariate time-series anomaly detection via graph attention network. *IEEE International Conference on Data Mining (ICDM)*, pages 841–850, 2020.

[64] H. Ögüt, M.M Doğanay, and R. Aktaş. Detecting stock price manipulation in an emerging market: The case of turkey. *Expert Systems with Applications*, 36(9):11944–11949, 2009.