Abstract

Despite the fact that cryptocurrencies themselves have experienced an astonishing rate of adoption over the last decade, cryptocurrency fraud detection is a heavily under-researched problem area. Of all fraudulent activity regarding cryptocurrencies, pump and dump schemes are some of the most common. Though some studies have been done on these kinds of scams in the stock market, the lack of labelled stock data and the volatility unique to the cryptocurrency space constrains the applicability of studies on the stock market toward this problem domain. Furthermore, the only work done in this space thus far has been either statistical in nature, or has been concerned with classical machine learning models such as random forest trees. We propose the novel application of two existing neural network architectures to this problem domain and show that deep learning solutions can significantly outperform all other existing pump and dump detection methods for cryptocurrencies.

1 Introduction

In 2021 alone, cryptocurrencies (or “crypto”) experienced over $14 trillion worth of trading volume, representing nearly a 700% increase over the previous year. In that same year, the Binance cryptocurrency exchange (or just “exchange”) was responsible for over two-thirds of that volume (Khatri, 2021). As crypto and the use of these exchanges enters the mainstream, the conversation surrounding their regulatory hurdles has intensified (Cumming et al., 2019). Of those hurdles, the detection of fraudulent activities at scale is one of the most pressing, given the rapid growth of the space. Despite these concerns, the amount of regulation in crypto pales in comparison to more mainstream flows of money, such as stocks. This allows fraud to plague the crypto space at a level unheard of in the stock market.

Among all types of fraud in the crypto space, pump and dump (P&D) schemes are some of the most popular, and some of the easiest, to execute on (Twomey and Mann, 2020) – generally resulting from the concerted effort of just a single online P&D planning group (Hamrick et al., 2021). Despite that, no studies have been done to-date on the application of deep learning to P&D detection in crypto. Though some studies involving the application of deep learning have been done in the context of traditional securities such as stocks (Leangrun et al., 2018), these studies not only lacked the amount of data now freely available for crypto (via the blockchain), but also have no guarantee of applicability to the much more volatile world of crypto.

This paper presents two novel applications of existing deep learning methods to detect P&D schemes in crypto – specifically, for small, volatile cryptocurrencies also known as altcoins. Our work focuses on taking advantage of the bulk of freely available data by using deep learning to drive performance gains, since previous works in this space have thus far only applied either classical machine learning techniques such as random forest trees (La Morgia et al., 2020), or more basic techniques such as statistic analyses (Kamps and Kleinberg, 2018) to the problem. All of our work is reproducible, and a link to our code can be found in the footnote below.

2 Related Work

2.1 Application of Classical Machine Learning and Statistical Models to Crypto P&D Detection

To-date, only classical machine learning models such as random forest trees have been applied in the

1https://github.com/Derposoft/crypto_pump_and_dump_with_deep_learning
Figure 1: An example of an organized P&D scheme on the REAP token. Red stars indicate visual signs of the accumulation phase in the form of subtle buying pressure, while the blue star indicates the pump and dump phases of the scheme. This data was manually obtained directly from the KuCoin exchange (https://www.kucoin.com/trade/REAP-USDT), although the same information can be found on other exchanges as well. Publicly-available videos on YouTube can confirm that this fluctuation in the REAP token’s price was a result of a planned P&D operation (Coffeezilla, 2021).

crypto space to detect P&D schemes (La Morgia et al., 2020). Statistical models have also been applied in this domain (Kamps and Kleinberg, 2018). These models often aggregate trade data at a high level and use these coalesced features to predict when a P&D scheme will occur. As these are the only works that have been done in this niche, we hope that our own work in this paper will help build a better foundation in this domain going forward.

