Anomaly Detection in Real-Time Gross Settlement Systems

Ron Triepels[1,3], Hennie Daniels[1,2] and Ronald Heijmans[3]
[1]*Tilburg University, Tilburg, The Netherlands*
[2]*Erasmus University, Rotterdam, The Netherlands*
[3]*De Nederlandsche Bank, Amsterdam, The Netherlands*

Keywords: Anomaly Detection, Neural Network, Autoencoder, Real-Time Gross Settlement System.

Abstract: We discuss how an autoencoder can detect system-level anomalies in a real-time gross settlement system by reconstructing a set of liquidity vectors. A liquidity vector is an aggregated representation of the underlying payment network of a settlement system for a particular time interval. Furthermore, we evaluate the performance of two autoencoders on real-world payment data extracted from the TARGET2 settlement system. We do this by generating different types of artificial bank runs in the data and determining how the autoencoders respond. Our experimental results show that the autoencoders are able to detect unexpected changes in the liquidity flows between banks.

1 INTRODUCTION

Financial market infrastructures play a vital role in the smooth functioning of the economy. They facilitate the clearing and settlement of monetary and other financial transactions. A particular type of system in these infrastructures is the Real-Time Gross Settlement (RTGS) system. An RTGS system is a system that settles transactions of its participants individually (gross) and immediately (real-time).

It is important to supervise the activities of banks in financial systems. This is because financial systems tend to exhibit a robust-yet-fragile nature (Gai and Kapadia, 2010). Banks may be well capable of absorbing financial imbalances. However, if they start experiencing liquidity issues, than these issues can quickly propagate to many other banks due to their interconnectedness. For this reason, much research has recently been devoted to study the topology of payment networks and its impact on systemic risk. Systemic risk is the risk associated with any event that threatens the stability of a financial system as a whole (Berndsen et al., 2016). It is commonly measured by concepts of network theory, e.g. the centrality or degree of a payment network.

In this paper, we present a different approach to measure systemic risk. We apply anomaly detection on the payment data generated by a RTGS system. Generally, an anomaly is defined as a pattern that does not conform to expected behavior (Chandola et al., 2009). Accordingly, anomaly detection is the task of automatically identifying anomalies in data. In our case, anomalies are particular configurations of a payment network that deviate considerably from the expected norm. They are caused by financial stress or unwanted payment behavior.

The application of anomaly detection on payment data is promising. Payment data provide an accurate and system-wide overview of how banks manage their liquidity over time. Analyzing this data with anomaly detection allows to automatically identify unusual payment behavior and may help supervisors to initiate timely interventions. To the best of our knowledge, this is the first time such analysis is applied on payment data.

We define the anomaly detection task and discuss how it can be performed by an autoencoder. An autoencoder is a feed-forward neural network that learns features from data by compressing it to a lower dimensional space, and accordingly, reconstructing it back in the original space. Furthermore, we evaluate the performance of two types of autoencoders in a series of bank run simulations. The simulations show that the autoencoders are able to detect unexpected changes in the liquidity flows between banks.

The remainder of this paper is organized as follows. Section 2 provides a brief overview of related literature on payment systems and anomaly detection. Section 3 defines the autoencoder and discusses how it can be applied for anomaly detection. Section 4

Triepels, R., Daniels, H. and Heijmans, R.
Anomaly Detection in Real-Time Gross Settlement Systems.
DOI: 10.5220/0006333004330441
In *Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017) - Volume 1*, pages 433-441
ISBN: 978-989-758-247-9

describes the experimental setup of the bank run simulations. Section 5 discusses the results of the simulations. Finally, section 6 concludes the paper.

2 RELATED LITERATURE

Many explanatory studies have been conducted on payment data to study the properties and behavior of banks in settlement systems. For example, algorithms have been developed to detect loans in unsecured interbank markets (Furfine, 1999; Armantier and Copeland, 2012; Arciero et al., 2016). Moreover, network analysis has been applied to study the topology of payment networks and its impact on financial contagion and systemic risk (Allen and Gale, 2000; León and Pérez, 2014; Berndsen et al., 2016). The insights gained from such studies are applied to develop indicators of liquidity and systemic risk, see e.g. (Heijmans and Heuver, 2014).

Simulations are also commonly applied to study settlement systems. A well-known simulation environment is the Bank of Finland simulator. It implements many types of settlement systems and allows to resettle payment data under different conditions (Leinonen and Soramäki, 2005). This is usually done to measure the impact on a settlement system when characteristics of liquidity flows, such as their value or timing, are changed (Laine et al., 2013).

Anomaly detection, in particular, has been successfully applied on other types of financial data such as stock market data. The goal of stock fraud detection is to detect trades that violate securities laws. This includes the detection of unprofitable trades by brokers (Ferdousi and Maeda, 2006) and abnormal stock price changes caused by stock price manipulation (Kim and Sohn, 2012). Stock market data has also been combined with options data and news data to detect trades that were made based on information that was not available to the general public (Donoho, 2004). Multiple techniques were applied to detect these 'insider' transactions, including decision trees and neural networks.

