Isolation Forest Based Anomaly Detection with Learning to Hash

By

Haolong Xiang

A thesis submitted to Macquarie University

for the degree of Doctor of Philosophy

Faculty of engineering and science

School of computing

April 2024

MACQUARIE
University
SYDNEY·AUSTRALIA

This thesis is submitted to Macquarie University in fulfilment of the requirement for the Degree of Doctor of Philosophy.

The work presented in this thesis is, to the best of my knowledge and belief, original except as acknowledged in the text. I hereby declare that I have not submitted this material, either in full or in part, for a degree at this or any other institution.

Haolong Xiang

Declaration of contributions to publications

I declare that this chapter outlines my contributions to the publications that have arisen from the research presented in this Ph.D. thesis. The publications and contributions are as follows:

1. **OptIForest: Optimal Isolation Forest for Anomaly Detection**

   Authorship: I am the primary author of this publication, responsible for conceiving the research idea, designing experiments, collecting and analyzing data, and writing the manuscript.

2. **Deep Optimal Isolation Forest with Genetic Algorithm for Anomaly Detection**

   Authorship: I am the primary author of this publication, contributing significantly to the conceptualization, methodology, data collection, analysis, and manuscript preparation.

3. **DeepiForest: A Deep Anomaly Detection Framework with Hashing Based Isolation Forest**

   Authorship: I am the primary author of this publication, playing a substantial role in formulating the research questions, designing experiments, conducting experiments, analyzing data, and drafting the manuscript.

4. **Edge computing empowered anomaly detection framework with dynamic insertion and deletion schemes on data streams**

Authorship: I am the primary author of this publication, playing a substantial role in formulating the research questions, designing the method, conducting the experiments, data analysis, and manuscript revision.

5. **Isolation forest based anomaly detection framework on non-IID data**

   Authorship: I am the primary author of this publication, responsible for conceiving the research idea, designing experiments, collecting and analyzing data, and writing the manuscript.

6. **OPHiForest: order preserving hashing based isolation forest for robust and scalable anomaly detection**

   Authorship: I am the primary author of this publication, responsible for conceiving the research idea, designing experiments, collecting and analyzing data, and writing the manuscript.

In all the aforementioned publications, I have complied with the ethical guidelines and standards set by the respective journals or conferences. The contributions made by other co-authors have been acknowledged appropriately within the publications.

I affirm that these publications are directly related to the research presented in this Ph.D. thesis and have been crucial in advancing the knowledge and understanding of the field of study.

Acknowledgements

Upon the completion of the dissertation, I would like to give my heartfelt thanks to all those who have contributed to the completion of this dissertation. This journey has been both challenging and rewarding, and I could not have reached this milestone without the support and assistance of many individuals and organizations.

First of all, I am especially grateful to my supervisors, Dr. Xuyun Zhang and Prof. Mark Dras, who provided me with informative instructions and great patience in the course of writing and revision of the dissertation. Without their participation and cooperation, I could not have finished this dissertation. Dr. Xuyun Zhang's keen and meticulous academic observation acquainted me with academic norms, broadened my horizons of literary theories, and enlightened me in my future studies. Prof. Mark Dras is always supportive of my research problems and professional to help me with any academically relevant discussion. My deepest gratitude goes to Dr. Xuyun Zhang, who has provided me with the opportunity for Ph.D. study, and helped me with my scholarship application, Guaranteeing my study and life without worries.

I have been honored to have worked with so many outstanding researchers during my Ph.D. period, including Prof. Amin Beheshti, Prof. Wanchun Dou, Prof. Kotagiri Ramamohanarao, Prof. Xiaolong Xu, Dr. Meng Liu, Dr. Hongsheng Hu, and Dr. Jiayu Wang. Their expertise and diverse perspectives have enriched the content and methodology of my research.

I am grateful to my colleagues at Macquarie University and fellow researchers at our data science lab, who provided an intelligent collision environment and offered valuable

insights during group discussions and technical seminars. Your friendship and support have played a pivotal role in my academic journey.

I extend my appreciation to the staff and administrators at Macquarie University for their administrative assistance, logistical support, and access to resources that have facilitated my research efforts. I am also grateful that I was financially supported by the International Macquarie University Research Excellence Scholarship, which enabled me to pursue my doctoral studies.

I would like to thank all my friends in Sydney, in China, and even in other countries. My friends have been the pillars of strength, offering not only encouragement but also understanding when I had to prioritise my studies over social activities. I believe that we can continue to work together after graduation and pursue a bright academic future.

Finally, I dedicate this dissertation to my parents and family members, whose sacrifices and unending love have been the bedrock of my perseverance. The unwavering trust of my family, even when I may doubted myself, has enabled me to persevere throughout my doctoral journey and complete successful academic research projects.

List of Publications

Related to the thesis:

1. **Xiang, H.**, Zhang, X., Hu, H., Qi, L., Dou, W., Dras, M., Beheshti, A. and Xu, X., "OptIForest: Optimal Isolation Forest for Anomaly Detection", *International Joint Conference on Artificial Intelligence (IJCAI)*, 2023. [Core Ranking A^*]

   Chapter 4 is based on the above publication and solves the problem of finding the optimal isolation tree structure for accurate anomaly detection.

2. **Xiang, H.**, Zhang, X., Xu, X., Dras, M., Beheshti, A. and Dou, W, "Deep Optimal Isolation Forest with Genetic Algorithm for Anomaly Detection", *IEEE International Conference on Data Mining (ICDM)*, 2023. [Core Ranking A^*]

   Chapter 6 is based on the above publication. The main contribution of this chapter is to analyse the search space of isolation trees under specific data instances and address the challenges in finding optimal isolation forest.

3. **Xiang, H.**, Hu, H. and Zhang, X., "DeepiForest: A Deep Anomaly Detection Framework with Hashing Based Isolation Forest", *IEEE International Conference on Data Mining (ICDM)*, pp. 1251-1256, 2022. [Core Ranking A^*]

   Chapter 5 is based on the above publication and it is the first work that extends isolation forest into deep paradigm for anomaly detection.

4. **Xiang, H.**, Salcic, Z., Dou, W., Xu, X., Qi, L. and Zhang, X., "OPHiForest: order preserving hashing based isolation forest for robust and scalable anomaly

detection", *ACM international conference on information & knowledge management (CIKM)*, pp. 1655-1664, 2020. [Core Ranking *A*]

Chapter 3 is based on the above publication. This chapter introduces the learning to hash for anomaly detection and changes the data-independent hash scheme to data-dependent hash scheme with better accuracy for anomaly detection.

5. **Xiang, H.** and Zhang, X., "Edge computing empowered anomaly detection framework with dynamic insertion and deletion schemes on data streams", *World Wide Web Journal*, 25(5), pp.2163-2183, 2022. [JCR Q1]

6. **Xiang, H.**, Wang, J., Ramamohanarao, K., Salcic, Z., Dou, W. and Zhang, X., "Isolation forest based anomaly detection framework on non-IID data", *IEEE Intelligent Systems*, 36(3), pp.31-40, 2021. [JCR Q1]

   Publication 5 and 6 provide much background and guidance for Chapter 1 and 2, and these publications broaden the content of the distortion.

Abstract

Due to rapid technological advancements, Artificial Intelligence (AI) has become an integral part of our daily lives, impacting various industries such as healthcare, finance, and transportation. In AI industries, anomaly detection plays a crucial role as it is vital in preventing significant losses in various applications such as cybersecurity intrusion detection, financial risk detection, and human health monitoring. With the advancements in AI techniques, anomaly detection has become an essential tool to identify rare items that deviate from the most normal items. A variety of unsupervised anomaly detection methods, ranging from shallow to deep, have been proposed. Notably, a category based on the isolation forest mechanism stands out for its simplicity, effectiveness, and efficiency, e.g., iForest is frequently employed as a state-of-the-art detector in real applications. However, the existing isolation forest based approaches are data-independent and fail to effectively learn the information of data instances, which significantly impairs the effectiveness and robustness of anomaly detection. In this dissertation, we aim to solve the above challenges at both shallow and deep levels through the implementation of appropriate learning techniques.

Specifically, we analyse the limitations of traditional isolation forest based methods in learning data information to build isolation trees. To solve the problem, we adopt the learning to hash schemes to isolation forest and extend the order preserving hashing (OPH) for anomaly detection by designing a two-step learning scheme. Our analysis further establishes the critical role played by isolation tree structures in determining the overall performance of these detection methods. Herein, we investigate the

optimization problem of isolation tree structure concerning the branching factor and establish a theory on the optimal isolation forest, simultaneously designing a practical optimal isolation forest. Thirdly, we have the observation that deep anomaly detection (DAD) methods relying on deep neural networks (DNNs) are restricted by the intrinsic shortcomings of deep learning, such as an overabundance of parameters and fixed training layers. Inspired by the deep forest, we aim to solve the above learning problems at a deep level without relying on DNNs. We deepen the isolation forest into the cascaded forest to improve the detection performance of anomaly detection. Next, the isolation tree structure is optimised using a powerful deep model based on the genetic algorithm. Extensive experiments on both synthetic datasets and a series of real-world datasets demonstrate that our approaches can achieve better detection accuracy and robustness than the state-of-the-arts.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Research Background

Artificial Intelligence (AI) [166] is a field of computer science that aims to create intelligent machines capable of simulating human-like cognitive functions. The concept of AI dates back to the mid-20th century [216]. Over the years, it has evolved from theoretical ideas to practical applications that are currently an integral part of modern society, involving smart governance, traffic management and transportation, healthcare and public health, etc [47]. A crucial technique in AI called machine learning (ML) facilitates the development of algorithms and statistical models that enable computers to learn from and make predictions or decisions based on data without being explicitly programmed for each specific task [260].

The core principle of ML is to enable computer systems to automatically learn

and improve through experience and find the fundamental statistical-computational-information-theoretic laws that govern all learning systems, including computers, humans, and organisations [79]. This is achieved by learning good representations to build ML models, which can iteratively train datasets, tune parameters, and refine their internal models. According to the form of learning, ML can be divided into supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning [125]. Due to the ability to tackle complex and data-intensive tasks that would be difficult to solve with traditional rule-based techniques, ML has been applied to a variety of real-world applications, such as natural language processing, speech recognition, recommendation systems, fraud detection, and medical diagnosis [205, 57, 12, 22].

There is a hot topic in AI called anomaly detection or outlier detection, which aims to detect anomalous patterns or events that deviate significantly from normal behaviour [27]. Anomaly refers to 'non-conforming patterns' different from expected behaviors [59]. Although anomalies often represent a small fraction of the data or occur infrequently, they can lead to grave consequences like cascading failures in manufacturing and even loss of life in healthcare fatalities if not detected promptly. Currently, anomaly detection has become an omnipresent aspect of our lives and has been a fundamental problem in various fields, including cybersecurity, finance, healthcare, and industrial monitoring, because anomaly detection plays a crucial role in identifying financial fraud, detecting system failures of industrial equipment, and ensuring the integrity of complex AI systems [146]. Anomaly detection has gained broad attention and applied in a variety of real-world applications, including network intrusion detection [18], fraud detection in financial transactions [6], data leakage prevention [176], the identification of failures in complex systems [169], and diagnosis in the medical domain [43]. Specifically, network intrusion detection aims to find malicious activities in the operating system, network communication, and other user actions. Fraud detection often refers to finding anomalous actions in financial fraud, such as credit card fraud. Anomaly detection technology can monitor the data from particular locations or tremendous money transactions to identify anomalous user behaviour. In addition, failures in complex systems often happen in the sensor events, which require anomaly

detection to prevent accidents in the manufacturing systems. In many medical applications, anomaly detection is essential because of its crucial importance in detecting special data that reflects disease conditions.

With the rapid advancement of ML, anomaly detection is commonly merged with a variety of ML techniques [180, 19], including linear regression, ensemble learning, dimensional reduction, gradient decent, etc., which improves the ability of detection models to learn complex patterns automatically and identifies different types of anomalies. The advancement of machine learning technology has led to a significant rise in application data and diverse demands for identifying anomalies by users. In response, researchers have introduced a variety of unsupervised anomaly detection techniques [164], ranging from shallow to deep approaches, which are tailored to different types of anomalies and data formats. Collecting labelled data, especially for anomalies, is a difficult and expensive task. That is why researchers often turn to unsupervised machine learning techniques for anomaly detection in practical design [58]. For the supervised anomaly detection methods and the semi-supervised anomaly detection methods, we refer readers to other works [141, 56, 123, 202, 85]. In this dissertation, we focus on designing unsupervised anomaly detection methods from shallow to deep that help to achieve satisfactory effectiveness and robustness.

1.2 Research Problems

With the rapid development of AI techniques, the size of application data has increased dramatically in the last decades, resulting in a strong demand for algorithms suitable for these crucial large-scale data. In addition, the large-scale datasets tend to be high-dimensional, which increases the difficulty of finding anomalies. This is the first challenge caused by the "Volume" dimension of big data because large-volume datasets deactivate most existing detection methods in real-world applications, which are often of high computational complexity [158]. Furthermore, various data types in different applications generate the "Variety" dimension of big data, rendering the specific anomaly detection methods ineffective [128]. Therefore, it causes the second

challenge of consistently processing different kinds of data types by a specific anomaly detection method. Most existing anomaly detection methods fall short in terms of their robustness and applicability, limiting their use to only certain types of anomalies or applications. In this dissertation, we aim to design a generic framework with efficient and effective performance for anomaly detection in big data environments. The framework is designed to accommodate a wide range of data types, making it a versatile solution for anomaly detection.

