Towards enhancing unsupervised anomaly detection by improving complexity, dimensionality and class-boundary properties

Kasra Babaei

Faculty of Science and Engineering
University of Nottingham Malaysia

This dissertation is submitted for the degree of
Doctor of Philosophy

School of Computer Science October 2021

To my parents
and my wife
who supported me unconditionally.

Publications

K. Babaei, Z. Chen, T. Maul, "Detecting point outliers using prune-based outlier factor (PLOF)," *The 4th International Conference on Computing, Mathematics and Statistics*, 2019

K. Babaei, Z. Chen, T. Maul, "A Study of Fraud Types, Challenges and Detection Approaches in Telecommunication," *Journal of Information Systems and Telecommunication (JIST)*, vol. 4, no. 28, pp. 248 - 261, 2020

K. Babaei, Z. Chen, T. Maul, "AEGR: A simple approach to gradient reversal in autoencoders for network anomaly detection," *Soft Computing*, pp. 1 - 12, 2021

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Kasra Babaei
October 2021

Acknowledgements

I would like to express my sincerest gratitude to my supervisors Dr Zhi Yuan Chen and Dr Tomas Maul, for their unconditional guidance and support. This work would not have been completed without their untiring efforts and patience, which made this work possible.

I would like to thank my parents and my wife for their continuous and unquestioning love, care and support that enabled me to overcome difficulties through the years. There are no words to express my gratitude adequately. This work is dedicated to you.

Abstract

Any observation that follows a pattern other than the expected one, i.e., the normal behaviour, is considered abnormal behaviour (also known as an anomaly). Abnormal behaviour is witnessed in various areas— for instance, a previously unseen high temperature during winter in a naturally cold environment.

Prior research shows that anomalies can result in negative impacts such as financial losses in telecommunications or human fatalities in aviation accidents. Despite the advances made in the area of anomaly detection, detection methods underperform due to the challenges that affect and hinder the process of anomaly detection. In this work, three novel anomaly detection approaches are introduced each, of which aims at addressing one problem that debilitates the performance of anomaly detection.

One of the problems in anomaly detection is having access to an ample amount of anomalous examples; therefore, the proposed methods in this work are all unsupervised as this type of learning is needless of having access to a labelled training set. The first contribution of this research is focused on reducing the execution time of a density-based method while maintaining the performance at a high level by applying a novel pruning-based preprocessing step.

In density-based methods, measuring the density plays an important role, and as the dimensionality increases, the definition of density becomes harder. By using dimensionality reduction methods, it is possible to transform the high-dimensional input data into a low-dimensional form while maintaining essential features. In the second contribution, a novel dimensionality reduction method is introduced that is needless of having access to an anomaly and noise-free training set.

When using One-Class Classifier methods, the performance varies as the size of the training set changes. Having access to a training set that includes more normal examples can improve the performance as the class-boundary becomes less ambiguous. The final contribution of this work is focused on improving the definition of class-boundary by proposing a data augmentation approach. The proposed approach generates augmented examples while simultaneously reduces the dimensionality of the input data.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

AD	Anomaly Detection
ADASYN	Adaptive Synthetic Sampling
ADMM	Alternating Direction of Method of Multipliers
AE	Autoencoders
AEGR	AutoEncoders with Gradient Reversal
ANN	Artificial Neural Networks
AUC	Area Under ROC
CAE	Clustering-based deep AutoEncoder
cGAN	Conditional Generative Adversarial Network
CNN	Convolutional Neural Network
DA	Data Augmentation
DAE	Denoising AutoEncoders
DBN	Deep Belief Networks
GAN	Generative Adversarial Network
GMM	Gaussian Mixture Model
IF	Isolation Forest
KDE	Kernel Density Estimation
KSAE	k-means Shrink AutoEncoder

LDOF	Local Distance-based Outlier Factor
LOF	Local Outlier Factor
LRD	Local Reachability Distance
LSH	Local Sensitive Hashing
MSE	Mean Square Error
OCC	One-Class Classifier
OCSVM	One Class Support Vector Machine
PCA	Principal Component Analysis
PLOF	Prune-based Local Outlier Factor
PR	Precision-Recal
PV	Photovoltaic
PVLOF	Photovoltaic Local Outlier Factor
R-SAE	Robust Stacked AutoEncoders
RandNet	Randomised Neural Network for Anomaly Detection
RBM	Restricted Boltzmann Machine
RC	Robust Covariance
RE	Reconstruction Error
RNN	Recurrent Neural Networks
ROC	Receiver Operating Characteristics
S-SAE	Standard Stacked AutoEncoders
SAE	Stacked AutoEncoders
SGD	Stochastic Gradient Descent
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support Vector Machines
ZCA	Zero Component Analysis

Chapter 1

Introduction

Every year, the speed of generating data increases. This abundant amount of data contains substantial information that can be challenging to detect and requires special techniques and methods. For instance, the data can be an indication of a faulty part in the engine of a passenger plane or a malignant cell in a patient's lung. The key difference between this type of data with the rest is in which the pattern does not follow what the majority of data conforms to. The small fraction of data that its pattern deviates from the pattern of the majority of the data is known as anomalies, and Anomaly Detection (AD) refers to the process in which the aim is to identify these observations [1–3]. In other words, the whole data can be divided into two sets. One set contains a large portion of the whole data, known as the majority, and all the data instances in this set follow a well-known pattern, i.e., the normal pattern. In the second set, a small fraction of the whole data is placed, and objects within this set follow any pattern but not the pattern of the majority.

Anomaly detection has gained a tremendous amount of attention from various areas, such as financial fraud detection, image processing, and sensor networks. For instance, an unusual international money transaction from a foreign country that is different from where the transactions used to be carried out is a good sign of fraudulent activities. In Fig. 1.1, data instances of an artificially generated dataset are plotted in which red points are considered anomalies as they are placed far from the majority due to their distinct pattern. However, several obstacles hinder the detection process, such as the curse of dimensionality, skewed distribution, concept drift, or lack of labelled data.

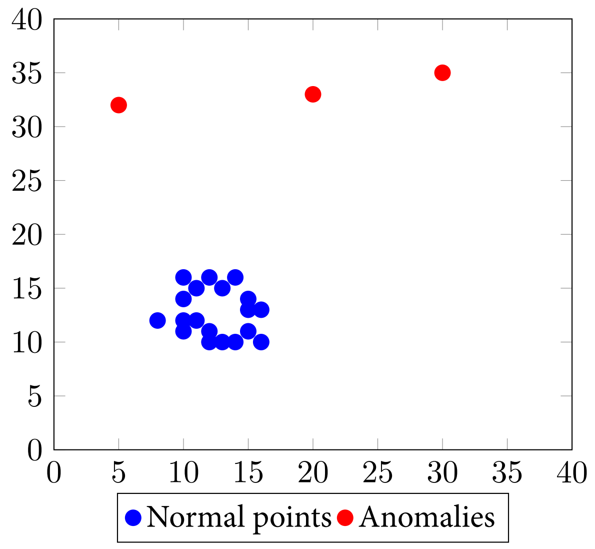


Fig. 1.1 An example of a dataset that contains anomalies

1.1 Anomalies

Having a clear understanding of anomalies is important. Anomalies, outliers, novelties, noises, deviation, exceptions, aberrations, surprises, peculiarities, or contaminants are correlated terms that in some research work have been used interchangeably. An anomaly is a data instance that is inconsistent and deviates from the normal pattern of the rest of the data it belongs to [4][5]. Anomaly detection or outlier detection refers to the process of identifying and often removing anomalies from the given data set, which has been widely used in many applications such as fraud detection in telecommunication, health-insurance, or credit card, intrusion detection for cyber-security, fault detection in crisis management systems, and military surveillance for an enemy or terrorist activities detection [1].

1.2 Types of Anomalies

Understanding the nature of anomalies is essential to find the optimum outlier detection method. According to [1], there are three types of anomalies: point anomalies, contextual anomalies, and collective anomalies.

1.2.1 Point Anomalies

This type is known as the simplest type of anomaly, and most of the research works in anomaly detection have tried to detect point anomalies in which a data instance is identified as an anomaly with respect to the rest of the data. In Figure 1.1, each of the red points is considered as a point anomaly. A real-world example of a point anomaly is any unusual financial transaction in credit card fraud detection or an abnormal call in telecommunication fraud detection.

1.2.2 Contextual Anomalies

Contextual anomaly, also known as a conditional anomaly, refers to an individual data instance that is an anomaly only in a specific context but not otherwise [6]. Contextual outlier detection has been widely utilised in application domains with time-series data, and spatial data [1]. An example of a contextual anomaly is the temperature of the environment at two different times. To explain further, a high temperature during a hot season is considered normal, while witnessing the same temperature during a cold season is unexpected.

1.2.3 Collective Anomalies

A collective anomaly is very similar to a point anomaly except for the fact that a collection of related data instances are compared to the rest of the data as opposed to juxtaposing a single data instance [1]. Collective anomalies have been applied to different types of data, such as sequence data, graph data, and spatial data. An example of a collective anomaly can be a series of actions that are considered normal in case the actions follow the correct sequence. However, if the actions occur in an unusual order, the whole series can be considered an outlier.

1.3 Anomalies and Common Definition Issues

Despite the fact that there is a consensus over the definition of an anomaly in the literature, it often becomes confusing because the definition overlaps with the definition of other concepts such as noise. Also, there are different terms for an anomaly that are used interchangeably in the literature. This section covers the differences between anomalies and noise and also goes through various terms for an anomaly while mentioning the difference between anomalies and novelties.

1.3.1 Noise vs Anomalies

It is hard to differentiate anomalies from noise as both have one similar crucial characteristic, which is deviating from the pattern that the majority follow. To separate anomalies from noise, one should look at a few key factors. Firstly, the cause of noise is different from anomalies. To explain further, noise can be generated by, for instance, a dusty lens or a faulty microphone, while an anomaly in the mentioned examples would be an unprecedented object in the image or a meaningful nuance in the track. Secondly, while anomalies implicate meaningful and important, noise is something undesired that the analyst is not interested in and tries to clean the data from noise, which is known as noise removal, before carrying out any experiment on the data [1]. Nonetheless, in some contexts, such as the study conducted by Ha et al. [3], anomaly detection is considered the same as noise removal.

1.3.2 Novelties vs Anomalies

There are various terms such as outlier, deviation, or exception detection that all refer to the anomaly detection [4]. Although most research works have claimed that novelty detection is merely another term for anomaly detection, some such as [2] have defined novelties slightly different from outliers. In novelty detection, the aim is to discover data instances that are totally unprecedented. While novelty detection in its definition is very similar to anomaly detection, the key distinction is that the observed pattern here is later appended to the normal behaviour [1]. For instance, a new shopping pattern can emerge when buying Christmas presents. It is worth mentioning that as these concepts are very related, the methods for dealing with them are very similar and, in many cases, applied in both concepts.

1.4 Motivation

Anomaly detection is being used in various fields. Application domains can be divided into several categories, and there is no universally accepted classification. However, it is possible to categorise them into the followings: financial fraud detection, computer and network intrusion detection, healthcare detection, industrial damage detection, image processing, text data, and sensor networks.

The practicality and popularity of anomaly detection models and the importance of their performance cannot be overstated. Despite the recent advances in this area, anomaly detection methods still suffer from some well-known problems such as the curse of dimensionality, and the performance does not meet the expected level. Therefore, this thesis is motivated by the importance of investigating and understanding more about anomaly detection to gain deeper insights about the challenges that affect the performance and the benefits of developing novel methods that push the performance limits for achieving better models that can outperform their predecessors.

1.5 Problems

Traditional approaches are categorised into different groups such as distribution-based, density-based or distance-based. Even though the computational power of computers is increasing significantly every year, some of these methods suffer from high computation cost that even the current generation of computers struggle to carry out tasks. There have been a few works in which a more efficient variant of the traditional methods was introduced to reduce the computation cost; however, they disregard the performance, i.e., fail at keeping the detection rate at an acceptable or same level.

This thesis also investigates the problem of high dimensionality in datasets and the effect of the curse of dimensionality on the anomaly detection process. In a high-dimensional dataset, the data points are more spread out, which affects the density and convex hull [4]. There are various works in the literature that target this problem as it is not an anomaly detection problem but rather a known challenge in machine learning. Classic methods such as PCA have been used in various areas for reducing dimensionality while keeping valuable information. Deep learning has also shown outstanding results in recent years when used for dimensionality reduction. However, when applied in the area of anomaly detection, the training phase can become a challenge as the network also learns how to extract information about anomalies instead of excluding them.

Lastly, this research studies the impact of tuning the class boundary in a model that is trained to make a binary decision. The process of anomaly detection is a binary decision in which the majority of data points belong to the normal class, and a minor fraction of the dataset goes under the other class that includes merely anomalies. Finding the right boundary between the two classes can be challenging. While a too tight boundary can cause a high false-positive rate, i.e., many normal instances will be labelled as anomalies, a slack boundary can produce a higher false-negative rate, i.e., anomalies will be classified as inliers. When the training set contains inadequate amount of normal instances, the model struggles to find the right class boundary that separates inliers from outliers.

1.6 Proposed Solutions

This thesis introduces three novel methods, each of which targets one of the problems that is mentioned in Section 2.2. This section summarises the three proposed methods.

1.6.1 Efficiency and Performance

In Chapter 3, a novel variant of a state-of-the-art density-based approach is proposed that is known for its high computation cost. The main research goal of the proposed method in Chapter 3 is to introduce a new model that is capable of detecting anomalies with less computation while retaining performance. The proposed solution employs a novel method for calculating the neighbourhood density of each data point and then applies a pruning procedure based on the computed score. One key advantage of the proposed approach is that, unlike its predecessor, it does not sacrifice the detection performance for lower computation cost.

1.6.2 Insensitive to Anomalies

Also, in Chapter 4, the impact of dimensionality reduction is studied to gain more insights about traditional approaches and the current state-of-the-art methods for reducing dimensionality. Then, a novel deep learning model is proposed that tries to reduce dimensionality while automatically excluding anomalies in the training phase based on its gradient updates in an unsupervised setting. The advantage of the proposed approach compared to other works found in the literature is that the training set does not need to be anomaly or noise-free as opposed to other works that must be trained with a clean training set. After reducing the dimensionality of the dataset, various widely used anomaly detection methods are used to separate anomalies from normal datasets.

1.6.3 Reproduction

In Chapter 5, after investigating the importance of defining a better class boundary and its influence on the performance of a One-Class Classifier, a novel method is introduced that not only reduces the dimensionality but at the same time reproduces augmented data out of the training set to be used for training the model. Simplicity is one of the advantages of the proposed model. The proposed method investigates the effect of reproduction, i.e., data augmentation, on defining a better class boundary to improve the performance of the model in anomaly detection.

1.7 Overview

This thesis is organised as follows. Chapter 2 studies previous works and their findings. In Chapter 3, a novel prune-based approach is introduced to reduce the complexity of a state-of-the-art density-based method while maintaining its performance.

Chapter 4 presents a variant of a deep learning method that tries to deal with the curse of dimensionality while making the network insensitive towards anomalies and noise. Then, Chapter 5 introduces a data augmentation approach that addresses the challenge of defining the class boundary in One-Class Classifiers. Next, Chapter 6 summarises the findings of this thesis while providing a general discussion, limitations and possible future works. Finally, Chapter 7 summarises the contributions of this thesis.

Chapter 2

Literature Review

In this chapter, the literature on anomaly detection is reviewed to explain various approaches and challenges in detecting anomalies.

2.1 Anomaly Detection Approaches

In the literature, there is no consensus on categorising anomaly detection approaches. Different researches have come up with various categorisations. Anomaly detection methods can be divided into the following three groups: supervised, unsupervised, and semi-supervised. The aim of this section is to briefly summarize the difference between the three groups but not to get into the details because it is deemed as out of the scope.

2.1.1 Supervised Learning

In supervised learning, a portion of the data set in which the observations are labelled to different classes such as normal and abnormal, is used as a training set. Firstly, the model is trained using the training set, and then unlabelled data is given to the model to perform prediction. There are two main schemes of supervised learning models: classification and regression. In a classification model, the outcomes are discrete, and the model tries to map unseen observations into predefined classes. Alternatively, when the outcomes are continuous, regression models are used.

Classification can be divided into one-class classification and multi-class classification. While in multi-class classification, the purpose of the algorithm is to classify unseen observations into one of the predefined classes, in one-class classification, the model is trained merely by one of the classes, which is referred to as the target class or the positive class [7]. In anomaly detection, the target class is the normal class, and the non-target class is the abnormal class. This approach is functional when the number of instances from the non-target class is low due to the difficulty of measuring abnormal behaviour. After training the model with the instances of the normal class, every instance that does not belong to the normal class is considered an anomaly.

One-class classification suffers from problems of multi-class classification as well, for instance, error rates approximation, the curse of dimensionality (explained in 2.2.1), or generalisation, and selecting observations that can precisely define the boundary between the normal and abnormal classes is often hard especially when this boundary is vast, and the data is non-convex [7].

Li et al. [8] proposed a one-class classification for intrusion detection using Support Vector Machines (SVM); however, their approach was unable to detect intrusion detection online because it required a training set in prior to the prediction phase. In most previous work, such as Zhang et al. [9] and Patnaik et al. [10], a variation of One-Class SVM (OCC) have been applied for anomaly detection.

2.1.2 Unsupervised Learning

In unsupervised learning, having a labelled training set is needless as an unsupervised approach tries to separate observations into clusters of data point with similar characteristics. In some context, this is considered a better approach when the majority of the data sets is negative, i.e., normal; however, it can also produce a high false alarm rate if this assumption is not met [1]. There are several unsupervised learning methods that, based on their calculation nature, can be categorised in the following groups [11]:

- Distance-based methods: firstly, the distance between all the observations is measured, and observations with d_{min} distance from p percentage of observations in the data sets are considered as anomalies. Distance-based methods are used in detecting point anomalies. One example of a distance-based method is the k-nearest neighbours algorithm.

- Clustering-based methods: it is possible to separate the data into different clusters, i.e., each cluster holding data instances with similar characteristics, and label those data instances that belong to no cluster as anomalies.

- Density-based methods: the density of each observation is measured using a density estimation method such as Kernel Density Estimation (KDE), and observations that their local density vary from their neighbours are determined as outliers. An example of this type of anomaly detection is the Local Outlier

Factor (LOF) method, which is based on the local density of each observation [12].

- Depth-based methods: based on computing various layers of k-d convex hulls, and observations outside of the hulls are determined as outliers [13].
- Distribution-based methods: a statistical approach determines a data point as an outlier if it substantially deviates from the underlying distribution.

Although distance-based algorithms are simple and fast, their performances debilitate when the data set has various degrees of density [3]. Also, density-based algorithms underperform when there are low-density patterns inside the data set [14]. Likewise, depth-based approaches struggle when the data set becomes huge because they depend on calculating k-d convex hulls that are composed of a lower boundary complexity of $\phi(m^{k/2})$ for m instances [15]. A prior understanding of the distribution is needed when using a distribution-based algorithm. For instance, the optimum number of components is often unknown and needs to be determined [16].

2.1.3 Semi-Supervised Learning

Semi-supervised learning lies between supervised learning and unsupervised learning. Semi-supervised learning is used under various circumstances such as paucity of training data or lack of certainty about all instances labels [17]. This method uses both labelled data and unlabelled data for the learning process [18], which makes the method very suitable for outlier detection where the number of positive instances in the data sets is very low [19].

There is also another type of approach, known as rule-based systems, that is worth mentioning. A Rule-Based technique is a straightforward approach in which an alarm is triggered when a particular criterion is met. It is not considered a supervised approach; however, the fact that it requires labelled data for defining the rules makes it very similar to a supervised approach. Although these systems are simple, effective and efficient, they come with some deficiencies, which are [20, 21]:

- Vulnerable to unknown anomalies.
- Rules should be programmed precisely for every possible anomaly.
- Setting new rules requires a field expert and prior knowledge.
- Setting new rules is not immune to human error.

- Defining new rules is time-consuming and often complicated.

Rule-based systems have been widely employed in many application domains, such as fraud detection. However, shortcomings of rule-based systems give adversaries the opportunity to adapt and change their attacking methods and avoid triggering the alarm, which makes rule-based fraud detection systems based ineffective against new attacking patterns.

2.1.4 Neural Networks and Recent Development in Anomaly Detection

The strength of artificial neural networks, which have been widely employed in other areas such as image and speech processing, has proven to be useful in anomaly detection as well. However, no approach comes without shortcomings, and as suggested by Ahmed et al. [22], the downside of using neural networks is the high computational requirement. Nevertheless, neural networks are being used in anomaly detection either alone or alongside another approach, such as a statistical model. For instance, Cao et al. [23] proposed a new variant of AutoEncoders (AE) to reduce the dimensionality of the datasets and then applied other methods such as One-Class SVM, kernel density estimation or local outlier factor to separate anomalies from inliers. They used autoencoders for reducing the dimensionality of the dataset, which is a feed-forward neural network made of two parts, namely the encoder and the decoder [24] (autoencoder is explained in details in Chapter 4). Pérez et al. [25] compared the performance of autoencoders with PCA for feature extraction and dimensionality reduction for the purpose of detecting anomalies. Their work was focused on detecting network intrusion and used other models, namely LOF, One-Class SVM, Isolation Forest (IF), and Robust Covariance (RC), for separating anomalies. The results of their work showed that the linear method, i.e., PCA, did not improve the performance and, in some scenarios, even worsened it while the models performed better when the feature extraction was carried out by the autoencoders.