Another related application is credit card fraud detection. In this case, anomaly detection is applied on credit card transactions to detect suspicious spending patterns. Many techniques have been proposed for this task including: neural networks (Ghosh and Reilly, 1994; Maes et al., 2002), Bayesian networks (Maes et al., 2002), self-organizing-maps (Zaslavsky and Strizhak, 2006; Quah and Sriganesh, 2008), association rules (Sánchez et al., 2009), and hidden Markov models (Srivastava et al., 2008).

3 ANOMALY DETECTION IN RTGS SYSTEMS

In this section, we define the problem of detecting anomalies in RTGS systems based on lossy compression. Moreover, we describe how an autoencoder can be employed for this task.

3.1 Definitions

Let $B = \{b_1, \dots, b_n\}$ be a set of n banks participating in a RTGS system. The banks initiate payments to each other which are settled by the settlement system in real-time and without any netting procedures. Furthermore, let $T = < t_1, \dots, t_m >$ be an ordered set of m time intervals, where $t_1 = [\tau_0, \tau_1)$, $t_2 = [\tau_1, \tau_2)$, and so on. We assume that the time intervals in T are consecutive and of equal duration. They might, for example, denote the operating hours or minutes of the settlement system.

The liquidity that banks transmit to each other though the settlement system over time is recorded. Let $D = \{\mathbf{A}^{(1)}, \dots, \mathbf{A}^{(m)}\}$ be a set of m matrices where each $\mathbf{A}^{(k)} \in D$ is the n by n matrix:

$$\mathbf{A}^{(k)} = \begin{bmatrix} a_{11}^{(k)} & \cdots & a_{1n}^{(k)} \\ \vdots & \ddots & \vdots \\ a_{n1}^{(k)} & \cdots & a_{nn}^{(k)} \end{bmatrix} \tag{1}$$

Each element $a_{ij}^{(k)} \in [0, \infty)$ denotes the total amount of liquidity that b_i sends to b_j in time interval t_k. Notice that this also includes $a_{ii}^{(k)}$. This liquidity flow denotes the total liquidity transmitted by b_i at time interval t_k between its own accounts. We also call $\mathbf{A}^{(k)}$ a liquidity matrix. It can be interpreted as a payment network consisting of banks (nodes) which are interconnected by liquidity flows (edges). The elements of $\mathbf{A}^{(k)}$ define the weights associated with the edges of the payment network at time interval t_k. For analysis purposes, we derive from $\mathbf{A}^{(k)}$ the vectorized representation:

$$\mathbf{a}^{(k)} = [a_{11}^{(k)}, \dots, a_{n1}^{(k)}, \dots, a_{1n}^{(k)}, \dots, a_{nn}^{(k)}]^T \tag{2}$$

where, $\mathbf{a}^{(k)}$ is a n^2 column vector consisting of all columns of $\mathbf{A}^{(k)}$ vertically enumerated. We also call $\mathbf{a}^{(k)}$ a liquidity vector.

3.2 Anomaly Detection Task

Anomalies in RTGS systems can be detected by determining how well liquidity vectors can be reconstructed after being compressed by lossy compression. Lossy compression is a particular form of compression in which data may not be entirely recovered. Instead, only the essential features of data are compressed. This makes lossy compression useful to detect anomalous liquidity vectors. If the reconstruction error of a liquidity vector is low, then it fits some frequently recurring pattern that the compression model has learned to compress well. However, if the reconstruction error is large, then the model does not recognize the liquidity flows and fails to reconstruct their values. One may conclude from this observation that the liquidity vector is generated by a different underlying process, and thus, is an anomaly.

Let M be a lossy compression model. We measure the reconstruction error of $\mathbf{a}^{(k)}$ after it is compressed and reconstructed by M by function RE:

$$\mathrm{RE} : D \rightarrow [0, \infty) \tag{3}$$

$\mathrm{RE}(\mathbf{a}^{(k)})$ is the non-negative reconstruction error aggregated over all liquidity flows between the banks at time interval t_k. $\mathbf{a}^{(k)}$ is considered anomalous if the reconstruction error is high, i.e. $\mathrm{RE}(\mathbf{a}^{(k)}) \geq \varepsilon$ where $\varepsilon > 0$ is a threshold. The objective of our anomaly detection task is to find all liquidity vectors in D whose reconstruction error is higher than ε. We define this task as:

Definition 1 (Anomaly Detection Task)**.** *Given a set of liquidity vectors D, a lossy compression model* M_θ *with parameters* θ*, and threshold* ε*, find the anomaly set* $F = \{\mathbf{a}^{(k)} \in D | RE(\mathbf{a}^{(k)}) \geq \varepsilon\}$.

3.3 Autoencoder

We employ a three-layer autoencoder (Rumelhart et al., 1985; Hawkins et al., 2002) for M to compress liquidity vectors. An autoencoder is an artificial feedforward neural network that is trained to reconstruct the input layer at the output layer. It does this by processing the input through a bottleneck layer in which a set of neurons form a representation of the input in a lower dimensional space. The architecture of the autoencoder is depicted by Figure 1.