To address the above challenges of the "Volume" dimension and the "Variety" dimension, the subsampling-based ensemble technique is a promising way to increase efficiency and robustness for anomaly detection in big data environments. Notably, a category based on the isolation forest mechanism stands out for its simplicity, effectiveness, and efficiency, e.g., iForest [110] is frequently employed as a state-of-the-art detector in real applications. However, most isolation forest based approaches fail to learn the data information effectively, which significantly impairs the effectiveness and robustness of anomaly detection. To tackle this challenge effectively, we can consider both shallow and deep techniques for anomaly detection. It is imperative to incorporate feature learning and tree structure learning while applying learning to hash scheme [209]. Herein, we divide the research problem into four sub-problems: How to learn the isolation forest and how to optimise the learning process; What is the optimal isolation tree structure and how to learn this optimal tree structure? The framework of the research problems in our dissertation is shown in Figure 1.1.

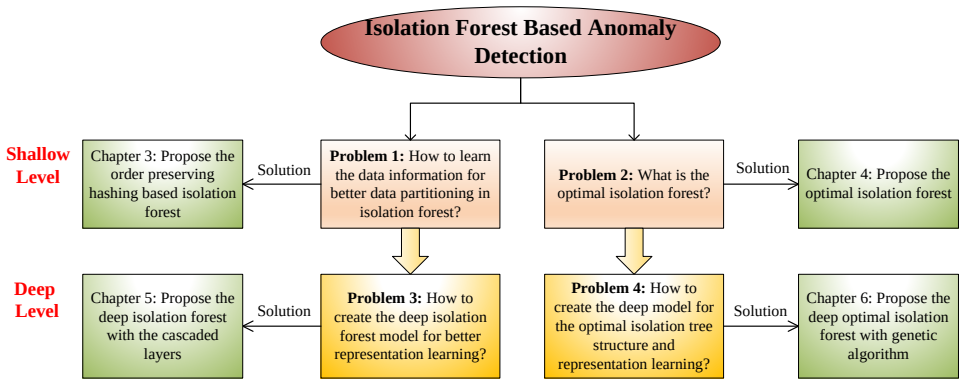


FIGURE 1.1: The framework of the research problems in our dissertation.

The four main problems are described and analysed in detail as follows. Liu et al. [110] first proposed an isolation-based ensemble anomaly detection method named iForest, combining the subsampling-based ensemble approach with the isolation scheme to quickly partition data instances in the isolation forest. Benefiting from the ensemble feature, iForest performs significantly better than the distance-based methods in terms of the execution time, especially in large datasets. As a variant, Liu et al. [111] also proposed another iForest-based scheme, called SCiForest, to detect the clustered anomalies while iForest targets scattered anomalies. Combining the hashing scheme with the isolation forest technique has been proven by researchers to effectively improve the detection accuracy while retaining the superior features of the isolation forest like fast execution speed. Typically, Zhang et al. [249] designed a generic anomaly detection framework named LSHiForest, which takes full advantage of locality-sensitive hashing (LSH) scheme [16] and iForest technique to detect anomalies in the big data environment. The LSHiForest framework provides four instances that utilise various distances: angular, Manhattan, Euclidean, and Kernel distances [91]. It is worth noting that the iForest and SCiForest are specific instances of LSHiForest and, hence, have been proven highly reliable and effective. Although LSH has probabilistically guaranteed that nearer data instances have a more significant probability to be mapped into the same binary codes and farther data instances are more likely to be mapped into the different binary codes, this kind of data-independent hashing scheme fails to learn data information to construct a more nature isolation forest for anomaly detection [208, 240]. Besides, this limitation is inherent to all the iForest-based anomaly detection methods using a data-independent hashing scheme. To effectively capture the relationship between data instances or structural information of isolation trees, we leverage the powerful learning to hash scheme [209]. This approach can potentially extract more information from the data to significantly enhance anomaly detection. There are numerous types of learning to hash schemes available [209]. The research challenge is to properly learn the hash values of data instances and construct isolation trees for highly effective and robust anomaly detection. Specifically, **Problem 1** in our dissertation is described as follows.

Problem 1: How to appropriately learn the isolation forest to achieve the higher effectiveness and robustness of anomaly detection? (This problem has been addressed in Chapter 3)

While most isolation forest based methods following iForest use the binary tree structure for data isolation, the framework LSHiForest [249] producing multi-fork isolation trees with the use of similarity hash functions has demonstrated better detection performance. Although multi-fork isolation trees can be elegantly constructed in LSHiForest and better detection performance can be achieved, the theoretical understanding of this phenomenon is still missing. Besides the tree structure, it is still a challenge on how much information isolation forest should learn from the data to facilitate data partition at each internal node. For instance, iForest learns the minimum and maximum of the selected feature to determine the random splitting value, and its variant SCiForest [111] learns more information to determine the splitting hyperplane and achieve better performance. But it is worth noting that more learning does not mean better detection performance while the bias of a single base detector can be reduced, according to the bias-variance trade-off theory in ensemble learning [5]. LSHiForest is data-independent, i.e., no learning is incorporated in LSHiForest for data isolation, but it achieves better detection accuracy than iForest. Accordingly, an interesting problem is identified as follows.

Problem 2: What is the optimal isolation tree, and how to learn this optimal tree structure for anomaly detection? (This problem has been addressed in Chapter 4)

In recent years, deep learning excels at learning expressive representations of complex data, such as high-dimensional, temporal, spatial, and graph data, and has greatly pushed the boundaries of different learning tasks [238]. Almost all current deep learning models apply neural network (NN) as the core mechanic, which uses the backpropagation algorithm to train parameters in their nonlinear modules [97]. DNNs can train high-accuracy models, but they are often plagued by long execution times, very deep layers, and high memory consumption [263]. These problems are associated with the inherent drawbacks of deep learning, such as too many parameters and deep training layers. To remedy the above drawbacks, an unsupervised non-neural network deep

model is explored for anomaly detection based on the mechanism of deep forest [263].

Problem 3: How can we design a deep anomaly detection (DAD) model with a non-neural network scheme to produce good feature representations and achieve a high detection accuracy? (This problem has been addressed in Chapter 5)

DAD methods are considered effective approaches for addressing complex anomaly detection problems, but they often suffer from issues like complex parameter learning, expensive computations, etc. Then, a recent work [232], named DIF, involved the isolation forest to deep isolation forest, which applies random neural networks to learn powerful representations in each layer of the isolation trees and achieves fast detection with high accuracy. DIF explores new deep models based on the isolation forest and leverages the power of deep isolation forest to learn stronger representations for anomaly detection. However, the existing deep isolation forest approaches are all based on representation learning while ignoring the learning of the inherent tree structure. Our proposed OptiForest approach [225] has theoretically proved the crucial role of the tree structure in isolation forest based methods. Herein, a corresponding problem arises as follows.

Problem 4: How to learn the optimal isolation tree structure through a deep model in a practical algorithm? (This problem has been addressed in Chapter 6)

1.3 Research Contribution

As analysed in the research problems, the two big data challenges (Volume and Variety) imposed on anomaly detection have not been tackled adequately. In this dissertation, we aim to design a generic framework with the learning to hash scheme to provide efficient and effective performance for anomaly detection in the big data environment. To be generic, this framework can cope with various types of data for a wide range of applications. By fully exploiting the power of deep learning, our framework has been enhanced significantly to overcome the big data challenges and deliver superior detection performance. Overall, the contributions of this dissertation are summarised as follows:

- In Chapter 2, we categorise and provide taxonomies for the types of anomalies and existing unsupervised anomaly detection methods, from shallow to deep methods, by a comprehensive review. The key anomaly detection methods for tabular data can be categorised into statistical methods, proximity-based, ensemble-based, and deep learning-based methods. Based on the literature review, we analyse the advantages and limitations of the proposed methods and then provide the future directions of isolation-based anomaly detection. Finally, we summarise the basic information of benchmarking datasets used in previous work and the commonly-adopted performance metric, which will also be extensively employed in our evaluations.

- In Chapter 3, we analyse the order preserving hashing (OPH), one type of learning to hash schemes and extend OPH to anomaly detection by changing the learning mode in the nearest neighbour search. In the learning process of OPH, we use two steps of learning to improve the efficiency and effectiveness of OPH: One is coarse-learning, to learn the coarse hash function and determine the optimised direction, and the other is fine-learning, to obtain the optimal solution. Then, an isolation forest based anomaly detection method, called OPHiForest, is proposed by combining the OPH scheme with the isolation forest without sacrificing either the computational efficiency or detection accuracy. Finally, we provide guiding insight on how the parameters of our method influence the prediction accuracy.

- In Chapter 4, we are the first to formally investigate the optimality problem of isolation tree structure with respect to the branching factor and establish a theory on the optimal isolation forest, offering a theoretical understanding for the effectiveness of the isolation forest mechanism. Then, we innovatively propose a practical optimal isolation forest, named OptIForest, which can enhance detection performance and computational efficiency by designing tailored clustering-based learning to hash on an excellent bias-variance trade-off. We have conducted extensive experiments on a variety of benchmarking datasets for both ablation and comparative studies, and the results have confirmed the effectiveness and

efficiency of OptIForest.

- In Chapter 5, we apply deep forest for deep anomaly detection and deepen the isolation forests into the cascaded forest. Herein, we propose an unsupervised DAD method called DeepiForest, which has few parameters, short training time, and robust detection accuracy. Specifically, we improve the representation learning by combining the features of label representation and tree-embedding representation. This process enriches the diversity of features and adds more enhanced features to the original data instances. Extensive experiments illustrate the efficiency and robustness of our method. Besides, we conduct a series of ablation studies on how tree-embeddings, the number of trees in a forest, and the number of cascaded layers influence the performance of the proposed anomaly detection method.

- In Chapter 6, we conduct the first work to optimise the isolation tree structure through the deep learning model. We make a theoretical analysis for the search space in our problem and design a genetic algorithm-based deep model to solve the discrete optimisation problem. Specifically, we design two mutation schemes for isolation tree evolution and use the isolation efficiency to select the optimal isolation forest in our approach. Finally, extensive experimental results illustrate the effectiveness and robustness of our method. We conduct a series of ablation studies on how the number of layers and the number of trees in a forest influence our proposed method.

- We have created the open-source repositories of our proposed methods, which are available in https://github.com/xiagll/OptIForest.

1.4 Thesis Structure

In this dissertation, we conduct a series of research works about isolation forest based anomaly detection, which is organised as follows. Firstly, we introduce the background

of AI and ML, which are closely related to anomaly detection, and analyse the problems still to be solved based on previous research for anomaly detection. Then, we summarise the contributions made to address the identified problems. In Chapter 2, we conduct a literature review on the types of anomalies and existing unsupervised anomaly detection methods, from shallow to deep methods. In Chapter 3, we propose a novel anomaly detection method, named OPHiForest, with the use of the order preserving hashing-based isolation forest, which can handle complicated anomalies like local anomalies and achieve high detection accuracy. In Chapter 4, we investigate an interesting problem: What is the optimal branching factor for an isolation tree and establish a theory on the structure optimality of an isolation tree with respect to the branching factor by introducing the notion of isolation efficiency? In Chapter 5, we design an unsupervised non-neural network deep model for anomaly detection based on the experience of deep forest, because the proposed deep anomaly detection methods have the inherent drawbacks of deep learning, such as too many parameters and fixed training layers. In Chapter 6, we analyse the search space of isolation trees under specific data instances and address the challenges in finding optimal isolation forest by designing a deep model. Finally, Chapter 7 summarises the dissertation, discusses the contributions and provides the sightseeing of future directions.

2

Literature Review

In this chapter, we first introduce the fundamental machine learning (ML) techniques and their practical applications in detecting anomalies. Then, we describe the categories of anomaly types and the detailed content of different anomalies. Anomaly detection models can be broadly categorised into four types based on the ML models used: statistical methods, proximity-based methods, ensemble-based methods, and deep learning-based methods. Herein, we make a literature review of different anomaly detection methods and analyse their characteristic. We focus on the isolation forest based anomaly detection methods, categorised into the ensemble-based methods, which have been approved to be efficient and effective in various applications. Finally, we summarise most, if not all, the datasets and metrics used in different anomaly detection methods, and the links of the open-source codes.

2.1 Fundamental Techniques of Machine Learning

ML techniques are increasingly adopted to anomaly detection, their powerful capabilities of model training and solving complex problems. ML provides the autonomous learning ability of the computer and makes an effort to "automate the process of knowledge acquisition from examples" [20]. Currently, the most popular classification criteria divide ML into four categories, including supervised learning, semi-supervised learning, unsupervised learning, and reinforcement learning. Herein, we will describe some detailed methods of four categories and analyse the relationship between these techniques and anomaly detection.

Supervised Learning. In this approach, the ML model is trained on labelled data, and the parameters are trained by the labels of the input data. The goal is to learn a mapping function that can accurately predict the output labels for new data [166]. Given a training dataset $D = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, each training example x is represented by a vector, sometimes called a feature vector, and y represents a label. Through iterative optimization of an objective function $y = f(\boldsymbol{x}; \boldsymbol{\theta})$, supervised learning algorithms learn a function $y = f(\boldsymbol{x}; \boldsymbol{\theta}')$ that can be used to predict the output associated with new inputs. When the output y is discrete, the ML algorithm is called classification algorithm [131]; otherwise, the regression algorithm will produce the continuous output [7]. Common supervised learning algorithms for anomaly detection include supervised neural networks, support vector machines (SVM), k-nearest neighbours, Bayesian networks, and decision tree [49].