2.2 Application of Deep Learning to General Anomaly Detection

Anomaly detection in a general setting unrelated to crypto is a well-researched field. More specifically, studies involving the application of deep learning to time-series anomaly detection problems have been done involving multiple network architectures: LSTMs (Malhotra et al., 2016), convolutional networks (Kwon et al., 2018), and various combinations of the two (Kim and Cho, 2018); and more recently, multiple variations of attention-based methods such as RNN attention (Brown et al., 2018) or the anomaly transformer (Xu et al., 2021) have also been explored. As of writing this paper, deep learning architectures are at the forefront of time-series anomaly detection due to their strength in making predictions using spatiotemporal relationships, which are key to strong anomaly detection models (Kim and Cho, 2018). In this work, we implement, modify, and tune some of these latest architectures in an attempt to adapt them to the crypto domain.

3 Background

While P&D is heavily fined by the SEC in traditional financial markets (e.g. stocks), crypto remains to be regulated at a federal level. This opens the opportunity for various P&D groups to openly plan P&D events over online messaging platforms such as Telegram or Discord (La Morgia et al., 2020). Typically, a P&D scheme consists of three phases: the accumulation phase, the promotion (or “pump”) phase, and the distribution (or “dump”) phase.

During the accumulation phase, the group organizing the scheme slowly accumulates a significant position in the asset of interest. Once ready, the promotion phase begins: excitement is drummed up via social media promotion for the asset, and bullish sentiment is falsified through fraudulent re-
ports. This causes retail investors to rally behind and invest in the asset, thereby driving the price up. Finally, during the distribution phase, the perpetrator of the scheme liquidates their position in the asset over a very short period of time. Since the position being liquidated generally represents a significant portion of the asset itself, this inevitably causes a crash in the price of the original asset – leaving most of the retail investors who bought during the promotion phase at a significant loss on their positions over an extremely short time.

The graph in Fig. 1 is an example of a real P&D scheme on the REAP token which occurred near the start of April 2021. In this case, the entirety of the pump and dump phases lasted only a few minutes, and the accumulating party dumped all of their coins as soon as the value of the token shot up. Investors who bought during the peak of the pump would have been down nearly 90% in the span of just a few minutes.

Successful P&D detection algorithms have the potential to alert investors of P&D schemes before they occur, and could even enable regulatory bodies to police this fraudulent activity in more cost and time-effective manners. Ultimately, this could help to establish crypto as a much more legitimate trading option, as well as save investors and exchanges millions from potential scams and fraudulent activity.

4 Method

We propose the novel application of two architectures that have scored well on standard anomaly detection datasets to the now-burgeoning financial data also available in the cryptocurrency space: the C-LSTM model (Kim and Cho, 2018), and the Anomaly Transformer model (Xu et al., 2021).

Since P&D schemes often have multiple phases that generally occur over vastly different lengths of time (e.g. the accumulation phase may last for up to a month, whereas the pump or dump phases can last for as little as a minute), both of the models that we have chosen have the ability to capture both longer-term anomalies – otherwise known as “trend” anomalies – and much shorter-term anomalies – otherwise known as “point” anomalies. This capability is important for the detection of P&D schemes, since models that are only capable of one or the other could potentially be fooled by the volatility inherent in crypto markets.

4.1 Models

4.1.1 C-LSTM

Our first model is the C-LSTM model, originally introduced by (Kim and Cho, 2018) for learning anomaly detection by treating data as spatiotemporal in nature. The model consists of a series of convolutional/ReLU/pooling layers to encode the input sequence, followed by a set of LSTM layers, with decoding done via a set of feedforward layers. In this model, convolutional layers help to capture spatial information within the dataset, thereby helping to identify point anomalies; on the other hand, the LSTM layers help to capture temporal information, and help to identify trend anomalies. This simple model has shown success in the detection of varying types of web traffic anomalies (Kim and Cho, 2018), as well as anomalies across a number of stocks in the Chinese stock market (Yang et al., 2020). The following image is a visual representation of the C-LSTM model.