For input $\mathbf{a}^{(k)}$, the autoencoder estimates a reconstruction $\hat{\mathbf{a}}^{(k)}$ of $\mathbf{a}^{(k)}$ via a hidden layer consisting of l neurons. The reconstruction mapping is composed of two functions ϕ and ψ:

$$\phi : \mathbb{R}^{n^2} \rightarrow \mathbb{R}^l \tag{4}$$

$$\psi : \mathbb{R}^l \rightarrow \mathbb{R}^{n^2} \tag{5}$$

Here, ϕ is also called the encoder function and ψ the decoder function. First, $\mathbf{a}^{(k)}$ is encoded by ϕ in l-dimensional space by processing it through the hidden layer:

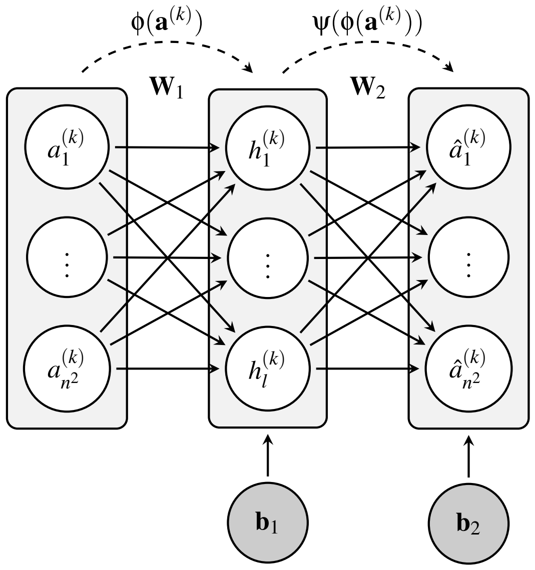

Figure 1: The architecture of an autoencoder consisting of an: input layer (left), hidden layer (middle), and output layer (right).

$$\phi(\mathbf{a}^k) = f^{(l)}(\mathbf{W}_1\mathbf{a}^{(k)} + \mathbf{b}_1) \tag{6}$$

where, $\mathbf{W}_1$ is a l by n^2 matrix of weights, $\mathbf{b}_1$ is a vector of l bias terms, and $f^{(l)}(\mathbf{y}) = [f(y_1), \ldots, f(y_l)]$ is a set of activation functions that are applied to $\mathbf{y}$ element-wise. Potential functions for f are the linear (identity) function or sigmoid function. The result of the encoding $\phi(\mathbf{a}^{(k)}) = [h_1, \ldots, h_l]$ is a vector of l hidden neuron activations forming a (compressed) representation of $\mathbf{a}^{(k)}$ in $\mathbb{R}^l$. Then, $\phi(\mathbf{a}^{(k)})$ is decoded back by ψ in n^2-dimensional space by processing it through the output layer:

$$\psi(\phi(\mathbf{a}^{(k)})) = g^{(n^2)}(\mathbf{W}_2\phi(\mathbf{a}^{(k)}) + \mathbf{b}_2) \tag{7}$$

where, $\mathbf{W}_2$ is a n^2 by l matrix of weights, $\mathbf{b}_2$ is a vector of n^2 bias terms, and $g^{(n^2)}(\mathbf{y})$ is a set of activation functions. The result of the decoding $\psi(\phi(\mathbf{a}^{(k)})) = [\hat{a}_1^{(k)}, \ldots, \hat{a}_{n^2}^{(k)}]$ is a vector of n^2 outputs forming a reconstruction of $\phi(\mathbf{a}^{(k)})$ in $\mathbb{R}^{n^2}$.

The goal of the autoencoder is to learn a mapping from the input layer to the output layer such that $\mathbf{a}^{(k)} \approx \psi(\phi(\mathbf{a}^{(k)}))$ for all $\mathbf{a}^{(k)} \in D$. The quality of the reconstruction depends on the number of neurons in the hidden layer. If the number of neurons is too large, then the autoencoder may approximate the identity mapping and simply copy inputs from the input layer to the output layer. However, if the number of neurons is limited, than the autoencoder is forced to compress the liquidity vectors and approximate their intrinsic structure.

Several variations have been proposed to prevent autoencoders from approximating the identity mapping. These include: sparse autoencoders (Ng, 2011), denoising autoencoders (Vincent et al., 2008), and contractive autoencoders (Rifai et al., 2011). However, as a first attempt, we only consider the classic implementation of an autoencoder in this paper.