Semi-supervised Learning. This technique occupies a middle ground between unsupervised learning, which lacks labelled training data, and supervised learning, which relies on fully labelled training data [160]. Although some training examples lack labels, researchers in machine learning have discovered that utilizing unlabelled data in combination with a small amount of labelled data can significantly enhance learning accuracy. Semi-supervised learning deals with training labels that can be noisy, limited, or imprecise. A key challenge in anomaly detection is the lack of quality training instances. For example, accessing labelled instances for normal behaviour is generally

easier, but getting access to a labelled set of anomalous instances is challenging. Under these cases, semi-supervised learning can achieve good anomaly detection performance through learning from a small amount of labelled data [202].

Unsupervised Learning. The unsupervised learning algorithms learn from training data that has not been labelled, classified or categorised [246]. Because the characteristics of the anomalies are rare and difficult to label, the unsupervised ML methods have become the most commonly used approaches in anomaly detection [58]. Many semi-supervised techniques can be adapted into the unsupervised modes through using unlabelled data instances as the training data. Such adaptation assumes that there are very few anomalies in the testing data and these few anomalies are robust to the model training [141]. Common unsupervised learning algorithms for anomaly detection [51] include the one-class support vector machine (OCSVM), unsupervised neural networks, local outlier factor (LOF), isolation forest based methods, etc.

Reinforcement Learning. This technique focuses on guiding software agents to make decisions within an environment to maximise cumulative rewards [201]. The versatility of reinforcement learning has made it widely applied in various fields [188], including game theory, simulation-based optimization, multi-agent systems, swarm intelligence, genetic algorithms, etc. Reinforcement learning adopts the framework of a Markov decision process (MDP) to represent the environment and dynamic programming techniques are commonly applied in numerous reinforcement learning algorithms. In anomaly detection, reinforcement learning is always used to enhance the detection performance of an ML model and keeps evolving with the growth of anomaly detection experience [73].

2.2 Types of Anomaly

An anomaly can be a data point that is significantly different from the remaining data and also can be a group of data points that are deviated from the remaining data. Thus, the categories of anomalies can be diverse depending on different data representations. Currently, anomalies can be classified into three main categories according to

the different types of data [72]:

1. **Point Anomaly:** If the sample data point is too far away from other data instances in the geographic space, this sample point is regarded as an anomalous point. In terms of the dimensions of the anomaly point, the value of most dimensions is different to that in normal data instances. This kind of anomaly is often found in credit fraud detection and medical anomaly detection.

2. **Contextual Anomaly:** It refers to the fact that the data instance is anomalous in its context. So, the continuous data instances must be given, and the relationship between these data instances can be defined. For example, when detecting the weather of continuous days (or months), you can find the contextual anomaly that abnormal weather happens on a specific day (or month). Contextual anomaly detection has been widely used in detecting anomalies of time-series data and spatial data [214].

3. **Collective Anomaly:** In collective anomaly, only a group of data can be used to determine whether the behaviour of data instance is abnormal, while it is indistinguishable from finding whether the individual data is normal. Specifically, individual data instances are normal in a dataset but can have abnormal behaviours when linked together. This property is always applied to detect spatial outliers [178].

Because of the variety of anomalous types, one specific anomaly detection method can not perform well on all kinds of applications, which confirms the proverb that there is no free lunch in the world. In this dissertation, we focus on point anomaly, which has drawn lots of attention in anomaly detection research and many other types of anomalies can be transformed into point anomalies, like trajectory anomalies. The following discussion of related works also focuses on this type of anomaly.

2.3 Methods for Anomaly Detection

In this section, we introduce the unsupervised anomaly detection methods on tabular data or hybrid data types, which are relevant to our research. Specifically, the reviewed anomaly detection methods can be categorised into four types, including statistical methods, proximity-based, ensemble-based, and deep learning-based methods. To thoroughly understand the area, we introduce a hierarchical taxonomy to classify the relevant anomaly detection methods into four main categories and ten fine-grained categories from the modelling perspective. An overview of the taxonomy of these methods is shown in Figure 2.1. Specifically, for papers in the category of four main types, we further categorise them based on specific types of the target ML models. We further divide papers in the statistical methods category based on whether the model is parametric or nonparametric. For papers in the category of proximity-based methods, we further categorise them into cluster-based methods, distance-based methods, and density-based methods. For papers in the category of ensemble-based methods, we further categorise them into isolation-based ensembles and other ensembles. For papers in the category of deep learning-based approaches, we further categorise them based on three conceptual paradigms: deep learning for feature extraction, learning feature representations of normality, and end-to-end anomaly score learning.

In addition to classifying all related papers on unsupervised anomaly detection as portrayed in Figure 2.1, we further highlight a few representative papers, showcasing their attributes in Table 2.1. Each of these chosen papers is the representative work of each type of anomaly detection method. In contrast to the information provided in Figure 2.1, Table 2.1 furnishes a more comprehensive overview of each paper, thus providing readers with an enhanced comprehension and comparison of each contribution. Specifically, for each paper enumerated in Table 2.1, we provide details concerning the year of publication. The explanation of each metric and a comprehensive summary of each dataset can be located in Section 2.5 and Table 2.3, respectively. In the following four subsections, we review the above four paradigms of unsupervised anomaly

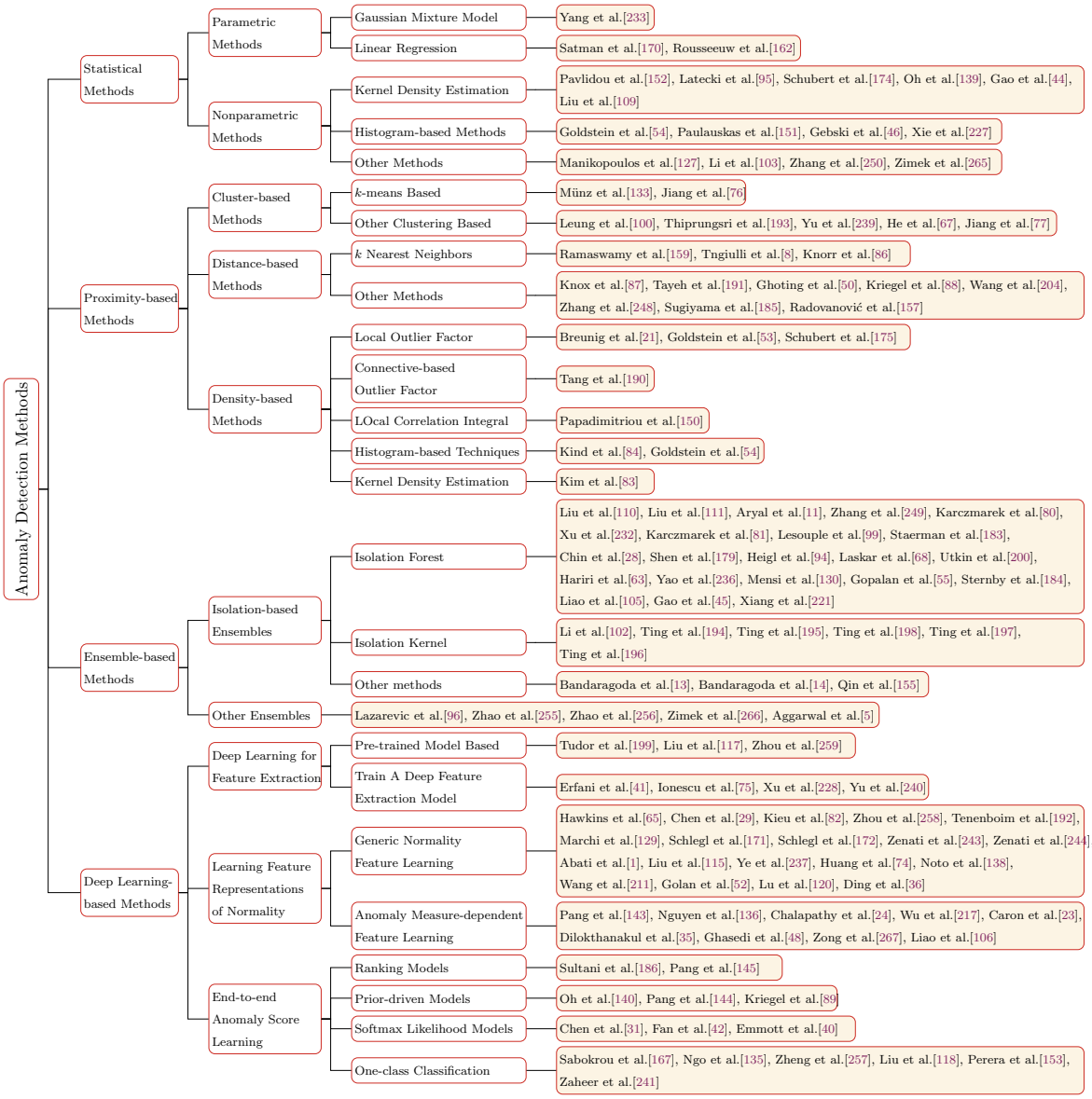


FIGURE 2.1: Taxonomy of unsupervised anomaly detection methods.

detection methods in detail.

2.3.1 Statistical Model-based Anomaly Detection

The statistical model-based methods are the earliest works that have been proposed for anomaly detection. The fundamental principle underlying these anomaly detection methods is that anomalies are commonly found in regions with low probabilities, whereas the opposite is typically valid for regular data instances. These approaches

Table 2.1: Summary of the representative work of unsupervised anomaly detection.

Method	Ref.	Year	Venue	Type	Metric	Code URL
OCSVM	[173]	1999	NIPS	Shallow	Visualization	sklearn library
KNN	[159]	2000	SIGMOD	Shallow	Detected anomalies	sklearn library
LOF	[21]	2000	SIGMOD	Shallow	Visualization	sklearn library
COF	[190]	2002	PAKDD	Shallow	Visualization	https://link.springer.com/chapter/10.1007/3-540-47887-6_53
ORCA	[17]	2003	SIGKDD	Shallow	Detected anomalies	None
PCA	[181]	2003	Technical report	Shallow	Precision, Recall	sklearn.decomposition.PCA
iForest	[110]	2008	ICDM	Shallow	ROC-AUC	sklearn.ensemble.IsolationForest
SCiForest	[111]	2012	ECML-PKDD	Shallow	ROC-AUC	https://github.com/david-cortes/isotree
HBOS	[54]	2012	KI	Shallow	ROC-AUC	http://madm.dfki.de/rapidminer/anomalydetection
iNNE	[13]	2014	ICDM Workshop	Shallow	ROC-AUC	https://github.com/zhuye88/iNNE
LSHiForest	[249]	2017	ICDE	Shallow	ROC-AUC	https://github.com/xuyun-zhang/LSHiForest
EIF	[63]	2019	TKDE	Shallow	ROC-AUC, PR-AUC	https://github.com/sahandha/eif
IDK	[195]	2020	SIGKDD	Shallow	ROC-AUC	https://github.com/IsolationKernel/Codes
ECOD	[103]	2022	TKDE	Shallow	ROC-AUC, PR-AUC	https://github.com/yzhao062/pyod
Replicator	[65]	2002	DaWaK	Deep	Detected anomalies	None
DEC	[226]	2016	ICML	Deep	Accuracy	https://github.com/piiswrong/dec
APE	[31]	2016	IJCAI	Deep	ROC-AUC, PR-AUC	https://github.com/wyxingyuX/APE
RandNet	[29]	2017	SDM	Deep	ROC-AUC	https://github.com/jinghuichen/autoencoder
DeepSVDD	[163]	2018	ICML	Deep	Visualization, ROC-AUC	https://github.com/yzhao062/pyod
REPEN	[143]	2018	SIGKDD	Deep	ROC-AUC	https://git.io/JfZRg
AEHE	[42]	2018	CIKM	Deep	ROC-AUC, MAP	https://github.com/googlebaba/CIKM2018-AEHE
DAGMM	[267]	2018	ICLR	Deep	Precision, Recall, F1-score	https://git.io/JfZR0
PUP	[140]	2019	SIGKDD	Deep	Precision, Recall, F1-score	None
AE-1SVM	[136]	2019	ECML-PKDD	Deep	ROC-AUC, PR-AUC	https://git.io/JfGgl
SDOR	[145]	2020	CVPR	Deep	Visualization, ROC-AUC	https://github.com/GuansongPang/SOTA-Deep-Anomaly-Detection
RDP	[206]	2020	IJCAI	Deep	ROC-AUC, PR-AUC	https://git.io/RDP
GLocalKD	[124]	2022	WSDM	Deep	ROC-AUC	https://git.io/GLocalKD
DIF	[232]	2023	TKDE	Deep	ROC-AUC, PR-AUC	https://github.com/GuansongPang/deep-iforest

are usually categorised into two main groups, parametric and nonparametric methods, based on their assumption of the knowledge of underlying distribution [26]. The central differentiation is that parametric methods operate based on the assumption that the data conforms to a predetermined parametric distribution. Consequently, the fitting procedure entails acquiring knowledge about the parameters intrinsic to this assumed

parametric distribution. Many advanced parametric methods, including using Gaussian mixture models (GMM) and linear regression, have been proposed for anomaly detection, which is fully reviewed in [233, 170, 162].

Histogram-based techniques represent some of the simplest and more instinctive nonparametric statistical algorithms. These methods are composed of two steps: Firstly, the algorithm constructs histograms for every dimension. Then, the anomaly score is derived by aggregating the bin heights in each dimension to which the data point belongs. The classical histogram-based anomaly detection methods include [54, 151, 46, 227]. Another type of nonparametric statistical method relies on the utilization of kernel functions. Kernel-based functions serve to approximate probability density, and instances of data with low probability density are identified as anomalies. The related works are discussed in [152, 95, 174, 139, 44, 109]. Furthermore, Other nonparametric algorithms can refer to [127, 103, 250, 265] for more detail.

