Anomaly Detection in Industrial Software Systems
Using Variational Autoencoders

Tharindu Kumarage, Nadun De Silva, Malsha Ranawaka,
Chamal Kuruppu and Surangika Ranathunga
Department of Computer Science and Engineering, University of Moratuwa, Katubedda, Sri Lanka

Keywords: Anomaly Detection, Industrial Software Systems, Variational Autoencoder, VAE, DBSCAN.

Abstract: Industrial software systems are known to be used for performing critical tasks in numerous fields. Faulty conditions in such systems can cause system outages that could lead to losses. In order to prevent potential system faults, it is important that anomalous conditions that lead to these faults are detected effectively. Nevertheless, the high complexity of the system components makes anomaly detection a high dimensional machine learning problem. This paper presents the application of a deep learning neural network known as Variational Autoencoder (VAE), as the solution to this problem. We show that, when used in an unsupervised manner, VAE outperforms the well-known clustering technique DBSCAN. Moreover, this paper shows that higher recall can be achieved using the semi-supervised one class learning of VAE, which uses only the normal data to train the model. Additionally, we show that one class learning of VAE outperforms semi-supervised one class SVM when training data consist of only a very small amount of anomalous samples. When a tree based ensemble technique is adopted for feature selection, the obtained results evidently demonstrate that the performance of the VAE is highly positively correlated with the selected feature set.

1 INTRODUCTION

Software systems are becoming increasingly common in industrial applications, being utilized for solving various complicated problems. Most of the applications of industrial software systems are in critical scenarios where any failure would result in huge losses.

An industrial software system is a combination of multiple components, each consisting of a large number of attributes. Each attribute in a component is a statistical measure of a certain aspect of the component. For example, queue size of requests received by a component is an indication of the network connection to the component, as well as the memory usage of the component. Monitoring these attributes manually or using a rule based system for anomaly detection is not feasible or maintainable in the long run. Hence, automated mechanisms for anomaly detection are required.

One widely incorporated approach for detecting faults automatically in industrial software systems is based on machine learning techniques. The techniques that have been discovered over the years can be divided into supervised, semi-supervised, and unsupervised (Agrawal and Agrawal, 2015).

Among them, supervised and semi-supervised methods have proven to be the best performing anomaly detection techniques. However, in order to apply a supervised technique for an industrial software system in a production environment, large datasets containing system statistics need to be labeled as anomalous and non-anomalous before the training phase (Ranaweera et al., 2017). Moreover, new datasets would require being labeled periodically to ensure that the models used for detecting anomalies are relevant to the current conditions.

Thus, unsupervised techniques are the most practical solution that can be used for anomaly detection in industrial software systems. Most of the unsupervised clustering and deep learning techniques introduced over the years have been used for anomaly detection in general (Willsky, 1976; Tan et al., 2012; Schneider et al., 2015). Out of the unsupervised deep learning techniques, Autoencoders are the most known method for anomaly detection in many domains (Sakurada and Yairi, 2014). However, these novel deep learning techniques have not been used for anomaly detection in the industrial software systems domain.

This paper presents a machine learning framework

Kumarage, T., Silva, N., Ranawaka, M., Kuruppu, C. and Ranathunga, S.
Anomaly Detection in Industrial Software Systems - Using Variational Autoencoders.
DOI: 10.5220/0006600304400447
In *Proceedings of the 7th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2018)*, pages 440-447
ISBN: 978-989-758-276-9

based on Variational Autoencoders (VAE) for detecting anomalies in industrial software systems. VAE shares the same decoder and encoder structure of a normal Autoencoder. However, here the encoder acts as a variational inference network, which makes the VAE outperform normal Autoencoders (Kingma and Welling, 2014; An and Cho, 2015).