Our Implementation Our own model uses the following architecture: 1 set of convolutional/ReLU/pooling layers with a convolution kernel size of 3 with a stride of 1, and a pooling kernel size of 2 with a stride of 1; 1 LSTM layer with an embedding dimension of 350; 1 feedforward layer which directly projects the last hidden state of the LSTM to a dimension of 1; and a sigmoid layer which constrains the output of our classifier between 0 and 1.

This setup resulted in a C-LSTM model with 997,851 learnable parameters which was trained for 200 epochs.

4.1.2 Anomaly Transformer

The second model we explored was the Anomaly Transformer introduced by (Xu et al., 2021). Unlike a standard transformer, the Anomaly Transformer uses a custom anomaly attention module to improve its performance in anomaly detection scenarios. To that end, the model introduces two novelties: an anomaly attention module, and a minimax optimization strategy. Additionally, on top of this model, we introduced our own, original novelty
The Anomaly Transformer model was chosen because it is the current state-of-the-art anomaly detection model for general time-series anomaly detection (Xu et al., 2021), consistently achieving strong results across a variety of standard datasets, including server sensor data (Su et al., 2019), rover data from NASA (Keogh et al., 2021), and the NeurIPS 2021 time-series benchmark (Lai et al., 2022).

The first of the two novelties introduced by the Anomaly Transformer which makes it particularly good at detecting anomalies is the replacement of the standard self-attention computation with two, internally-computed association values that influence the attention block: series association and prior association. These two associations make up the “anomaly attention module” and are as described below:

1. **Series association** at each layer $l$, denoted $S^l$, is a simple self-attention computation on the data before multiplying by the value matrix $V$ which is standard to a normal attention mechanism. Series association focuses on longer-term trends, and helps in the detection of trend anomalies.

$$S^l = \text{Softmax} \left( \frac{QK^T}{\sqrt{d_{\text{model}}}} \right)$$

2. **Prior association** at each layer $l$, denoted $P^l$, computes a Gaussian kernel for each point with respect to the input sequence. A rescale operation is done to convert the result to discrete distributions for each layer $l$. Prior association focuses more on local points, and helps the model in the detection of point anomalies.

$$P^l = \text{Rescale} \left( \mathcal{N}(j | \mu = i, \sigma) \right)_{i,j \in 1, \ldots, N}$$

The second of the two novelties introduced by this model is the minimax optimization process which takes place after computing the values $S^l$ and $P^l$. This process consists of two phases: minimize and maximize. During the minimize phase, the series association is moved towards the prior association using the symmetric KL divergence formula. Therefore, during the minimize phase, the series association approximates the series association. Then, during the maximize phase, the series association pays more attention to non adjacent points since it is enlarged by the prior association. As a result, a sequence generated by prior and series associations will lower the attention value at any anomaly. The total amount of difference between the two associations captured during this process is encapsulated by the Association Discrepancy function, denoted as $AD$. The data is then reconstructed after the minimax optimization via multiplying the series association $S^l$ with the value matrix standard to an attention module, $V$. The minimax strategy is visualized below.

Combining these novelties means that the reconstructed series, denoted $\hat{X}$, will greatly differ from the original series if an anomaly is present, making the discrepancy greater and giving the model an easier time to identify anomalies. After integrating in the reconstruction loss, the losses derived from each of these two phases are summarized through the following equations, where $\lambda > 0$ and $L_{\text{tot}}$ represents the total loss:

MinPhase: $L_{\text{tot}}(\hat{X}, P, S_{\text{detach}}, -\lambda; X)$

MaxPhase: $L_{\text{tot}}(\hat{X}, P_{\text{detach}}, S, \lambda; X)$
Our Implementation Since our own problem was supervised, we modified the optimization strategy and loss function used, since the Anomaly Transformer was originally built for unsupervised data. We propose and implement the following changes in order to adapt this model to a supervised anomaly detection setting:

1. Keep the maximize phase the same, but replace the minimize phase: instead, we move the series association towards the ground truth.