Although the statistical model-based anomaly detection methods are always very simple and intuitive, it is hard for these approaches to solve complicated problems. Specifically, histogram-based methods cannot capture anomalies that are dependent on multiple features, and Kernel-based methods are computationally expensive and sensitive to parameters.

2.3.2 Proximity-based Anomaly Detection

The proximity-based anomaly detection methods utilise the local neighbourhood information surrounding each data point. Such methods can be grouped into three categories: cluster-based, density-based, and distance-based methods.

The intuition of the cluster-based anomaly detection methods is that every data point within the dataset is categorised as either belonging to a cluster or representing an anomaly [3]. The classic works that belong to this category contain [133, 76, 100, 193, 239, 67, 77]. The cluster-based methods are easily operated in unsupervised setting and can deal with complex and different data types. However, these approaches heavily rely on clustering algorithm design and are time-consuming in high-dimensional datasets.

Distance-based techniques evaluate objects by measuring their proximity to neighbouring items, and objects significantly distant from their neighbours are classified as outliers. Among distance-based methods, the anomaly score of a given data point is generated by summing the distances separating the point from its k-nearest neighbours. The most representative methods for anomaly detection based on distance is k Nearest Neighbors (k-NN) [159] and its extensions, including [8, 86]. Besides, there are other distance-based anomaly detection methods, which are described in [87, 191, 50, 88, 204, 248, 185, 157]. These methods are all easy to implement and straightforward, and also independent to data distribution. However, implementing them in practical applications is challenging due to their high computational cost.

The last type is the density-based technique, which contains the intuition that the density of the anomaly is significantly different from the normal instance, i.e., an anomaly can be found in a low-density region, whereas non-outliers are assumed to appear in dense neighbourhoods [168]. The classical density-based anomaly detection method is the local outlier factor (LOF) [21]. This method calculates the local density of a data point based on the distance to its k nearest neighbours and then compares it to the local densities of its neighbours to determine if it is an outlier. The distances of these neighbouring points are then employed to support the estimation of local densities. Any data point showcasing significantly lower local density compared to its neighbours can be identified as an outlier. Then, a few variants of LOF are proposed, including Connective-based Outlier Factor (COF) [190], LOcal Correlation Integral (LOCI) [150] and FastLOF [53]. Other density-based anomaly detection methods are described in [84, 54, 83]. These methods are very intuitive and perform well in detecting local anomalies, but a considerable drawback of employing these anomaly detection algorithms lies in their computational cost, stemming from the computation of pairwise distances.

Proximity-based approaches are largely nonparametric since they avoid assuming any parametric data distribution. This quality can offer a significant advantage in anomaly detection, as it demands minimal prior information on the probability distribution of datasets. This intrinsic quality renders these methods both intuitive and

easily graspable. However, this type of approach often comes with high computational costs, sensitivity to hyperparameter tuning and vulnerability to the curse of dimensionality.

2.3.3 Ensemble-based Anomaly Detection

Ensemble-based methods amalgamate outputs from various foundational anomaly detectors to produce results of heightened robustness [168]. The intuition for anomaly detection is similar to that in classification/regression tasks (e.g., bagging, boosting, random forests). Benefit from the outstanding robustness and effectiveness, ensemble scheme has been widely applied in anomaly detection, such as the classical isolation forest [110], isolation kernel [194], and LSHiForest [249]. As the first instance, iForest [110] has been widely recognised in academia and deployed in real applications, e.g., it has been included in *scikit-learn*, a commonly-used machine learning library in Python. Then, some extensions to iForest, like SciForest [111] and extended isolation forest [63], have improved the hash learning scheme or anomaly score calculation to achieve better detection accuracy. Typically, a generic anomaly detection framework, named LSHiForest, takes full advantage of locality-sensitive hashing (LSH) scheme and iForest technique to detect anomalies in big data environment [249]. Except for the isolation forest, the isolation kernel scheme is another type of ensemble-based anomaly detection technique [195]. Ting et al. [195] proposed an isolation distributional kernel as a new way to detect both point and group anomalies, which is based on a data dependent point kernel. Other ensemble-based methods can refer to [111, 11, 63, 80, 81, 232, 99, 183, 94, 68, 200, 28, 179, 105, 130, 236, 45, 55, 184, 221, 13, 155, 102, 198, 197, 196] for more detail.

Typically, ensemble-based anomaly detection methods demonstrate effective and efficient results, even in the context of datasets with a large number of dimensions or large-scale sizes. However, these methods are vulnerable to the base detectors, which proposes a big challenge on algorithm design.

2.3.4 Deep Learning-based Anomaly Detection

Deep neural networks (DNNs) make use of complex compositions of linear and non-linear functions, which are symbolically depicted using a computational graph, facilitating the acquisition of representations that are both intricate and expressive [132]. The fundamental building blocks of DNNs are activation functions and layers. Activation functions play a pivotal role in computing the output of all nodes (i.e., neurons in neural networks) through the given inputs. They can adopt either linear or non-linear forms, such as linear, sigmoid, tanh, Rectified Linear Unit (ReLU) and its variants. Neural networks comprise layers, which denote arrangements of neurons. Widely utilised layer types include fully connected, convolutional, pooling, and recurrent layers, which serve as the foundation for constructing a variety of widely recognised neural network structures. Given a dataset $D = \{x_1, x_2, ..., x_N\}$ with $x_i \in \mathbb{R}^d$, let $\mathbb{S} \in \mathbb{R}^k (k << d)$ be a representation space, then deep anomaly detection (DAD) aims at learning a feature representation mapping function $\phi(\cdot) : D \rightarrow \mathbb{S}$ or an anomaly score learning function $h(\cdot) : D \rightarrow \mathbb{R}$ in a way that anomalies can be distinctly separate from the normal data instances in the mapping space produced by either the ϕ or h function, where ϕ and h denote mapping functions driven by neural networks, each encompassing $H \in L$ hidden layers, alongside their weight matrices denoted as $W = \{w^1, w^2, ..., w^H\}$. The difference between $\phi(\cdot)$ and $h(\cdot)$ feature mapping is that the anomaly score needs to be calculated for each data instance within the mapping representation space of $\phi(\cdot)$ scenario. Conversely, $h(\cdot)$ can directly infer the anomaly scores using original data inputs.

Currently, DNNs are classified into three main categories according to three conceptual paradigms: Deep Learning for Feature Extraction, Learning Feature Representations of Normality, and End-to-end Anomaly Score Learning [146]. In the first paradigm, deep learning techniques are used as some independent feature extractors only. After the feature extraction, anomaly scores are measured based on these new representations. In the second paradigm, feature extraction and anomaly score measures depend on each other in some form, with the objective of learning expressive representations of normality. The last paradigm fully unifies the above two modules, in which

the methods are dedicated to learning anomaly scores via neural networks in an end-to-end fashion [146]. Some representative examples with neural networks extraction include the use of generative adversarial networks (GANs) [118, 171, 243], autoencoders (including variational autoencoders) [29, 82, 258], and reinforcement learning [147, 140]. There are more unsupervised DAD methods, which have been introduce in [120, 36, 129, 244, 1, 115, 237, 74, 138, 192, 211, 52, 143, 136, 24, 217, 23, 35, 48, 267, 106, 186, 145, 89, 31, 42, 40, 167, 135, 257, 118, 153, 241]

Overall, DAD approaches can produce effective and robust detection performance in complex applications, especially in scenarios involving large datasets. However, the computational cost associated with most DAD methods is substantial. Moreover, these methods often entail intricate hyperparameter tuning and pose challenges in terms of their interpretability of deep nets.

In this dissertation, we focus on isolation forest based anomaly detection, which has shown excellent simplicity, effectiveness, and efficiency for big data anomaly detection. Herein, the following context will review the isolation forest based approaches from shallow to deep.

2.4 Isolation Forest Based Anomaly Detection

In the current big data era, datasets characterised by their extensive volume, high dimensionality, heterogeneity, wide geographical distribution, and rapid evolution present a significant challenge on traditional data mining and analytical methods, including anomaly detection [219]. A number of unsupervised anomaly detection methods have been proposed to tackle the above challenges, as analysed in Section 2.3. By analyzing and comparing these anomaly detection methods, we found that a category of detection methods based on the isolation forest mechanism [5, 63] stands out of the shallow models due to its excellent simplicity, effectiveness, and efficiency, therefore being promising for big data anomaly detection.

The first instance of isolation forest techniques is called iForest [110], which has developed many extensions and has been widely applied in industry. The construction

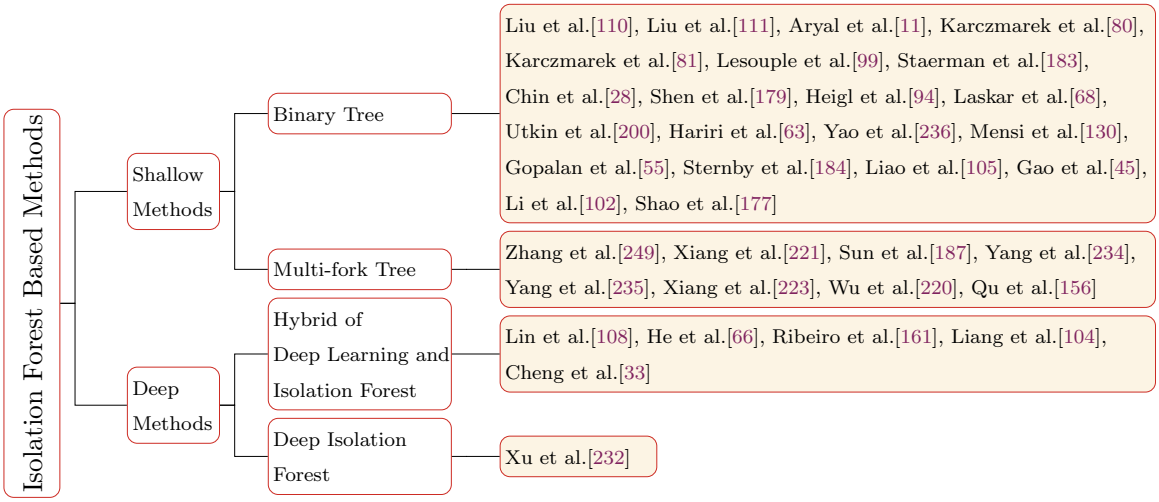


Figure 2.2: Taxonomy of isolation forest based anomaly detection methods

of an isolation forest involves composing an ensemble by aggregating multiple isolation trees. In each isolation tree, the data instances are randomly split into nodes within the randomly selected attributes, eventually ending in singleton nodes containing one instance. Remarkably, the tree branches containing anomalies are noticeably less deep because these data instances are located in sparse regions. Consequently, the outlier score is determined based on the path length from the leaf to the root. In the final step of the process, the aggregation of outcomes is achieved through the computation of the average path lengths for data points in different trees within the isolation forest. Although iForest is very fast and can achieve high accuracy in some specific applications, it is hard to detect local anomalies and complex datasets limited by its axis-parallel partitioning. Herein, more extensions to iForest and even deep learning-based isolation forest methods have been developed for anomaly detection. In this section, we make a comprehensive review of the isolation forest based anomaly detection methods from shallow to deep, shown in Figure 2.2.

Isolation forest based anomaly detection methods can be categorised into two types: Shallow methods and deep methods. Specifically, for papers in the category of shallow methods, we further categorise them into binary trees and multi-fork trees, based on the branching factor of isolation trees. For papers in the category of deep methods,

we further categorise them into two types according to the structure of the depth models. The methods of "Hybrid of Deep Learning and Isolation Forest" focus on feature learning by neural networks-based deep models and anomaly score training by traditional isolation forest models. Besides, "Deep Isolation Forest" methods tend to expand the isolation forest into deep paradigms without neural networks.

2.5 Metrics and Datasets for Anomaly Detection

In this section, we first summarise the metrics for evaluating the effectiveness performance of anomaly detection. Then, we summarise common datasets used in the existing works of unsupervised anomaly detection methods and provide relevant links to the open-source datasets.

2.5.1 Metrics

In anomaly detection, the Area Under Receiver Operating Characteristic Curve (AUC-ROC) and the Area Under Precision-Recall Curve (AUC-PR) are the most popular criteria for accuracy evaluation[92, 207]. These metrics are derived from the confusion matrix, which is a kind of visual display tool to evaluate the quality of the classification model in machine learning. Specifically, each row of the matrix represents the sample case predicted by the model and each column of the matrix represents the true situation of the sample. The confusion matrix is shown in Table 2.2.

Confusion Matrix		Actual Label	
		Positive	Negative
Predicted Label	Positive	True Positives (TP)	False Positives (FP)
	Negative	False Negatives (FN)	True Negatives (TN)

TABLE 2.2: Confusion Matrix

It can be seen from Table 2.2 that four basic concepts are proposed, indicating the true class of the sample is a positive class, and the result of model prediction is also a

positive class (TP); the true class of the sample is positive, but the model recognises it as a negative class (FN); the true class of the sample is negative, but the model recognises it as a positive class (FP); the true class of the sample is negative, and the model recognises it as a negative class (TN), respectively. Then, four indicators are derived from the confusion matrix, including Recall, Precision, True Positive Rate (TPR), and False Positive Rate (FPR). These four indicators can be calculated as:

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}},$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}},$$

$$\text{True Positive Rate} = \frac{\text{TP}}{\text{TP+FN}},$$

$$\text{False Positive Rate} = \frac{\text{FP}}{\text{FP+TN}}.$$

The total number of data instances denoted as S, is formalised as $S = TP + FP + TN + FN$. The ROC curve takes TPR as the Y-axis and FPR as the X-axis, so the value of AUC-ROC is the area under the ROC curve. Similarly, the PR curve summarises the relation between Precision and Recall. The value of AUC ranges from 0 to 1, where a larger AUC result indicates better performance. The result of AUC has been extensively adopted in many anomaly detection works and has become an essential measure of accuracy in correlational research.