The industrial software system we use is a commercial stock trading platform. We show that VAE outperforms DBSCAN (Density Based Spatial Clustering of Applications with Noise), which is a density based clustering method popularly used in anomaly detection. Moreover, this paper presents the application of one class learning method to improve the recall of the VAE. According to this method, VAE is trained only using the normal data and then used to detect previously unseen anomalous data points. This semi-supervised VAE outperforms the semi-supervised one class SVM classification technique when the anomaly ratio of the training data is trivial. Furthermore, previous research that employed Autoencoders has not exploited any feature selection since Autoencoders are said to be usable in high dimensional problems (Sakurada and Yairi, 2014; Aygun and Yavuz, 2017). However, in contrary, our research demonstrates how the performance of the VAE can be extensively increased by selecting discriminative features, which supports modeling the latent variables more accurately.

The rest of the paper is organized as follows. Section 2 describes the related work corresponding to anomaly detection in industrial software systems, and section 3 describes the industrial software system used for the experiments. Anomaly detection is outlined in section 4, and section 5 discusses the importance of feature selection for the VAE. Section 6 presents the evaluation, followed by the conclusion in Section 7.

2 RELATED WORK

2.1 Anomaly Detection

Traditional methods for anomaly detection involved probabilistic (Guo et al., 2006), and statistical models (Idé and Kashima, 2004). Although these methods exhibited agreeable results, they were not scalable due to the increasing complexity of the systems concerned.

With the breakthrough of machine learning techniques used in various domains, new supervised, semi-supervised and unsupervised anomaly detection techniques for detecting anomalies were also introduced. Supervised classification techniques such as Decision trees, K-nearest neighbors algorithm, Random Forest algorithms (Alonso et al., 2011), Naive Bayes and Support Vector Machines (Hu et al., 2003) have been used for anomaly detection. All these supervised techniques have been proven to be quite powerful in detecting anomalies.

Many semi-supervised learning techniques have also been used for detecting anomalies in several domains. Among these techniques, one class SVM and Autoencoders with one class learning have shown promising results (Eskin et al., 2002; Heller et al., 2003; An and Cho, 2015). One class SVM uses only the normal data and within a given ratio, anomalies are predicted. This method has been successfully adopted in intrusion detection systems (Wang et al., 2004), as well as in high dimensional and large scale anomaly detection problems (Erfani et al., 2016).

However, due to the practical problem of labeling data for supervised anomaly detection, unsupervised methods such as clustering and deep learning neural networks came into use (Amer and Abdennadher, 2011; Dean et al., 2012). However these methods highly depend on the features relevant to the domain of execution.

2.2 Anomaly Detection in Industrial Software Systems

Anomaly detection in industrial software system domain has been conducted using both supervised and unsupervised machine learning techniques. When considering the supervised techniques, SVM classification based anomaly detection has displayed good results on an industrial trading software system (Ranaweera et al., 2017).

Moreover, there is some research on machine learning based approaches for unsupervised anomaly detection in industrial software systems such as Tree Augmented Naive Bayesian networks (Willsky, 1976; Tan et al., 2012), Hidden Markov models (Willsky, 1976; Alonso et al., 2011), and restricted Boltzmann machines (Schneider et al., 2015).

2.3 Variational Autoencoders for Anomaly Detection

Among the unsupervised learning techniques, deep learning neural network based techniques are currently gaining the momentum (Xu et al., 2017). Autoencoder neural networks are one of the most prominent deep learning based anomaly detection technique that had been used in applications such as network intrusion detection, medical diagnosis, and credit card fraud detection (Sakurada and Yairi, 2014; Aygun and Yavuz, 2017). One of the most important features of the Autoencoder is that it can employ nonlinear dimensionality reduction while learning features. It is shown that Autoencoders perform better than the traditional principal component analysis (PCA) approach for anomaly detection using dimensionality reduction (Sakurada and Yairi, 2014).

There had been different variations of Autoencoders introduced over the years. VAE (Kingma and Welling, 2014) is one variation that had shown promising results in anomaly detection. The advantage of a VAE over the traditional Autoencoder is that it uses theoretically sound variance inference for generating stochastic latent variables rather than deterministic latent variables as in Autoencoders (An and Cho, 2015; Walker et al., 2016; Doersch, 2016).