2. Replace the first term of the loss with the popular MSE loss between the outputted labels and the ground truth labels.

Our loss function for our maximize phase therefore becomes the following:

\[ L_{\text{tot}}(\hat{X}, P, S, \lambda; X) = \frac{1}{L} \sum_{l=1}^{L} SKL(P_{i_l}, S_{i_l}) \]

\[ \hat{X} = S^l V \]

\[ \hat{X} = S^l V \]

**4.2 Metrics**

During our experiments, we collect the precision, recall, and F1 scores of our models on the validation set at the end of each epoch. Since cryptocurrency P&D detection is an anomaly detection problem, we have chosen not to collect the accuracy of our model, because accuracy measurements do not reflect the strength of a model in anomaly detection scenarios. Specifically, since anomaly detection scenarios have a large class imbalance heavily favoring negative labels, the accuracy of any model is almost always over 99.9% even if it just outputs 0 (the negative class). Therefore, we optimize our models for the best possible F1 score for the reason that F1 scores generally reflect the strength of an anomaly detection model accurately. This is the standard used by most other anomaly detection models and helps us compare our results to previous work more effectively.

The precision is defined as the percentage of predicted anomalies which were correctly classified, and the recall is defined as the percentage of actual anomalies that were correctly classified as anomalies. The F1 score is the harmonic mean of these two values. Given that both the precision and recall are important in the context of anomaly detection, it stands to reason that the F1 score, which is a mix of the two values, is a good metric with which to compare two different models at the end of the day. The following equations are used to calculate the precision, recall, and F1 scores respectively:

\[ \text{Precision} = \frac{N_{TP}}{N_{TP} + N_{FP}} \]

\[ \text{Recall} = \frac{N_{TP}}{N_{TP} + N_{FN}} \]

\[ F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \]

where the terms \( N_{TP}, N_{FP}, \) and \( N_{FN} \) refer to the number of True Positives, False Positives, and False Negatives respectively.

After training each model, we choose the metrics corresponding to the epoch in which the highest validation F1 score was obtained instead of the metrics from the final epoch. This ensures that our metrics reflect a model that hasn’t overfit the training data.

**5 Experiments**

**5.1 Dataset**

The dataset used in this paper consists of manually-labelled, raw transaction data from the Binance cryptocurrency exchange, first introduced by (La Morgia et al., 2020). The transactions of various cryptocurrencies that experienced known occurrences of P&D were collected.

To generate the dataset, the authors first joined several cryptocurrency P&D Telegram groups that were well-known for planning and executing on P&D schemes. Then, over a period of two years, the researchers collected the timestamps for official "pump signals" which were announced in each of these groups by group administrators. Using these timestamps and the Binance API, the authors were able to collect every transaction for
the pumped cryptocurrency for up to 1 week preceding and succeeding the pump, depending on what was available to access. In this fashion, 343 P&D occurrences’ worth of data was collected.

After collecting this raw data from the Binance API, the authors further preprocessed the data by aggregating transactions into 5-second, 15-second, and 25-second “chunks”, thereby forming three different aggregated datasets. Each of these aggregated datasets, in turn, contains the following 15 features:

1. **Date, HourSin, HourCos, MinuteSin, MinuteCos**: The date and hour- and minute-based positional encoding of a given chunk.

2. **PumpIndex, Symbol**: The 0-based index of the pump, numbering it out of the 343 available pumps, and the ticker symbol of the coin on which the pump took place.

3. **StdRushOrder, AvgRushOrder**: The moving standard deviation and average percent change of the number of rush orders.

4. **StdTrades**: The moving standard deviation of the number of trades, both buy and sell.

5. **StdVolume, AvgVolume**: The moving standard deviation and average percent change in the order volume.

6. **StdPrice, AvgPrice, AvgPriceMax**: The moving standard deviation, average percent change, and average maximum percent change in the price of the asset.