Also, there are various research works in which a neural network was used alone for detecting anomalies. For instance, Chen et al. [26] used a variant of autoencoders, i.e., Convolutional Autoencoders, for network anomaly detection in which the network was designed to reduce the dimensionality and capture non-linear features. In contrast, the reconstruction error of the network was later used for separating anomalies. To explain further, anomalies in the test data cause a more significant reconstruction error because the network is not familiar with their pattern; therefore, the encoder and decoder do a poor job in capturing features and regenerating the input data, respectively. One of the advantages of this type of approach for separating anomalies is its simplicity and low computation cost, as the complexity of calculating the reconstruction error for the test data is only $O(n)$ where n is the number of data instances. However, finding the suitable threshold is always challenging and has a substantial effect on the performance of the model. Defining a threshold that is too low can cause a high false-positive rate as the model flags most of the instances as anomalies, and a too high threshold can cause the opposite, which is a high false-negative rate, i.e., the model labels a high portion of the anomalies as normal. For this reason, Castellini et al. [27] used two different thresholds for detecting fake Twitter accounts. The first threshold was created by computing the maximum reconstruction error of the training set, which was originally proposed by Dau et al. [28]. They used real Twitter accounts for training a denoising autoencoder network, and the maximum reconstruction error of the training set was used for anomaly detection based on the assumption that the probability of the network producing any more significant value than the threshold for a normal data point is much lower than for an anomaly. The second threshold was calculated by taking the difference between the produced reconstruction error of two successive values in the reconstruction errors ranking. Even though they did not thoroughly discuss the reason, they stated that the latter method for computing the threshold produced better results compared to the first approach. One possible explanation could be that the second approach does a better job at avoiding local optima.

Aside from threshold-based anomaly detection approaches that use the reconstruction error of the autoencoders, there are other works in which the framework did not apply another model, such as SVM on the extracted features for separating anomalies. Akcay et al. [29] proposed a conditional generative adversarial network that consisted of an extra encoder, i.e., changing the architecture of the network to be an encoder-decoder-encoder pipeline, to extract meaningful latent variables from images. Then, instead of using another model to separate the anomalies, they used a threshold-based approach for detecting anomalies. The binary classification was based on an anomaly score value that was computed for each test example. To compute the score values, they considered using the difference between the latent variables from the first encoder and the second encoder in which the assumption was that an anomaly would output a higher score compared to a normal score due to the intrinsic features of anomalies that vary from inliers. In another work, Zhou et al. [30] proposed a denoising autoencoder based on the concept of robust principal component analysis that can be trained without having access to a noise or anomaly free training set. In their proposed approach, the network split the data into $X = L_D + S$ in which L_D contains the latent variables while S captures the anomalies and noise. The process is carried out by optimizing the objective function, which is solved by combining concepts from backpropagation and Alternating Direction of Method of Multipliers (ADMM) [31].

Despite the fact that autoencoders can capture non-linear features, which is a substantial advantage over linear models such as PCA, the network tries to transform the whole input data into one cluster regardless of the nature of the characteristics of the input data, which might have come from separate clusters. Bui et al. [32] tried to tackle this problem by proposing a model named k-means Shrink AutoEncoder (KSAE) that applies k-means clustering algorithm to split the training data into several clusters and then trained a variant of autoencoder that was originally introduced by Cao et al. [23] for each model. For detecting anomalies, they proposed two different approaches. The first one took a similar approach to the training phase in which, after splitting the test set by applying k-means, latent features were obtained by the corresponding autoencoder. In the second approach, all the trained autoencoders models were applied. For separating the anomalies, One-Class SVM was applied to the latent features. One of the drawbacks of their approach was in the method they used for clustering the data as k-means is a linear approach and cannot take into account the non-linear features when clustering the input data.

Even though the normal data in many scenarios fall under the same cluster as they tend to have a high resemblance, there are cases in which the data might belong to various clusters. For instance, when dealing with network data, each example can be formed by data coming from different network services. Nguyen1 et al. [33] proposed a hybrid approach called Clustering-based deep AutoEncoder (CAE), in which two regularisers were added to the loss function of an autoencoder to push the normal data points in the training set to the centre of the cluster and after extracting latent features from the test set, they used Centroid (CEN) method to detect anomalies. In their approach, the parameter k that defines the number of clusters is chosen empirically, which, arguably, has a substantial effect on the performance of the model. Finding the optimum k value requires either prior knowledge about the data or should be computed using another approach.

There are other variants of artificial neural networks that have been used for anomaly detection aside from autoencoders. For instance, Convolutional Neural Network (CNN), which is a supervised method unlike autoencoders, have shown promising results in other areas such as image processing, object detection or activity detection. In Sabokrou et al. [34], a variant of CNN, i.e., Fully Convolution Neural Networks (FCN), was proposed for processing crowded scenes and detecting abnormal regions, i.e., abnormal behaviour, in videos. Their proposed model was essentially an improvement over what was proposed by Sabokrou et al. [35] who also used a variant of CNN for anomaly detection for detecting abnormal behaviour from crowd scenes. The difference between the two models was that the model proposed by Sabokrou et al. [34] consisted of two main parts, the trainable part and the fixed part in which the fixed part is used as a reference for normal behaviour that is trained by a single class, i.e., the normal class. This fixed part is, in fact, copied from the Alexnet model [36]. For the trainable part, they employed a sparse autoencoder to extract latent variables from the normal class. During the testing phase, any frame deviating from the normal reference model was considered as abnormal behaviour, i.e., anomalies. One problem with this kind of approach, as mentioned by Oza et al. [37], is that training such models is generally hard.

It is worth mentioning that CNN is often but not always used when the input data is of type of image or video. Using CNN alongside autoencoders has been experimented in various research works. Hasan et al. [38] proposed a framework for detecting temporal regularity in videos, i.e., anomalies, that was a combination of a fully-connected autoencoder and a fully-convolutional autoencoder. However, the training set used during the experiment phase was done by passing handcrafted features, and the anomaly detection was done by using the reconstruction error as the anomaly score.

As mentioned, various versions of deep learning methods have been investigated and studied for the purpose of anomaly detection, and promising results were achieved. Besides the variants that were mentioned above, many other architectures such as Restricted Boltzmann Machine (RBM) [39, 40], Recurrent Neural Networks (RNN) [41] and Deep Belief Networks (DBM) [42, 43] have also been studied.

2.2 Challenges

Anomaly detection is a challenging task due to various reasons. There are several challenges that hinder the process of detecting anomalies. In this section, these challenges are briefly explained. The suitable approach for dealing with these challenges is not discussed as it is deemed out of the scope of the section.

2.2.1 Curse of Dimensionality

As mentioned in Section 1.4, anomaly detection is applied to various domains in which many of them come with an enormous amount of data. For instance, telecommunication companies produce a large amount of data every day [44]. One aspect of this is the number of features, which is known as the "curse of dimensionality". In a high-dimensional space, data instances become more spread out, leading to decreased density, which in turn causes the convex hull to become stretched and difficult to distinguish [4]. High-dimensional datasets are very complicated, require larger amounts of memory and cause longer computing time that makes the detection process extremely difficult and time-consuming [17, 45].

The goal of a dimensionality reduction algorithm is to extract features from the initial data and transform that into a lower dimension while retaining valuable information and discarding dispensable features. Algorithms such as Principal Component Analysis (PCA) [46], Latent Dirichlet Allocation [47] and Latent Semantic Indexing with Singular Value Decomposition [48] are some of the well-known and widely-used ones for reducing dimensionality. While algorithms such as PCA are using a linear approach, there are non-linear algorithms such as autoencoders [49, 50] that are based on training multilayer neural networks. This study has used a novel variant of autoencoders for reducing dimensionality. The details of the used neural network are explained more in details in Section 2.1.4 and Section 4.2.

2.2.2 Imbalanced Data Distribution

A common problem in real-world datasets is that distributions are often imbalanced (also known as skewed data distributions). In an imbalanced binary dataset, the instances are not equally distributed amongst classes as one class, known as the majority class, includes more instances than the other class, which is called the minority class [51]. For instance, in a data set that is related to medical diagnosis, there might be only a few cases that have cancer, with many cases being normal. This is a severe problem for supervised learning algorithms where often there are only a few abnormal instances for training, which makes training more challenging due to the resulting skewed distribution [52]. In a typical imbalanced dataset, the ratio between the minority and majority classes can be, for instance, 1 to 100, 1 to 1,000, 1 to 10,000 or even more [53].

2.2.3 Availability of Data

The paucity of publicly accessible data to perform research on is one of the issues that hinders doing research in this area [54, 55]. Companies are in many cases not keen on providing their data to researchers due to the confidential information that the data contain. Also, some laws prevent companies from furnishing researchers with data for experimental purposes. By using data augmentation methods, it is possible to regenerate similar samples from the original data with a minor variation to overcome this challenge.

In this research, unsupervised learning is chosen due to the fact that a well-known problem in the anomaly detection area is not having an ample amount of examples for anomalous cases. Unsupervised methods are needless of having access to a labelled training set, which makes them suitable for this area.

2.2.4 Concept Drift

Concept drift refers to the condition of an online supervised learning system where the distribution of the input and output changes, which will affect the prediction model, and can be defined as [56]:

$$\exists X : p_{t_0}(X, y) \neq p_{t_1}(X, y) \tag{2.1}$$

where p_{t_0} is the joint distribution at time t_0, X refers to the input features, and y refers to the output. In supervised learning, the model is trained with the input features X and the respective output y. In the prediction phase, a new set of (previously unseen) input features X is given, and the aim is to predict the output y. Concept drift can happen when normal behaviours keep evolving or altering, for example, when the purchasing behaviour of customers changes on special occasions such as the new year. Hence, the model cannot perform accurate predictions since, under a more general perspective of drift, the relationship between the input features and the output has changed. Therefore, concept drift requires either updating the model incrementally or re-training it with recent batches of data [56, 57]. Adaptive learning is a solution to the concept drift problem where classical learning is not suitable. It is an advanced method of incremental learning in a non-stationary environment where the system has the capability of adapting to the stream of data [58, 56].

2.2.5 Noisy Data

Most real-world datasets are incomplete, noisy, and contain redundant or obsolete records [17]. Therefore, many researchers tend to apply a preprocessing step before designing their model to clean the dataset and transform it into a suitable form. Noisy data can cause severe effects on the anomaly detection process. Noise is known as meaningless data that can cause variations in observations [59].

2.2.6 Misclassification Costs

Misclassification happens when a normal instance is incorrectly classified as an anomaly (also known as a false positive), or when anomaly is classified as a normal instance (also known as a false negative). In some domains such as fraud detection, the cost of a false positive misclassification is unequal to the cost of a false negative misclassification [60]. To explain further, the cost of a false negative is more expensive than a false positive because a false positive can be classified correctly after further investigation, but a false negative means that the fraudster has managed to stay undetected and can continue committing fraud.

Besides the aforementioned challenges, there are other issues that should be noted. An essential requirement of a supervised learning approach is labelled data; however, its availability is often an issue; moreover, labelling can be costly, time-consuming, and requires an expert [4]. Also, an anomaly can have various meanings in different application domains as some have a more generic form while others have a specific form [1], and often it is tough and expensive in some areas to provide labels for anomalous cases such as failures in aircraft engines [61]. Moreover, the performance of fraud detection systems depends heavily on the sources of data, and data often originates from different sources with different formats and standards [62]. For instance, attributes can be binary, categorical, continuous, or a mixture of these.

Chapter 3

Prune-based Local Outlier Factor

As mentioned in Section 1.6 and with more details in Section 1.6.1, the contribution of this work is divided into three parts in which the first part is about addressing the issue of efficiency and performance. In this chapter, one of the widely used density-based methods for anomaly detection is reviewed along with its intrinsic problems that weaken its performance. Then the main contribution of this chapter is elaborated, which is to improve the efficiency and performance of the method, followed by the details and the results of the experiments that were conducted.

3.1 Introduction

As explained in 2.1.2, unsupervised approaches for anomaly detection can be categorised into various groups such as density-based methods, distribution-based or cluster-based. Arguably, there is no perfect approach that achieves the global solution. In other words, each approach comes with some deficiencies. Nonetheless, many published works such as the studies published by Cao et al. [23, 63] have used density-based approaches for anomaly detection as it performs well unless the data has some characteristics such as having regions that vary in density [1].

Some density-based algorithms, such as what Stein et al. [64] proposed, aim at computing the density of the whole data, which yields a global density estimation. Based on this estimation, and by defining a threshold, a portion of the data set that has low density is separated and labelled as anomalies. The main problem with these algorithms is that often the users are merely interested in finding anomalies with respect to the local instability [65].

3.2 Local Outlier Factor

A widely used density-based anomaly detection method is Local Outlier Factor (LOF), in which the assumption is that anomalies tend to stay outside of dense neighbourhoods because of their peculiar characteristics that distance them from inliers [66]. The method generates a score that shows the outlierness of each data point based on its local density. A low score indicates that the query point is an inlier while a high score shows that the query point is an anomaly. The algorithm has one parameter, k, which is the minimum number of neighbours that each data point is expected to have inside its neighbourhood.

It is possible to divide the anomaly detection process of LOF into three steps. In the first step, LOF tries to find the minimum distance, called $k\text{-}distance$, to hold at least k neighbours. Next, the algorithm measures the reachability distance defined as:

$$reach\text{-}dist_k(p, o) = max\{k\text{-}distance(o), d(p, o)\} \tag{3.1}$$

which is equal to $k\text{-}distance$ of point o if the query point is also inside point o's neighbourhood, otherwise it is equal to the actual distance between the two points. In the second step, the local reachability distance (LRD), which is the inverse of the average reachability distance of the data point p from its neighbours, is measured. LRD is defined as:

$$LRD_k(p) = 1/\left(\frac{\sum\limits_{o \in N_k(p)} reach\text{-}dist_k(p,o)}{|N_k(p)|}\right) \tag{3.2}$$

in which k denotes the number neighbours in data point p's neighbourhood that is used to detect anomalies. In the final step, the LRD of the query point is compared with the LRD of its neighbours using the following equation:

$$LOF_k(p) = \frac{\sum\limits_{o \in N_k(p)} \frac{lrd_k(o)}{lrd_k(p)}}{|N_k(p)|} \tag{3.3}$$

If the density of point p is very close to its neighbours, then the value of LOF stays around 1 while for inliers it is less than 1 and for anomalies, it is more significant than 1. It is worth mentioning that it is prevalent to apply a simple threshold-based clustering here to separate anomalies.

3.2.1 Problems of LOF

Most of the published studies have tried to overcome the well-known problems of LOF, which are its expensive computation and also dealing with data sets that are constituted of micro clusters with different densities.

Zhang et al. [67], argued that the value that LOF produces to show outlierness of each object is not smooth, which can result in discontinuities in measuring the local outlierness. Besides, they argued that the accuracy of LOF is very sensitive to the selection of its parameters. To overcome these problems, they proposed an adaptive approach in which they employed a Gaussian Kernel function and managed to increase the discrepancy between normal cases and anomalies.

In another approach, LOF was used to detect faults in monitoring photovoltaic (PV) systems [68]. They claimed that although LOF could perform well on large-scale PV systems, its performance was unsatisfactory when utilised for detecting faults in small-scale PV systems, especially when there is irradiance. Therefore, they presented a new modified LOF, namely PVLOF, based on the conventional LOF. In their approach, the length of each PV string is considered in calculating a new density estimation based on LOF's outlier score. Despite the fact that their modification suited well in their application, it is not generalisable to other applications.

Another problem of LOF is that it only focuses on the object's density with reference to its neighbourhood without considering the sparsity and the degree of dispersion between objects. For instance, when a sparse cluster is located near a dense cluster, that can result in the wrong estimation. This becomes a bigger problem, especially when dealing with big data sets in which the objects are scattered. Su et al. [65] tried to take into account the degree of dispersion of each object with respect to its neighbourhood. In their approach, they considered using the distribution of the distance between objects in computing the local density. A new neighbourhood dispersion was introduced, which is a ratio between each object's distribution and its k-nearest neighbourhood average distance.

Jin et al. [69] proposed finding both neighbours and reverse neighbours of each object for computing the local density. In their approach, called INFLO, the local density of each object is estimated based on a symmetric neighbourhood relationship. They also introduced a micro-clustering step before computing k-nearest neighbours, which caused a reduction in the computational cost.

3.3 Proposed Solutions in the Literature

Breunig et al. [15] proposed a density-based method known as LOF in which a value is given to each data instance, which shows the outlierness for that individual. Then, the top-n points with the highest LOF value are considered as the anomaly. Often, the points with LOF value above 1 are regarded as the anomaly [70]. One disadvantage of LOF is its complexity, i.e., $O(N^2)$ where N is the size of the data [71]. Chiu et al. [72] proposed three improvements: LOF', LOF'' and $GridLOF$ to simplify the computation of the original LOF. In LOF', merely the ratio of $k{-}dist$ of the query object and its neighbour is used to compute the LOF value. They argued that this ratio is sufficient, and it is needless to compute the reachability distance and local reachability density. Furthermore, in LOF'', they employed two k instead of just one to enhance the performance of LOF. Their last enhancement, $GridLOF$, is a pruning-based approach that removes dense areas so that computing LOF for data instances inside the dense areas is not required anymore. However, the drawback of $GridLOF$ is that it requires a manual grid setting, which is not always feasible.

In a previous study, Goldstein et al. [70] randomly divided the data set into several chunks and then computed the k-nearest neighbours of each data instance only based on the instances within the query point's chunk. Next, using the computed nearest neighbours, they generated LRD and LOF for all the data instances. However, this requires a precise neighbour selection, and also the efficiency of their approach aggravates as not all the anomalies may be detected.

Another pruning approach was proposed by Pamula et al. [73] in which a clustering algorithm was applied in advance, and then dense clusters are pruned based on this assumption that they contain no anomalies. Next, they employed a Local Distance-based Outlier Factor (LDOF) to the sparse and small clusters to detect anomalies. Earlier, Pamula et al. [74] proposed a similar method that after clustering the data set by using k-means, instances that are close to the centroid of the cluster, which they belong to, were pruned, and LDOF was used merely on instances that were away from the centroid, i.e., outside of a predefined radius.

Rizk et al. [75] claimed that their approach could reduce not only the calculation rate of LOF but also minimises the false-negative rate. Their approach has two stages. In the first stage, the data instances are clustered using k-medoids, which they believe its robustness against anomalies and noise is more than k-means. Next, after computing a local cut-off value based on the size of the cluster, instances that are outside of the radius are considered potential anomalies. In the final stage, LOF value is computed only for potential anomalies that were obtained from the previous stage.

Poddar et al. [76] presented a generic method for reducing the execution time of many density-based and distance-based anomaly detection algorithms. In their method, a new density estimation called $devToMean$ was introduced that, based on its value, normal data instances were pruned, then the anomaly detection method was applied only on the rest of the data. However, the computation of $devToMean$ is expensive; therefore, they used k-means to divide the data set into small clusters and then calculated the value of $devToMean$ for each instance within its cluster.

3.4 Prune-based LOF

One common drawback of methods that were proposed in the literature is that they all need to cluster the data set in advance and then prune a portion of the data instances based on a metric. This kind of approach brings in the complexity of the clustering method into LOF. In our proposed method, there is no need to cluster the data set, which means it is simpler than similar works.

To understand the proposed method, first should look at the density equation:

$$\rho = \frac{m}{v} \tag{3.4}$$

where m is the mass and v refers to the volume. According to Xiong et al. [77], in order to calculate the density of a data instance, firstly, the number of neighbours within a radius of k-distance should be counted. As depicted in Figure 3.1, the point p has 21 neighbours, which is interpreted as the mass and k-distance is interpreted as the volume. Therefore, the density of point p is defined as:

$$Den_{p,n} = \frac{|N|}{k\text{-}dist(p,n)} \tag{3.5}$$

where $|N|$ is the number of neighbours of point p, i.e., the mass, and k-distance in which point p has at least k neighbours.