Apart from the above metrics, $F_\beta - Score$ is another popular type of metrics for anomaly detection [38, 251, 252], which can be calculated by:

$$F_\beta - Score = \frac{(1+\beta^2) * Precision * Recall}{\beta^2 * Precision + Recall},$$

the meaning of F_β is a weighted average of the precision and the recall, and in the process of merging, the weight of the recall is β times the precision. Specifically, The F_1 score considers the recall and the precision equally important. The F_2 score considers the recall twice as important as the precision, which emphasises the model's ability to recognise positive samples. Besides, the $F_{0.5}$ score considers the recall half as important as the precision, which emphasises the model's ability to distinguish negative samples.

2.5.2 Evaluation Datasets and Open-source

In this part, we present a comprehensive summary of the widely used tabular datasets in the anomaly detection domain. Refer to Table 2.3 for detailed information. The datasets are categorised into 16 types according to the field of original data acquisition, including healthcare, network, finance, image, sociology, botany, web, astronautics, linguistics, physical, chemistry, oryctognosy, forensic, biology, document, and NLP. We summarise the information on data size and anomaly rate and also provide the open-source link of each dataset.

Table 2.3: Summary of publicly accessible real-world datasets with real anomalies.

Dataset	Samples	Features	Rate(%)	Category	URL
Annthyroid	7,200	6	7.42	Healthcare	UCI Machine Learning Reposiroty [1]
Arrhythmia	452	274	14.60	Healthcare	UCI Machine Learning Reposiroty
Cardio	1,831	21	9.61	Healthcare	UCI Machine Learning Reposiroty
Breastw	683	9	35.00	Healthcare	UCI Machine Learning Reposiroty
Pima	768	8	34.90	Healthcare	UCI Machine Learning Reposiroty
Mammography	11,183	6	2.32	Healthcare	UCI Machine Learning Reposiroty
Thyroid	3,772	6	2.47	Healthcare	Kaggle Reposiroty [2]
Backdoor	95,329	196	2.44	Network	https://github.com/GuansongPang/ADRepository-Anomaly-detection-datasets
KDDCup99	494,021	38	1.77	Network	Kaggle Reposiroty
Campaign	41,188	62	11.27	Finance	https://github.com/Minqi824/ADBench/tree/main/adbench
Creditcard	284,807	29	0.17	Finance	https://github.com/GuansongPang/ADRepository-Anomaly-detection-datasets
Fraud	284,807	29	0.17	Finance	https://github.com/Minqi824/ADBench/tree/main/adbench
bank	41,188	10	11.27	Finance	https://github.com/GuansongPang/ADRepository-Anomaly-detection-datasets
Celeba	202,599	39	2.24	Image	https://github.com/GuansongPang/ADRepository-Anomaly-detection-datasets
Mnist	7,603	100	9.21	Image	https://github.com/Minqi824/ADBench/tree/main/adbench
Skin	9,521	3	3.67	Image	Kaggle Reposiroty
ALOI	49,534	27	3.04	Image	https://github.com/Minqi824/ADBench/tree/main/adbench
InternetAds	1,966	1,555	18.72	Image	UCI Machine Learning Reposiroty
CIFAR10	5,263	512	5.00	Image	Kaggle Reposiroty
MNIST-C	10,000	512	5.00	Image	Kaggle Reposiroty
Census	299,285	500	6.20	Sociology	https://github.com/GuansongPang/ADRepository-Anomaly-detection-datasets
Donors	619,326	10	5.92	Sociology	https://github.com/GuansongPang/ADRepository-Anomaly-detection-datasets
AD	3,279	1,555	14.0	Sociology	https://github.com/GuansongPang/ADRepository-Anomaly-detection-datasets
Cover	286,048	10	0.96	Botany	https://github.com/GuansongPang/ADRepository-Anomaly-detection-datasets
Wilt	4,819	5	5.33	Botany	https://github.com/Minqi824/ADBench/tree/main/adbench
Http	567,498	3	0.39	Web	https://github.com/Minqi824/ADBench/tree/main/adbench
Smtp	95,156	3	0.03	Web	https://github.com/Minqi824/ADBench/tree/main/adbench
Probe	64,759	6	6.43	Web	KaggleReposiroty
U2R	60,821	6	0.37	Web	KaggleReposiroty
Satellite	6,435	36	31.60	Astronautics	UCI Machine Learning Reposiroty
landsat	6,435	36	20.71	Astronautics	Kaggle Reposiroty
Shuttle	49,097	9	7.15	Astronautics	UCI Machine Learning Reposiroty
Vowel	1,456	12	3.43	Linguistics	UCI Machine Learning Reposiroty
speech	3,686	400	1.65	Linguistics	https://github.com/Minqi824/ADBench/tree/main/adbench
Waveform	3,505	21	4.62	Physics	UCI Machine Learning Reposiroty
Fault	1,941	27	34.67	Physical	https://github.com/Minqi824/ADBench/tree/main/adbench
Magic.gamma	19,020	10	35.16	Physical	UCI Machine Learning Reposiroty
Wine	5,318	11	4.53	Chemistry	UCI Machine Learning Reposiroty
musk	3,062	166	3.17	Chemistry	UCI Machine Learning Reposiroty
Ionosphere	351	32	35.90	Oryctognosy	UCI Machine Learning Reposiroty
Glass	214	7	4.21	Forensic	UCI Machine Learning Reposiroty
Yeast	1,484	8	34.16	Biology	UCI Machine Learning Reposiroty
SpamBase	4,207	57	39.91	Document	UCI Machine Learning Reposiroty
Amazon	10,000	768	5.00	NLP	https://github.com/Minqi824/ADBench/tree/main/adbench
Yelp	10,000	768	5.00	NLP	https://github.com/Minqi824/ADBench/tree/main/adbench

3

Order Preserving Hashing Based Isolation Forest for Robust and Scalable Anomaly Detection

In the big data era, most applications demand fast and versatile anomaly detection capability to handle various types of increasingly huge-volume data. However, existing detection methods are either slow due to high computational complexity, or unable to deal with complicated anomalies like local anomalies. Besides, as we mentioned in the first research problem in Chapter 1, most isolation forest based anomaly detection methods fail to fully learn the data information and produce high effectiveness. In this chapter, we propose a novel anomaly detection method named OPHiForest with the use of the order preserving hashing based isolation forest. The core idea is to learn

the information from data to construct better isolation forest structure than the state-of-the-art methods like iForest and LSHiForest, which can achieve robust detection of various anomaly types. Finally, extensive experiments on both synthetic and real-world datasets demonstrate that our method is highly robust and scalable.

3.1 Overview

Anomaly detection is a technique used to identify abnormal patterns that do not meet the expected behavior, or to find objects that are different from most objects [122]. Currently, anomaly detection, also called outlier detection, is the most adopted data mining and analytic task in a range of fields such as credit card transaction analysis, network intrusion detection, diagnosis in the medical domain, traffic anomaly detection, etc [182, 64]. The applications in these fields always have ultrahigh dimensionality or produce large volumes of data, which raises significant challenges on the accuracy and efficiency of traditional anomaly detection methods, e.g., Local Outlier Factor (LOF) [21], k Nearest Neighbors (kNN) [215], one-class Support Vector Machines (OCSVM) [32]. Specifically, the traditional anomaly detection methods can not produce accurate results in customers' tolerable time, due to the limitation of huge-volume data[218, 37]. Some researchers have optimised the traditional methods, but they still can not meet the requirement of Low latency and high accuracy in most applications.

To deal with the above situation, isolation forest (iForest) is introduced to enhance the accuracy and efficiency of anomaly detection [110]. Specifically, iForest lowers the high-dimensions of datasets by randomly selecting one dimension to partition anomalous and normal data instances each time. Besides, the sampling technique is applied in iForest, which tremendously decreases the training set of big data but improving the accuracy by averaging the sample results [16]. With the promotion of random partition and sampling, iForest has been demonstrated to be 100 times faster than other existing anomaly detection methods [112]. Based on this remarkable result, a following anomaly detection method called SCiForest, which is specialised in detecting

clustered anomalies, was proposed to detect specific anomaly type [111]. These methods obtain considerable efficiency and accuracy in specific areas, but the applied range of these methods is limited, e.g., iForest suffers from incapability in detecting axis-parallel anomalies and SCiForest is closely designed for clustered anomaly detection. To expand the detection range and keep the superior features of iForest, some people combine the hashing scheme with the isolation forest technique.

Typically, a generic anomaly detection framework, named LSHiForest, takes full advantage of locality-sensitive hashing (LSH) scheme and iForest technique to detect anomalies in big data environment [249]. Compared with other ensemble-based anomaly detection methods, LSHiForest demonstrates to be faster and more accurate for anomaly detection in big data, and not limited to some specific data types. Although LSH has probabilistically guaranteed that nearer data instances (or normal points) have a larger probability to be mapped into the same binary codes and farther data instances (or anomalies) are more likely to be mapped into the different binary codes, this kind of data-independent hashing scheme loses much data information resulting in the loss of detection accuracy [208, 240]. Also, the drawback is similar to all the iForest based anomaly detection methods using a data-independent hashing scheme. So the data-dependent hashing draws some attention for iForest based anomaly detection method, despite the difficulty to learn a suitable data-dependent hash function.

In view of the data-dependent hashing approach (or learning to hash approach), it is the similarity preserving hashing, which learns hash functions from the dataset so that similar data instances in the original space are mapped into the same hash codes in the projection space [209]. There are many learning to hash algorithms, including [113, 208, 254, 78, 212, 116], which have been widely-studied solutions to the approximate nearest neighbor search. Learning to hash scheme is not so effective if directly applied to anomaly detection, since the core idea of hashing is to map similar items into the same hash codes rather than isolate the different items. So we focus on finding the proper learning to hash scheme and combine it with isolation forest to enhance the separation efficiency.

In an attempt to address the above issues, we find a learning to hash based approach called order preserving hashing (OPH), which showed excellent performance over existing state-of-the-art hashing techniques for nearest neighbor search [208]. Specifically, OPH always needs shorter code length or fewer hash tables to learn the hash function compared with other hashing schemes [208]. In this chapter, we adapt OPH to anomaly detection by designing a fast two-step learning process, and combining it with isolation forest to construct a fast and accurate anomaly detection method, named OPHiForest. The basic idea is to learn hash functions through minimizing the gap between the similarities computed in the original space and the hash coding space. Then, iForest is used to rapidly isolate data instances and compute anomalous factors according to the hash indices. The experimental results prove this method has better robustness and scalability compared to existing iForest based anomaly detection methods. The main contributions of this chapter are three-fold: 1) We analyse the OPH scheme and extend OPH to anomaly detection by changing the learning mode in the nearest neighbor search. In the learning process of OPH, we use two steps of learning to improve the efficiency and effectiveness of OPH: one is coarse-learning, to learn the coarse hash function and determine the optimised direction, and the other is fine-learning, to obtain the optimal solution. 2) We design an isolation forest based anomaly detection method, named OPHiForest, by using the order preserving hashing scheme. Then, we elaborate on the stages of our method, and explain how OPH can be learned and combined with the isolation forest without sacrificing either the computational efficiency or detection accuracy. 3) We analyse how the number of trees influences experimental results and demonstrate that our proposed method performs well in detecting anomalies on various types of data. Then, we provide guiding insight on how parameters of our method influence the prediction accuracy.

3.2 Related Work

Much work on anomaly detection has been done in recent years, including applied field research, general technology, formal methods, etc. This chapter will delve into the use

of ensemble methods for anomaly detection, and demonstrate how the hashing scheme can be effectively applied to this task. We will provide an overview of the research work closely relevant to our approach, including the most effective methods used for anomaly detection and the hashing scheme.

The main idea behind the ensemble-based anomaly detection methods is to improve the efficiency of detecting anomalies in big data environments and mitigate the influence of errors caused by individual judgment. In the first place, Lazarevic and Kumar [96] proposed the feature bagging scheme, which is the origin to the subspace ensemble method. This method combines the anomaly scores, generated from lots of subspaces of the data instances, to calculate the average score as the final result. Then, Zimek et al. [266] adopted the subsampling-based method to speed up the ensemble anomaly detection. In this approach, the feature bagging scheme is extended to a big data environment, where the performance in runtime and accuracy is better than that in the traditional feature bagging method. Furthermore, Aggarwal et al. [5, 2] have exploited that the subsampling method is useful to reduce the processing time for ensemble-based anomaly detection, through fixing the size of the subsample. But this method is still time-consuming in a big data environment since each data instance should be tested in the evaluation stage.