3.4 Reconstruction Error

The autoencoder estimates the reconstruction error at different aggregation levels. For an individual liquidity flow $a_{ij}^{(k)}$, the reconstruction error is estimated by taking the squared difference between its reconstructed transaction value and original transaction value:

$$\mathrm{RE}(a_{ij}^{(k)}) = \frac{1}{2}(\psi(\phi(\mathbf{a}^{(k)}))_{i+n(j-1)} - a_{ij}^{(k)})^2 \quad (8)$$

Similarly, the reconstruction error of all liquidity flows combined at time interval t_k can be estimated by taking the squared ℓ_2-norm of the difference between the reconstructed liquidity vector and original liquidity vector:

$$\mathrm{RE}(\mathbf{a}^{(k)}) = \frac{1}{2}||\psi(\phi(\mathbf{a}^{(k)})) - \mathbf{a}^{(k)}||^2 \quad (9)$$

Finally, by taking the mean of the reconstruction error of all liquidity vectors in D we obtain the overall Mean Reconstruction Error (MRE):

$$\mathrm{MRE}(D) = \frac{1}{m}\sum_{k=1}^{m}\mathrm{RE}(\mathbf{a}^{(k)}) \quad (10)$$

3.5 Model Learning

The parameters $\theta = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2\}$ of the autoencoder are estimated from historic liquidity flows. We do this by minimizing the MRE, i.e.:

$$\theta = \underset{\mathbf{W}_1,\mathbf{W}_2,\mathbf{b}_1,\mathbf{b}_2}{\arg\min}\ \mathrm{MRE}(D) \quad (11)$$

There are many approaches to solve the optimization problem in 11. One approach is to apply (stochastic) gradient descent in conjunction with back-propagation to efficiently calculate all gradients during the optimization process (Werbos, 1982; Bottou, 2004). In this case, the parameters are iteratively updated proportional to the negative gradient of the MRE. This process is repeated until the parameters converge to a configuration for which the MRE is (locally) minimum.

4 EXPERIMENTAL SETUP

In this section, we describe an experiment in which two autoencoders were evaluated on real-world payment data. We elaborate on the characteristics of the payment data, the implementation of the autoencoders, and the way they were evaluated in a series of bank run simulations.

4.1 Payment Data

A sample of payment data was extracted from the TARGET2 settlement system. TARGET2 is the RTGS system of the Eurosystem. It facilitates the settlement of large domestic and cross-border payments in euros for most European countries.

The sample focused on the Dutch part of the settlement system. It included about two million client payments which were settled between January and October 2014 among the twenty largest banks[1]. These are payments that were initiated in TARGET2 by the banks on behave of their customers. The accounts of the Dutch Ministry of Finance and De Nederlandsche Bank were excluded from the sample.

We aggregated the payments over 8,561 time intervals that each spanned fifteen minutes and derived the corresponding liquidity vectors. The liquidity flows in these vectors were transformed by a log transformation to make their highly skewed distribution less skewed. Min-max normalization was in turn performed to normalize their values to the $[0,1]$ interval.

Accordingly, we partitioned the liquidity vectors in separate sets for training and evaluation purposes. The parameters of the autoencoders were learned from a training set. This set contained 5081 liquidity vectors corresponding to six months (March until August). A holdout set containing 1680 liquidity vectors corresponding to the first two months (January and February) was set aside to optimize the number of neurons of the autoencoders. Finally, we evaluated the autoencoders on a test set which contained 1800 liquidity vectors corresponding to the final two months (September and October).

4.2 Implementation

We implemented two autoencoders: one having linear activations in the hidden layer and sigmoid activations in the output layer, and the other, having sigmoid activations for both the hidden layer and output layer. We refer to these networks as the linear autoencoder

[1]The size of the banks were determined based on their total turnover.

and sigmoid autoencoder respectively. The initial parameters of the networks were sampled from a normal distribution with zero mean and a variance of 0.1 for symmetry breaking. Stochastic gradient descent (Bottou, 2004) in conjunction with back-propagation was in turn applied to learn the parameters from the training set. This was performed for 30 iterations through the training set with a fixed learning rate.

The number of hidden neurons was optimized by a grid-search. During the grid search, a set of autoencoders having a different number of neurons $l \in \{10, 20, \ldots, 400\}$ in the hidden layer were learned from the training set. The MRE of these networks were evaluated on the holdout set. In particular, we investigated the point were adding more neurons did not yielded a better error.

Moreover, we determined whether the autoencoders approximated the identity mapping by evaluating their MRE on a set of uniformly sampled liquidity vectors. In the optimal case, the MRE of the autoencoders on these random liquidity vectors equals a lower bound. Let $X = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\}$ be a set of m liquidity vectors for n banks. Each $\mathbf{x}^{(k)} \sim U(0,1)$ is sampled from a uniform distribution. An autoencoder achieves the lowest error when it reconstructs the mean $\hat{\mathbf{x}}^{(k)} = [\frac{1}{2}, \ldots, \frac{1}{2}]$ for all $\mathbf{x}^{(k)} \in X$. The corresponding lower bound of the MRE is:

$$\mathrm{MRE}_l = \frac{1}{2}\mathbb{E}\left(\frac{1}{m}\sum_{k=1}^{m} ||\mathbf{c} - \mathbf{x}^{(k)}||^2\right) \tag{12}$$

$$= \frac{1}{2}\int_0^1 \frac{1}{m}\sum_{k=1}^{m} ||\mathbf{c} - \mathbf{x}^{(k)}||^2 d\mathbf{x}^{(k)} \tag{13}$$

$$= \frac{1}{24}n^2 \tag{14}$$

where, $\mathbf{c} = [\frac{1}{2}, \ldots, \frac{1}{2}]^T$ is a column vector of n^2 elements. In the case an autoencoder achieves an error $\mathrm{MRE}(X) < \mathrm{MRE}_l$, it is modeling noise because X is random. This is only possible if it is approximating the identity mapping.