### 5.2 Implementation Details

Given an input data sequence \( X = X_1, ..., X_N \), our first step splits the data into a separate train and validation set using an 80:20 ratio without shuffling. Following this split, train data undergoes the following preprocessing steps: First, data is broken into \( M \) contiguous subsequences \( X = Y_1, ..., Y_M \), where \( Y_i \) corresponds to the \( i \)th pump in the dataset as determined by the PumpIndex feature. Separating the data out by pump at this stage ensures that during training time, models aren’t fed information from 2 different pumps at the same time, which may hinder their training process. Of these pumps, all pump sequences \( Y_i \) with fewer than 100 chunks are discarded, since pumps that only have a few chunks do not make for exemplary training samples. This leaves us with a training subset of \( m < M \) pump sequences.

Then, we further prepare the data for each of these pumps by splitting them into segments of size \( s \) via taking a sliding window over the chunks of each pump \( Y_i \) and adding reflection padding to the start of size \( s - 1 \) so that the resulting count of windows is equal to the original count of chunks. Together, the chunks in each window are considered to be one “segment.” In this scheme, segments are inputs to our models. The models then predict the probability of a pump occurring during the last chunk of the segment. At this point, the segments from all pumps can once again be safely shuffled and batched together without fear of information from one pump “leaking” into the training data for another pump.

We choose to segment our data for several reasons. First, segmenting and predicting in this way avoids the possibility for models to make predictions based on future values – in other words, in our setup, models can only predict whether or not a pump will occur based on currently-known information. This ensures that models trained under this scheme have the capability to be deployed in a real-world, realtime anomaly detection scenarios on real exchanges. Additionally, the C-LSTM model incorporates an LSTM layer which faces performance limitations for longer sequence lengths (Vaswani et al., 2017). The data for each pump can also span several days, making it difficult to train models directly on a whole pump sequence due to memory constraints. Having a fixed segment length as opposed to the variable lengths of pump data also simplifies data loading.

Finally, segmenting our data into chunks makes it straightforward to perform undersampling on our data, which has been shown to boost model performance in anomaly detection scenarios by addressing the class imbalance – even in cases where the reduction does not bring the class ratio all the way down to 50:50 (Hasanin and Khoshgoftaar, 2018). Undersampling also naturally speeds up training, since it reduces the number of training samples. We therefore apply undersampling to our train data by choosing to keep only a random subset of proportion \( u \) of all segments that do not contain any anomaly labels. At the same time, we unconditionally include all segments that have an anomaly label anywhere.

Validation data undergoes a similar segment-
ing process, with the following exceptions: small pumps are not thrown out, and no undersampling is performed. This avoids unfairly skewing metrics in favor of our models.

Via hyperparameter tuning, we found that a segment length value of \( s = 15 \) resulted in the best validation performance as measured by F1 score. An undersampling proportion of \( u = 0.05 \) was used for all experiments except for the C-LSTM on the 15-second chunked dataset, in which \( u = 0.1 \) was used. For the C-LSTM model, batch sizes of 1200, 600, and 600 were used on the 5-second, 15-second, and 25-second chunked datasets respectively. For the Anomaly Transformer model, a batch size of 32 was used on all datasets. Likewise, the C-LSTM model uses precision-recall thresholds of 0.5, 0.4, and 0.65 on each the three datasets respectively, while the Anomaly Transformer model uses a threshold of 0.48 on all datasets.

In order to keep results consistent across all models, baseline results were also recomputed following the same train-validation split as our own models.

5.3 Baselines
The baseline results that we use to compare our models against are those of the random forest model employed by (La Morgia et al., 2020). For more context, the results of the statistical model introduced by (Kamps and Kleinberg, 2018) are also included in our results table, since they were used as a baseline for the random forest model originally. The metric that we will use to compare our models to the baseline results will be the aforementioned F1 score of each experiment.