3 INDUSTRIAL SOFTWARE SYSTEM

The anomaly detection approach of this research uses the data corresponding to an industrial trading software system, in which three predefined components are used. Namely, the components incorporated are Native Gateway (NG), Sequencer, and Matching Engine (ME). The Native Gateway component handles the communication between external parties and the trading software system, while Sequencer controls and manages the sequence of the multiple inputs provided by the Native Gateway and passes them to the Matching Engine. Matching Engine is the component that is responsible for matching the input orders with the stocks. Hence, ME handles a larger part of the computing process within the trading system. Moreover, due to the complex functionality of these components, each of these consists of a large number of attributes and parameters as shown in Table 1. These three components work together in the trading system, to perform the necessary calculations required for the use cases of the trading system. Moreover, these components work together to achieve the tasks by communicating over a network.

Table 1: Main components and dimensionality.

Component	Dimension
Native Gateway	78
Sequencer	110
Matching Engine	357

4 METHODOLOGY

4.1 System Overview

We present an extension of a previous anomaly detection work done on the same industrial trading software system (Ranaweera et al., 2017). Figure 1 demonstrates the architecture used in the previous research, and we are exploiting the same architecture with the modification of replacing the supervised SVM algorithm used in different classifiers in the component anomaly detector (CAD) with unsupervised DBSCAN and VAE anomaly detectors.

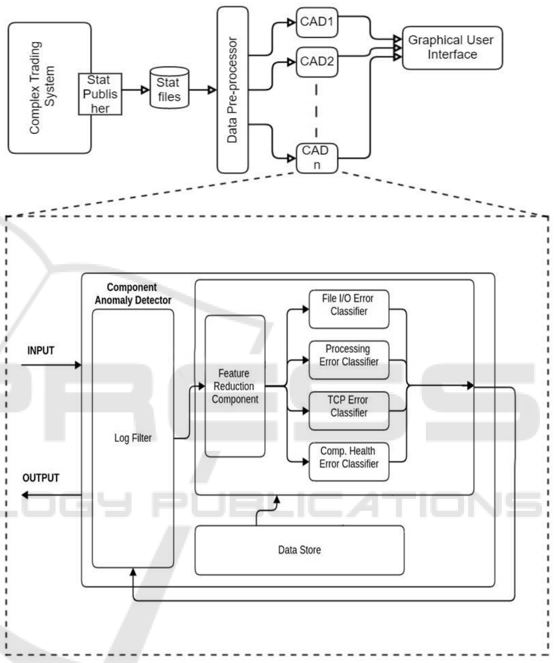

Figure 1: Architecture of the anomaly detection framework (Source: (Ranaweera et al., 2017)).

4.2 Variational Autoencoder (VAE)

4.2.1 Structure of the VAE

VAE has the structure of a normal Autoencoder and consists of three main components (Kingma and Welling, 2014; An and Cho, 2015).

1. Encoder neural network (Recognition model) - This network maps the given data instances x onto latent variable distribution z, given parameters ϕ. This is achieved by modeling the parameters of the approximate posterior $q(z|x,\Phi)$.
2. Decoder neural network (Generative model) - This network maps the randomly sampled latent

variable z onto a distribution of x, given parameters θ. This is achieved by modeling the parameters of the probabilistic graphical model p(x|z,θ), which implies the the likelihood of the data x given the latent variable z.

3. Loss function as given in equation 1.

$$L(i) = E_{q_\phi(z)}\left[\ln\left(p\left(x|z,\theta\right)\right)\right] - KL(q_\phi(z|x)||p_\theta(z)) \tag{1}$$

Here, the first term in equation 1 helps in minimizing the error between reconstructed output and the original input, as in a normal Autoencoder. The second term, by taking the Kullback-Leibler (KL) divergence, forces the approximate posterior q_ϕ(z|x) and prior distribution p_θ(z) to become similar values, which prevents the stochastic hidden variables from being deterministic as in the normal Autoencoder.