Based on the theory of subsampling, Liu et al. [112] first proposed an isolation-based ensemble anomaly detection framework named iForest, which isolates anomalous and normal data instances to different branches in a tree. The anomaly scores of data instances are determined by the height of the node where the data is located. The core idea of iForest is to identify the anomalies in shorter path lengths. This method is really fast for anomaly detection in big data environments since it combines the sampling scheme to reduce the data size and the random hashing scheme to speed up the projection. However, Zhang et al. [249] proved that iForest suffers from incapability in detecting complicated anomalies such as axis-parallel anomalies. This limits the applicability of iForest to detect global and clustered anomalies. Although Liu et al. proposed another anomaly detection method called SCiForest to detect clustered anomalies, it is limited to some specific applications and does not perform well on other

big size datasets [111, 249]. So Zhang et al. [249] proposed a generic framework called LSHiForest, which combines the locality sensitive hashing scheme with isolation forest technique. Due to the wide range of LSH family, LSHiForest can be customised and applied to many cases, which broadens the applicability of LSHiForest. Simultaneously, the processing time of LSHiForest is very short. Besides, Bandaragoda et al. [13] combined the traditional anomaly detection method with iForest scheme to construct iNNE (isolation using Nearest Neighbour Ensemble) framework, which does not exhibit better performance than the hashing-based isolation forest framework.

However, LSH, as a data-independent hashing scheme, has been proven to be weaker than data-dependent hashing methods [208]. In the data-dependent hashing scheme, most algorithms aim to learn the best hash functions to map data instances to the hamming space, through keeping the similarities of data instances between two spaces. Liu et al. [113] proposed a deep supervised hashing for searching semantically similar images. This approach aims to learn the compact similarities of a pair of images computed from the original space and the Hamming space. Then, Zhao et al. [254] proposed a new hashing scheme called deep semantic ranking-based hashing, which learns the deep hash functions through the multilevel similarity information provided by the image ranking list. This is one kind of multiwise similarity preserving scheme that constructs the loss function by minimizing the loss of the similarity orders over more than two items [209]. However, the similarities between the original and Hamming spaces are ambiguous. Thus, the algorithms that focus on pursuing effective space partitioning without explicit similarity preserving were proposed. Typically, random maximum margin hashing trains purely random splits of the data to generate the independent hash functions [254]. This approach showed excellent performance in Kernel space. In learning to hash technique, there are hashing algorithms that exploit quantization information to improve performance. Wang et al. [208] proposed a supervised quantization scheme that aims to minimise the quantization error to find the closest images.

Although many learning to hash methods have been proposed, there is still a lack of a good approach to combining learning to hash schemes with iForest like LSHiForest.

Zhang et al. [249] have leveraged the LSH structure as the core part of LSHiForest, which shows pretty good effectiveness and efficiency for anomaly detection in big data environments. That means learning to hash structure can imitate LSH pattern to be leveraged as the core role for anomaly detection, where this structure may have better performance than LSHiForest. Besides, Wang et al. [208] proposed a new learning to hash based method called order preserving hashing (OPH), which shows better performance on nearest neighbour search compared to other state-of-the-art methods. The learning process of OPH is representative of minimizing the penalizing misalignment rate to keep the order of data instances between the original space and the Hamming space. But whether OPH or other learning to hash methods can be designed for anomaly detection is still under research.

3.3 Preliminaries and Formulation

Our generic method uses an extended OPH scheme to learn hash code and integrate it into isolation forest to isolate anomalies quickly. Herein, we formulate all the parameters and objects to shape the method in this section. The symbols and notations used throughout this chapter are described in Table 3.1.

First, given a dataset $\boldsymbol{X} = \{\boldsymbol{x_1}, \ldots, \boldsymbol{x_N}\}$ with $\boldsymbol{x} \in \mathbb{R}^d$, we hash each point $\boldsymbol{x}$ into a m-dimensional binary vector $\boldsymbol{v}$ through a hash function $f : \mathbb{R}^d \rightarrow \mathbb{H}^m$, where $\mathbb{H}^m$ denotes the Hamming space with the hamming distance $d_H(\cdot)$ and space dimension m.

In Order Preserving Hashing, "order alignment" and "bucket balance" are used to guarantee the matching attributes of the ordered lists computed from the hamming space and the original space. "order alignment" is separated into "point order alignment" and "category order alignment" [208]. For example, given a query point $\boldsymbol{x}$, the similarity order of each point in the hamming space is $\boldsymbol{O^H} = (\boldsymbol{a_1}, \ldots, \boldsymbol{a_N})$, and the order in the original space is $\boldsymbol{O^E} = (\boldsymbol{b_1}, \ldots, \boldsymbol{b_N})$. Here. the original space is commonly referred to Euclidean space, which is ubiquitous in real life. The points are expected to satisfy the rule of "point order alignment", which means $\boldsymbol{a_j} = \boldsymbol{b_j}$,

TABLE 3.1: Symbols and notations

Symbol	Meaning
X	the set of data
N	Size of dataset, $N = \|X\|$
x	a data instance
H	expected height of a tree
M	the number of dimensions
$h(\cdot)$	hash function
P_{th}	path length
L	limit of the tree height
t	the number of tree
ψ	Subsample size, $\psi \in X$
a	Number of anomalous data
b_r	Number of the average branching factor of the root node
W	projection metric
η	penalty parameter
d_H	Hamming distance
O^H	similarity order in Hamming distance
C_m^H	the set of data instances whose Hamming distance is smaller than m
S_i^H	the union set of C_m^H, $0 =< m <= i$
$P(h(\cdot); X)$	penalizing misalignment rate

$\forall j(1 <= j <= N)$. Besides, there are a limited number $(M + 1)$ of distance values (integer values) for data in hamming space $\boldsymbol{H^m}$, so the order $\boldsymbol{O^H}$ has $M + 1$ categories. These categories are defined as $(\boldsymbol{C_0^H}, \ldots, \boldsymbol{C_M^H})$, where $\boldsymbol{C_m^H} = \{\boldsymbol{a_1^m}, \ldots, \boldsymbol{a_{Nm}^m}\}$, $0 <= m <= M$. Accordingly, $\boldsymbol{O^E}$ is divided into $M + 1$ categories, $(\boldsymbol{C_0^E}, \ldots, \boldsymbol{C_M^E})$. The categories are expected to satisfy the rule of "categories order alignment", which means $\boldsymbol{CqH} = \boldsymbol{CqE}$, $\forall q(1 <= q <= M)$.

"Bucket balance" means that the points are uniformly distributed in all hash buckets and is helpful to guarantee search efficiency. Unlike nearest neighbour search, we

need to isolate anomalies in the individual bucket in anomaly detection, so "bucket unbalance" is used to isolate anomalous attributes of the ordered lists.

Finally, all the reference data points are assigned to one of $(M+1)$ target categories. To penalise the misalignment rate, we transform the multi-class classification problem to M binary-class classification problems [208]. Specifically, the target categories are partitioned into M sets of binary super-categories, $(\boldsymbol{S_0^H}, \widetilde{\boldsymbol{S_0^H}}), \ldots, (\boldsymbol{S_{M-1}^H}, \widetilde{\boldsymbol{S_{M-1}^H}})$, where $\boldsymbol{S_i^H} = \cup_{j=0}^{i} \boldsymbol{C_j^H}$ and $\widetilde{\boldsymbol{S_i^H}} = \boldsymbol{X} - \boldsymbol{S_i^H}$, $0 <= i <= M$. Each pair of super-categories $(\boldsymbol{S_i^H}, \widetilde{\boldsymbol{S_i^H}})$ forms a binary-class classification problem. Herein, the super-categories $(\boldsymbol{S_i^E}, \widetilde{\boldsymbol{S_i^E}})$ in the original space is similarly built. In OPH, the function of penalizing misalignment rate is defined as:

$$\begin{aligned} P(h(\cdot); \boldsymbol{X}) &= \sum_{n=1}^{N} \sum_{i=0}^{M-1} P(h(\cdot); \boldsymbol{x_n}, i) \\ &= \sum_{n=1}^{N} \sum_{i=0}^{M-1} (|\boldsymbol{S_i^E} - \boldsymbol{S_i^H}| + \gamma |\widetilde{\boldsymbol{S_i^E}} - \widetilde{\boldsymbol{S_i^H}}|) \\ &= \sum_{n=1}^{N} \sum_{i=0}^{M-1} (|\boldsymbol{S_i^E} - \boldsymbol{S_i^H}| + \gamma |\boldsymbol{S_i^H} - \boldsymbol{S_i^E}|), \end{aligned} \tag{3.1}$$

where $|\boldsymbol{S_i^E} - \boldsymbol{S_i^H}|$ are the errors for the binary super-categories between Euclidean space and Hamming space, γ is the trade-off parameter to balance the penalties for these binary super-categories.

According to the aforementioned merits of OPH and isolation forest based anomaly detection scheme, an interesting question occurs: Can we further utilise the speciality of OPH to construct robust and scalable isolation forest for anomaly detection? The answer is *yes*. Concretely, we combine the OPH scheme with isolation forest to build the baseline for our method. Then, we design a fast two-step learning process for the OPH scheme to enhance anomaly detection robustness and accuracy.

3.4 The Proposed Method: OPHiForest

In this section, we propose the isolation forest based method OPHiForest for anomaly detection and explain three stages of our method. All the content is separated into four parts: Section 3.4.1 presents a general description of the overall method; Section 3.4.2 makes a detailed analysis of the core part in our method and gives the formulas;

Section 3.4.3 describes the concrete stages of the method and constructs algorithms; In Section 3.4.4, we compute the algorithm complexity of the method and compare it with LSHiForest.

3.4.1 Model-based Representation for The Method

The core hashing scheme in our method is to learn hash function with order preservation to minimise the order alignment error between original data instances and projected data instances. Then, the hash codes of data instances are used to construct the isolation trees, since anomalies can be quickly separated from the others through recursively splitting the dataset [110]. OPHiForest method is separated into three stages: pre-training stage, training stage and evaluation stage, which are shown in Figure 3.1.

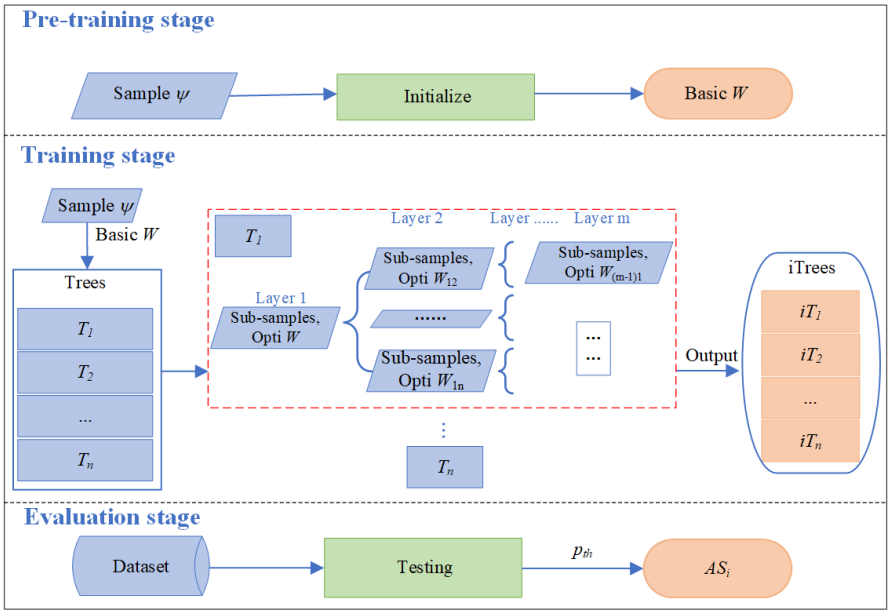


Figure 3.1: The stages of OPHiForest method. There are three stages in OPHiForest, including the pre-training stage, the training stage and the evaluation stage.

From Figure 3.1, we can find that OPHiForest method is organised into three stages. Specifically, we initialise projection matrix $\boldsymbol{W}$ (consisting of several projection vectors), and make a sample ψ to optimise $\boldsymbol{W}$ to get the basic $\boldsymbol{W}$ for the next stage. In the training stage, we use the sample and $\boldsymbol{W}$ to train the data instances in the first layer

of a tree. After training, this sample is separated into several sub-samples and the best projection matrix $\boldsymbol{W}$, which minimises the penalizing misalignment rate, is kept in layer 1. Then, each sub-sample as the input of layer 2 is used to train the best projection matrix $\boldsymbol{W_2}$ and get their new sub-samples. The training stage will stop when all the sub-samples in the leaf have no new sub-sample. Finally, based on the OPHiTrees, we can test the dataset and recursively calculate the path length p_{th}, which is used to calculate the anomaly score AS.

3.4.2 Analysis of Two Steps of Learning

The hashing scheme aims to map similar objects into the same hash code efficiently and accurately so that the anomalous objects can be separated from the normal objects. First, the hash function is formally defined as $y = h(x)$, where it can be linear projection, kernel function, spherical function, etc. In this part, we use the widely-used linear projection:

$$\begin{aligned} y = h(x) &= sign(\boldsymbol{W^T x} + \boldsymbol{B}) \\ &= [sign(\boldsymbol{w_1^T x} + b_1) \dots sign(\boldsymbol{w_m^T x} + b_m)]T, \end{aligned} \tag{3.2}$$

where $sign(x) = 1$ if $x > 0$ and $sign(x) = 0$ otherwise, $\boldsymbol{W}$ is the projection matrix of size $d \times m$, each $\boldsymbol{w_k} \in \mathbb{R}^d$ is a projection vector and satisfy the condition $\boldsymbol{w_k^T w_k} = 1$, and $\boldsymbol{B}$ is the bias variable.