4.3 Bank Run Simulation

We evaluated the performance of the autoencoder in a series of simulations. In particularly, we evaluated how well they were able to detect different types of artificial bank runs.

We simulated a bank run as follows. First, choose a bank b_i as the subject bank. Then, add additional liquidity to the outgoing liquidity flows from b_i to the remaining banks according:

$$a_{ij}^{(k)} := a_{ij}^{(k)} + P(k) \cdot C(k) \tag{15}$$

where, $P(x) \in \{0,1\}$ denotes whether liquidity is added at time interval t_x and $C(x) \in [0, \infty)$ denotes the amount of additional liquidity. We sampled $P(x)$ randomly from a binomial distribution with probability p_x. This probability increased exponentially during the bank run:

$$p_x = \begin{cases} p_s + (p_e - p_s)(\frac{x-s}{d})^r, & \text{if } s \leq x \leq s+d \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

Here, s is the time interval t_s at which the bank run starts, d is the duration of the bank run, r controls the rate of increase, and p_s and p_e are start and end values of p_x respectively. Moreover, we sampled $C(x)$ from an exponential distribution with rate parameter λ_x. This parameter also increased exponentially during the bank run:

$$\lambda_x = \begin{cases} \lambda_s + (\lambda_e - \lambda_s)(\frac{x-s}{d})^r, & \text{if } s \leq x \leq s+d \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

Here, λ_s and λ_e are the rate parameters at the start and end of the bank run respectively.

Table 1: The parameters of the simulated bank runs.

Bank Run	d	r	p_s	p_e	λ_s	λ_e
A	196	2	0	0.8	10^{-4}	10^{-7}
B	196	6	0	0.8	10^{-4}	10^{-7}
C	392	2	0	0.8	10^{-4}	10^{-7}
D	392	6	0	0.8	10^{-4}	10^{-7}

Table 1 summarizes the parameters of the simulated bank runs. We chose a large commercial bank as the subject bank. All bank runs started at the end of the test set and lasted $d = 196$ or $d = 392$ time intervals. This corresponds to seven and fourteen operational days respectively. Furthermore, we applied a rate parameter of $r = 2$ to mimic a bank run which developed slowly over time and $r = 6$ to mimic a bank run which developed more radically over time. The probability that liquidity was added during the bank runs increased from $p_s = 0$ to $p_e = 0.8$.

The liquidity rate parameters were adjusted to the magnitude of the liquidity flows of the subject bank. It increased from $\lambda_s = 10^{-4}$ to $\lambda_e = 10^{-7}$ in all simulations. This means that at the start of the bank run on average 10,000 euro was added to the outflow of the subject bank which increased to 10,000,000 euro in the final time interval.

5 RESULTS

Figure 2a shows the MRE of the holdout set estimated by the autoencoders while having a different number of neurons in the hidden layer. Initially, the