4.2.2 Anomaly Detection Procedure of the VAE

Basic training procedure of the VAE is that, given the training data X:{x(1), x(2), ..., x(m)} where x(i) ε R^D, network tries to regenerate the given input at the output $\hat{X}$:{$\hat{x}$(1), $\hat{x}$(2), ..., $\hat{x}$(m)} by identifying the latent variables Z that correspond to the data. If it is assumed that there exists a significant difference between the normal samples and anomalous samples, the error in reconstructing anomalous samples is higher than that of the normal samples (Sakurada and Yairi, 2014). Due to this reason, after training the VAE using the training dataset, and while testing the model on test dataset, anomalous data tend to give a higher reconstruction error, which is calculated according to the equation 2. Moreover, by defining a threshold value for the reconstruction error, anomalous data can be identified as the data points that have a reconstruction error above the defined threshold.

$$E(i) = \sqrt{\sum_{j=0}^{D}\left(x_j(i) - \hat{x}_j(i)\right)^2} \tag{2}$$

4.2.3 Semi-supervised One Class Learning

According to equation 2, data instances that tend to give a higher reconstruction error are classified as anomalies and the reason for higher reconstruction error is that the VAE is unable to reconstruct those given instances correctly. In order to exploit this characteristic of the VAE, we adopted a one class learning technique.

In this one class learning technique, all the datasets corresponding to the three components were divided into normal and anomalous based on the labels, and only the normal proportion of data was used for unsupervised training of the VAE.

4.3 Density based Spatial Clustering of Applications with Noise (DBSCAN)

Density based techniques had been best at identifying clusters of arbitrary shapes and therefore they are the best performing algorithms for detecting local outliers. Moreover, density based techniques are quite robust in the presence of noise. This makes density based techniques the best clustering based technique for anomaly detection (Kotsiantis and Pintelas, 2004).

DBSCAN had been one of the most used density based algorithms and had performed well over the years (Loh and Park, 2014; Shekhawat and Sharma, 2017). There had been many variants of DBSCAN that performed well in different domains. However, the key underlying concept of anomalies had been the same in all of these techniques.

In DBSCAN, a data point is clustered based on whether the number of data points that exist within a predefined radius is higher than a predefined number of data points (Yang et al., 2014). This marks the data point clusters that have less number of data points than the required amount within the defined radius, as anomalous data points.

4.3.1 Anomaly Detection Procedure of DBSCAN

When classifying new data points as anomalous or not, the number of data points within the predefined radius from the new data point is counted, and if the number of data points is higher than the predefined number of data points, the new data point is labeled as non-anomalous, and vice versa.

4.4 Dealing with High Dimensionality

The industrial software system components used in this research consist of a large number of features as shown in Table 1, which makes the anomaly detection procedure quite complex.

In order to overcome this problem, according to a previous research executed on the same industrial trading software system (Ranaweera et al., 2017), a tree ensemble method based on randomized decision trees was utilized to select the most suitable features that correspond to each component.

As shown in Table 2, we obtained a reduced feature set by using the tree based ensemble.

Table 2: Dimensionality of the reduced feature set.

Component	Dimension
Native Gateway	5
Sequencer	10
Matching Engine	10

5 VAE WITH FEATURE SELECTION

Latent variables inferred by the VAE hold a great importance in the anomaly detection procedure, since these latent variables are the building blocks of the generative model. If the inferred latent variable distribution is capable of identifying the different classes within the data, differentiating anomalous from normal data evidently becomes more accurate. Most of the supervised machine learning techniques try to achieve this using labeled data. When the latent variable distribution becomes accurate, result from unsupervised anomaly detection tends to become as accurate as in supervised learning algorithms.

According to the literature, VAE has shown good performance for many domains and high dimensional benchmarking datasets (An and Cho, 2015). However, we argue that the performance of the VAE can be improved by identifying the best features in the dataset, which would lead to a more expressive latent model of the data. Hence, a feature selection is adopted to extract the most suitable set of features, which leads to a higher information gain while maximizing the posterior probability of P(X|z, θ). As a result, the recognition model of the VAE tends to discover the best approximation of the latent variable distribution, and consequently probability of each data instance X of the training dataset, as shown in equation 3 gets maximized. This is mainly due to the accuracy increases of the generative model with respect to the high expressiveness of the latent variables, which results in increased likelihood of the data X given latent variable z. Thus, a VAE that has a high performing generative model can be designed with the support of feature selection, leading to a higher accuracy of detecting anomalies in industrial software systems.