5.4 Results
Both deep learning models were able to beat all previous classical and statistical approaches by a statistically significant margin \((p < 0.025)\) except for the C-LSTM model on the 15-second chunked dataset, which matched previous results. This demonstrates the effectiveness of deep learning in this previously unexplored area. State-of-the-art results across all models are bolded in our table. On average, we found that predictions using the 5-second chunked dataset are much less accurate than those on the 15-second and 25-second chunked dataset, which suggests that predicting anomalies using smaller chunk sizes corresponds to a harder problem in general. This corroborates findings from previous works (La Morgia et al., 2020).

| Model         | Chunk Size | Precision | Recall | F1   |
|---------------|------------|-----------|--------|------|
| Kamps (Init.) | 1 Hour     | 15.6%     | 96.7%  | 26.8%|
| Kamps (Bal.)  | 1 Hour     | 38.4%     | 93.5%  | 54.4%|
| Kamps (Strict)| 1 Hour     | 50.1%     | 75.0%  | 60.5%|
| RF            | 5 Secs     | 97.7%     | 71.6%  | 82.6 ± 0.0% |
| RF            | 15 Secs    | 98.0%     | 81.9%  | 89.2 ± 0.0% |
| RF            | 25 Secs    | 94.5%     | 83.8%  | 88.8 ± 0.0% |
| C-LSTM        | 5 Secs     | 91.2%     | 77.5%  | 83.7 ± 1.0% |
| C-LSTM        | 15 Secs    | 94.2%     | 84.9%  | 89.3 ± 0.4% |
| C-LSTM        | 25 Secs    | 94.2%     | 85.0%  | 89.3 ± 0.5% |
| Anom. Trans.  | 5 Secs     | 91.0%     | 87.7%  | 89.3 ± 0.4% |
| Anom. Trans.  | 15 Secs    | 93.0%     | 94.2%  | 93.6 ± 0.8% |
| Anom. Trans.  | 25 Secs    | 88.4%     | 90.0%  | 89.2 ± 0.3% |

To show the variance of our models, the F1 metrics are followed by a 95% confidence interval computed over \( n = 10 \) runs. Variance is measured after our efforts to maintain reproducibility via setting seeds. The random forest (RF) model’s lack of variance is due to our setting seeds in a similar manner to (La Morgia et al., 2020). Since the Kamps model is statistical in nature, it does not have variance across runs.

A comparison of the two deep learning models that we have employed shows that the transformer was able to beat the LSTM in all but the 25-second chunk size. This is unsurprising for two reasons: firstly, transformers are state-of-the-art for time-series anomaly detection problems to begin with, and secondly, a basic transformer design without the use of the Anomaly Transformer novelties already outperforms the random forest model results – albeit just barely.

We believe that the theoretical backing for the relative performance of the LSTM and Anomaly Transformer lies in the fact that for 5-second and 15-second chunk sizes, longer sequences are required to capture the same amount of information on average when compared to the larger 25-second chunks. Consequently, this implies a drop in the amount of temporal information present in the input for smaller chunk sizes, given a fixed segment length \( s = 15 \) as was used in this paper. We hypothesize that this difference is responsible for the LSTM’s faster drop in efficacy at smaller chunk sizes. However, more experiments must be done to reveal our models’ sensitivity to the aggregation chunk size of the dataset before confirming this.

6 Conclusion and Future Work
This paper studies the application of deep learning models to the crypto fraud detection problem space. We propose two separate models that can each reach state-of-the-art performance on the data available: the C-LSTM, and the Anomaly Trans-
former. Our results consequently show that both LSTMs and Transformers have the ability to outperform both classical ML models and statistical models on this dataset with relatively little effort. Future work includes fine-tuning these models to better-account for the volatility generally found in crypto, and exploring the potential for these models to detect pumps ahead of time as opposed to when they begin. Furthermore, the dataset provided to the community by (La Morgia et al., 2020), while incredibly useful, has been preprocessed to be optimized in some ways for use by their random forest model; for instance, an LSTM could make use of the actual price of an asset (as opposed to the average percent change in price) in ways that a random forest model cannot. Therefore, the release of the raw dataset, if possible, would push the capabilities of deep learning models in this domain even further.