As depicted in Figure 3.2, in cases where most of the neighbours of the point p make a very dense and small cluster, if the number of neighbours does not reach the k value, the radius needs to increase in order to have at least k neighbours. In such cases, k-distance becomes inaccurate due to extreme values. To address this issue, we calculate the average k-distance using the following equation:

$$Average = \frac{\sum_{i=0}^{n} dist(p,i)}{|N|} \tag{3.6}$$

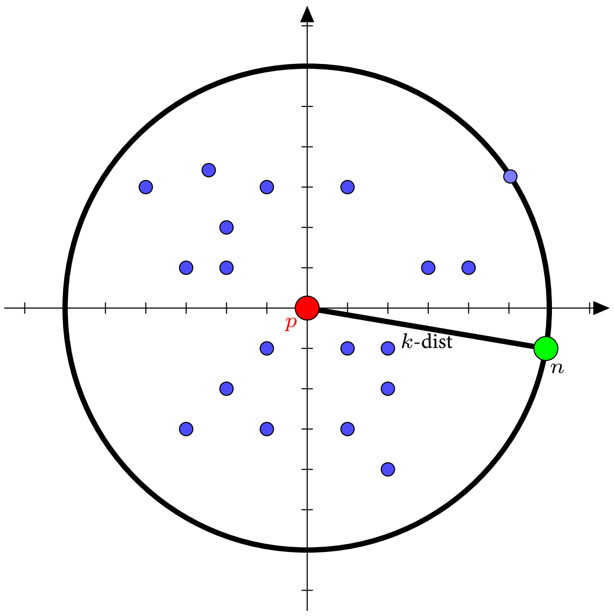


Fig. 3.1 Neighbourhood of p based on k-distance

where $dist(p, i)$ is the distance between p and its i-th neighbour. Having Equation 3.5 and 3.6, can make the following equation:

$$\delta = \frac{|M|^2}{\sum\limits_{i=0}^{n} dist(p, i)} \tag{3.7}$$

where $|M|$ is the cardinality of p's neighbours and $dist(p, i)$ is the distance between p and its i-th neighbour.

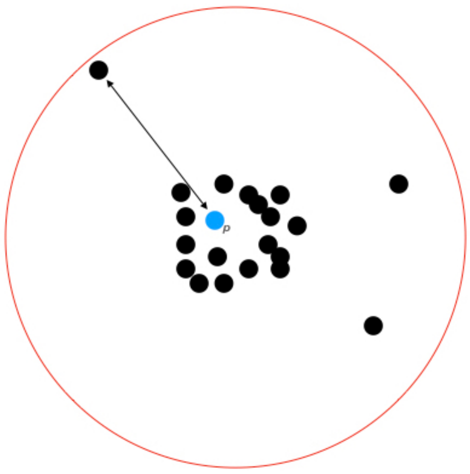


Fig. 3.2 Inaccurate k-distance

The Equation 3.7 is used to compute the density of each data instance. Having a set of δ values, the median is determined and based on the assumption that anomalies should have a low δ value (i.e., low density), all the instances that have a δ value more significant than the median are pruned. To remove the effect of extreme values [65], the largest and the smallest values of δ are eliminated. This simple median-based pruning method removes the problem of finding a reasonable threshold. It is worth mentioning that pruned instances are still used for computing reachability-distance, local-reachability-distance, and local outlier factor for not pruned instances. Finally, the LOF value for points that their δ value is less than the median is assigned to 0. The pseudo-code of PLOF is given in Algorithm 1.

Algorithm 1: Prune-based LOF Algorithm

```
input  : $D$: a data set with $n \times m$ dimension
output: $\lambda$: a $1d$ array
1  initialise: $k = minimum\ neighbours$
2  $S$ = $\delta$ values
3  for each point $x_i$ in $D$ do
4      $NN_i$ = find the $k$-th neighbour $x_j$ for $x_i$
5      $\delta_i$ = compute $\delta$ for $x_i$ given $NN_i$
6      append $\delta_i$ to $S$
7  end
8  eliminate extreme values of $\delta$
9  $med$ = median of $S$
10 for each $x_i$ in $D$ do
11     if $\delta_i > med$ then
12         prune $x_i$
13     end
14 end
15 for each $x_i$ in $D$ do
16     if $x_i$ not pruned then
17         compute reachability-distance of $x_i$
18         compute local reachability density of $x_i$
19         $\rho_i$ = compute LOF value of $x_i$
20         append $\rho_i$ to $\lambda$
21     else
22         append 0 to $\lambda$;
23     end
24 end
25 return $\lambda$
```

3.4.1 Complexity Analysis

As mentioned, one of the goals of this contribution is to improve efficiency; therefore, it is important to explain the complexity of PLOF. The complexity depends on the use of KD-tree. In the case of using KD-tree, the time complexity of finding the nearest neighbours is $O(N * \log N)$, where N is the number of instances in the data set. In the case of using a Brute-Force for getting the nearest neighbour, the complexity becomes $O(N^2)$.

The complexity of computing δ value of each data instance is $O(N)$ while the complexity of calculating the median of δ values is $O(1)$. Furthermore, the complexity of LOF is $O(N^2)$ [65]. By pruning, the size of N reduces, i.e., ($N' \ll N$); therefore, the complexity of computing LOF is substantially reduced.

3.5 Analysis of PLOF

The proposed approach is compared with the original LOF algorithm [15], and also the following two state-of-the-art similar approaches: FastLOF [70] and devToMean [76]. The two latter approaches are selected because their primary purpose is to reduce the complexity and execution time of LOF. The original LOF requires merely one parameter, which is the minimum number of neighbours (k). But, it is possible to define a threshold and pick data instances whose LOF value exceeds the threshold. During the experiment, this parameter stayed fixed for every approach.

3.5.1 Datasets

The proposed method is carried out on seven real data sets obtained from UCI Machine Learning Repository [78], which are all publicly available. In the context of anomaly detection, the data sets that are chosen in this paper are considered as the benchmark and have been widely used in the literature. The detail of each data set that is used in our experiment is given in Table 3.1. It is worth mentioning that because most of these data sets are originally used in classification, some of the data sets are modified slightly, e.g., merging one or two major classes to form the normal class.

3.5.2 Evaluation Metric

Various evaluation metrics have been used in the literature; however, the most effective and widely used metric, especially for evaluating unsupervised methods, is

Table 3.1 The details of the 6 datasets used in

Data set	No. of features	No. of anomalies	No. of data
Wine	13	10	129
Lymphography	18	6	148
Glass	9	9	214
Ionosphere	33	126	351
WBC	30	21	278
Heart	22	15	187
Breast	9	239	683

Receiver Operating Characteristics (ROC) curve, which basically shows the true-positive rate versus the false-negative rate at different threshold [12]. By calculating the Area Under ROC (AUC) curve, it is possible to show the effectiveness of the algorithm by a single value. The value of AUC can vary between 0 to 1 in which any value closer to 1 indicates better performance, while 0.5 shows random decision making. In this paper, we have used AUC, and also other evaluation metrics such as accuracy, precision, and recall [79]. The experiment on each data set is carried out five times, and the averaged outcome is reported.

3.6 Results

In terms of execution time, our approach came second after FastLOF. However, as it was stated previously, our objective is to decrease the computation cost without sacrificing the performance. Therefore, before making any conclusion merely based on the execution time, one should consider other metrics as well.

Accuracy is another prevalent evaluation metric. In terms of accuracy, devToMean showed a better average accuracy. Despite the fact that our method produced the second-best accuracy on average, it is worth noting that devToMean performed better by a very small margin, i.e., 0.034. Also, by calculating the standard deviation of PLOF's accuracy scores on different data sets, it can be deduced that PLOF yielded satisfactory results on all seven data sets with a small variation.

While FastLOF showed the worst results based on precision, PLOF dominated by a good margin, i.e. 0.067, from the second-best model, which was devToMean (0.339). Also, PLOF was able to produce the best result in terms of recall (0.842), while LOF came second (0.481).

Table 3.5 shows the area-under-curve for models that were used in these experiments. It is worth noting that in the context of anomaly detection, AUC is one the most common and reliable evaluation metrics and its value is widely used for eval-

Table 3.2 The execution time in seconds

Data set	PLOF	LOF	devToMean	FastLOF
Wine	0.156	0.229	0.335	0.067
Lymphography	0.207	0.396	0.215	0.079
Glass	0.415	0.576	0.425	0.113
Ionosphere	1.167	1.297	1.081	0.214
WBC	1.320	1.719	1.243	0.238
Heart	0.340	0.445	0.358	0.125
Breast	4.336	5.163	4.335	0.633
Average	1.134	1.403	1.141	**0.209**

Table 3.3 Accuracy of PLOF, LOF, devToMean and FastLOF

Data set	PLOF	LOF	devToMean	FastLOF
Wine	0.783	0.946	0.922	0.682
Lymphography	0.743	0.743	0.926	0.743
Glass	0.734	0.659	0.921	0.715
Ionosphere	0.877	0.638	0.661	0.678
WBC	0.757	0.646	0.915	0.685
Heart	0.572	0.519	0.578	0.540
Breast	0.851	0.613	0.630	0.848
Average	0.759	0.680	**0.793**	0.698

Table 3.4 Precision of PLOF, LOF, devToMean and FastLOF

Data set	PLOF	LOF	devToMean	FastLOF
Wine	0.263	0.714	0.500	0.030
Lymphography	0.136	0.136	0.143	0.056
Glass	0.125	0.000	0.100	0.036
Ionosphere	0.895	0.495	0.706	0.566
WBC	0.186	0.000	0.000	0.062
Heart	0.536	0.446	0.778	0.481
Breast	0.701	0.463	0.150	0.857
Average	**0.406**	0.322	0.339	0.298

Table 3.5 AUC of PLOF, LOF, devToMean and FastLOF

Data set	PLOF	LOF	devToMean	FastLOF
Wine	0.882	0.882	0.637	0.416
Lymphography	0.866	0.866	0.562	0.547
Glass	0.808	0.344	0.534	0.479
Ionosphere	0.849	0.589	0.537	0.627
WBC	0.871	0.342	0.485	0.520
Heart	0.552	0.498	0.532	0.518
Breast	0.885	0.624	0.487	0.809
Average	**0.816**	0.592	0.539	0.559

Table 3.6 Recall of PLOF, LOF, devToMean and FastLOF

Data set	PLOF	LOF	devToMean	FastLOF
Wine	1.0	1.0	0.300	0.100
Lymphography	1.0	1.0	0.167	0.333
Glass	0.889	0.000	0.111	0.222
Ionosphere	0.746	0.413	0.095	0.444
WBC	0.905	0.000	0.000	0.333
Heart	0.357	0.298	0.083	0.298
Breast	1.0	0.661	0.013	0.678
Average	**0.842**	0.481	0.109	0.344

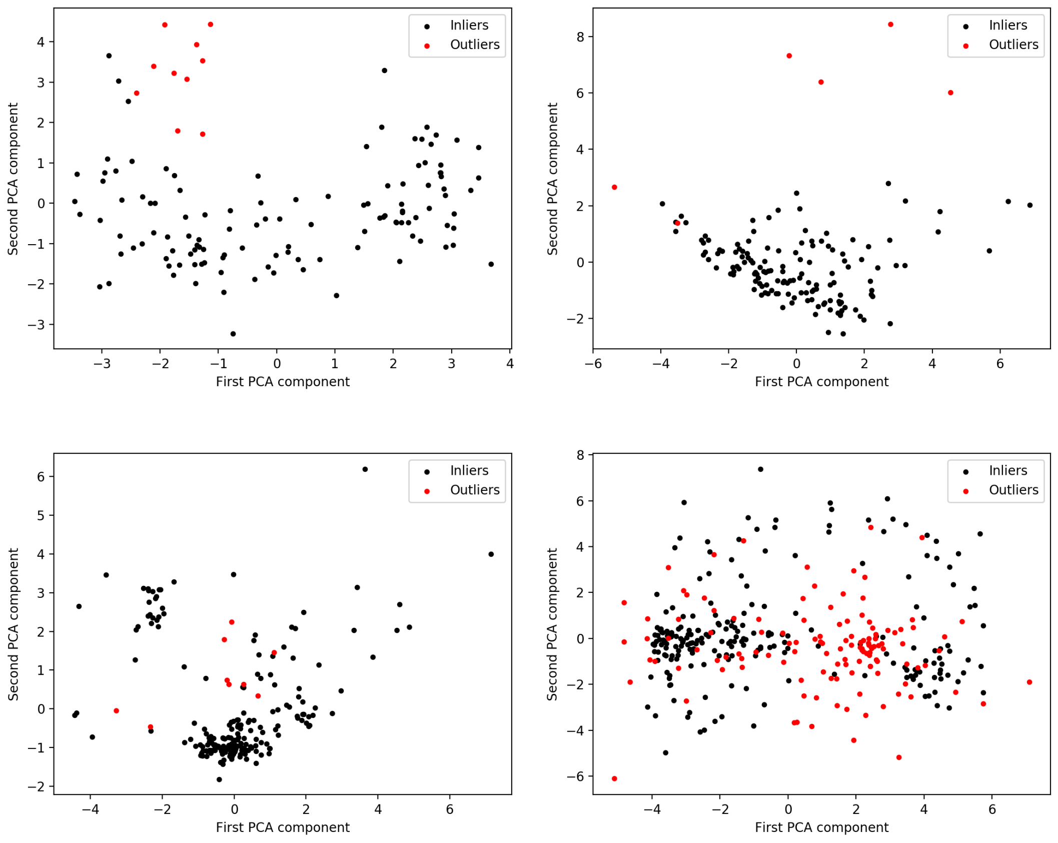


Fig. 3.3 Showing inliers and anomalies in the following four data sets: (A) Wine, (B) Lymphography, (C) Glass, and (D) Ionosphere. During the experiments, no dimensionality reduction method was applied and merely for better visualisation. The first two principal components are shown here.

uation in similar works such as the study of published by Cao et al. [23, 80]. PLOF was found to be the best model, i.e., performing better not only on average but also on every data set. On average, PLOF achieved 0.816 in AUC. In Figure 3.3, for the purpose of better visualisation, the two first principal components of the datasets used during the experiments are selected.

3.7 Summary

In this chapter, a pre-processing method was proposed, called PLOF, which aims at improving the efficiency of LOF by reducing the execution time while maintaining the performance. The proposed method firstly computes a value for each data instance which shows the outlierness of that point. Next, by employing a simple median-based threshold clustering, data instances whose density value is higher than the median are pruned. Finally, LOF is merely applied to the rest of the data that require further investigation. The results of the experiment show that PLOF can reduce the execution time while producing better precision, recall and AUC.

In the following two chapters, the two other contributions of this work are presented. While this chapter focused on improving LOF on the two aforementioned aspects, in Chapter 4, a novel variant of autoencoders is presented to improve the performance of anomaly detection methods, including LOF, when the dimensionality of the input data is increased, which causes various problems that are discussed in the chapter. Then, in Chapter 5, after discussing the importance of having an ample amount of normal data instances and its effect on defining a better class boundary, a new approach is proposed for generating augmented data while decreasing the dimensionality to improve the performance of LOF and other anomaly detection methods.

Chapter 4

AutoEncoders with Gradient Reversal

The previous chapter presented the first out of the three contributions of this research work that are pointed out in Section 1.6, which was about improving LOF efficiency and performance. However, as it will be explained in more details in this chapter, there are other problems that affect the performance of LOF. So, this chapter explains the details of the second contribution of this work in which a novel variant of autoencoders is presented that is capable of reducing the dimensionality of the input data without the need for a noise and anomaly free training set. After going through the importance of reducing the dimensionality of data and its impact on methods such as LOF, the proposed solution is elaborated, followed by extensive experimentation to demonstrate its effectiveness.

4.1 Introduction

The data captured from areas such as social networking websites or telecommunication companies often come in large volumes. Traditional methods, LOF in particular, are found to be less effective and efficient when performed on datasets that have a large dimension [81, 82]. Most of the approaches in the literature are incapable of producing high performance when applied to large-scale and high-dimensional data, and those methods that can, often require ample storage space for the intermediate obtained data [23, 83]. Consequently, such data should be reduced in dimensionality before applying other machine learning methods. This is achieved by performing a pre-processing step known as dimensionality reduction, which tries to remove irrelevant data while keeping essential features and converting it into a lower dimension.

Moreover, parameter tuning for classical approaches such as clustering models in large-scale datasets is a problematic task [84]. Consequently, the dimensionality of such datasets should be reduced before applying anomaly detection methods. This is achieved by converting the input data to low-dimensional data, which improves classification and data storage [24]. Besides high dimensionality, the lack of labelled datasets is another problem in this context. Many supervised learning algorithms have been employed to detect anomalies; however, an essential requirement of a supervised learning approach is labelled data. While the availability of labelled datasets is often a problem, labelling can also be costly, time-consuming, and requires a field expert [4]. Lastly, in many application domains, such as telecommunication fraud and computer network intrusion, companies are often very conservative and protective of their data due to privacy issues and tend to resist providing their data [83].

4.2 AutoEncoders

AutoEncoders, previously known as auto-association, are unsupervised artificial neural networks that contain two components [24]. The first component, known as the *encoder*, tries to transform the high-dimensional input data into a low-dimensional feature space, known as the *bottleneck*, while the second component, known as the *decoder*, attempts to reconstruct the input data from the bottleneck. The difference between the reconstructed and input data is called the *reconstruction error*. The dimension of the middle layer is lower than the input and output even when it has more number of nodes in which a constraint is applied to activate or deactivate some of those nodes. In each training iteration, the network measures the reconstruction error, computes the gradient of the error with respect to network parameters (e.g. weights), and backpropagates these gradients through the network in order to update the weights to minimise the reconstruction error, i.e., to increase the resemblance between the generated output and the input. AEs can adopt various structures. In Figure 4.1, the structure of an AE, known as Stacked Autoencoder (SAE), is shown in which multiple layers are stacked to form a deep neural network.

As shown in Figure 4.2, the most basic AE with only one hidden layer tries to transform input x into latent vector z using an encoder represented by function u. Next, the latent vector u is reconstructed by a decoder represented as function v into output y where the dissimilarity between y and x is called reconstruction error. Having a training set, $D_u = \{x_1, x_2, x_3, ..., x_n\}$, where n is the number of data points in D and x_i is the ith data point with m features, the encoder is then defined as:

$$z = u(x) = s(Wx + b) \tag{4.1}$$

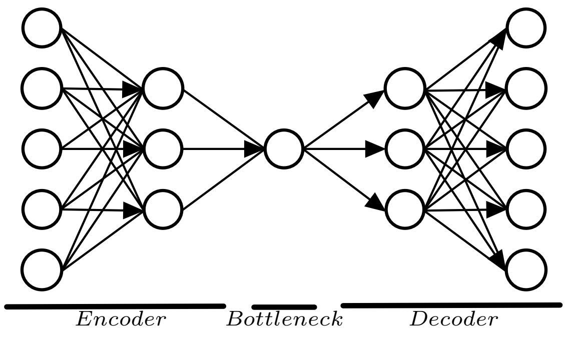


Fig. 4.1 The structure of a deep undercomplete autoencoder

while the decoder is defined as:

$$y = v(z) = s'(W'z + b') \tag{4.2}$$

where both s and s' represent the activation functions that are most often non-linear, W and W' denote the weight matrices while b and b' represent the bias vectors.

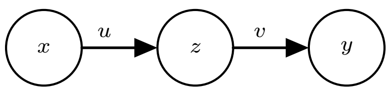


Fig. 4.2 The structure of a basic autoencoder

It is worth mentioning that, in general, certain restrictions or regularisation techniques should be applied to an AE in order to prevent the network from learning the identity function. Otherwise, the network will copy the input through the network. A common solution to overcome this issue is by using a denoising AE. In a Denoising AutoEncoder (DAE), the network tries to reconstruct inputs that are partially corrupted. Bottlenecks and sparsity constraints applied to the main hidden layer are additional ways to avoid learning the identity function. In terms of network structure, autoencoders come in various types such as under-complete, over-complete, shallow or deep. A comprehensive review was published by Charte et al. [85].

4.2.1 Problems of AE in Anomaly Detection

Unlike other dimensionality reduction methods such as Principal Component Analysis (PCA) that use linear combinations, AEs perform a nonlinear dimensionality reduction and, according to the literature, demonstrate better performance compared to PCAs [50]. Sakurada et al. [86] used AE for anomaly detection and compared its performance to linear PCA and kernel PCA on both synthetic and real data, and based on the result, they concluded that AE can extract more subtle anomalies than PCA. Another disadvantage of statistical algorithms such as PCA or Zero Component Analysis (ZCA) is that as the dimensionality increases, more memory is required to calculate the covariance matrix [87].

As Qi et al. [88] mentioned, the performance of AEs diminishes when the data exhibits noise and anomalies. To explain further, the network tries to reduce the reconstruction error by learning how to reconstruct noise and anomalies. During the training phase, anomalies tend to produce a more significant reconstruction error as their patterns deviate from the distribution that the majority of data points follow. The network, thus, disproportionately backpropagates the corresponding error gradients and performs a substantial weight update to be able to reproduce these patterns with lower error. However, in the context of anomaly detection, this is not desired. In fact, in previous works, various approaches have tried to prevent the network from becoming more accurate in reproducing anomalies. Most of these approaches, such as DAE require access to noise and anomaly-free set for training the network, while the proposed model is needless of such a training set.