In this chapter, we design a fast two-step learning process to learn the best hash functions with order preservation, which can most accurately map anomalous and normal objects into different hash codes. This process is separated into two learning parts: one is to learn the basic hash functions in the pre-training stage, called coarse-learning; another is to learn the optimised hash functions in the training stage, called fine-learning. The key of both stages is utilizing the penalizing misalignment rate to minimise the projection gap between Euclidean space and Hamming space.

A. Coarse-learning

To learn the basic hash functions is to learn the basic projection matrix $\boldsymbol{W}$. Herein, the function of penalizing misalignment rate should be simplified by $\boldsymbol{S_{ni}^H}$, which is

formed by the points ($\boldsymbol{x'} \in \boldsymbol{S^E_{ni}}$) whose Hamming distances to the point $\boldsymbol{x_n}$ are not larger than i. $|\boldsymbol{S^E_{ni}} - \boldsymbol{S^H_{ni}}|$ means the gap that the points $\boldsymbol{x'} \in \boldsymbol{S^E_{ni}}$ but $\boldsymbol{x'} \notin \boldsymbol{S^H_{ni}}$, so $d_H(\boldsymbol{h}(\boldsymbol{x_n}), \boldsymbol{h}(\boldsymbol{x'})) > i$. Accordingly, $|\boldsymbol{S^H_{ni}} - \boldsymbol{S^E_{ni}}|$ means the gap that the points $\boldsymbol{x'} \in \boldsymbol{S^H_{ni}}$ but $\boldsymbol{x'} \notin \boldsymbol{S^E_{ni}}$, so it is easy to get the relationship that $d_H(\boldsymbol{h}(\boldsymbol{x_n}), \boldsymbol{h}(\boldsymbol{x'})) <= i \Leftrightarrow i + 1 - d_H(\boldsymbol{h}(\boldsymbol{x_n}), \boldsymbol{h}(\boldsymbol{x'})) > 0$. Combining the solving scheme of $|\boldsymbol{S^E_{ni}} - \boldsymbol{S^H_{ni}}|$ and $|\boldsymbol{S^H_{ni}} - \boldsymbol{S^E_{ni}}|$, the objective function of our scheme can be calculated by:

$$\begin{aligned} P(h(\cdot); \boldsymbol{X}) = & \sum_{\boldsymbol{x'} \in \boldsymbol{S^E_{ni}}} sign(d_H(\boldsymbol{h}(\boldsymbol{x_n}), \boldsymbol{h}(\boldsymbol{x'})) - i) \\ & + \gamma \sum_{\boldsymbol{x'} \notin \boldsymbol{S^E_{ni}}} sign(i + 1 - d_H(\boldsymbol{h}(\boldsymbol{x_n}), \boldsymbol{h}(\boldsymbol{x'}))). \end{aligned} \tag{3.3}$$

In coarse-learning, the basic projection matrix $\boldsymbol{W}$ can be learned by randomly quantifying $\boldsymbol{W}$ to reach the minimum $P(\boldsymbol{W}; \boldsymbol{x_n}, i)$, since it can learn a relatively accurate hash function in a short time and find the direction of function optimization. The time cost of this process is just $\Theta(N)$. Then, we can get the minimum of the objective function:

$$\min P(\boldsymbol{W}; \boldsymbol{x_n}, i) = \begin{cases} P(\boldsymbol{W'}; \boldsymbol{x_n}, i) \text{ if } P(\boldsymbol{W}; \boldsymbol{x_n}, i) > P(\boldsymbol{W'}; \boldsymbol{x_n}, i), \\ P(\boldsymbol{W}; \boldsymbol{x_n}, i) \text{ otherwise.} \end{cases} \tag{3.4}$$

To illustrate the details of coarse-learning, an example is given here. We use the "make blobs" method to generate a 2-dimensional synthetic dataset ($-10 < x < 10, -10 < y < 10$), which contains 1000 normal and 20 anomalous data instances. Besides, the projection matrix $\boldsymbol{W}$ just contains one vector. We set the iteration for learning to 25 times since it can provide a good hash function without wasting too much time in pre-training. The learning process of this example is shown in Figure 3.2.

Figure 3.2 shows the optimization of hash functions. Line L_1 represents the initial hash function, lines L_2, L_3, L_4 and L_5 state the optimization in coarse-learning. So in 25 iterations, the objective function is improved 4 times. According to Equation 3.4, there is a conclusion that $\min P(\boldsymbol{W}; \boldsymbol{x_n}, i) = P(\boldsymbol{W_5}; \boldsymbol{x_n}, i) < P(\boldsymbol{W_4}; \boldsymbol{x_n}, i) < P(\boldsymbol{W_3}; \boldsymbol{x_n}, i) < P(\boldsymbol{W_2}; \boldsymbol{x_n}, i) < P(\boldsymbol{W_1}; \boldsymbol{x_n}, i)$.

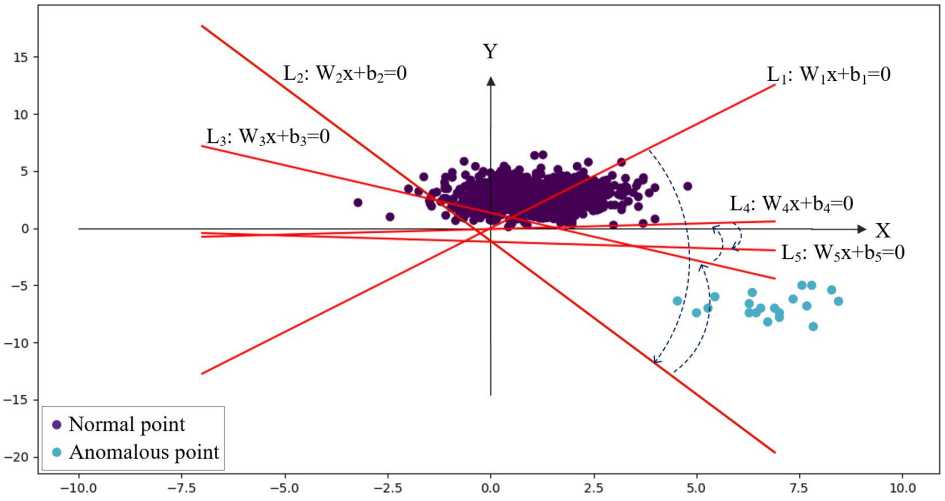


Figure 3.2: An example that illustrates the process of coarse-learning.

B. Fine-learning

We use fine-learning to learn the best hash function by optimizing $\boldsymbol{W}$, obtained from the coarse-learning. In this process, the stochastic gradient descent approach is used to optimise the objective function. Different to OPH, only a single cycle rather than a nested loop is used in stochastic gradient descent of fine-learning, since we have found the optimised direction in coarse-learning. This process can tremendously decrease the learning cost of our method.

We use a Sigmoid function $\phi_\alpha(x) = \frac{1}{1+e^{-\alpha x}}$ to relax the function sign$(\cdot)$ in Equation 3.2. Here, α is the equilibrium factor, which means a larger α results in a better approximation, but a more difficult optimization. After relaxation, we can calculate the overall objective function, as described in OPH:

$$\begin{aligned} \min\ P(\boldsymbol{W}) &= \min \frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{M-1} P(\boldsymbol{W}; \boldsymbol{x_n}, i) \\ &\cong \min \frac{1}{N} \sum_{n=1}^{N} \sum_{i=0}^{M-1} (\sum_{\boldsymbol{x'} \in \boldsymbol{S_{ni}^E}} \phi_{\alpha_1}(d_{sig}(\boldsymbol{x_n}, \boldsymbol{x'}) - i) \\ &+ \gamma \sum_{\boldsymbol{x'} \notin \boldsymbol{S_{ni}^E}} \phi_{\alpha_2}(i + 1 - d_{sig}(\boldsymbol{x_n}, \boldsymbol{x'}))), \\ &s.t.\ \boldsymbol{w_k^T w_k} = 1, \forall 1 <= k <= m\ \&\ k \in \mathbb{N}, \end{aligned} \tag{3.5}$$

where $d_{sig}(\boldsymbol{x_n}, \boldsymbol{x'}) = ||\phi_\alpha(\boldsymbol{W^T x_n}) - \phi_\alpha(\boldsymbol{W^T x'})||_2^2$, α_1 and α_2 are the equilibrium factors.

According to OPH, the above nonlinear constrained optimization problem is transformed into a simpler linear problem by the quadratic penalty method [137], which is

calculated by:

$$\begin{aligned}\min P(\boldsymbol{W}) &\cong \min\frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{M-1}\hat{P}(\boldsymbol{W};\eta)\\ &= \min(\frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{M-1}P(\boldsymbol{W};\boldsymbol{x_n},i)\\ &+ \frac{\eta}{4}\sum_{k=1}^{M}(\boldsymbol{w_k^T w_k}-1)^2),\end{aligned} \tag{3.6}$$

where $\eta > 0$ is the penalty parameter, which means a bigger η results in more severe penalty on constraint violations. Then, we consider a sequence of values η_t with $\eta_t \to \infty$ as $\eta_t + 1 = \tau\eta_t(\tau > 1)$. Taking the derivative of $\hat{P}(\boldsymbol{W};\eta)$, the gradient is defined as:

$$\begin{aligned}\nabla_k\hat{P}_t &= \frac{\partial\hat{P}(\boldsymbol{W};\eta_t)}{\partial w_k}\\ &= \frac{1}{N}\sum_{n=1}^{N}\sum_{i=0}^{M-1}\frac{\partial P(W;\eta_t)}{\partial w_k} + \eta_t(\boldsymbol{w_k^T w_k}-1)\boldsymbol{w_k},\end{aligned} \tag{3.7}$$

where $\frac{\partial P(\boldsymbol{W};\eta_t)}{\partial \boldsymbol{w_k}} = \frac{\partial(\sum_{\boldsymbol{x'}\in\boldsymbol{S_{ni}^E}}\phi_{\alpha_1}(d_{sig}(\boldsymbol{x_n},\boldsymbol{x'})-i)+\gamma\sum_{\boldsymbol{x'}\notin\boldsymbol{S_{ni}^E}}\phi_{\alpha_2}(i+1-d_{sig}(\boldsymbol{x_n},\boldsymbol{x'})))}{\partial\boldsymbol{w_k}}$.

When calculating the penalizing misalignment rate, we need to go through all the N points for each point $\boldsymbol{x_n}$, so the time cost is $O(N^2)$. In a big data environment, the size of the dataset $\boldsymbol{X}$ is huge, resulting in a big-time cost. Thus, sampling technology is used in our method. In the training stage, we randomly select a small subset L to build a tree. Here, $L \subseteqq \boldsymbol{X}$, and $|L| = L' << N$.

In OPH, "active set" is introduced to reduce the points in $\boldsymbol{S_{ni}^E}$, so that the computing cost can be saved. Herein, we optimise the points in $\boldsymbol{S_{ni}^E}$ to $(\boldsymbol{S_{ni}^E} - \boldsymbol{S_{n(i-\bar{i}_1)}^H})$. Accordingly, we can eliminate some irrelevant points from the range $\boldsymbol{x'} \notin \boldsymbol{S_{ni}^E}$ to get a smaller range $\boldsymbol{x'} \in \boldsymbol{S_{n(i+\bar{i}_2)}^H} - \boldsymbol{S_{ni}^E}$. In our implementation, $\bar{i}_1 = \bar{i}_2 = 1$. Finally, the gradient descent function can be simplified as:

$$\nabla_k\hat{P}_t = \frac{1}{L'}\sum_{n=1}^{L'}\sum_{i=0}^{M-1}\frac{\partial P(\boldsymbol{W};\eta_t)}{\partial\boldsymbol{w_k}} + \eta_t(\boldsymbol{w_k^T w_k}-1)\boldsymbol{w_k}, \tag{3.8}$$

where $\partial P(\boldsymbol{W};\eta_t)/\partial\boldsymbol{w_k} = \partial(\sum_{\boldsymbol{x'}\in(\boldsymbol{S_{ni}^E}-\boldsymbol{S_{n(i-1)}^H})}\phi_{\alpha_1}(d_{sig}(\boldsymbol{x_n},\boldsymbol{x'})-i)+\gamma\sum_{\boldsymbol{x'}\in(\boldsymbol{S_{n(i+1)}^H}-\boldsymbol{S_{ni}^E})}\phi_{\alpha_2}(i+1-d_{sig}(\boldsymbol{x_n},\boldsymbol{x'})))/\partial\boldsymbol{w_k}$.

The core idea of the stochastic gradient descent algorithm is to perform iteratively to determine the parameters. So $\boldsymbol{w_k}$ is iterated to be the optimal parameter as:

$$\boldsymbol{w_k} \leftarrow \boldsymbol{w_k} - \rho(\frac{1}{L'}\sum_{n=1}^{L'}\sum_{i=0}^{M-1}\frac{\partial P(\boldsymbol{W};\eta_t)}{\partial\boldsymbol{w_k}} + \eta_t(\boldsymbol{w_k^T w_k}-1)\boldsymbol{w_k}). \tag{3.9}$$

To follow the same example in coarse-learning, we continue to optimise the objective function $P(\boldsymbol{W}; \boldsymbol{x_n}, i)$ through fine-learning. The detailed learning process is shown in Figure 3.3.