References

Andy Brown, Aaron Tuor, Brian Hutchinson, and Nicole Nichols. 2018. Recurrent neural network attention mechanisms for interpretable system log anomaly detection. In Proceedings of the First Workshop on Machine Learning for Computing Systems, pages 1–8.

Coffeezilla. 2021. I joined a pump and dump scheme so you don’t have to - youtube.

Douglas J. Cumming, Sofia Johan, and Anshum Pant. 2019. Regulation of the crypto-economy: Managing risks, challenges, and regulatory uncertainty. Journal of Risk and Financial Management, 12(3).

J.T. Hamrick, Farhang Rouhi, Arghya Mukherjee, Amir Feder, Neil Gandal, Tyler Moore, and Marie Vasek. 2021. An examination of the cryptocurrency pump-and-dump ecosystem. Information Processing & Management, 58(4):102506.

Tawfiq Hasanin and Taghi Khoshgoftaar. 2018. The effects of random undersampling with simulated class imbalance for big data. In 2018 IEEE International Conference on Information Reuse and Integration (IRI), pages 70–79.

J. Kamps and B. Kleinberg. 2018. To the moon: defining and detecting cryptocurrency pump-and-dumps. Crime Sci.

Eamonn J. Keogh, Taposh Roy, Naik U, and Agrawal A. 2021. Multi-dataset time-series anomaly detection competition, competition of international conference on knowledge discovery & data mining.

Yogita Khatri. 2021. Crypto exchanges saw over $14 trillion in trading volume this year.

Tae-Young Kim and Sung-Bae Cho. 2018. Web traffic anomaly detection using c-lstm neural networks. Expert Systems with Applications, 106:66–76.

Donghwoon Kwon, Kathiravan Natarajan, Sang C Suh, Hyunjoo Kim, and Jinoh Kim. 2018. An empirical study on network anomaly detection using convolutional neural networks. In ICDCS, pages 1595–1598.

Massimo La Morgia, Alessandro Mei, Francesco Sassi, and Julinda Stefa. 2020. Pump and dumps in the bitcoin era: Real time detection of cryptocurrency market manipulations. In 2020 29th International Conference on Computer Communications and Networks (ICCCN), pages 1–9.

Kwei-Herng Lai, Daochen Zha, Junjie Xu, Yue Zhao, Guanchu Wang, and Xia Hu. 2022. Revisiting time series outlier detection: Definitions and benchmarks.

Teema Leangarun, Poj Tangamchit, and Suttipong Thajchayapong. 2018. Stock price manipulation detection using generative adversarial networks. In 2018 IEEE Symposium Series on Computational Intelligence (SSCI), pages 2104–2111.

Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vir, Puneet Agarwal, and Gautam Shroff. 2016. Lstm-based encoder-decoder for multi-sensor anomaly detection. arXiv preprint arXiv:1607.00148.

Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. 2019. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD ’19, page 2828–2837, New York, NY, USA. Association for Computing Machinery.

David Twomey and Andrew Mann. 2020. Fraud and manipulation within cryptocurrency markets. Corruption and fraud in financial markets: malpractice, misconduct and manipulation, 624.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. CoRR, abs/1706.03762.

Jiehui Xu, Haixu Wu, Jianmin Wang, and Ming-sheng Long. 2021. Anomaly transformer: Time series anomaly detection with association discrepancy. CoRR, abs/2110.02642.

Wenjie Yang, Ruofan Wang, and Bofan Wang. 2020. Detection of anomaly stock price based on time series deep learning models. In 2020 Management Science Informatization and Economic Innovation Development Conference (MSIEID), pages 110–114.