4.2.2 Proposed Solutions in the Literature

A well-trained autoencoder should generate minor Reconstruction Errors (REs) for each data point; however, autoencoders fail at replicating anomalies because their patterns deviate from the pattern that the majority of data instances follow. In other words, the reconstruction errors of anomalies are significantly above the REs for normal data. In some studies such as what Aygun et al. [89] conducted, the reconstruction error was used as a score in a threshold-based classification approach to separate normal data from anomalies, i.e., the data instances that have a reconstruction error above the threshold are identified as anomalies. In contrast, anything below the threshold is considered normal. In another similar approach, Schreyer et al. [90] generated an anomaly score based on reconstruction error and individual attribute probabilities in large-scale accounting data.

Chen et al. [91] proposed an approach in which ensembles of AEs were used to improve the robustness of the network. In their approach, named Randomised Neural Network for Anomaly Detection (RandNet), instead of using fully connected layers, the connections between layers are randomly dropped, and anomalies are separated from normal data using the reconstruction error. Their approach showed superior performance compared to four traditional anomaly detection methods, including LOF.

To enhance the performance of AEs, Sun et al. [83] added new regularisers to the loss function that encourage the normal data to create a dense cluster at the bottleneck layer while the anomalies tend to stay outside of the cluster. Next, they employed various OCC methods such as LOF, Kernel Density Estimation (KDE), and One Class Support Vector Machine (OCSVM) to divide the data into anomalies and normal data. They also investigated the influence of altering the training set size on the performance of their models and the result showed consistency across different training sizes.

As the number of hidden layers increases, the backpropagated gradients to the lower layers tend to attenuate, which is known as the vanishing gradient problem and culminating in the weights of lower layers showing the limited change. To overcome the vanishing gradient issue and discover better features, a pretraining phase was proposed by Yousefi-Azar et al. [87] in which a stacked Restricted Boltzmann Machine (RBM) was used to obtain suitable weights for initialising the AE. The model was applied to the NSL-KDD dataset and achieved high accuracy (83.34%).

The performance of AEs diminishes when datasets contain anomalies and/or noise which is very prevalent in real-world datasets. By using Denoising AEs (DAEs), it is possible to enhance the accuracy of the network. DAE is an extension of AE in which the network is trained on an anomaly and noise-free set while random noise is added to the input and the AE tries to regenerate the original input, i.e., without the added noise [92]. Sakurada et al. [86] used a DAE to obtain meaningful features and decrease data dimensionality. Their result proved that DAE outperforms statistical methods such as PCA.

DAEs require an anomaly and noise-free training set; however, such a training set is not always available. In related work, Zhou et al. [30] proposed an AE, called Robust Deep Autoencoder (RDA), that can extract satisfactory features without having access to an anomaly or noise-free training set. Inspired by Robust PCA (RPCA), they added a penalty-based filter layer to the network that uses either L_1 or $L_{2,1}$ norms, and managed to separate anomalies and noise from normal data.

A common criterion used in AEs is Mean Square Error (MSE). As mentioned before, anomalies and noise tend to cause more significant reconstruction error, i.e., larger MSE; consequently, the network carries out a substantial weight update. Therefore, it debilitates the accuracy of AEs as they tend to learn to regenerate anomalies and noise. A possible remedy to this problem is using a criterion that is insensitive to anomalies and noise. Qi et al. [88] proposed a new approach, called Robust Stacked AutoEncoder (R-SAE), in which they used the maximum correntropy criterion to prevent substantial weight updates and make the model robust to anomalies and noise. They tested their model on the MNIST dataset contaminated with non-

Gaussian noise and achieved 39%-63% lower REs compared to what was obtained from a Standard Stacked AutoEncoder (S-SAE).

4.3 AutoEncoder with Gradient Reversal

The proposed AEGR (AutoEncoder with Gradient Reversal) approach consists of two main components. The first component is an AE which transforms the high dimensional data into compressed data while preserving important latent variables for the following component. The second component employs a basic LOF approach on the features obtained from the first component to separate anomalies from normal data points.

As stated by Qi et al. [88], the performance of AEs diminishes when the data exhibits noise and anomalies. To explain further, the network tries to reduce the reconstruction error by learning how to reconstruct noise and anomalies. During the training phase, anomalies tend to produce a greater reconstruction error as their patterns deviate from the distribution that the majority of data points follow. The network thus disproportionately backpropagates the corresponding error gradients and performs a substantial weight update to be able to reproduce these patterns with lower error. However, in the context of anomaly detection, this is not desired. In fact, in previous works, various approaches have tried to prevent the network from becoming more accurate in reproducing anomalies. Most of these approaches, such as DAE require access to a noise and anomaly-free set for training the network, while the proposed model is needless of such a training set. Ganin et al. [93] tried to add a reversal layer to their network for the purpose of domain adaption. The approach proposed in this work is different in several ways, including the context within which it is applied and the fact that unlike the approach proposed by Ganin et al. [93], it is based on the reconstruction error.

In AEGR, the AE component tries to make the network insensitive to anomalies by manipulating gradients. First, as shown in Algorithm 2, at each epoch, a gradient score is given to each data point (or each mini-batch in the case of using mini-batch gradient descent) based on the gradients in the bottleneck. This is measured by using the Frobenius norm, which is defined as:

$$||X||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|x_{i,j}|^2} \tag{4.3}$$

where X denotes a $m \times n$ matrix and x_{ij} represents the element at the ith row and jth column. At the bottleneck layer, the Frobenius norm is computed. Having an

Algorithm 2: AEGR

```
input  : $D$: a dataset with $n \times m$ dimension
         $e \leftarrow$ number of epochs
output: $D'$: a dataset with $n \times m'$ dimension
1  $GS$ = gradient score;
2  $LGS$ = list of gradient scores;
3  for each epoch do
4      for each point/mini-batch in $epoch_j$ do
5          if bottleneck then
6              $GS_j \leftarrow ||n||_F$
7              $LGS \leftarrow GS_j$
8          end
9      end
10     $Max_{GS} \leftarrow$ Max $GS$ in $LGS$ ;
11     Perform backpropagation;
12     Bacpropagate inverted $Max_{GS}$;
13     Shuffle the batches;
14 end
15 $D' \leftarrow$ the bottleneck of the last epoch
```

AE with three layers in which the number of nodes in layer one to three is $49, 12, 49$, respectively, the values of m and n would be 12 and 49, respectively. The bottleneck is the layer that latent variables are extracted from to be used in LOF; therefore, the network should be penalised based on the magnitude of its gradients in this layer rather than every layer. Then, the norm of layer l is measured and stored as:

$$\{l_1, l_2, l_3, ..., l_n\} \tag{4.4}$$

where n denotes the number of data points (or mini-batches in case of using mini-batch gradient descent) in the network, the approach has a single parameter, k, which is the epoch number when the network starts reversing its gradients.

Assuming that anomalies produce greater REs, at each epoch, the gradient of the data point that holds the most significant gradient score is picked, and the inverse of its gradient is used to perform a new weight update, i.e., reverting the substantial weight update caused by this data point. In order to avoid reverting the gradients of the same data points at each epoch, batches are shuffled before carrying out the next epoch. There is no artificial noise added in the training phase to make the network insensitive to noise. Noise in data can be very similar to anomalies, except the analyst is not interested in noise [1]. It is assumed that the network avoids learning reproducing noise based on the fact that they also impose a big gradient update on the network. Consequently, by applying the same method that is applied to anomalies, the AE is expected to stay weak in replicating noise. While the input of Algorithm 2 is a matrix of $n \times m$, where n is the number of rows and m is the number of columns, it outputs a matrix with $n \times m'$ dimension in which $m' < m$ since the network tries to reduce the dimensionality by reducing the number of features.

In the second phase, of which LOF is used to separate anomalies from normal data points, three approaches are proposed. The first approach merely uses the latent variables that are extracted from the previous stage without any changes, while in the second approach, a threshold-based clustering method based on the reconstruction error of the training set is applied to prune data points that have generated an error above the mean value. The assumption behind this approach is that the training set becomes more homogeneous, and consequently, LOF can perform more robustly. However, this approach also reduces the number of training points that can weaken the performance of one-class classifiers [7]. Therefore, a third approach is proposed in which the pruned training data is augmented by adding random Gaussian noise. The assumption is that Data Augmentation (DA) can homogeneously increase the training set's size and improve anomaly detection performance. More details about data augmentation is available in Chapter 5.

4.3.1 Complexity Analysis

In terms of the algorithm's time complexity, Step 4 to 11 has a complexity of $O(n)$ in which n is either the number of data points or the number of batches. The time complexity of Step 13 is $O(1)$. In this algorithm, the input size is constant, which is equal to n, and the space complexity is $O(n)$.

4.4 Analysis of AEGR

In this section, the performance of the proposed model is presented, evaluated and compared with 1) the basic stand-alone LOF algorithm; 2) a deep AE in which the reconstruction error is used for separating anomalies, and 3) a deep AE in which the latent variables are fed to LOF for detecting anomalies.

The number of layers was set to 5 for all the datasets as suggested in [94] with the number of nodes in the bottleneck being set to $m = [1 + \sqrt{n}]$, where n is the number of features of the input [80]. Table 4.1 shows the number of nodes in the bottleneck for each dataset.

For datasets with more than 2000 instances, the mini-batch size was set to 64 and 16 otherwise. To avoid overfitting, a simple early stopping heuristic was imple-

Table 4.1 The size of the middle layer for each dataset

Dataset's name	Number of nodes in the bottleneck
PenDigits	5
Shuttle	4
Spambase	8
InternetAds	40
Arrhythmia	17
NSLKDD	12
UNSW-NB15	15
CTU13	7

mented to stop the training process when the network was no longer learning after a certain number of iterations or when the learning improvement was insignificant.

The loss function used in this experiment was $Smoothl1Loss$, which is essentially a combination of $L2$ and $L1$ terms, i.e., it uses $L2$ if the absolute element-wise error is less than one and $L1$ term if not. To minimise the loss function, Stochastic Gradient Descent (SGD) was used. Although the main focus here is not optimisation, it is worth mentioning that SGD has shown to perform robustly and can avoid converging to bad local optima if a reasonable learning rate is chosen [95].

All the datasets were normalised into the range of $[-1, +1]$, and the hyperbolic tangent function was chosen as the activation function. The activation function is defined as:

$$f_{tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{4.5}$$

where the output range is $(-1, +1)$. There are various activation functions, and choosing the right one can be challenging. Not every activation function performs well in AEs. For instance, ReLU is a well-known activation function in deep learning; however, it can weaken the performance of AEs [85]. Using an empirical approach, it was realised that the hyperbolic tangent activation function shows the highest performance.

4.4.1 Datasets

As presented in Table 4.2, eight datasets were used that are publicly available and widely used in this context. It is worth mentioning that five network datasets that are very common in the domain of network anomaly detection were used besides three non-network datasets. Testing the model on various datasets makes it possible to evaluate performance more thoroughly. It is worth noting that the categorical features of two datasets, namely NSL-KDD and UNSW-NB15, were preprocessed by one-hot-encoder. Four datasets, namely Spambase, InternetAds, Arrhythmia and

Table 4.2 Details of the 8 datasets used in the experiments

datasets name	No. of features	Training set	Validation set	Testing set
PenDigits	16	1247	312	727
Shuttle	9	3267	817	14500
Arrhythmia	259	251	83	86
Spambase	57	2759	919	923
InternetAds	1558	1414	471	474
NSL-KDD(Probe)	122	6319	1580	12132
NSL-KDD(DoS)	122	9060	2266	17169
NSL-KDD(R2L)	122	6183	1546	12598
NSL-KDD(U2R)	122	5428	1358	9778
UNSW-NB15(Fuzzers)	196	5934	1484	74184
UNSW-NB15(Analysis)	196	4640	1160	58000
UNSW-NB15(Backdoor)	196	4619	1155	57746
UNSW-NB15(DoS)	196	5460	1366	68264
UNSW-NB15(Exploits)	196	7151	1788	89393
UNSW-NB15(Generic)	196	7680	1920	96000
UNSW-NB15(Reconnaissance)	196	5319	1330	66491
UNSW-NB15(Shellcode)	196	4570	1143	57133
UNSW-NB15(Worm)	196	4584	1146	56130
CTU13-13(Virut)	40	4315	1438	1440
CTU13-10(Rbot)	38	7331	2443	2445
CTU13-09(Neris)	41	12895	4298	4301
CTU13-08(Murlo)	40	8045	2681	2683

CTU13, have no separate training and test sets; therefore, 60% of each dataset was used for training while the rest was evenly split for validation and testing. For the remaining five datasets that come with a separated training and test set, 20% of the training set was used for validation. Also, as suggested in [23], the training sets of UNSW-NB15 and NSL-KDD are substantially larger than other datasets; therefore, only 10% of the training set was used. This work found it necessary to apply the same size reduction approach to the CTU13 and Shuttle dataset. The details of each dataset are shown in Table 4.2.

The following datasets were obtained from the UCI Machine Learning Repository: PenDigits, Shuttle, Spambase, and InternetAds [78]. The CTU13-08 dataset was released in 2011, which is a botnet dataset and publicly available [96]. The NSL-KDD dataset is a new version of its predecessor, KDD'99, which is considered a benchmark in the area of anomaly detection [97]. In NSL-KDD, some of the intrinsic issues of its old version, mentioned in [98], are resolved. This dataset has 41 features, of which three are categorical, and after applying a one-hot-encoder, the number of features increased to 122. A similar preprocessing step was carried out on the UNSW-NB15 dataset [99] with three categorical features, which increased the number of features from 42 to 190. While five datasets contain only a single type of anomaly, three network datasets, namely UNSW-NB15, NSL-KDD and CTU13, include different types of anomalies (i.e., network attacks). The experiment was carried out on each type of these attacks.

All the datasets in Table 4.2 come with labels; however, the labels are not used in training the models. However, the labels are necessary for evaluating the performance of the models. Therefore, they are used during the evaluation.

4.4.2 Evaluation Metrics

One of the widely used evaluation metrics in this area is Receiver Operating Characteristics (ROC) AUC [100]; however, Provost et al. [101] claimed that when the datasets are highly imbalanced, ROC AUC is not a suitable metric, and a better alternative would be the area under the Precision-Recall curve (PR) AUC, in particular, when working with high dimensional data in which the positive class, i.e., anomalies, is more important than the negative class, i.e., normal points. Nonetheless, no single evaluation metric dominates others; therefore, both ROC AUC and PR AUC were used for evaluation.

4.5 Results

The performance of AEGR-LOF was compared with three different approaches. Besides employing the traditional LOF, an autoencoder with LOF (AE-LOF) and an autoencoder with RE (AE-RE) were used. In AE-LOF, the network is not using gradient reversal, and its bottleneck is fed to the next stage for separating normal instances from anomalies. In AE-RE, the reconstruction error of the AE is used for detecting anomalies. It is worth mentioning that a denoising AE was not used in the experiment as it needs an anomaly-free training set, and the purpose of this research is to stay needless of a clean training set. Also, it is possible to achieve higher performance by employing LOF if only a clean training set is used in advance to fit LOF first and use it as a one-class classifier [23], which has already been done in the literature; however, in this work, LOF is trained based on the data which is extracted from the AE.

As mentioned earlier, the evaluation metrics used here are PR AUC and ROC AUC. The ROC AUC shows the area under the receiver operating characteristic

Table 4.3 The PR AUC and ROC AUC values for UNSW-NB15

Detection approach	Modification method	Fuzzers		Analysis		Backdoor		DoS		Exploits		Generic		Reconnaissance		Shellcode		Worms	
		PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC
Stand-alone LOF	None	0.793	0.562	0.983	0.702	0.984	0.664	**0.926**	**0.778**	0.667	0.603	0.583	0.499	0.872	0.590	0.986	0.618	**1.000**	0.846
AE-RE	None	0.722	0.329	0.97	0.593	0.976	0.580	0.853	0.584	0.605	0.443	0.687	0.624	0.873	0.572	0.981	0.589	0.998	0.091
AE-LOF	None	0.786	0.520	0.986	0.720	0.978	0.571	0.862	0.589	0.633	0.506	0.563	0.463	0.869	0.557	0.987	0.627	**1.000**	0.877
	Pruning	0.791	0.568	0.983	0.721	0.985	0.475	0.863	0.569	0.654	0.537	0.574	0.518	0.850	0.475	0.980	0.517	**1.000**	0.881
	Pruning+DA	0.646	0.205	0.940	0.381	0.976	0.527	0.815	0.422	0.569	0.374	0.377	0.012	0.858	0.538	**0.996**	**0.850**	**1.000**	0.750
AEGR-LOF	None	0.785	0.532	**0.998**	0.948	**0.996**	**0.907**	0.858	0.601	0.620	0.499	0.597	0.518	0.858	0.538	0.988	0.632	**1.000**	0.861
	Pruning	0.810	0.574	**0.998**	**0.955**	0.974	0.509	0.848	0.543	0.601	0.475	0.674	0.626	0.841	0.467	0.979	0.485	**1.000**	**0.902**
	Pruning+DA	**0.833**	**0.576**	0.986	0.698	0.976	0.527	0.876	0.697	**0.742**	**0.655**	**0.879**	**0.758**	**0.945**	**0.772**	0.991	0.726	0.999	0.342

Table 4.4 PR AUC and ROC AUC for NSL-KDD and CTU13

Detection approach	Modification method	DoS		Probe		R2L		U2R		Virut CTU13-13		Rbot CTU13-10		Neris CTU13-09		Murlo CTU13-08	
		PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC
Stand-alone LOF	None	0.549	0.555	0.830	0.635	0.810	0.687	0.994	0.530	0.644	0.781	0.845	0.465	0.708	0.060	0.940	0.594
AE-RE	None	0.547	0.440	0.800	0.056	0.790	0.543	0.994	0.569	0.431	0.463	0.815	0.057	0.864	0.003	0.920	0.001
AE-LOF	None	0.522	0.492	0.821	0.645	0.756	0.525	0.994	0.551	0.439	0.558	0.807	0.254	0.747	0.199	0.933	0.564
	Pruning	0.549	0.512	0.895	0.742	0.757	0.409	0.994	0.516	0.371	0.412	0.799	0.241	0.981	0.880	**0.988**	**0.908**
	Pruning+DA	0.687	0.781	0.683	0.296	0.766	0.548	0.993	0.451	0.518	0.639	0.895	0.523	0.966	0.900	0.800	0.068
AEGR-LOF	None	0.514	0.488	0.844	0.677	0.823	0.679	0.997	0.789	0.552	0.699	0.857	0.514	0.719	0.088	0.928	0.577
	Pruning	**0.925**	**0.883**	**0.947**	**0.841**	**0.859**	**0.685**	**0.998**	**0.817**	0.472	0.588	**0.899**	0.547	0.891	0.452	0.979	0.782
	Pruning+DA	0.643	0.695	0.797	0.598	0.674	0.287	0.993	0.550	**0.797**	**0.832**	0.859	**0.597**	**0.992**	**0.985**	0.917	0.332

Table 4.5 PR AUC and ROC AUC for data sets with single-type anomaly

Detection approach	Modification method	Spambase		InternetAds		PenDigits		Shuttle		Arrhythmia	
		PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC
Stand-alone LOF	None	0.596	0.418	0.857	0.498	0.554	0.529	0.700	0.343	0.638	0.626
AE-RE	None	0.606	0.324	0.838	0.188	0.483	0.368	0.792	0.007	0.558	0.289
AE-LOF	None	0.579	0.428	0.853	0.515	0.574	0.590	0.819	0.689	0.681	0.678
	Pruning	0.650	0.497	0.909	0.638	0.980	0.970	**0.839**	**0.720**	0.703	0.688
	Pruning+DA	0.503	0.334	0.859	0.538	0.315	0.078	0.635	0.105	0.605	0.616
AEGR-LOF	None	0.551	0.377	0.854	0.521	0.491	0.499	0.689	0.398	0.698	0.671
	Pruning	0.558	0.388	**0.928**	**0.687**	**0.998**	**0.998**	0.696	0.414	**0.728**	**0.732**
	Pruning+DA	**0.682**	**0.646**	0.854	0.538	0.970	0.958	0.812	0.559	0.512	0.484

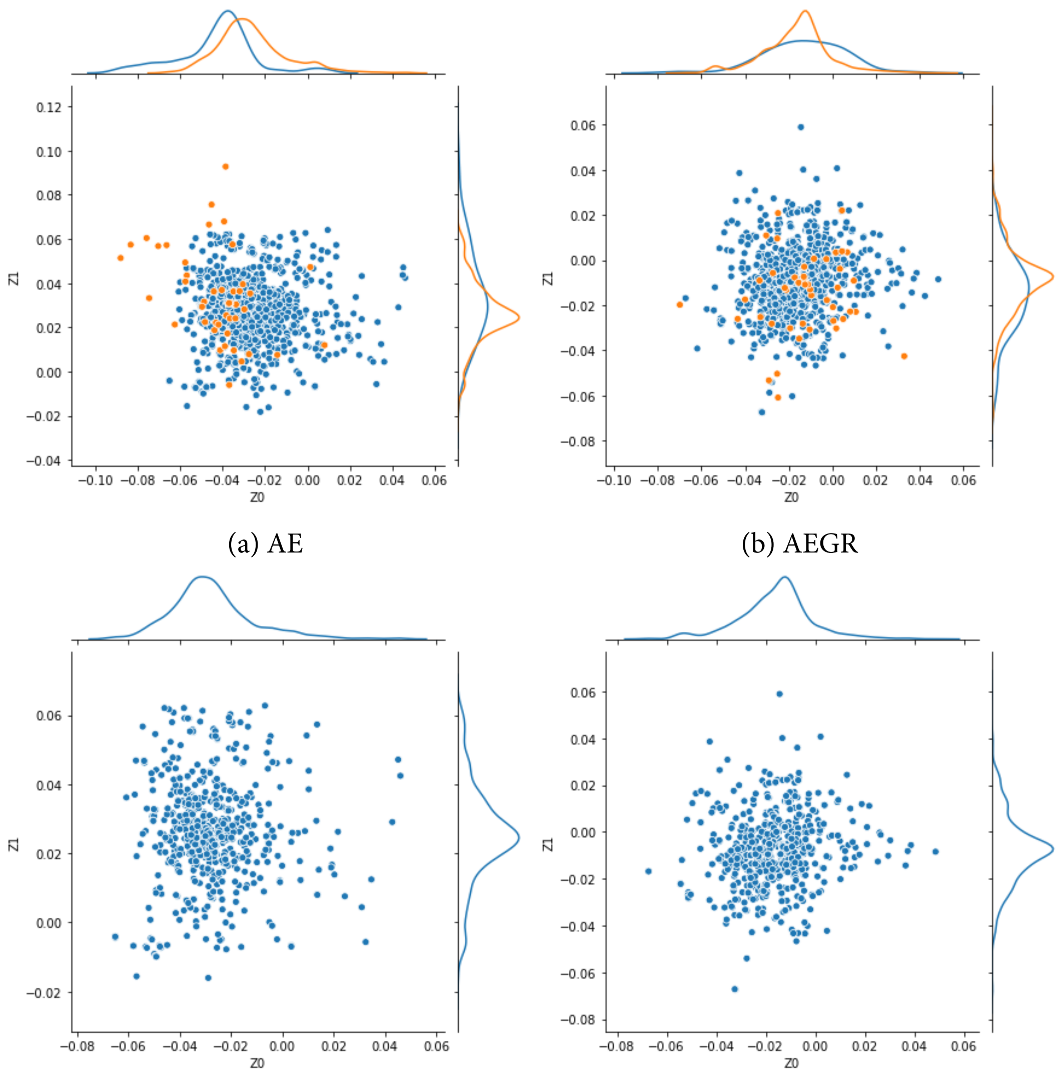
curve, which presents the true-positive rate versus the false-negative rate at various thresholds. The range of AUC is $[0, 1]$ where any value closer to 1 shows that the model is performing better. In PR AUC, the true negatives have no impact on the metric. Instead, it reports the relationship between precision and recall at various thresholds. Similar to ROC AUC, the range is $[0, 1]$ in which values closer to 1 indicate improved performance.