In next section, we empirically show that VAE does perform better with a reduced feature set.

$$P(X) = \int P(X|z,\theta)P(z)dz \tag{3}$$

6 EVALUATION

6.1 Dataset Generation

The datasets used in the experiments were generated in a controlled environment using an industrial software system that is used by a company in the trading systems domain. As mentioned earlier, controlled environment contained three main components running to generate the required datasets; Native Gateway, Sequencer, and Matching Engine. Datasets for each component were separated into training and test datasets. The test datasets were labeled so that they can be used for evaluating the anomaly detection techniques. The dataset that was generated is identical to the datasets used in the experiments done for the research conducted on a supervised approach for detecting anomalies (Ranaweera et al., 2017). Dataset corresponding to each component contains 20000 data samples on average.

6.2 Experimental Results

Results were obtained for the three components separately. The training sets were used for training the VAE and DBSCAN models, after which both models were evaluated using the test datasets.

A ZeroR classifier was used as a baseline classifier (assigning normal tag to all the data entries) in order to mark a lower bound for the accuracy. Results obtained from the experiments were analyzed based on the Recall, Precision, F1 Score, and the Accuracy. For each component, the scores were compared, under these different experiments. The experiment procedure of this research is as below.

- Tested both multi-class learning (noted as M-c in the table) and one class learning (noted as as O-c in the table) of the VAE with respect to the dataset. Performance evaluation of the two learning methods can be found in Table 3. Moreover, one class and multi-class VAE were tested upon the selected features from the tree based ensemble. This evaluation also can be found in Table 3.
- Tested both multi-class VAE and DBSCAN with and without feature selection(FS). Evaluation of the obtained performance can be found in Table 4.
- Tested a similar semi-supervised anomaly detection technique known as one class SVM against the one class learning of VAE in order to evaluate the usability in a practical environment. This experiment was done with and without feature selection. Results are shown in Table 5.

Table 3: Performance evaluation of Variational Autoencoder

Comp	Desc	Recall		Precision		F1		Accuracy		
		M-c	O-c	M-c	O-c	M-c	O-c	M-c	O-c	Base
ME	Without FS	0.35	0.77	0.66	0.42	0.46	0.55	0.35	0.77	0.71
	With FS	0.96	**0.97**	0.96	**0.92**	0.94	**0.94**	0.97	**0.97**	0.71
SEQ	Without FS	0.59	0.74	0.56	0.63	0.57	0.68	0.59	0.74	0.72
	With FS	0.99	**1.00**	0.85	**0.85**	0.91	**0.92**	0.99	**0.99**	0.72
NG	Without FS	0.81	0.83	0.32	0.33	0.46	0.48	0.92	0.92	0.70
	With FS	0.97	**0.98**	0.39	**0.39**	0.56	**0.56**	0.93	**0.93**	0.70

Table 4: Variational Autoencoder and DBSCAN Comparison.

Comp	Desc	Recall		Precision		F1		Accuracy		
		VAE	DBSCAN	VAE	DBSCAN	VAE	DBSCAN	VAE	DBSCAN	Base
ME	Without FS	0.35	0.39	0.65	0.45	0.46	0.42	0.35	0.40	0.71
	With FS	**0.96**	0.95	0.91	**0.91**	**0.93**	0.93	**0.97**	0.95	0.71
SEQ	Without FS	0.59	0.55	0.55	0.51	0.57	0.53	0.58	0.54	0.72
	With FS	0.99	**0.99**	**0.85**	0.80	**0.91**	0.89	0.99	**0.99**	0.72
NG	Without FS	0.81	0.62	0.32	0.04	0.46	0.08	0.91	0.92	0.70
	With FS	**0.97**	0.81	**0.38**	0.05	**0.55**	0.09	**0.92**	0.52	0.70

Table 5: One class learning of Variational Autoencoder and one class SVM Comparison.