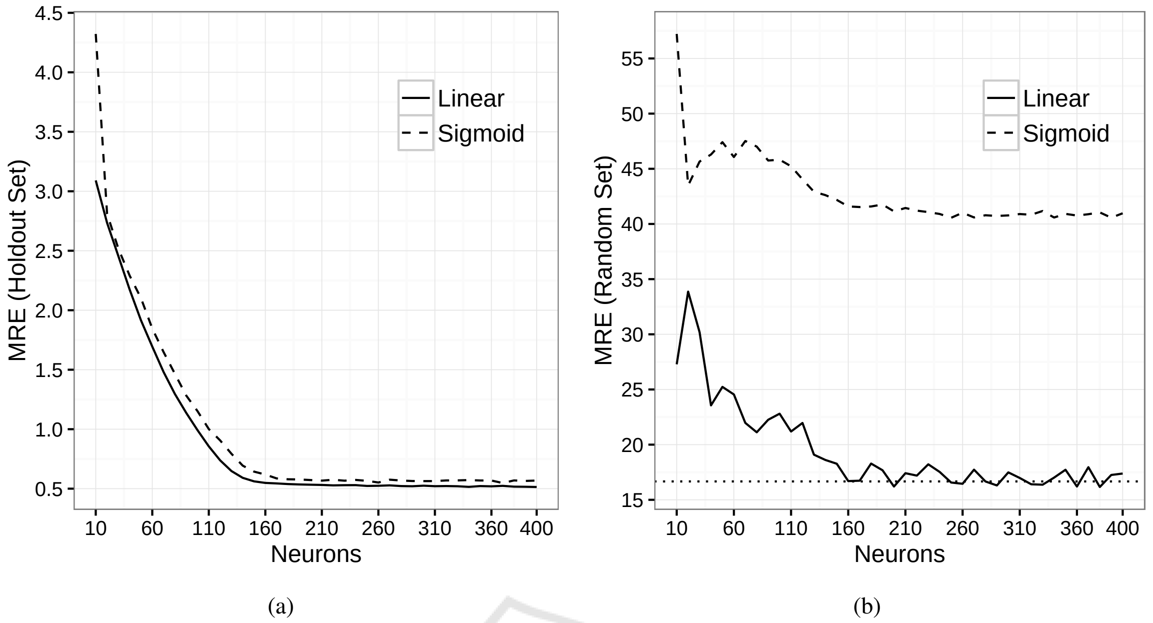

Figure 2: (a) The MRE of the holdout set estimated by the linear and sigmoid autoencoder having a different number of neurons in the hidden layer. (b) The same graph as (a) but instead estimated on a set of random liquidity vectors. The dotted line represents the lower bound of the MRE.

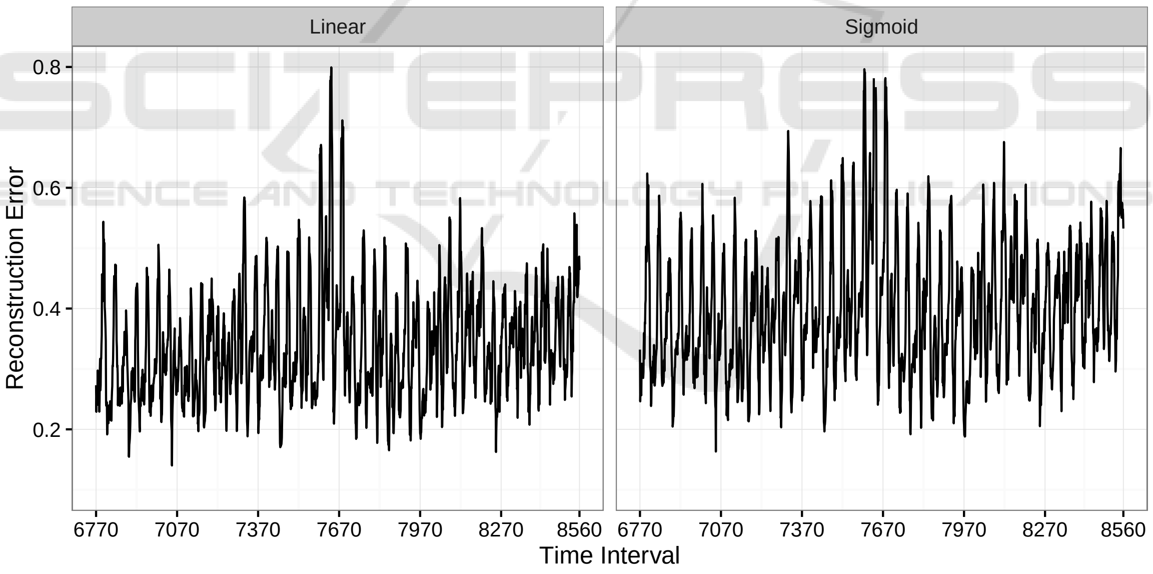

Figure 3: The reconstruction error of the original test set estimated by the linear and sigmoid autoencoder for each time interval. The error curves are smoothed with a rolling average of ten time intervals to make their trend more clearly visible.

MRE quickly decreased when increasing the number of neurons. Then, after about 160 neurons, it saturated. Figure 2b shows the same graph estimated on the random set. The linear autoencoder reconstructed the random set better than the sigmoid autoencoder. After 160 neurons, it achieved an optimal reconstruction as its MRE closely approximated the lower bound. It even crossed the lower bound several times which suggests that, in these particular cases, it was approximating the identity mapping. Given these results, we chose to use 160 neurons in the hidden layer of the autoencoders.

Moreover, the results show that liquidity vectors contain distinctive payment patterns which an autoencoders is able to pick up very well. If we compare the MRE of the holdout set and the random set, then we see that the autoencoders achieve a much lower error on the holdout set. This would not be possible if the