It can be seen in Table 4.3 that the proposed model arguably outperformed other approaches in every type (i.e., different types of network attack) in terms of both metrics. Stand alone LOF only showed superior results when applied to detect DoS attacks while also showed good results alongside other approaches when used to identify Worms. The UNSWB-NB15 dataset is a high dimensional dataset, and the results can be used to support the assumption that LOF performs poorly when applied to this type of dataset. The NSL-KDD and CTU13 are two network datasets that are prevalent in the literature. As shown in Table 4.4, the proposed model outperformed other approaches except in one case, i.e., when applied to CTU13-08.

Table 4.5 shows the performance when applied to datasets with a single type of anomaly. Based on the results, the proposed approach outperformed other models except when applied to the Shuttle dataset. In particular, AEGR-LOF produced higher PR AUC and ROC AUC when detecting anomalies from the two single-type network datasets, i.e., Spambase and InternetAds.

Figure 4.3 illustrates the latent variables produced by the simple AE and the AEGR model. While Figure 4.3a and Figure 4.3c show the latent variables extracted from the simple AE's bottleneck, Figure 4.3b and Figure 4.3d present the latent variables generated by the proposed model. By looking at Figure 4.3a and Figure 4.3b, which show the latent variables before pruning, it can be seen that while the simple AE managed to regenerate some anomalies correctly and separate them from the normal data points, the AEGR model failed at separating anomalies due to the constraint applied, i.e., they stayed close to the dense area. However, this failure means that the AEGR model should have generated high reconstruction errors for anomalies. After applying a simple pruning based on the reconstruction error (Figure 4.3c and Figure 4.3d), the latent variables created by the AEGR model made a denser cluster compared to the simple AE, which led to getting a better performance from LOF as it can define a better class boundary when the training set is very dense and homogeneous.

Overall, AEGR-LOF with pruning and pruning and data augmentation outperformed other approaches except in 5 out of 22 cases. Therefore, it can be concluded that AEGR-LOF is significantly better than other approaches, which shows the importance of regularisation in AEs when used in anomaly detection problems. To sup-



(a) AE (b) AEGR

(c) AE (Pruned) (d) AEGR (Pruned)

Fig. 4.3 Visualisation plot of the latent variables (the first two features) extracted by a simple AE and also AEGR on InternetAds. The orange points represent anomalies while the blue points represent the normal points. The curves at the top and right side of each figure indicate the KDE curves.

port this conclusion further, Wilcoxon signed-rank test [102] was carried out to verify whether the improvement was significant or not. Therefore, the NSL-KDD(DoS) dataset was randomly selected, and the test was carried out on both LOF and AEGR-LOF with pruning and data augmentation 20 times, i.e., $N = 20$. By feeding PR AUCs to the Wilcoxon test, it was confirmed that the results are significantly different. It is worth recalling that carrying out repeated Wilcoxon tests on multiple algorithms is not recommended as it increases the chance of rejecting a certain proportion of the null hypotheses merely based on random chance [103].

4.6 Summary

In this chapter, a new model is proposed to detect network anomalies, particularly in large datasets, that traditional algorithms such as LOF are incapable of dealing with. In the proposed approach, a novel autoencoder called AEGR is utilised to reduce the dimensionality of large datasets, transforming the data into a lower-dimensional space while minimising the loss of vital features. Regular AEs fail to produce satisfactory results when the data is polluted with noise and anomalies because the network learns to replicate them together with normal data instances. Unlike other approaches that either try to use an insensitive loss function or train the network by injecting noise, the unsupervised model presented in this work, at each epoch, finds the data instances that caused the highest weight update and then manipulates the inverted backpropagated gradients to counter that update. Finally, the original LOF is applied to the extracted features from the autoencoder's bottleneck to separate normal instances from anomalies. Based on the results that were achieved from the experiments conducted on seven datasets, it was shown that the AEGR-LOF model is capable of achieving better results compared to the traditional LOF and other similar approaches such as a standard stacked Autoencoder followed by a threshold-based classifier. The performance of the proposed model was evaluated using two metrics in which, overall, the AEGR model showed superior results.

In the next chapter, the last contribution out of the three contributions explained in Section 1.6 of this work is explained. As it was recapped above, AEGR is suitable for datasets with high dimensionality. The next chapter aims at dealing with the problem of class boundary and the lack of ample amount of training data. A novel approach is presented that tries to deal with the high dimensionality problem while augmenting more positive data points for the purpose of training and improving the definition of class boundary.

Chapter 5

Data Augmentation by AutoEncoders

As mentioned in Section 1.6, the contribution of this work is divided into three parts. The previous two chapters covered the first two contributions. In Chapter 3, a new variant of a density-based anomaly detection method known as LOF was proposed that focused on increasing the efficiency of the algorithm in terms of computation consumption while maintaining and improving the performance. The problem of working with high dimensional data in which the definition of density becomes harder and complicated was dealt with in Chapter 4. A new dimensionality approach based on a variant of an AE network was proposed that is needless of having an anomaly and noise-free training set as it is made insensitive towards noise an anomaly during the training phase. This chapter covers the final contribution of this work. After recapping the concept of imbalanced distribution, data augmentation and its effect on defining class boundary are explained. Then, the proposed solutions in the literature are reviewed, followed by a new proposed approach. Finally, the novel approach is tested and compared with other methods to demonstrate its effectiveness.

5.1 Introduction

As mentioned in the previous chapter, deep neural networks have demonstrated their effectiveness and managed to improve the state-of-the-art in various application areas such as image recognition. As explained in Chapter 2.2, anomaly detection suffers from several challenges, for instance, high dimensionality, concept drift and, in particular, imbalanced data distribution (also known as skewed distribution). To briefly recap the imbalanced distribution problem, having a dataset with two classes in which a high portion, e.g., 90%, of the data comes from the first class while the rest of the data comes from the second class. In this example, the distribution of the dataset is skewed, and this weakens the performance of the model and can cause a high misclassification rate [1]. In particular, supervised approaches suffer more because it is hard to obtain examples of anomalies for training the model. Therefore, unsupervised models or one class classifier algorithms appear to be more suitable. In an OCC algorithm, the model is trained with a training set that only includes data instances from one class (known as the target class), and the model is expected to separate data instances of the target class from non-target class instances in the test set [7]. In order to employ an OCC, it is necessary to have access to a training set that includes merely normal examples. It is worth noting that in some scenarios, the training set includes a small number of data points from other classes as well. This training set is used to set the threshold at which non-target and target points will be divided. In an anomaly detection problem, the goal is to separate anomalies from normal points; therefore, an anomaly-free training set is required for tuning the OCC algorithm.

5.2 Data Augmentation

The performance of OCC methods depends on various factors, including the size of the training set. These methods can perform better when the training set includes more data points as it makes it feasible to compute a less ambiguous class boundary for dividing data points of the target class from the rest [7]. Data augmentation refers to the process in which the training instances are synthetically regenerated to balance the dataset and ultimately improve the model's performance [104]. This approach is widely used in machine learning tasks and very effective in deep learning when working with small size datasets. However, the benefits of data augmentation have not been tested in various areas, and it has been mostly used and shown positive results when applied to images [105].

5.2.1 Proposed Solutions in the Literature

There are several methods to overcome the issue of imbalanced class distributions that can be categorised into data-level, algorithmic-level, and cost-sensitive methods [106]. In a data-level method, the goal is to bring balance to the dataset prior to performing classification by oversampling, undersampling, or a hybrid approach [107]. Two widely used oversampling methods are Synthetic Minority Oversampling Technique (SMOTE) and Adaptive Synthetic Sampling (ADASYN) [108]. Oh et al. [109] proposed a new approach in which a Generative Adversarial Network (GAN) is used to enhance classification performance in imbalanced datasets. They used a variant of GAN that can also detect outliers in the majority class, which prevents creating a biased classification boundary. In a similar approach, Douzas et al. [106] proposed a Conditional Generative Adversarial Network (cGAN) for oversampling the minority class in a binary classification problem. In their approach, the network is conditioned on external information, i.e., class labels, to approximate the actual class distribution. In the area of anomaly detection, Lim et al. [110] claimed to be the first to use data augmentation in an unsupervised approach for detecting anomalies. They employed a variant of a GAN to obtain latent variables and selectively oversampled instances close to the head and tail of the latent variables by an approach similar to SMOTE. They argued that their approach addresses the problem of scarcity of infrequent normal instances, which can reduce the performance of density-based anomaly detection algorithms.

When the type of input data is not images, DA becomes more challenging, and by reviewing the literature, one can deduce that apart from using methods such as SMOTE, most of the proposed approaches use only injecting noise to generate synthetic samples [105].

5.3 Data Augmentation by AutoEncoders

The basics of AutoEncoders are explained in Section 4.2. In this section, the proposed method is elaborated upon. The purpose of this approach is to use an AE to augment the training set for the purpose of detecting anomalies. This augmented training set is later used in an OCC method that tries to detect anomalies in the test set. The performance of the OCC depends on how well its threshold is defined. The assumption here is that by augmenting the training set, it is possible to compute a better class boundary for the OCC method and ultimately improve the performance.

Having a training set, the AE tries to minimise the reconstruction error by updating its weights through error backpropagation. Traditionally, once the network is fully trained, latent variables are extracted from the bottleneck of the AE and used to train the OCC algorithm. However, in the proposed approach, the latent variables of the last ν epochs are extracted and concatenated to form an augmented training set. The value of ν is a parameter that needs to be tuned. The assumption is that the network is almost well trained during the last few epochs and that the corresponding latent variables possess an excellent representation of the original input. The algorithm of the proposed approach is summarised in Algorithm 3, in which $n' > n$ as the model is augmenting the training set while $m' < m$ as at the dimensionality is being shrunk. Another advantage of the proposed model is that, unlike other approaches, it is not necessary to perform any extra computation (in addition to reducing the dimensionality) for augmenting the training set.

Algorithm 3: Over-sampling by AE

```
   input  : $D$: a dataset with $n \times m$ dimension
   output: $D'$: a dataset with $n' \times m'$ dimension
1  $n\_epochs \leftarrow$ number of epochs;
2  $\nu \leftarrow$ portion of epochs to use;
3  for each epoch $i$ in $n\_epochs$ do
4      Reconstruct the input;
5      Compute the reconstruction error;
6      Backpropagate the loss;
7      Take a gradient step;
8      if $i \geq ((1 - \nu) \times n\_epochs)$ then
9          $z \leftarrow$ latent variables in the bottleneck;
10         $D' \leftarrow$ append $z$;
11     end
12 end
13 return $D'$;
```

5.3.1 Complexity Analysis

The elegance of the proposed approach is that it is not adding a computationally expensive step to the feature extraction process. The data augmentation starts during the last $\nu\%$ epochs (refer to Step 8 to 11 in Algorithm 3), and its complexity is $O(n)$ where n is the number of epochs where the if statement returns true.

5.4 Analysis of Data Augmentation by AutoEncoders

This section presents the evaluation of the proposed approach for augmenting the training set in order to improve the performance of anomaly detection with OCC algorithms. To demonstrate the effectiveness of the proposed approach, three OCC algorithms, namely Local Outlier Factor (LOF), Kernel Density Estimation (KDE), and Isolation Forest (ISF), were used in the experiments. The chosen OCC algorithms are widely used in the area of anomaly detection.

All the datasets were normalised using the same approach explained in Section 4.4. In addition, the same activation function and loss function used in Section 4.4 were applied here as well. As for the number of nodes inside the bottleneck, the same values that are tabulated in Table 4.1 were used here too.

5.4.1 Datasets

The experiments were carried out on eight publicly available datasets [78, 96] which are widely used in this domain. The datasets used in this chapter are the same ones used in the previous chapter (refer to Section 3.5.1).

5.4.2 Evaluation Metrics

The performance of the proposed approach is compared to the state-of-the-art. In particular, the two widely used oversampling methods, namely SMOTE and ADASYN, were used to increase the size of the training set. Also, the training set was oversampled by adding random Gaussian noise to the latent variables. In short, the performance of the proposed model is compared with the following four different approaches:

1. Latent variables are extracted from the AE, without generating augmented data and OCC algorithms are applied to detect anomalies
2. Latent variables are augmented by SMOTE, and then OCC algorithms are used
3. Similar to the previous approach except ADASYN is used instead of SMOTE for augmenting the latent variables
4. By adding random Gaussian noise, the latent variables are augmented

The fourth approach, i.e., augmenting data by adding artificial noise, was carried out to test whether the proposed approach was acting as a simple regularizer similar to a simple noise injection or whether it had a more distinctive contribution. The hypothesis was that by simply adding artificial noise, it is impossible to achieve the same level of meaningful augmented data that the proposed approach can obtain. In other words, the assumption was that this simple transformation, i.e., adding noise, does not provide as much meaningful information as using latent representations from different epochs.

As for evaluation metrics, the same evaluation metrics used in Section 4.4 were used here, namely Receiver Operating Characteristics AUC and Precision-Recall curve AUC.

5.5 Result

On each dataset, the experiment was repeated ten times. To eliminate the effect of extreme values [111], the largest and smallest values were removed, and the average performance was documented. In Table 5.1, the performance of all four different models on the UNSW-NB15 dataset is tabulated in terms of PR AUC and ROC AUC. The values that are shown in bold indicate the best performance. In the UNSW-NB15 dataset, as stated in the previous chapter, there are 9 different types of anomalies, and the experiment was carried out on each one individually. As listed in Table 5.1, the proposed data augmentation method dominated both in terms of PR AUC and ROC AUC in almost every type of anomaly. By looking at the performance of models when applied to Exploits and Reconnaissance, in terms of ROC AUC, it can be seen that the proposed model came second after the approach in which artificial noise was used for data augmentation; however, by only 0.044 and 0.003 respectively. It is worth mentioning that all approaches performed competitively well on the Worms dataset, while the proposed approach showed the second-best result in terms of PR AUC.

As depicted in Table 5.2, the proposed approach outperformed other approaches on NSL-KDD and CTU13 datasets. Also, it is noticeable that almost all the approaches performed equally well on the NSL-KDD (U2R) dataset, especially in terms of PR AUC. It can be deduced that the extracted latent variables from the AE already do a good training set for OCC algorithms, and it is relatively easy to compute the class boundary. Therefore, one can argue that data augmentation on similar scenarios is needless.

Augmenting the training set using the AE completely dominated other approaches when applied on the datasets with a single-type anomaly, i.e., PenDigits, Spambase, InternetAds and Arrhythmia; except when applied to Shuttle in which adding artificial noise with the use of KDE showed a higher ROC AUC value. While LOF outperformed Isolation Forest and KDE when working with Arrhythmia, KDE showed superior results when applied to PenDigits and Spambase. Also, it is worth mentioning that even though the proposed method dominated when carried out on InternetAds, the performance showed a considerable drop when compared to other datasets. For instance, the PR value was 0.428 achieved by using the proposed method and ISF, which was the lowest value among the best PRs throughout the experiment. The assumption is that this is due to the high sparsity of this dataset and the fact that it has the highest number of features, i.e., 1558, compared to other datasets. Thus, to capture useful latent variables without losing essential information, it requires a different AE; however, in order to make sure the performance stays comparable to other datasets, it is important to maintain impartiality in every model. Therefore, the same approach for designing the AE architecture that was used for other datasets was followed on InternetAds.