Comp	Desc	Recall		Precision		F1		Accuracy		
		VAE	SVM	VAE	SVM	VAE	SVM	VAE	SVM	Base
ME	Without FS	0.77	0.32	0.42	0.23	0.54	0.27	0.77	0.33	0.72
	With FS	0.97	**0.97**	**0.92**	0.50	**0.94**	0.66	0.97	**0.97**	0.72
SEQ	Without FS	0.74	0.72	0.63	0.36	0.68	0.48	0.74	0.71	0.72
	With FS	**1.00**	0.99	**0.85**	0.22	**0.92**	0.36	**0.99**	0.99	0.72
NG	Without FS	0.83	0.99	0.33	0.24	0.48	0.39	0.92	0.92	0.70
	With FS	0.98	**1.00**	**0.39**	0.23	**0.56**	0.38	0.93	**0.95**	0.70

6.2.1 Result Analysis

Recall is the most important score in anomaly detection, since even a slight fault in an industrial software system would result in massive losses. However, precision is also important as many false positives would result in a huge wastage of resources in reacting to the anomaly alerts.

When comparing the scores from the experiments performed using the multi-class VAE and DBSCAN, the VAE outperforms, or is quite close to DBSCAN in all the components of the industrial software system. This shows that the VAE is better than or equal to DBSCAN, which is an established technique.

When comparing the scores from the experiments performed using the VAE one class learning, the VAE has shown to give higher recall values than that of the multi-class model. Moreover, one class learning VAE outperformed the well known semi-supervised anomaly detection technique, one class SVM with a high margin. Hence one class learning of VAE provides a suitable semi-supervised anomaly detection framework for industrial software system domain. However, VAE could not outperform the supervised SVM classification based anomaly detection technique executed in the previous research on the same dataset (Ranaweera et al., 2017).

When comparing the impact of the feature selection technique on the scores of the VAE, we can observe that the performance of all techniques had improved when feature selection was performed. Previous research that employed Autoencoders has not exploited any feature selection, since Autoencoders are said to be usable in high dimensional problems (Sakurada and Yairi, 2014; Aygun and Yavuz, 2017). However, according to the results obtained from the feature selection, we show that feature selection can improve the performance of VAE substantially. Moreover, the experiments support the claims that had been made in previous research that DBSCAN does not perform well in high dimensional datasets (Berkhin et al., 2006).

However, when comparing the overall results of the three components, Native Gateway has lower results than the other two components. When analyzing the dataset, We found that compared to Sequencer and Matching Engine, data points have a sparse distribution in density wise, which can cause recognition network of the VAE to identify normal points as anomalous due to the high reconstruction error.

7 CONCLUSION

This paper presented the use of a deep learning neural network known as Variational Autoencoder (VAE) to detect anomalies in an industrial software system containing components with a large number of features. Since the anomaly detection framework was designed for industrial software systems with critical executions, higher recall is expected. According to the experiments executed, higher recall values were obtained and these results were shown to improve extensively with the feature selection mechanism adopted in the research. Moreover, performance of VAE is shown to outperform DBSCAN, which is a well-established clustering based technique for anomaly detection. Furthermore, the semi-supervised one class learning based VAE outperforms the one class SVM, and consequently proves that it is better suited for a production environment.

One of the main characteristics of the VAE is that it can be improvised for high dimensional anomaly detection problems. However, this paper emphasized the importance of identifying discriminative features before utilizing the VAE, which leads to improved generative power and accuracy in detecting anomalies.

As future work, we intend to experiment with unsupervised feature selection techniques such as Joint Embedding Learning and Sparse Regression (JELSR), Robust Joint Graph Sparse Coding, and Exemplar Convolutional Neural Networks, instead of the currently used supervised feature selection technique tree based ensemble. This way, multi-class VAE can be successfully used as a fully unsupervised anomaly detection technique in industrial software systems.

ACKNOWLEDGEMENTS

The authors thank MillenniumIT Software (Private) Ltd. for providing the datasets for industrial system evaluation.

REFERENCES

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

Alonso, J., Belanche, L., and Avresky, D. R. (2011). Predicting software anomalies using machine learning techniques. In *Proceedings of the 10th IEEE International Symposium on Network Computing and Applications*, pages 163–170. IEEE.

Amer, M. and Abdennadher, S. (2011). Comparison of unsupervised anomaly detection techniques. *German Research Center for Artificial Intelligence (DFKI GmbH)*.