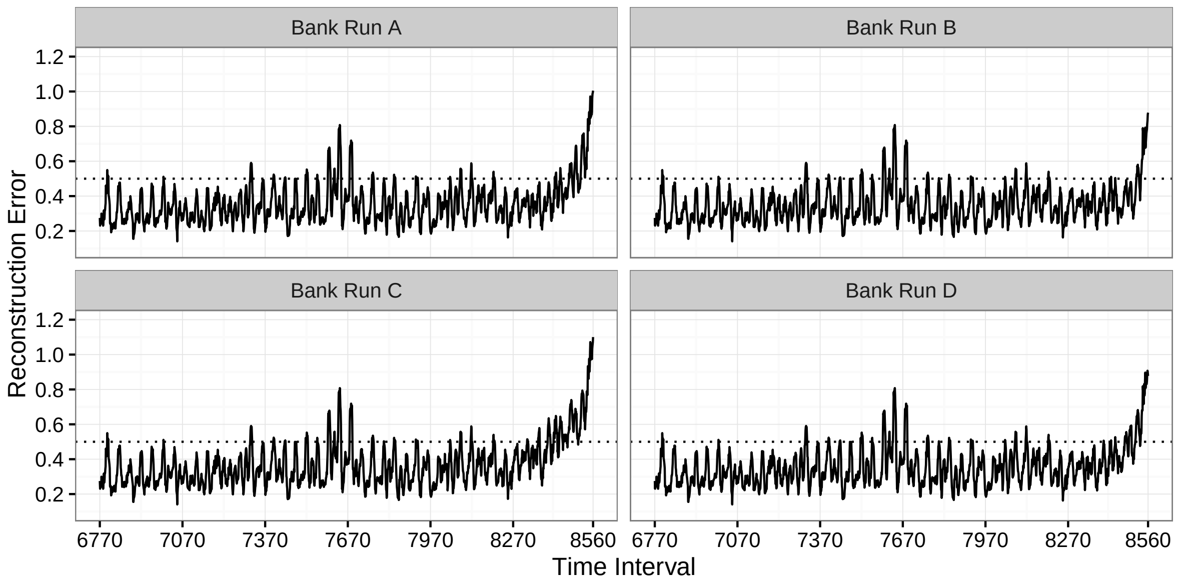

Figure 4: The reconstruction error of the manipulated test sets estimated by the linear autoencoder for each time interval. We simulated the bank runs at the final time intervals. The error curves are smoothed with a rolling average of ten time intervals to make their trend more clearly visible. Moreover, the dotted line represents the anomaly threshold $\varepsilon = 0.5$.

holdout set was also randomly generated.

Before running the actual simulations, we determined how well the autoencoders were able to reconstruct the original test set. Figure 3 shows the reconstruction error estimated by the autoencoders for each time interval. The error curves of the autoencoders are quite similar and exhibit fluctuations. Moreover, they display a short period of time, around time interval 7670, were both autoencoders had difficulties reconstructing the liquidity vectors. Closer inspection revealed that these high errors were caused by a few very large liquidity flows. No clear signs of stress could be identified. Instead, we suspect that the large errors were caused by random influences that are subject to the financial intermediation process.

We simulated the bank runs at the end of the test set. Figure 4 shows the reconstruction error of the manipulated test sets estimated by the linear autoencoder for each time interval[2]. The error curves were smoothed by a rolling average of ten time intervals to make their trend clearly visible. In addition, we applied an anomaly threshold of $\varepsilon = 0.5$.

The error curves clearly highlight the artificial bank runs at the final time intervals. During these time intervals, the reconstruction error increased rapidly as the payment network started to change unexpectedly. In particular, the stressed bank became more centrally positioned having high outgoing liquidity flows to the remaining banks in the payment network. We can see this when inspecting the reconstruction error of the final liquidity matrices, see Figure 5. The high outgoing liquidity flows of the stressed bank could not be accurately reconstructed and caused a high reconstruction error during the bank runs.

[2]The error curves of the sigmoid autoencoder are very similar and omitted for brevity.

6 CONCLUSIONS

We have introduced a method to detect system-level anomalies in a RTGS system. This method involves training an autoencoder to reconstruct a set of liquidity vectors. Our experimental results show that liquidity vectors contain distinctive features of a payment network which an autoencoder is able to capture very well. Furthermore, the reconstruction error made by a well-trained autoencoder after compressing and reconstructing the liquidity vectors reflects anomalous changes in the liquidity flows between banks.

In the future, we plan to further improve our work in several aspects. This includes: evaluating the proposed method on larger payment networks that are subject to real-world stress, explaining the most likely cause of anomalies, and incorporating time dependencies between liquidity vectors.

ACKNOWLEDGEMENTS

We would like to thank Ron Berndsen for his helpful suggestions and feedback.

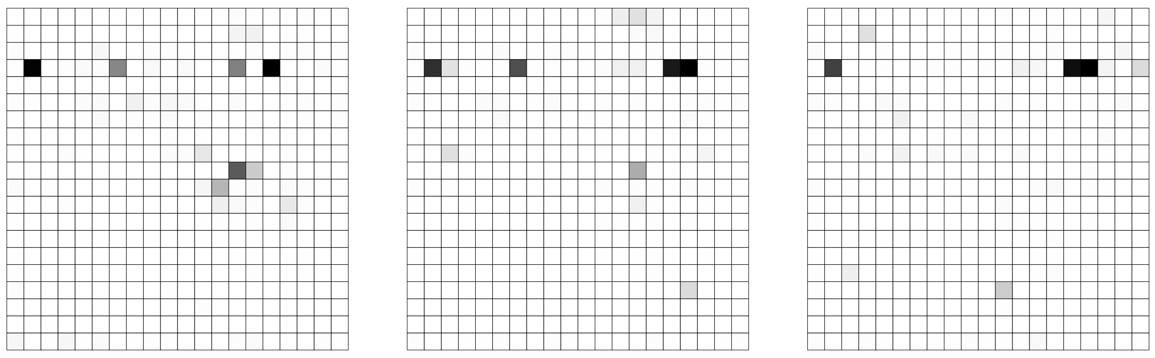

Figure 5: The reconstruction error of the liquidity matrices at (a) t_{8528}, (b) t_{8529}, and (c) t_{8530} of bank run C estimated by the linear autoencoder. The intensity of each element indicates the error made by the autoencoder for the corresponding liquidity flow. The fourth row from the top represents the outgoing liquidity flows of the stressed bank.