Table 5.1 The PR AUC and ROC AUC values for UNSW-NB15

Data augmentation method	OCC method	Fuzzers		Analysis		Backdoor		DoS		Exploits		Generic		Reconnaissance		Shellcode		Worms	
		PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC
None	ISF	0.825	0.478	0.995	0.782	0.996	0.813	0.949	0.689	0.758	0.502	0.939	0.878	0.914	0.531	0.990	0.476	**0.999**	0.603
	KDE	0.821	0.456	0.993	0.800	0.995	0.787	0.956	0.682	0.744	0.412	0.810	0.671	0.921	0.552	0.992	0.519	0.998	0.421
	LOF	0.837	0.437	0.996	0.847	0.985	0.475	0.843	0.402	0.693	0.515	0.921	0.552	0.877	0.349	0.985	0.332	0.998	0.328
SMOTE	ISF	0.811	0.390	0.992	0.718	0.995	0.844	0.942	0.664	0.838	0.624	0.929	0.834	0.909	0.467	0.984	0.280	**0.999**	**0.748**
	KDE	0.794	0.393	0.989	0.685	0.993	0.726	0.945	0.718	0.785	0.527	0.927	0.817	0.928	0.502	0.987	0.349	0.997	0.363
	LOF	0.792	0.285	0.962	0.240	0.971	0.311	0.853	0.436	0.710	0.466	0.718	0.622	0.841	0.275	0.982	0.295	0.996	0.104
ADASYN	ISF	0.848	0.489	0.992	0.810	0.995	0.784	0.964	0.748	0.718	0.427	0.921	0.822	0.894	0.525	0.985	0.470	**0.999**	0.475
	KDE	0.848	0.489	0.992	0.810	0.995	0.784	0.964	0.748	0.718	0.427	0.921	0.822	0.894	0.525	0.985	0.470	**0.999**	0.475
	LOF	0.837	0.440	0.969	0.376	0.987	0.585	0.834	0.352	0.734	0.385	0.662	0.523	0.891	0.346	0.986	0.341	0.998	0.527
Noise	ISF	0.828	0.480	0.994	0.792	0.994	0.737	0.965	0.776	0.864	**0.693**	0.929	0.856	0.938	0.555	0.994	0.620	0.998	0.366
	KDE	0.896	0.506	0.992	0.817	0.996	0.833	0.917	0.627	0.798	0.612	0.933	0.874	0.944	0.562	0.989	0.473	**0.999**	0.632
	LOF	0.850	0.483	0.990	0.622	0.987	0.539	0.864	0.383	0.731	0.462	0.842	0.784	0.949	**0.721**	0.991	0.488	**0.999**	0.484
Proposed method	ISF	0.892	0.566	**0.996**	**0.851**	**0.998**	**0.924**	0.970	0.826	**0.873**	0.681	0.970	0.943	0.948	0.591	**0.995**	**0.680**	**0.999**	0.713
	KDE	**0.932**	**0.668**	0.995	0.814	**0.998**	0.905	**0.974**	0.821	0.868	0.649	**0.975**	**0.959**	**0.954**	0.641	0.988	0.467	**0.999**	0.605
	LOF	0.875	0.540	0.992	0.801	0.995	0.842	0.972	**0.830**	0.792	0.642	0.861	0.787	**0.954**	0.718	0.990	0.496	0.995	0.133

Table 5.2 PR AUC and ROC AUC for the NSL-KDD and CTU13 data set

Data augmentation method	OCC method	Probe		DoS		R2L		U2R		Virut		Rbot		Neris		Murlo	
		PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC
No oversampling	ISF	0.982	0.927	0.830	0.808	0.839	0.691	**0.999**	0.828	0.384	0.100	0.783	0.024	0.749	0.100	0.064	0.094
	KDE	0.975	0.893	0.828	0.801	0.824	0.667	**0.999**	0.821	0.364	0.092	0.704	0.046	0.746	0.270	0.041	0.068
	LOF	0.683	0.188	0.539	0.535	0.657	0.179	0.981	0.140	0.840	0.809	0.718	0.082	0.882	0.426	0.108	0.671
SMOTE	ISF	0.976	0.900	0.728	0.765	0.950	0.829	0.997	0.715	0.374	0.067	0.776	0.030	0.738	0.155	0.696	0.797
	KDE	0.967	0.880	0.795	0.863	0.938	0.785	0.997	0.698	0.370	0.116	0.698	0.018	0.756	0.260	0.061	0.297
	LOF	0.627	0.082	0.382	0.132	0.626	0.187	0.980	0.078	0.585	0.651	0.718	0.065	0.834	0.312	0.233	0.865
ADASYN	ISF	0.831	0.574	0.872	0.924	0.906	0.736	0.998	0.762	0.374	0.067	0.729	0.017	0.729	0.117	0.218	0.225
	KDE	0.953	0.862	0.856	0.910	0.938	0.775	0.997	0.808	0.364	0.089	0.696	0.004	0.751	0.286	0.117	0.682
	LOF	0.625	0.097	0.369	0.070	0.636	0.222	0.977	0.047	0.455	0.312	0.696	0.004	0.750	0.222	0.044	0.166
Noise	ISF	0.940	0.806	0.897	0.858	0.926	0.785	0.991	0.468	0.504	0.261	0.859	0.039	0.824	0.344	0.213	0.776
	KDE	0.965	0.885	0.857	0.833	0.948	0.816	0.997	0.737	0.433	0.287	0.701	0.029	0.903	0.509	0.047	0.143
	LOF	0.670	0.165	0.434	0.256	0.629	0.178	0.981	0.084	0.898	**0.873**	0.838	0.612	0.890	0.475	0.547	0.878
Proposed method	ISF	0.986	0.941	0.930	0.917	0.977	**0.918**	**0.999**	**0.881**	0.619	0.535	0.946	0.668	0.834	0.455	0.504	0.904
	KDE	**0.991**	**0.972**	**0.954**	**0.931**	**0.969**	0.887	0.998	0.862	0.491	0.391	0.834	0.617	0.928	0.600	**0.763**	0.797
	LOF	0.745	0.494	0.570	0.590	0.798	0.593	0.987	0.423	**0.906**	**0.873**	**0.993**	**0.961**	**0.928**	**0.682**	0.526	**0.922**

Table 5.3 PR AUC and ROC AUC for data sets with single-type anomaly

Data augmentation method	OCC method	PenDigits		Shuttle		Spambase		InternetAds		Arrhythmia	
		PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC
No oversampling	ISF	0.694	0.712	0.714	0.440	0.296	0.327	0.160	0.525	0.440	0.500
	KDE	0.878	0.826	0.821	0.712	0.317	0.398	0.182	0.527	0.416	0.446
	LOF	0.646	0.777	0.759	0.427	0.293	0.311	0.221	0.604	0.416	0.446
SMOTE	ISF	0.557	0.371	0.784	0.511	0.341	0.461	0.134	0.411	0.438	0.443
	KDE	0.602	0.587	0.628	0.130	0.318	0.351	0.171	0.525	0.440	0.50
	LOF	0.319	0.103	0.711	0.364	0.314	0.332	0.165	0.489	0.419	0.461
ADASYN	ISF	0.725	0.715	0.727	0.400	0.313	0.376	0.168	0.525	0.373	0.365
	KDE	0.454	0.452	0.690	0.356	0.348	0.400	0.148	0.443	0.377	0.392
	LOF	0.388	0.215	0.763	0.597	0.331	0.386	0.127	0.374	0.376	0.382
Noise	ISF	0.916	0.867	0.923	0.792	0.491	0.622	0.221	0.660	0.468	0.543
	KDE	0.893	0.535	0.877	**0.808**	0.400	0.547	0.168	0.567	0.460	0.604
	LOF	0.491	0.415	0.846	0.718	0.464	0.629	0.141	0.451	0.534	0.554
Proposed method	ISF	0.947	0.918	**0.926**	0.781	0.683	0.821	**0.428**	**0.738**	0.573	0.591
	KDE	**0.960**	**0.932**	0.872	0.723	**0.714**	**0.832**	0.255	0.694	0.583	0.604
	LOF	0.930	0.908	0.890	0.744	0.614	0.761	0.368	0.719	**0.604**	**0.616**

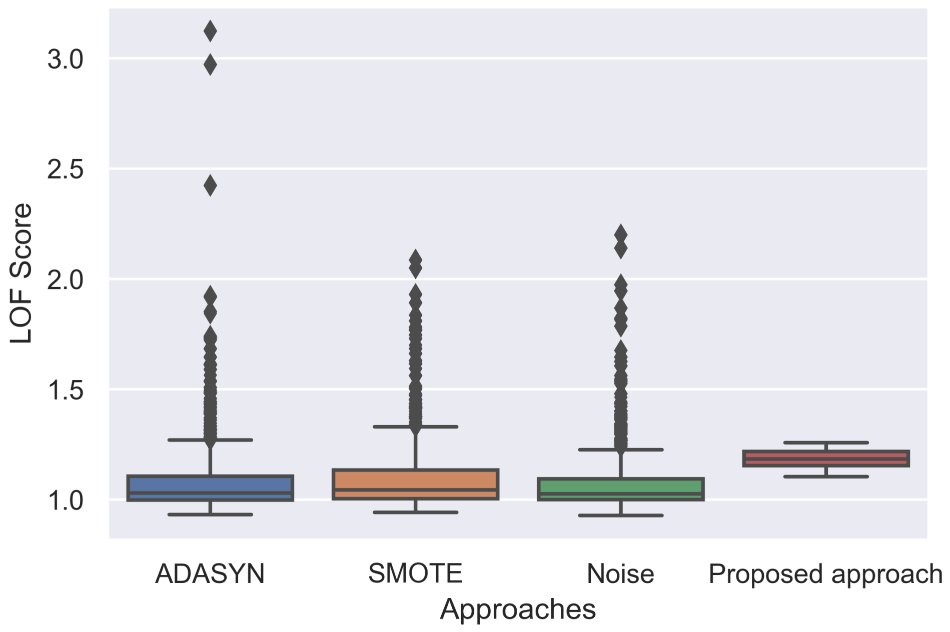


Fig. 5.1 Boxplots of computed LOF scores for PenDigits dataset after data augmentation with different approaches

In Figure 5.1, a simple statistical analysis is carried out using Boxplots to see the effect of different data augmentation models on the LOF scores obtained from the PenDigits dataset. Boxplots provide a good graphical representation, and as depicted in Figure 5.1, it can be deduced that the other three approaches caused LOF to compute a wider range of LOF scores with some outliers while the proposed approach computed LOF scores that are within a small range. By looking at Figure 5.1 and also the performance of data augmentation with the proposed model and using LOF for detecting anomalies in Table 5.3, it can be reasoned that the proposed model can augment the training set in a way that leads to finding a better class boundary for LOF. To verify whether the LOF scores are significantly different or not, a Wilcoxon signed-rank test [102] was performed between the proposed method and the adding noise approach. By feeding LOF scores to the Wilcoxon test, it was observed that the results are significantly different at $p < 0.05$. As mentioned in Section 4.5, carrying out repetitive Wilcoxon tests on multiple models is not recommended because it increases the chance of rejecting a certain proportion of the null hypotheses merely based on random chance [103].

5.6 Summary

This chapter studies the usefulness of using autoencoders to augment the training set in the feature-space to improve the performance of OCC algorithms in anomaly detection problems. When the training set is not large enough, OCCs perform poorly as finding a suitable class boundary becomes difficult. Once the AE is almost well-trained, it is possible to start deriving latent variables in each epoch and feed this augmented training set to the OCC algorithm.

By carrying out the proposed method on several well-known datasets in this domain, it was demonstrated that the proposed approach is capable of outperforming other data augmentation methods such as SMOTE and ADASYN. In terms of OCC methods, three state-of-the-art algorithms were employed to detect anomalies. According to the results, the proposed approach can produce a more robust performance compared to other approaches.

Chapter 6

General Discussion

6.1 Introduction

The aim of this work was to address a number of challenges that hinder the process of anomaly detection using unsupervised approaches. Anomaly detection suffers from various challenges, and, arguably, it was not feasible to address all of them; nonetheless, this work addressed three significant obstacles. In Chapter 3, the problem of complexity and accuracy was addressed, while in Chapter 4, high dimensionality was dealt with. Finally, in Chapter 5, another approach was proposed that targeted the problem of lack of labelled data for training. By going through the literature, one can see that there has not been a prior work that demonstrates an approach similar to what was proposed in this work for the purpose of anomaly detection.

In this chapter, a succinct review of the proposed methods is presented in which the novelty and limitations of each proposed method are discussed. Also, this chapter mentions what can be done in the future to overcome some of those limitations.

6.2 Methods

One of the unsupervised density-based anomaly detection methods that is widely used in the literature is LOF. However, it has its own limitations, such as being computationally expensive and its underperformance when applied to datasets with high dimensionality. For instance, Boniol et al. [112] reported that applying LOF to a large dataset exceeded the time-out of their experiment, which was eight hours. In another study, Xu et al. [113] tried using a Local Sensitivity Hashing (LSH) process to improve the efficiency of LOF and reduce its memory consumption. They also used a Gaussian Mixture Model (GMM) to compute LOF only for points that generated a low probability of belonging to the standard distribution model.

There is a trade-off between reducing the complexity of LOF and improving its performance. Approaches such as what Goldstein et al. [70] proposed, try to address one of the two mentioned issues while failing to cover the other one. Overcoming both the complexity of LOF and improving its performance is what PLOF attempted to demonstrate. In other words, PLOF tried to not only improve the performance of LOF but also aimed at reducing the computation cost.

As depicted in Figure 6.1, in PLOF, a novel pruning approach is applied to the dataset prior to using LOF to remove points that are deemed inliers. Then, instead of computing a LOF score for all the points, the score is calculated only for data points that are not pruned. This ensures that LOF is not unnecessarily computed for every data point, which arguably means less computation. The pruning process should make sure that anomalies are not wrongly pruned, i.e., it should preserve valuable information while marking points that are in a very dense area, which is an indication of being an inlier. Therefore, defining a suitable threshold that guarantees high accuracy is a challenge. In fact, this is very often a challenge in any machine learning problem. The focus of the research was not to address the issue of finding the optimum boundary here; thus, a simple median-based classification was used for separating inliers from outliers.

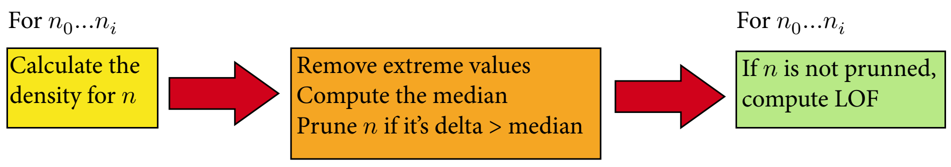

For $n_0 ... n_i$
Calculate the density for n
Remove extreme values
Compute the median
Prune n if it's delta > median
For $n_0 ... n_i$
If n is not prunned, compute LOF

Fig. 6.1 Overall structure of the PLOF model

Despite the fact that LOF shows high performance compared to other anomaly detection algorithms, no algorithm is perfect. As mentioned, one of the problems of LOF is the complexity which was address in Chapter 3. Another problem of LOF emerges when it is applied to datasets with high dimensionality. Most of the studies on LOF are centred on linearity issues and applicable to problems in which, for instance, the sparsity is low [114]. When dealing with nonlinearity issues, the performance of the traditional LOF method deteriorates as it cannot capture the underlying features and studies such as the work published by Yuxin et al. [115] tried to introduce variants of LOF that strengthen it to be able to derive the latent features. To explain further why LOF underperforms in a high dimensional dataset, the definition of sparsity warps as the dimensionality increases; also, computing the distance between points becomes more challenging as the number of features expands, which has a substantial effect on LOF [116]. As stated by O'Neil et al. [117], another problem that is caused by high dimensionality is that as the dimensionality increases, the distance between data points tends to go towards equidistance which, arguably, interferes with density computation.

As mentioned, detecting anomalies using methods such as LOF, which is a density-based technique, is challenging when working with high dimensional and complex datasets. To remedy this problem, various studies have investigated using dimension reduction techniques to transform the high dimensional data into a low dimensional space and then applying an anomaly detection method [118]. Despite the fact that this approach has shown promising results in many areas when it comes to anomaly detection, the dimension reduction approach plays a vital role. It is of great importance to make sure that the density distribution is preserved during the transformation. Therefore, choosing a suitable dimensionality reduction method has a significant effect on the performance of anomaly detection.

Deep learning has proved to be useful in anomaly detection problems [118]. Deep learning techniques come in various architectures. One of the types of deep learning architectures is autoencoders, which has been widely utilised for detecting anomalies. The details of autoencoders are explained in Section 4.2. In the literature, researchers have proposed variants of autoencoders that try to prevent the network from learning the identity function while detecting anomalies, such as denoising autoencoders or variational autoencoders. For instance, An et al. [119] proposed a variational autoencoder in which the reconstruction probability was used for separating anomalies from inliers. The anomaly decision in their approach was made by a simple threshold-based classification. It is also possible to use the reconstruction error for separating anomalies from inliers. This approach was adopted by Zenati et al. [120] in which they used the reconstruction error of a bi-directional generative adversarial network. Zenati et al. [120] applied the proposed approach on two datasets (i.e., KDD99 and Arrhythmia) and compared the results with other methods, including two classic methods, namely OCSVM and Isolation Forests, in which the results showed the supremacy of autoencoders. In many approaches, a regulariser is used to prevent the network from learning the identity function [121]. Cao et al. [23] penalised the loss function of the network by the magnitude of the latent variable in order to shrink the normal points and make them stay in a dense cluster. For separating anomalies from inliers, different methods such as LOF, OCSVM and KDE were applied.

Despite the fact that taking the reconstruction error of the network as an anomaly score has achieved results with high accuracy, it suffers from a few problems. One problem that could be referred to as a requirement is having access to a portion of the dataset that is clean. This portion should have no anomalies or even noise. The fraction is then used for training the network. Once the network knows how to reproduce normal data points, the test set is passed to the model in which there are anomalies. The network is incapable of reproducing anomalies as well as inliers because it is only familiar with what it was trained with, which is normal points. Therefore, anomalies show a high reconstruction error. Having access to a clean set is not always possible. Consequently, using models such as denoising autoencoders is not always possible [30].

To overcome the mentioned requirement, some have proposed using a loss function that is insensitive towards anomalies. In other words, during the training phase, the network generates a small error for anomalies instead of a large error that would normally make it learn how to regenerate anomalies. Qi et al. [88] used the same idea and applied a new loss function based on correntropy, which was less sensitive to non-Gaussian noise compared to other traditional loss functions such as Mean Square Error (MSE). Another correntropy-based loss function used in stacked autoencoders was proposed by Singh et al. [122] who introduced a new loss function, denoted as C_{loss}, that is a variant of MSE in which produces L_0 loss for data points with high errors while approximating the L_2 error for samples with minor errors.

The proposed solution in Chapter 4 tries to reduce the dimensionality of the dataset while preserving the vital features so that anomalies can be detected from the low dimensional version of the dataset. The model is needless of being trained by a clean training set, which is a crucial requirement in other approaches such as denoising autoencoders. As depicted in Figure 6.2, the structure of the proposed model is very similar to a standard stacked autoencoder, except AEGR tries to stop the network from learning the characteristics of anomalies by forcing an altered gradient after each epoch. To explain further, the model selects the data point that caused the highest gradient update and tries to undo that update by backpropagating the inverse of the gradient of that data point. The assumption here is that the selected data point has the highest gradient update because its characteristics are highly different from the rest of the data points. This behaviour is very common among anomalies. So, by reverting the gradient update, the network learns how to regenerate normal data points, but its efforts in learning how to reproduce anomalies are countered. Next, once the training is finished, the test set is passed to the network and the latent variables inside the bottleneck of the network are extracted, which is essentially the dataset in a low dimensional space. In the last step, another technique needs to be applied to the obtained latent variables in order to extract anomalies. As explained in Section 4.4, the proposed method was applied on several benchmark datasets to evaluate the enhancement compared to other approaches such as using a standard stacked autoencoder.

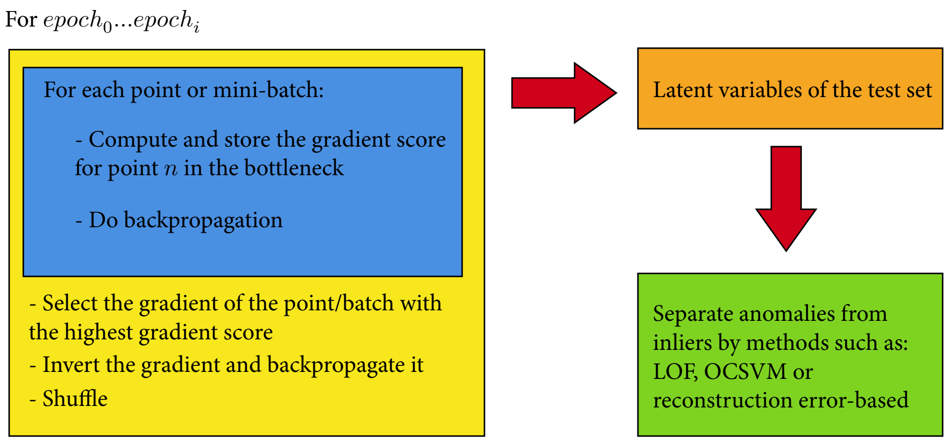

For $epoch_0 ... epoch_i$
For each point or mini-batch:
- Compute and store the gradient score for point n in the bottleneck
- Do backpropagation
- Select the gradient of the point/batch with the highest gradient score
- Invert the gradient and backpropagate it
- Shuffle
Latent variables of the test set
Separate anomalies from inliers by methods such as: LOF, OCSVM or reconstruction error-based

Fig. 6.2 Overall structure of the AEGR model

As emphasised before, the most significant advantage of AEGR is that a clean training set is not necessary. The proposed approach can filter out the points that are causing the most significant shift in the learning process to avoid learning anomalies or noise reproduction. However, it is important to consider how the network distinguishes noise from anomalies. During the training phase, both noise and anomalies are treated similarly as the purpose in AEGR is to make the network insensitive to them (the difference between noise and anomalies was explained earlier in Section 1.3.1, and it is deemed unnecessary to reiterate it here).

Several research works have used OCCs for separating instances, and some have proposed models that are tailored for anomaly detection, such as LOF. These methods are used when the negative class, e.g., anomalies, are absent or hard to define. However, these methods face various challenges, such as high-dimensionality and skewed distribution, that affect the performance. One of the challenges that was addressed in Chapter 5 is the size of the training set. In one-class classification, the model tries to learn the characteristics of a certain type of data instance and then, during the testing phase, separates examples of that category from the rest. In an anomaly detection problem, the model is trained with normal instances and is expected to separate anomalies in the test set from the inliers. However, some anomalies might have very similar characteristics to an inlier, which can cause the model to label them among the inliers, i.e., making a false-negative prediction. Defining a better classification boundary between the inliers and outliers can improve the performance of the model by accepting as many inliers as possible and rejecting as many anomalies as possible [123]. Due to the fact that during the training phase, the model has access to only one type of categories, defining the boundary becomes a challenge. For instance, the boundary should be snug enough to minimise the false-positive rate. Also, the feature selection impacts the process of defining the boundary.

To overcome the mentioned issue, a new approach was proposed in Chapter 5 in which, by using data augmentation, the size of the training set is increased to achieve a denser area filled with inliers for defining a better classification boundary. In other words, the effect of increasing the size of the training set by using data augmentation on the performance of OCC models for the purpose of anomaly detection was investigated. Data augmentation refers to a process in which synthetic data instances of a class is generated. However, traditional approaches such as statistical methods are incapable of producing high-quality instances that share the same characteristics that real data instances have with minor variations [124]. In the proposed approach, an AE was used to not only generate synthetic data of the normal instances but also to reduce the dimensionality of the dataset.