An, J. and Cho, S. (2015). Variational autoencoder based anomaly detection using reconstruction probability. *SNU Data Mining Center*.

Aygun, R. C. and Yavuz, A. G. (2017). Network anomaly detection with stochastically improved autoencoder based models. In *4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, pages 193–198. IEEE.

Berkhin, P. et al. (2006). A survey of clustering data mining techniques. *Grouping multidimensional data*, 25:71.

Dean, D. J., Nguyen, H., and Gu, X. (2012). Ubl: Unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems. In *Proceedings of the 9th international conference on Autonomic computing*, pages 191–200. ACM.

Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

Erfani, S. M., Rajasegarar, S., Karunasekera, S., and Leckie, C. (2016). High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. *Pattern Recognition*, 58:121–134.

Eskin, E., Arnold, A., Prerau, M., Portnoy, L., and Stolfo, S. (2002). A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. *Applications of data mining in computer security*, 6:77–102.

Guo, Z., Jiang, G., Chen, H., and Yoshihira, K. (2006). Tracking probabilistic correlation of monitoring data for fault detection in complex systems. In *Dependable Systems and Networks, 2006. DSN 2006*, pages 259–268. IEEE.

Heller, K. A., Svore, K. M., Keromytis, A. D., and Stolfo, S. J. (2003). One class support vector machines for detecting anomalous windows registry accesses. In *Proceedings of the workshop on Data Mining for Computer Security*, volume 9.

Hu, W., Liao, Y., and Vemuri, V. R. (2003). Robust support vector machines for anomaly detection in computer security. In *ICMLA*, pages 168–174.

Idé, T. and Kashima, H. (2004). Eigenspace-based anomaly detection in computer systems. In *Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 440–449. ACM.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Kotsiantis, S. and Pintelas, P. (2004). Recent advances in clustering: A brief survey. *WSEAS Transactions on Information Science and Applications*, 1(1):73–81.

Loh, W.-K. and Park, Y.-H. (2014). A survey on density-based clustering algorithms. In *Ubiquitous Information Technologies and Applications*, pages 775–780. Springer.

Ranaweera, L., Vithanage, R., Dissanayake, A., Prabodha, C., and Ranathunga, S. (2017). Anomaly detection in complex trading systems. In *Engineering Research Conference (MERCon), 2017 Moratuwa*, pages 437–442. IEEE.

Sakurada, M. and Yairi, T. (2014). Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the 2nd Workshop on Machine Learning for Sensory Data Analysis (MLSDA)*, page 4. ACM.

Schneider, C., Barker, A., and Dobson, S. (2015). Autonomous fault detection in self-healing systems using restricted boltzmann machines.

Shekhawat, M. and Sharma, I. (2017). Density based metrics for clustering–a comprehensive survey. *International Journal for Women Researchers in Engineering, Science and Management*, 1:6–9.

Tan, Y., Nguyen, H., Shen, Z., Gu, X., Venkatramani, C., and Rajan, D. (2012). Prepare: Predictive performance anomaly prevention for virtualized cloud systems. In *IEEE 32nd International Conference on Distributed Computing Systems (ICDCS)*, pages 285–294. IEEE.

Walker, J., Doersch, C., Gupta, A., and Hebert, M. (2016). An uncertain future: Forecasting from static images using variational autoencoders. In *European Conference on Computer Vision*, pages 835–851. Springer.

Wang, Y., Wong, J., and Miner, A. (2004). Anomaly intrusion detection using one class svm. In *Fifth Annual IEEE Information Assurance Workshop*, pages 358–364. IEEE.

Willsky, A. S. (1976). A survey of design methods for failure detection in dynamic systems. *Automatica*, 12(6):601–611.

Xu, D., Yan, Y., Ricci, E., and Sebe, N. (2017). Detecting anomalous events in videos by learning deep representations of appearance and motion. *Computer Vision and Image Understanding*, 156:117–127.

Yang, Y., Lian, B., Li, L., Chen, C., and Li, P. (2014). Dbscan clustering algorithm applied to identify suspicious financial transactions. In *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, pages 60–65. IEEE.