REFERENCES

Allen, F. and Gale, D. (2000). Financial Contagion. *Journal of Political Economy*, 108:1–33.

Arciero, L., Heijmans, R., Heuver, R., Massarenti, M., Picillo, C., and Vacirca, F. (2016). How to Measure the Unsecured Money Market? The Eurosystem's Implementation and Validation using TARGET2 Data. *International Journal of Central Banking*, March:247–280.

Armantier, O. and Copeland, A. (2012). Assessing the Quality of Furfine-based Algorithms. *Federal Reserve Bank of New York Staff Report*, 575.

Berndsen, R. J., León, C., and Renneboog, L. (2016). Financial Stability in Networks of Financial Institutions and Market Infrastructures. *Journal of Financial Stability*.

Bottou, L. (2004). Stochastic Learning. In *Advanced lectures on machine learning*, pages 146–168. Springer.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly Detection: A Survey. *ACM Comput. Surv.*, 41(3):15:1–15:58.

Donoho, S. (2004). Early Detection of Insider Trading in Option Markets. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 420–429. ACM.

Ferdousi, Z. and Maeda, A. (2006). Unsupervised Outlier Detection in Time Series Data. In *22nd International Conference on Data Engineering Workshops (ICDEW'06)*, pages 51–56. IEEE.

Furfine, C. (1999). The Microstructure of the Federal Funds Market. *Financial Markets, Institutions and Instruments*, 8:24–44.

Gai, P. and Kapadia, S. (2010). Contagion in financial networks. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, page rspa20090410. The Royal Society.

Ghosh, S. and Reilly, D. L. (1994). Credit card fraud detection with a neural-network. In *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on*, volume 3, pages 621–630. IEEE.

Hawkins, S., He, H., Williams, G., and Baxter, R. (2002). Outlier Detection Using Replicator Neural Networks. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 170–180. Springer.

Heijmans, R. and Heuver, R. (2014). Is this Bank Ill? The Diagnosis of Doctor TARGET2. *Journal of Financial Market Infrastructures*, 2(3).

Kim, Y. and Sohn, S. Y. (2012). Stock fraud detection using peer group analysis. *Expert Systems with Applications*, 39(10):8986–8992.

Laine, T., K., K., and & Hellqvist, M. (2013). *Simulations Approaches to Risk, Efficiency, and Liquidity Usage in Payment Systems*. IGI Global.

Leinonen, H. and Soramäki, K. (2005). Optimising Liquidity Usage and Settlement Speed in Payment Systems. In Leinonen, H., editor, *Liquidity, risks and speed in payment and settlement systems - a simulation approach.*, Proceedings from the Bank of Finland Payment and Settlement System Seminars 2005, pages 115–148.

León, C. and Pérez, J. (2014). Assessing financial market infrastructures' systemic importance with authority and hub centrality. *Journal of Financial Markt Infrastructures*.

Maes, S., Tuyls, K., Vanschoenwinkel, B., and Manderick, B. (2002). Credit Card Fraud Detection Using Bayesian and Neural Networks. In *Proceedings of the 1st international naiso congress on neuro fuzzy technologies*, pages 261–270.

Ng, A. (2011). Sparse autoencoder. *CS294A Lecture notes*, 72:1–19.

Quah, J. T. and Sriganesh, M. (2008). Real-time credit card fraud detection using computational intelligence. *Expert systems with applications*, 35(4):1721–1732.

Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. (2011). Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In *Proceedings of* *the 28th international conference on machine learning (ICML-11)*, pages 833–840.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning Internal Representations by Error Propagation. Technical report, DTIC Document.

Sánchez, D., Vila, M., Cerda, L., and Serrano, J.-M. (2009). Association rules applied to credit card fraud detection. *Expert Systems with Applications*, 36(2):3630–3640.

Srivastava, A., Kundu, A., Sural, S., and Majumdar, A. (2008). Credit Card Fraud Detection Using Hidden Markov Model. *IEEE Transactions on dependable and secure computing*, 5(1):37–48.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM.

Werbos, P. J. (1982). Applications of Advances in Nonlinear Sensitivity Analysis. In *System modeling and optimization*, pages 762–770. Springer.

Zaslavsky, V. and Strizhak, A. (2006). Credit Card Fraud Detection Using Self-Organizing Maps. *Information and Security*, 18:48.