As depicted in Figure 6.3, the data augmentation and dimensionality reduction are carried out during the training phase of the AE model. In other approaches such as AEGR, the latent variables are extracted in the last epoch in which training the model is going to finish. However, in the proposed approach, it starts earlier. The approach needs a threshold to be defined that determines when to start collecting latent variables. For instance, the extraction can start during the last ten epochs in which the model is very close to being fully trained. During the last n epochs, the latent variables of the network's bottleneck are expected to be very similar but different with some variations. In other words, the network is generating synthetic data points that are from the same class with the same characteristics.

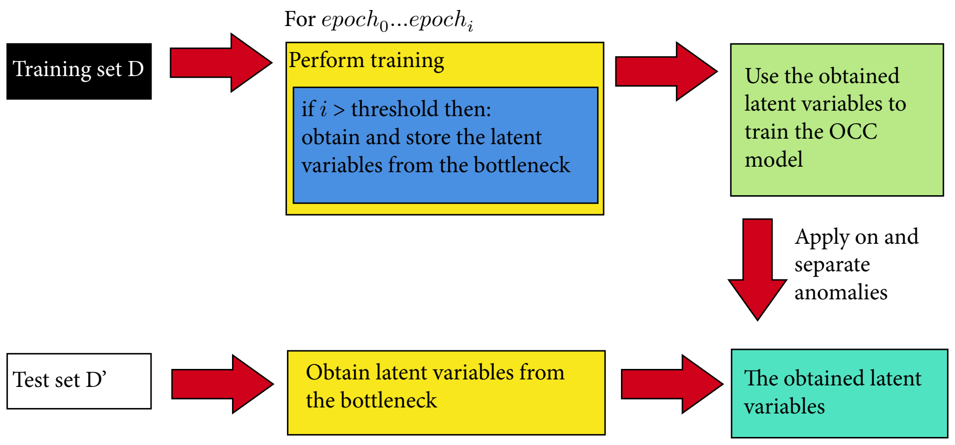

For $epoch_0 ... epoch_i$
Training set D
Perform training
if i > threshold then: obtain and store the latent variables from the bottleneck
Use the obtained latent variables to train the OCC model
Apply on and separate anomalies
Test set D'
Obtain latent variables from the bottleneck
The obtained latent variables

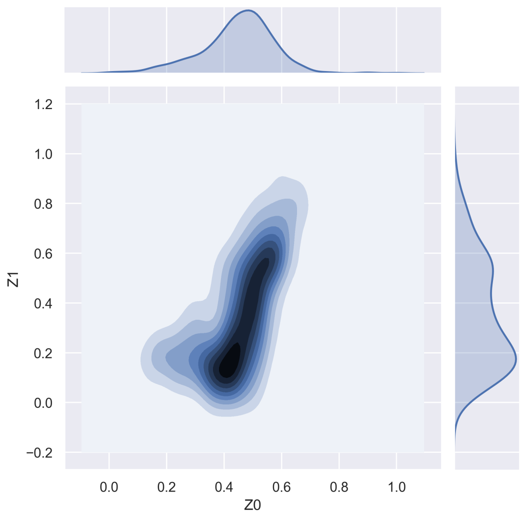
Fig. 6.3 Overall structure of the model that uses an AE for data augmentation

Once the training of the AE model is finished, the obtained latent variables from the bottleneck of the last n epochs are used to form a new low dimensional but larger training set, i.e., the training set has fewer features but more instances than the initial set. This new training set is used for the next step which is training the OCC model. In Chapter 5, different OCC models such as ISF and LOF were used to evaluate and see if the data augmentation has actually improved the performance. Next, the AE model is used to extract latent variables from the test set. It is worth mentioning that during this phase, no data augmentation is carried out as increasing the test size is unnecessary. Finally, the OCC model is applied to the latent variables that are extracted from the test set to detect anomalies.

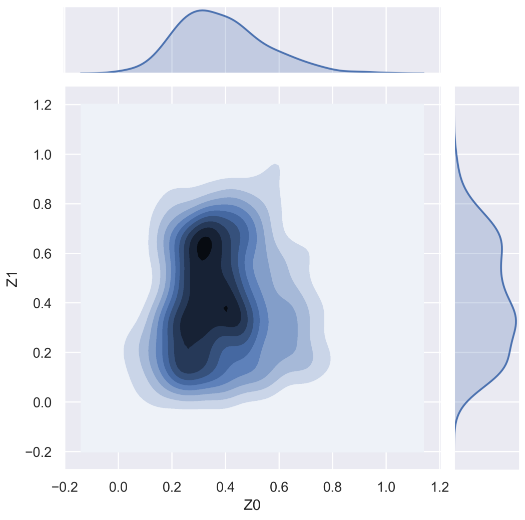
The performance of the proposed approach is reported and discussed in detail in Section 5.4 and Section 5.5, in which the data augmentation using autoencoders showed better results overall compared to the other approaches. By looking at Figure 6.4, it is possible to get more insight into how the proposed approach is overcoming the problem of not having enough samples for training OCC methods and generating an accurate classification boundary. The figure shows the density of the training set of the PenDigits dataset after being augmented by four different methods, including the one proposed in Chapter 5. A denser area means that the training set contains more uniform instances, while a sparse set includes data points that are more different in characteristics. As depicted in Figure 6.4b, using SMOTE for data augmentation produced a sparser training set compared to the other three, which shows the method did not fully take into account the intrinsic characteristics of the data instances when generating synthetic samples. By looking at Figure 6.4a and Figure 6.4c, it can be concluded using ADASYN and injecting noise for creating synthetic samples performed better than using SMOTE. Between injecting noise and using ADASYN, it is hard to point out which one performed better by merely looking at the figures. Nevertheless, by looking at Figure 6.4c, Figure 6.4a and Table 5.3, it can be deduced that injecting noise produced a training set slightly denser than using ADASYN. However, as shown in Figure 6.4c, using the proposed approach in which an AE was used for augmenting the data, generated the densest training set compared to the other three approaches. Therefore, it can be concluded that data augmentation using the AE managed to generate synthetic instances that are more uniform in characteristics, and so the OCC methods performed better at creating the classification boundary. Having a more accurate classification boundary should result in better anomaly detection performance and by looking at Table 5.3, the OCC methods performed better when the training set was augmented by the proposed approach.

6.3 Limitations and Future Work

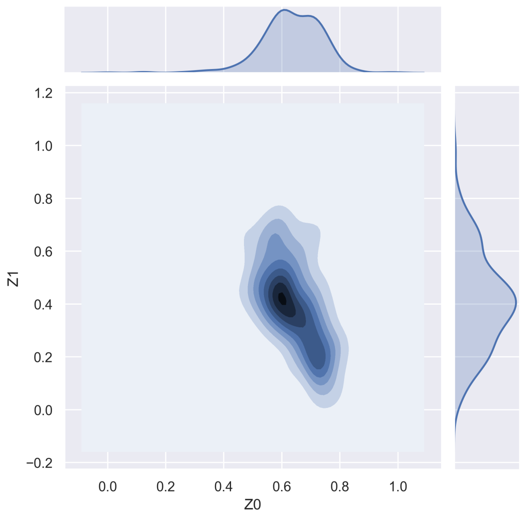
Despite the fact that the three proposed approaches showed promising results and overall managed to perform better than other state-of-the-art and some widely used



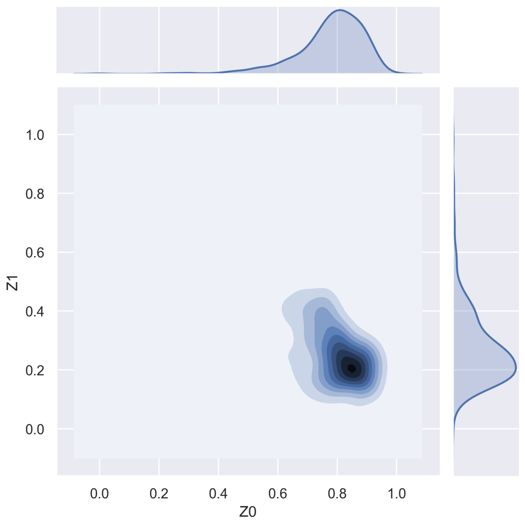
(a) DA with ADASYN



(b) DA with SMOTE



(c) DA with noise



(d) DA with AE

Fig. 6.4 Histogram plot with a kernel density estimation of the first two latent variables of the PenDigits dataset after applying various data augmentation methods on the training set. The training set contains only one type of data, i.e., inliers.

methods, they come with a few limitations that should be pointed out and possibly be addressed in the future.

As discussed, PLOF managed to improve both performance and efficiency. However, its performance heavily depends on one parameter, which is the threshold. The threshold decides which data instance should be pruned and which one needs to be flagged for later when the LOF score gets computed. In Chapter 3, the median of the δ values, which was computed by Equation 3.7, was used as the threshold. Arguably, parameter tuning is always a challenge in any machine learning problem, and an empirical approach in which different threshold values are experimented with is one of the solutions. One future work that can improve the performance of PLOF is studying and experimenting with parameter tuning approaches for generating a better threshold. Also, the pruning approach can be used in other anomaly detection methods that are density-based, which should be investigated in the future.

Using AEGR, it was possible to address the problem of detecting anomalies in high dimensional data and also made the network training phase insensitive to anomalies. One limitation of this approach is that the anomalies should have significant differences in characteristics compared to the inliers. In other words, in a dataset in which anomalies are very similar to inliers, the approach cannot perform as well as when applied on a dataset in which anomalies are spread far away from inliers, i.e., have different characteristics. Also, a good improvement that can enhance the performance of AEGR is taking into account the GS score of more than one mini-batch or data point in each epoch when carrying out gradient inversion. Moreover, as explained in Section 4.3, the obtained data from the autoencoder was pruned by using a threshold-based approach in which the threshold was based on the RE, followed by a data augmentation based on random Gaussian noise to increase the number of instances. Choosing a threshold is a challenge that can potentially have an undesirable impact on the final result. Stipulating a low threshold can cause elimination of, for instance, half of the training set that then can be replaced by synthetic data. But this would arguably interfere with the distribution of the real data and synthetic data generated by the data augmentation process. It is important to study the significance of this effect in the future. Lastly, the gradient update has an arguable effect on the learning process. Even though this study conducted an ample amount of experiments on numerous datasets and compared the results with other approaches, the effect of gradient reversal on the main task of the autoencoder, i.e., transforming high dimensional data into a low dimensional form, was not studied. The latent variable that AEGR produced, led to achieving better results from LOF but the difference between the latent variables coming from AEGR and the ones from a normal autoencoder was not measured. That is because, as discussed in Chapter 4, autoencoders perform poorly when the training set contains anomalies and noise. Therefore, the normal autoencoder needs to use use a clean training set while AEGR gets trained by a polluted training set, i.e., one that includes noise and anomalies. Despite the fact that passing two different inputs to the autoencoders can arguably make the experiment and consequently the results questionable, it is worth to investigate to what extent AEGR affects the process of transformation compared to a normal autoencoder.

As per data augmentation using autoencoders, one limitation is the need for a training set that is free of noise and anomalies. The proposed approach in Chapter 5 requires a training set that is clean of noise and anomalies, but it is also affected by the quality of data instances. To explain further, a training set containing more homogeneous inliers, i.e., examples that are very similar, can later improve the performance of the OCC algorithms because the augmented training set will be more uniform and dense.

Since PLOF showed promising results when applied to not high dimensional datasets, the use of PLOF with the other two proposed approaches should be studied. For instance, by applying AEGR, it is possible to reduce the dimensionality and then instead of applying the traditional LOF algorithm, anomalies can be separated by using PLOF. Also, it is possible to use the augmented training set using the proposed method in Chapter 5 and train PLOF with the augmented training set, which in theory should produce better results than merely applying the traditional LOF method.

Finally, none of the datasets used in this research were real-time datasets but benchmark datasets used by other research works. For instance, as Aldweesh et al. [125] stated, the NSL-KDD dataset was used by 37% of the published works that the authors reviewed. Although using a benchmark dataset makes the results comparable, this comparison can be questionable as other factors are being neglected. As Aldweesh et al. [125] mentioned, comparing deep-learning methods is a challenge because these methods can perform differently when a different pre-processing method is used, the hardware varies, the ratio between the size of the training set and the test set differs, or when the network is configured differently. It is deemed to conduct a more comprehensive experiment using the benchmark datasets and real-time datasets with more state-of-the-art methods.

6.4 Summary

This section briefly explained the approaches proposed in this work. While going through the problems that each approach tried to address, a few recent and related research works to each of the three proposed approaches were pointed out and referenced. Also, the performance of each approach was reviewed, as well as the limitations and future work.

Chapter 7 summarises on contributions of each of the three novelties proposed in this work to the field of anomaly detection.

Chapter 7

Conclusion

The present thesis looked at the challenges in the process of anomaly detection. The process of anomaly detection is found crucial in various application domains such as fraud detection, fault detection and sensor networks. The process of anomaly detection suffers from various challenges such as skewed distribution, lack of examples of anomalous cases, concept drift, and high dimensionality. Each of these challenges can hinder the process and deteriorate the performance of the machine learning model to a level that can cause catastrophic consequences. This thesis extended on the previous works from various angles. The contribution of the work can be split into three parts, each of which addressing one or more challenges of the anomaly detection process.

After investigating various anomaly detection models and reviewing the previous works, it was concluded that a density-based model called LOF, which is widely used in various application domains, can produce high performance. However, the algorithm's performance is affected by a few factors, including the dimensionality of the dataset. Also, LOF requires high computation power for separating anomalies from normal data points. There are various works in the literature that studied the mentioned problems. One of the contributions of this work was proposing PLOF. A variant of LOF that showed better performance with less computation cost. In PLOF, the dataset is pruned based on a novel density estimator, i.e., points that belong to a neighbourhood with a density higher than a pre-defined threshold are pruned, before computing the final anomaly score, i.e., the LOF value. In contrast to other similar works that reduced the computation but sacrificed the performance, the results of the experiment showed that PLOF can preserve the performance at a lower computation cost. The result also showed that PLOF not only reduced the execution time but, in some cases, even showed better performance compared to other state-of-the-art models.

This thesis made further investigations into the challenges of anomaly detection by looking into the problem of high dimensionality. Many machine learning algorithms cannot perform to their best or even norm when the dimensionality of the dataset increases, and LOF is no exception. In fact, according to the literature, LOF is based on a concept that requires a good definition of the density, which becomes opaque and hard to define as the dimensionality of the dataset grows. After studying various dimensionality reduction methods such as PCA, it was found out that a variant of neural networks known as autoencoders can produce better performance in capturing latent variables while reducing the dimensionality of the dataset. However, training autoencoders requires an anomaly free training set, which is not always available. Therefore, this thesis proposed a novel variant of autoencoders that can filter out anomalous data points during the training phase by taking into account the gamut of the gradient update caused by each data point during the training phase. In AEGR, the training set can include noise or anomalies as the training phase is insensitive towards them and the network's efforts in learning how to regenerate anomalies are countered by injecting the inverted gradient of the data points or batches that in each iteration caused the highest gradient update. Once the model is trained, the latent variables of the training set are obtained from its bottleneck and passed into a one-class classifier for separating outliers and inliers. The proposed model was tested on several publicly available datasets that are widely used in the domain of anomaly detection. The results of the experiment proved that the proposed dimensionality reduction method can lead to performance improvements.

Lastly, this research work proposed a novel method for not only reducing the dimensionality but also generating augmented data of the training set to be used by a one-class classifier. One-class classifiers can be used for anomaly detection if they are trained by normal examples. Then anything outside of the normal boundary is separated from the test set. The proposed approach was based on the assumption that improving the class boundary during the training phase can enhance the performance of OCCs as the model gets a better understanding of the characteristics of the normal data points. This thesis extended the past works by using an autoencoder for augmenting the training set while collecting important latent variables and reducing dimensionality. Expanding the training set means the OCC model can define a better class boundary; however, the augmented data points must show similar characteristics of the normal instances with minor variations. The autoencoder used in the proposed model begins augmentation when the network is almost trained and can reproduce the input from its bottleneck with a low RE. According to the results of the experiment, the proposed model outperformed other well-known methods in the literature. Therefore, it is conceivable that data augmentation using the proposed approach can enhance the class boundary definition and subsequently improving the performance of the OCC used for detecting anomalies.

References

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Comput. Surv.*, vol. 41, pp. 15:1—-15:58, jul 2009.

[2] E. R. Faria, I. Gonçalves, A. de Carvalho, and J. Gama, "Novelty detection in data streams," *Artificial Intelligence Review*, vol. 45, no. 2, pp. 235–269, 2016.

[3] J. Ha, S. Seok, and J.-S. Lee, "Robust outlier detection using the instability factor," *Knowledge-Based Systems*, vol. 63, pp. 15–23, 2014.

[4] V. J. Hodge and J. Austin, "A Survey of Outlier Detection Methodologies," *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, 2004.

[5] Z. Niu, S. Shi, J. Sun, and X. He, "A survey of outlier detection methodologies and their applications," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7002 LNAI, no. PART 1, pp. 380–387, 2011.

[6] X. Song, M. Wu, C. Jermaine, and S. Ranka, "Conditional Anomaly Detection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, pp. 631–645, may 2007.

[7] S. S. Khan and M. G. Madden, "A Survey of Recent Trends in One Class Classification," in *Artificial Intelligence and Cognitive Science: 20th Irish Conference, AICS 2009, Dublin, Ireland, August 19-21, 2009, Revised Selected Papers* (L. Coyle and J. Freyne, eds.), pp. 188–197, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[8] K.-L. Li, H.-K. Huang, S.-F. Tian, and W. Xu, "Improving one-class SVM for anomaly detection," in *International Conference on Machine Learning and Cybernetics*, vol. 5, pp. 3077–3081, 2003.

[9] M. Zhang, B. Xu, and D. Wang, "An anomaly detection model for network intrusions using one-class SVM and scaling strategy," *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, vol. 163, pp. 267–278, 2016.

[10] S. Patnaik, S. Subudhi, and S. Panigrahi, "Quarter-Sphere Support Vector Machine for Fraud Detection in Mobile Telecommunication Networks," *Procedia Computer Science*, vol. 48, pp. 353–359, 2015.

[11] V. Hautamäki, I. Kärkkäinen, and P. Fränti, "Outlier detection using k-nearest neighbour graph," in *Proceedings - International Conference on Pattern Recognition*, vol. 3, pp. 430–433, 2004.

[12] B. Tang and H. He, “A local density-based approach for outlier detection,” *Neurocomputing*, vol. 241, pp. 171–180, 2017.

[13] W. Jin, A. K. H. Tung, and J. Han, “Mining top-n local outliers in large databases,” in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 293–298, 2001.

[14] J. Tang, Z. Chen, A. W.-c. Fu, and D. W. Cheung, “Enhancing Effectiveness of Outlier Detections for Low Density Patterns,” in *Advances in Knowledge Discovery and Data Mining: 6th Pacific-Asia Conference, PAKDD 2002 Taipei, Taiwan, May 6–8, 2002 Proceedings* (M.-S. Chen, P. S. Yu, and B. Liu, eds.), pp. 535–548, Berlin, Heidelberg: Springer Berlin Heidelberg, 2002.

[15] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “LOF: Identifying Density-based Local Outliers,” *SIGMOD Rec.*, vol. 29, pp. 93–104, may 2000.

[16] F. Pernkopf and D. Bouchaffra, “Genetic-based EM algorithm for learning Gaussian mixture models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1344–1348, 2005.

[17] H. Farvaresh and M. M. Sepehri, “A data mining framework for detecting subscription fraud in telecommunication,” *Engineering Applications of Artificial Intelligence*, vol. 24, no. 1, pp. 182–194, 2011.

[18] X. Zhu and A. B. Goldberg, “Introduction to semi-supervised learning,” *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.

[19] A. Daneshpazhouh and A. Sami, “Semi-Supervised Outlier Detection with Only Positive and Unlabeled Data Based on Fuzzy Clustering,” *International Journal on Artificial Intelligence Tools*, vol. 24, no. 03, p. 1550003, 2015.

[20] K.-I. Kim, T. Kim, N.-W. Cho, and M. Kim, “Toll Fraud Detection of VoIP Service Networks in Ubiquitous Computing Environments,” *International Journal of Distributed Sensor Networks*, vol. 2015, 2015.

[21] J. Li, K.-Y. Huang, J. Jin, and J. Shi, “A survey on statistical methods for health care fraud detection,” *Health Care Management Science*, vol. 11, no. 3, pp. 275–287, 2008.

[22] M. Ahmed, A. Naser Mahmood, and J. Hu, “A survey of network anomaly detection techniques,” *Journal of Network and Computer Applications*, vol. 60, pp. 19 – 31, 2016.

[23] V. L. Cao, M. Nicolau, and J. McDermott, “Learning Neural Representations for Network Anomaly Detection,” *IEEE Transactions on Cybernetics*, vol. 49, no. 8, pp. 3074–3087, 2019.

[24] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[25] D. Pérez, S. Alonso, A. Morán, M. A. Prada, J. J. Fuertes, and M. Domínguez, "Comparison of network intrusion detection performance using feature representation," in *Engineering Applications of Neural Networks* (J. Macintyre, L. Iliadis, I. Maglogiannis, and C. Jayne, eds.), (Cham), pp. 463–475, Springer International Publishing, 2019.

[26] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau, "Autoencoder-based network anomaly detection," in *2018 Wireless Telecommunications Symposium (WTS)*, pp. 1–5, April 2018.

[27] J. Castellini, V. Poggioni, and G. Sorbi, "Fake twitter followers detection by denoising autoencoder," in *Proceedings of the International Conference on Web Intelligence*, WI '17, (New York, NY, USA), p. 195–202, Association for Computing Machinery, 2017.

[28] H. A. Dau, V. Ciesielski, and A. Song, "Anomaly detection using replicator neural networks trained on examples of one class," in *Asia-Pacific Conference on Simulated Evolution and Learning*, pp. 311–322, Springer, 2014.

[29] S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon, "Ganomaly: Semi-supervised anomaly detection via adversarial training," in *Computer Vision – ACCV 2018* (C. V. Jawahar, H. Li, G. Mori, and K. Schindler, eds.), (Cham), pp. 622–637, Springer International Publishing, 2019.

[30] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 665–674, ACM, 2017.

[31] S. Boyd, N. Parikh, and E. Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[32] T. C. Bui, V. L. Cao, M. Hoang, and Q. U. Nguyen, "A clustering-based shrink autoencoder for detecting anomalies in intrusion detection systems," in *2019 11th International Conference on Knowledge and Systems Engineering (KSE)*, pp. 1–5, 2019.

[33] V. Q. Nguyen, V. H. Nguyen, N.-A. Le-Khac, and V. L. Cao, "Clustering-based deep autoencoders for network anomaly detection," in *Future Data and Security Engineering* (T. K. Dang, J. Küng, M. Takizawa, and T. M. Chung, eds.), (Cham), pp. 290–303, Springer International Publishing, 2020.

[34] M. Sabokrou, M. Fayyaz, M. Fathy, Z. Moayed, and R. Klette, "Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes," *Computer Vision and Image Understanding*, vol. 172, pp. 88 – 97, 2018.

[35] M. Sabokrou, M. Fayyaz, M. Fathy, and R. Klette, "Deep-cascade: Cascading 3d deep neural networks for fast anomaly detection and localization in crowded scenes," *IEEE Transactions on Image Processing*, vol. 26, pp. 1992–2004, April 2017.

[36] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," 2014.

[37] P. Oza and V. M. Patel, "One-class convolutional neural network," *IEEE Signal Processing Letters*, vol. 26, no. 2, pp. 277–281, 2019.

[38] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, and L. S. Davis, "Learning temporal regularity in video sequences," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 733–742, 2016.

[39] Y. Zhang, P. Peng, C. Liu, and H. Zhang, "Anomaly detection for industry product quality inspection based on gaussian restricted boltzmann machine," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pp. 1–6, 2019.

[40] U. Fiore, F. Palmieri, A. Castiglione, and A. De Santis, "Network anomaly detection with the restricted boltzmann machine," *Neurocomputing*, vol. 122, pp. 13–23, 2013.

[41] J. Goh, S. Adepu, M. Tan, and Z. S. Lee, "Anomaly detection in cyber physical systems using recurrent neural networks," in *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)*, pp. 140–145, IEEE, 2017.

[42] M. Z. Alom, V. Bontupalli, and T. M. Taha, "Intrusion detection using deep belief networks," in *2015 National Aerospace and Electronics Conference (NAECON)*, pp. 339–344, IEEE, 2015.

[43] Y. Yang, K. Zheng, C. Wu, X. Niu, and Y. Yang, "Building an effective intrusion detection system using the modified density peak clustering algorithm and deep belief networks," *Applied Sciences*, vol. 9, no. 2, p. 238, 2019.

[44] C. S. Hilas and J. N. Sahalos, "User profiling for fraud detection in telecommunication networks," in *In: 5th Int. Conf. technology and automation*, pp. 382–387, 2005.

[45] L.-A. Gottlieb, A. Kontorovich, and R. Krauthgamer, "Adaptive metric dimensionality reduction," *Theoretical Computer Science*, vol. 620, pp. 105–118, 2016.

[46] H. Hotelling, "Analysis of a complex of statistical variables into principal components.," *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[47] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[48] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, pp. 391–407, 1990.

[49] D. DeMers and G. W. Cottrell, "Non-linear dimensionality reduction," in *Advances in neural information processing systems*, pp. 580–587, 1993.

[50] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, pp. 232–242, 2016.

[51] Y. Yan, M. Chen, M.-L. Shyu, and S.-C. Chen, “Deep Learning for Imbalanced Multimedia Data Classification,” in *Proceedings - 2015 IEEE International Symposium on Multimedia, ISM 2015*, pp. 483–488, 2015.

[52] S. Al-Stouhi and C. K. Reddy, “Transfer learning for class imbalance problems with inadequate data,” *Knowledge and Information Systems*, vol. 48, no. 1, pp. 201–228, 2016.

[53] N. V. Chawla, N. Japkowicz, and A. Kotcz, “Editorial: Special Issue on Learning from Imbalanced Data Sets,” *SIGKDD Explor. Newsl.*, vol. 6, pp. 1–6, jun 2004.

[54] A. H. Elmi, S. Ibrahim, and R. Sallehuddin, “Detecting SIM Box Fraud Using Neural Network,” in *IT Convergence and Security 2012* (J. K. Kim and K.-Y. Chung, eds.), pp. 575–582, Dordrecht: Springer Netherlands, 2013.

[55] R. Sallehuddin, S. Ibrahim, A. M. Zain, and A. H. Elmi, “Detecting SIM box fraud by using support vector machine and artificial neural network,” *Jurnal Teknologi*, vol. 74, no. 1, pp. 137–149, 2015.

[56] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A survey on concept drift adaptation,” *ACM Computing Surveys*, vol. 46, pp. 44:1—-44:37, mar 2014.

[57] A. Tsymbal, “The Problem of Concept Drift: Definitions and Related Work,” tech. rep., 2004.

[58] J. L. Lobo, J. Del Ser, M. N. Bilbao, I. Laña, and S. Salcedo-Sanz, “A probabilistic sample matchmaking strategy for imbalanced data streams with concept drift,” *Studies in Computational Intelligence*, vol. 678, pp. 237–246, 2017.

[59] M. Behdad, L. Barone, M. Bennamoun, and T. French, “Nature-Inspired Techniques in the Context of Fraud Detection,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, pp. 1273–1290, nov 2012.

[60] C. Phua, V. C. S. Lee, K. Smith-Miles, and R. W. Gayler, “A Comprehensive Survey of Data Mining-based Fraud Detection Research,” *CoRR*, vol. abs/1009.6, 2010.

[61] P. Gogoi, D. K. Bhattacharyya, B. Borah, and J. K. Kalita, “A Survey of Outlier Detection Methods in Network Anomaly Identification,” *The Computer Journal*, 2011.

[62] A. Bănărescu, “Detecting and Preventing Fraud with Data Analytics,” *Procedia Economics and Finance*, vol. 32, pp. 1827–1836, 2015.

[63] T. Kieu, B. Yang, and C. S. Jensen, “Outlier detection for multidimensional time series using deep neural networks,” in *2018 19th IEEE International Conference on Mobile Data Management (MDM)*, pp. 125–134, June 2018.

[64] B. van Stein, M. van Leeuwen, and T. Bäck, “Local subspace-based outlier detection using global neighbourhoods,” in *2016 IEEE International Conference on Big Data (Big Data)*, pp. 1136–1142, 2016.

[65] S. Su, L. Xiao, Z. Zhang, F. Gu, L. Ruan, S. Li, Z. He, Z. Huo, B. Yan, H. Wang, and Others, "N2DLOF: A New Local Density-Based Outlier Detection Approach for Scattered Data," in *High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2017 IEEE 19th International Conference on*, pp. 458–465, IEEE, 2017.

[66] M. M. Breuniq, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," *SIGMOD Record (ACM Special Interest Group on Management of Data)*, vol. 29, no. 2, pp. 93–104, 2000.

[67] L. Zhang, J. Lin, and R. Karim, "Adaptive kernel density-based anomaly detection for nonlinear systems," *Knowledge-Based Systems*, vol. 139, pp. 50–63, 2018.

[68] H. Ding, K. Ding, J. Zhang, Y. Wang, L. Gao, Y. Li, F. Chen, Z. Shao, and W. Lai, "Local outlier factor-based fault detection and evaluation of photovoltaic system," *Solar Energy*, vol. 164, pp. 139–148, 2018.

[69] W. Jin, A. K. H. Tung, J. Han, and W. Wang, "Ranking outliers using symmetric neighborhood relationship," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 577–593, Springer, 2006.

[70] M. Goldstein, "FastLOF: An Expectation-Maximization based Local Outlier detection algorithm," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pp. 2282–2285, nov 2012.

[71] A. Sinha and P. K. Jana, "Efficient Algorithms for Local Density Based Anomaly Detection," in *Distributed Computing and Internet Technology* (A. Negi, R. Bhatnagar, and L. Parida, eds.), (Cham), pp. 336–342, Springer International Publishing, 2018.

[72] A. L. M. Chiu and A. W.-c. Fu, "Enhancements on local outlier detection," in *Seventh International Database Engineering and Applications Symposium, 2003. Proceedings.*, pp. 298–307, jul 2003.

[73] R. Pamula, J. K. Deka, and S. Nandi, "Pruning based method for outlier detection," in *Proceedings - 2012 3rd International Conference on Emerging Applications of Information Technology, EAIT 2012*, pp. 210–213, 2012.

[74] R. Pamula, J. K. Deka, and S. Nandi, "An Outlier Detection Method Based on Clustering," in *2011 Second International Conference on Emerging Applications of Information Technology*, pp. 253–256, feb 2011.

[75] H. Rizk, S. Elgokhy, and A. Sarhan, "A hybrid outlier detection algorithm based on partitioning clustering and density measures," in *2015 Tenth International Conference on Computer Engineering Systems (ICCES)*, pp. 175–181, dec 2015.

[76] S. Poddar and B. K. Patra, "Reduction in Execution Cost of k-Nearest Neighbor Based Outlier Detection Method," in *Mathematics and Computing* (D. Ghosh, D. Giri, R. N. Mohapatra, E. Savas, K. Sakurai, and L. P. Singh, eds.), (Singapore), pp. 53–60, Springer Singapore, 2018.

[77] Z. Xiong, R. Chen, Y. Zhang, and X. Zhang, "Multi-density dbscan algorithm based on density levels partitioning," *JOURNAL OF INFORMATION &COMPUTATIONAL SCIENCE*, vol. 9, no. 10, pp. 2739–2749, 2012.

[78] D. Dheeru and E. Karra Taniskidou, "{UCI} Machine Learning Repository," 2017.

[79] G. O. Campos, A. Zimek, J. Sander, R. J. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle, "On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study," *Data Mining and Knowledge Discovery*, pp. 1–37, 2015.

[80] V. L. Cao, M. Nicolau, and J. McDermott, "A Hybrid Autoencoder and Density Estimation Model for Anomaly Detection," in *Parallel Problem Solving from Nature – PPSN XIV* (J. Handl, E. Hart, P. R. Lewis, M. López-Ibáñez, G. Ochoa, and B. Paechter, eds.), (Cham), pp. 717–726, Springer International Publishing, 2016.

[81] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A survey on deep learning for big data," *Information Fusion*, vol. 42, pp. 146–157, 2018.

[82] Y. Ma, P. Zhang, Y. Cao, and L. Guo, "Parallel auto-encoder for efficient outlier detection," in *Big Data, 2013 IEEE International Conference on*, pp. 15–17, IEEE, 2013.

[83] J. Sun, X. Wang, N. Xiong, and J. Shao, "Learning sparse representation with variational auto-encoder for anomaly detection," *IEEE Access*, vol. 6, pp. 33353–33361, 2018.

[84] Z. Sun and H. Sun, "Stacked Denoising Autoencoder With Density-Grid Based Clustering Method for Detecting Outlier of Wind Turbine Components," *IEEE Access*, vol. 7, pp. 13078–13091, 2019.

[85] D. Charte, F. Charte, S. García, M. J. del Jesus, and F. Herrera, "A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines," *Information Fusion*, vol. 44, pp. 78–96, 2018.

[86] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, p. 4, ACM, 2014.

[87] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 3854–3861, IEEE, 2017.

[88] Y. Qi, Y. Wang, X. Zheng, and Z. Wu, "Robust feature learning by stacked autoencoder with maximum correntropy criterion," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6716–6720, IEEE, 2014.

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

[90] M. Schreyer, T. Sattarov, D. Borth, A. Dengel, and B. Reimer, "Detection of anomalies in large scale accounting data using deep autoencoder networks," *arXiv preprint arXiv:1709.05254*, 2017.

[91] J. Chen, S. Sathe, C. Aggarwal, and D. Turaga, "Outlier detection with autoencoder ensembles," in *Proceedings of the 2017 SIAM International Conference on Data Mining*, pp. 90–98, SIAM, 2017.

[92] A. Miglani and N. Kumar, "Deep learning models for traffic flow prediction in autonomous vehicles: A review, solutions, and challenges," *Vehicular Communications*, vol. 20, p. 100184, 2019.

[93] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," *arXiv preprint arXiv:1409.7495*, 2014.

[94] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning," *Pattern Recognition*, vol. 58, pp. 121 – 134, 2016.

[95] L. Bottou, "Stochastic Gradient Learning in Neural Networks," *Proceedings of Neuro-Nımes*, vol. 91, no. 8, p. 12, 1991.

[96] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers & Security*, vol. 45, pp. 100–123, 2014.

[97] S. Garg, K. Kaur, N. Kumar, and J. J. P. C. Rodrigues, "Hybrid Deep-Learning-Based Anomaly Detection Scheme for Suspicious Flow Detection in SDN: A Social Multimedia Perspective," *IEEE Transactions on Multimedia*, vol. 21, pp. 566–578, mar 2019.

[98] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6, IEEE, 2009.

[99] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, pp. 1–6, nov 2015.

[100] J. Lian, W. Jia, M. Zareapoor, Y. Zheng, R. Luo, D. K. Jain, and N. Kumar, "Deep-Learning-Based Small Surface Defect Detection via an Exaggerated Local Variation-Based Generative Adversarial Network," *IEEE Transactions on Industrial Informatics*, vol. 16, pp. 1343–1351, feb 2020.

[101] F. Provost and T. Fawcett, "Analysis and Visualization of Classifier Performance: Comparison Under Imprecise Class and Cost Distributions," in *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, KDD'97, pp. 43–48, AAAI Press, 1997.

[102] D. S. Kerby, "The simple difference formula: An approach to teaching nonparametric correlation," *Comprehensive Psychology*, vol. 3, pp. 11—-IT, 2014.

[103] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

[104] M. A. A. Ghaffar, A. McKinstry, T. Maul, and T. T. Vu, “Data Augmentation Approaches for Satellite Image Super-Resolution,” *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, pp. 47–54, 2019.

[105] F. J. Moreno-Barea, J. M. Jerez, and L. Franco, “Improving classification accuracy using data augmentation on small data sets,” *Expert Systems with Applications*, vol. 161, p. 113696, 2020.

[106] G. Douzas and F. Bacao, “Effective data generation for imbalanced learning using conditional generative adversarial networks,” *Expert Systems with Applications*, vol. 91, pp. 464–471, 2018.

[107] P. Domingos, “MetaCost: A General Method for Making Classifiers Cost-sensitive,” in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’99, (New York, NY, USA), pp. 155–164, ACM, 1999.

[108] S. He, H., Bai, Y., Garcia, E., & Li, “ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In IEEE International Joint Conference on Neural Networks, 2008,” *IJCNN 2008.(IEEE World Congress on Computational Intelligence) (pp. 1322– 1328)*, no. 3, pp. 1322– 1328, 2008.

[109] J.-H. Oh, J. Y. Hong, and J.-G. Baek, “Oversampling method using outlier detectable generative adversarial network,” *Expert Systems with Applications*, vol. 133, pp. 1–8, 2019.

[110] S. K. Lim, Y. Loo, N. Tran, N. Cheung, G. Roig, and Y. Elovici, “DOPING: Generative Data Augmentation for Unsupervised Anomaly Detection with GAN,” in *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 1122–1127, nov 2018.

[111] S. Su, L. Xiao, Z. Zhang, F. Gu, L. Ruan, S. Li, Z. He, Z. Huo, B. Yan, H. Wang, and S. Liu, “N2DLOF: A New Local Density-Based Outlier Detection Approach for Scattered Data,” in *2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 458–465, IEEE, dec 2017.

[112] P. Boniol, M. Linardi, F. Roncallo, and T. Palpanas, “Automated anomaly detection in large sequences,” in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pp. 1834–1837, 2020.

[113] L. Xu, Y.-R. Yeh, Y.-J. Lee, and J. Li, “A hierarchical framework using approximated local outlier factor for efficient anomaly detection,” *Procedia Computer Science*, vol. 19, pp. 1174 – 1181, 2013. The 4th International Conference on Ambient Systems, Networks and Technologies (ANT 2013), the 3rd International Conference on Sustainable Energy Information Technology (SEIT-2013).

[114] X. Deng and L. Wang, “Modified kernel principal component analysis using double-weighted local outlier factor and its application to nonlinear process monitoring,” *ISA Transactions*, vol. 72, pp. 218 – 228, 2018.

[115] Y. Ma, H. Shi, H. Ma, and M. Wang, "Dynamic process monitoring using adaptive local outlier factor," *Chemometrics and Intelligent Laboratory Systems*, vol. 127, pp. 89 – 101, 2013.

[116] Y. Karadayi, M. Aydin, and A. Ögˇrenci, "A hybrid deep learning framework for unsupervised anomaly detection in multivariate spatio-temporal data," *Applied Sciences (Switzerland)*, vol. 10, no. 15, 2020. cited By 0.

[117] D. O'Neill, A. Lensen, B. Xue, and M. Zhang, "Particle swarm optimisation for feature selection and weighting in high-dimensional clustering," in *2018 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, 2018.

[118] P. Lv, Y. Yu, Y. Fan, X. Tang, and X. Tong, "Layer-constrained variational autoencoding kernel density estimation model for anomaly detection," *Knowledge-Based Systems*, vol. 196, p. 105753, 2020.

[119] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, pp. 1–18, 2015.

[120] H. Zenati, M. Romain, C. Foo, B. Lecouat, and V. Chandrasekhar, "Adversarially learned anomaly detection," in *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 727–736, Nov 2018.

[121] L. Vu, V. L. Cao, Q. U. Nguyen, D. N. Nguyen, D. T. Hoang, and E. Dutkiewicz, "Learning latent representation for iot anomaly detection," *IEEE Transactions on Cybernetics*, pp. 1–14, 2020.

[122] A. Singh, R. Pokharel, and J. Principe, "The c-loss function for pattern classification," *Pattern Recognition*, vol. 47, no. 1, pp. 441 – 453, 2014.

[123] S. S. Khan and M. G. Madden, "One-class classification: taxonomy of study and review of techniques," *The Knowledge Engineering Review*, vol. 29, no. 3, p. 345–374, 2014.

[124] M. Al Olaimat, D. Lee, Y. Kim, J. Kim, and J. Kim, "A learning-based data augmentation for network anomaly detection," in *2020 29th International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–10, 2020.

[125] A. Aldweesh, A. Derhab, and A. Z. Emam, "Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues," *Knowledge-Based Systems*, vol. 189, p. 105124, 2020.

[126] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[127] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, and J. Vanderplas, ""scikit-learn: Machine learning in python." journal of machine learning research 12 (oct), 2825–2830," 2011.

[128] N. Ketkar, "Introduction to keras," in *Deep learning with Python*, pp. 97–111, Springer, 2017.

Appendix A

Frameworks

This chapter lists all the frameworks that are used in this thesis.

A.1 Weka

Waikato Environment for Knowledge Analysis (Weka)[1] [126] is an open-source data mining software written in Java by the University of Waikato. It offers various machine learning algorithms and visualisation tools.

A.2 Scikit-learn

Scikit-learn (also known as sklearn)[2] [127], is a free open-source software written in Python that offers various machine learning methods, analytics and visulalization tools. It uses different libraries such as NumPy, matplotlit and SciPy.

A.3 PyTorch

PyTorch[3] [128] is a free open-source library with a Python interface (also comes with another available interface in C++) based on Torch library which is mainly developed by Facebook Inc.

[1]https://www.cs.waikato.ac.nz/ml/weka/index.html
[2]https://scikit-learn.org/
[3]https://pytorch.org/

A.4 Keras

Keras[4] [128] is a free open-source library with a Python interface that is based on TensorFlow. Keras provides various APIs for building deep neural networks such as autoencoders.

A.5 Sample Code of the Proposed Models

A sample code for each proposed model is freely available on GitHub[5].

[4]https://keras.io

[5]https://github.com/kasrababaei/anomaly-detection-models