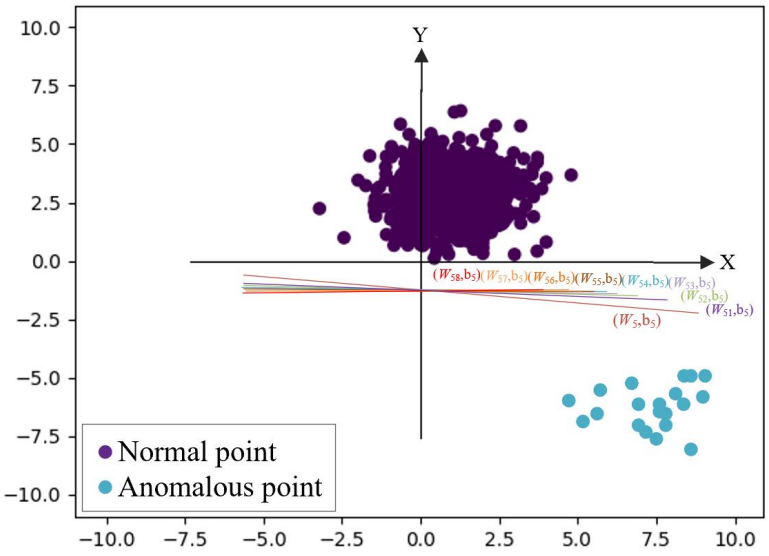


Figure 3.3: An example that illustrates the process of fine-learning.

Figure 3.3 shows that the iteration is performed 8 times, and each iteration makes the penalizing misalignment rate smaller, which means a better hash projection. So we can get the relationship of all penalizing misalignment rate is $P(\boldsymbol{W_{58}}; \boldsymbol{x_n}, i) < P(\boldsymbol{W_{57}}; \boldsymbol{x_n}, i) < P(\boldsymbol{W_{56}}; \boldsymbol{x_n}, i) < P(\boldsymbol{W_{55}}; \boldsymbol{x_n}, i) < P(\boldsymbol{W_{54}}; \boldsymbol{x_n}, i) < P(\boldsymbol{W_{53}}; \boldsymbol{x_n}, i) < P(\boldsymbol{W_{52}}; \boldsymbol{x_n}, i) < P(\boldsymbol{W_{51}}; \boldsymbol{x_n}, i) < P(\boldsymbol{W_5}; \boldsymbol{x_n}, i)$. The process of optimization follows the rule of stochastic gradient descent, which results in a minimum penalizing misalignment rate $P(\boldsymbol{W_{58}}; \boldsymbol{x_n}, i)$.

In the fine-learning scheme, we mainly use stochastic gradient descent to optimise W to get the best projection function. But different to traditional stochastic gradient descent, we do not need double nested iteration, since we have made coarse-learning before to guarantee the accuracy. The detailed algorithm is described in the next subsection.

3.4.3 Three Stages of OPHiForest

OPHiForest is an ensemble method that uses the values of hashing to generate multi-way isolation trees. Different from iForest, the isolation trees in OPHiForest are not random, so they can isolate anomaly and normal data more accurately. In this part, a detailed description of every stage in OPHiForest is carried out.

A. Pre-training stage

In this stage, we use coarse-learning to learn the basic hash function, which guarantees a certain accuracy to map different data instances into different hash indices. This stage can tremendously save time for the training stage, since the coarse-learning is fast with the only time cost $\Theta(N)$. In iForest scheme, Liu et al. [112] have proved that one can acquire a "sufficiently large and rich" hash family from a relatively small subset, which contains as few as 100 data instances. Thus, we adopt a fixed size of 100 elements ($\psi = 100$) as the subset for generating our hash family, which also shows the perfect performance on our experiment results. Then, we come to train the basic hash function in Algorithm 1.

Algorithm 1 Train The Basic Hash Function

Input: $\boldsymbol{X}$-Input Data, V-Dimension of the hash value, O_{ite}-Iteration times.

Output: $\boldsymbol{W}$-Projection matrix.

1: $\psi \leftarrow$ subsampling ($\boldsymbol{X}$);
2: **for** $i \leftarrow$ to V **do**
3: initialise a $\boldsymbol{W}$;
4: **end for**
5: calculate the penalizing misalignment rate $P(\boldsymbol{W}; \boldsymbol{x_n}, i)$;
6: **for** $j \leftarrow$ to O_{ite} **do**
7: initialise a $\boldsymbol{W}'$;
8: get the minimum $P(\boldsymbol{W}; \boldsymbol{x_n}, i)$ according to Equation 3.4.
9: **end for**
10: Return $\boldsymbol{W}$.

In lines 2 and 3, we initialise the projection matrix $\boldsymbol{W}$ according to the dimension

of the hash value. Here, the dimension of the hash value can be any number, but the bigger dimension means better accuracy and more time cost. In lines 6-9, the minimum penalizing misalignment rate is updated by $\boldsymbol{W}$. After iteration, we can get a good hash function to distinguish anomalous and normal objects.

B. Training stage

In the training stage, we will train the isolation trees and use these trees to construct the isolation forest. Each layer of a tree uses different hash functions to isolate data instances, this is different to the random separation scheme of traditional iForest. Next, three algorithms are used to define the details of the training stage.

First, OPH Forest is constructed in Algorithm 2. As shown in Line 2 of Algorithm 2, we use the same variable subsampling approach proposed by Aggarwal [4] to get the training data in a tree. To make the tree height more diverse, Zhang et al. [249] adopted a more reasonable subsample size of [64, 1,024] rather than [50, 1,000]. This is based on the assumption that tree height could be estimated as $\log_2 \psi$ (ψ is the number of data instances). However, we find that Szpankowski et al. proposed a more reasonable estimation of average tree height as $2\log_2 \psi + 0.833$, which generates the subsampling size distribution to [68, 1085].

Algorithm 2 Constructing OPH Forest

Input: $\boldsymbol{X}$-Input Data, $\boldsymbol{W}$-Projection matrix, L-Height Limit, t-Number of Trees.

Output: OPH Forest.

1: **for** $i \leftarrow$ to t **do**
2: $\quad \psi_i \leftarrow$ subsampling $(\boldsymbol{X})$;
3: $\quad L_i \leftarrow$ height _ limit $(|\boldsymbol{S_i}|)$;
4: $\quad \boldsymbol{W_i} \leftarrow$ use the basic projection matrix $\boldsymbol{W}$;
5: $\quad T_i \leftarrow$ OPHiTree $(\boldsymbol{S_i}, \mathcal{H}, L_i, 0)$;
6: **end forreturn** $\{T_i | i \in [1, t]\}$;

According to Szpankowski [189], assuming each data point is independent, the

expected height of a tree will follow:

$$H \approx \frac{2}{br} \ln n + \frac{\lambda - \ln 2}{br} + 1, \tag{3.10}$$

where br equals the number of branches in a tree, and $\lambda = 0.577$ is Euler constant. In our method, the OPH isolation tree will be constructed as a multi-way tree, which has different branch factors in different layers. Next, we will illustrate how to build an OPH isolation tree in Algorithm 3.

Algorithm 3 Building OPH Isolation Tree

Input: ψ-Sample Data, $\boldsymbol{W}$-Basic projection matrix, L-Height Limit, I-Current Height.

Output: T-OPH Tree.

1: **if** $|\psi| = 0$ **then**
2: **return** NULL;
3: **else if** $I > L$ **then**
4: **return** OPHNode { Size $\leftarrow |\psi|$, lChild $\leftarrow$ NULL, rChild $\leftarrow$ NULL, hash_index $\leftarrow I$ };
5: **else**
6: $\boldsymbol{W}' \leftarrow$ use $\boldsymbol{W}$ to do fine-grained learning of the hash function through Algorithm 4;
7: $\{V_1 : \psi_1, \cdots, V_{br} : \psi_{br}\} \leftarrow$ line_projection_split $(\psi, \boldsymbol{W}'_{\boldsymbol{I}_i})$, $\boldsymbol{W}'_{\boldsymbol{I}_i} \in \boldsymbol{W}'$;
8: **for** $i, 1 <= i <= br$ **do**
9: $Tree_i \leftarrow$ OPHiTree $(\psi_i, \boldsymbol{W}'_{\boldsymbol{I}_i}, L, I + 1)$;
10: Tree $\leftarrow \cup$ Tree $\{V_i : Tree_i\}$;
11: **end for**
12: **return** OPHNode{ Size $\leftarrow |\psi|$, Children $\leftarrow$ Tree, hash_index $\leftarrow I$ }
13: **end if**

As shown in Algorithm 3, we adopt a recursive method to build the OPH tree, where each node has several sub-trees. In lines 6-10 of Algorithm 3, the sample is separated into non-overlapping subsets associated with hash keys, which are generated

by hash functions. These hash functions are not random, on the contrary, they are learned by the order preservation scheme. We will learn a hash function in each node of the tree to isolate anomalies more accurately. A detailed fine-learning process is shown in Algorithm 4.

Algorithm 4 Learn The OPH Hash Function

Input: $\boldsymbol{D}$-Input Data, V-Dimension of the hash value, I_{ite}-Iteration times, $\boldsymbol{W}$-Basic projection matrix.

Output: $\boldsymbol{W}'$-Projection matrix after fine-learning.

1: Initialise η, γ, α_1, α_1;
2: $\boldsymbol{x_n} \leftarrow$ choose a reference point;
3: Construct $\boldsymbol{S_{ni}^H}$ according to OPH scheme;
4: **for** $j \leftarrow$ to I_{ite} **do**
5: Calculate $\boldsymbol{S_{ni}^E}$ and $\boldsymbol{S_{ni}^H}$ on dataset $\boldsymbol{D}$;
6: min $P(\boldsymbol{W}; \boldsymbol{x_n}, i) \leftarrow$ according to Equation 3.8;
7: $\boldsymbol{w_k} \leftarrow$ according to Equation 3.9;
8: **end forreturn** $\boldsymbol{W}'$;

In Algorithm 4, we realise the core process of fine-learning. The difficulty in fine-learning is to recursively update the minimum penalizing misalignment rate, which guarantees the accuracy of mapping the objects from the original space to the Hamming space. Lines 4-7 in Algorithm 4 represent the steps to get the best hash function through the stochastic gradient descent approach.

C. Evaluation stage

In this stage, we will calculate the anomaly score of each data instance. The bigger anomaly score represents the greater possibility that the data instance is anomalous. First, the height of the tree node can be calculated recursively through either the left child or right child, and stopping at the node that has no children or exceeds the limited height. Then, the anomaly score is obtained by the normalization function and node height. The algorithm of getting path length and computing anomaly score is similar to that discussed in LSHiForest [249], so the final anomaly score is calculated

by: $AS_x = \frac{1}{t}\sum_{i=1}^{t} 2^{-\frac{P_{th}}{H_x}}$.

3.4.4 Analysis of Algorithm Complexity

Since our method is a learning-based approach, it uses big time in the training stage to build the basic model. However, it is fast in the evaluation stage. These two features are the common sense for iForest-based approach. Next, we will make an analysis and comparison of algorithm complexity between OPHiForest and LSHiForest.

In the pre-training stage, the calculation of the objective function is linear. We just need to compute the objective function of each sample. So the time cost is only $\Theta(\psi)$, as analysed in Section 3.4.2.

In the training stage, the algorithm complexity of OPHiForest is determined by ensemble elements, subsample size, distance computation time, tree height and average branching factor. The average complexity is $\Theta\left(H \cdot br \cdot \psi^2 ta \log_{b_r} \psi\right)$ while that of LSHiForest is $\Theta\left(\psi ta \log_{b_r} \psi\right)$. Our method has higher time complexity in the training stage compared to LSHiForest, since the learning scheme is time-consuming and used in each node of a tree. But in real-world applications, we just consider the execution time rather than the time of training a model.

Then, we calculate the algorithm complexity in the evaluation stage by the height of a tree, ensemble elements, dataset size, and distance computation time. The average complexity of our method is similar to that of LSHiForest, which holds the complexity of $\Theta\left(tNa \log_{b_r} \psi\right)$.

Considering the practical applications, the algorithm complexity is associated with the execution time (in the evaluation stage), so our method is similar in time complexity compared with LSHiForest.

3.5 Empirical Evaluation

In this section, extensive experiments are conducted to verify the robustness and effectiveness of our method against state-of-the-art methods. Firstly, we evaluate how

the number of trees affects the detection quality. Then, we generate a group of synthetic datasets to compare all the methods in the ability to detect local anomalies. In the third part, we compare the performance of OPHiForest with our baseline and some ensemble-based anomaly detection methods in general real-world datasets. Specifically, we evaluate our approach with iForest, SCiForest, LSHiForest, iNNE [13], average k-NN distance ensemble method (EnKNN) [5] and LOF ensemble method (EnLOF) [266]. Finally, Section 3.5.4 contrasts the detection behaviour of OPHiForest and SCiForest in detecting the anomalies of clustered datasets.

As has been used extensively in many other works, including [92, 229, 249], we also use the Area Under receiver operating characteristic Curve (AUC) and execution time as the performance measures. The higher AUC means better performance. To exclude the contingency, we run all the experiments 20 times and take the arithmetic average as the final result.

In our empirical evaluation, the number of ensemble members t is set to 100, the same as that used in the existing isolation forest based methods. The effect of the number of trees is discussed in Section 3.5.1. To simplify the calculation, we use the subsampling technique discussed in LSHiForest for all methods and the sample size varies within the range [64, 1024]. In the pre-training stage, we use the subsample as 100 to train the basic hash function. The subsample here does not need to be large, or it is time-consuming.

All experiments are executed on a personal computer with Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz and 16 GB main memory.

3.5.1 Effects of the Number of Trees

Most isolation forest based anomaly detection methods use 100 isolation trees to construct their ensemble framework since 100 ensemble members have been proven to be effective and stable. Our method is also based on the iForest scheme but learns the hash function with order preservation. This learning scheme decreases the randomness in constructing isolation trees, so we study how the number of isolation trees affects the detection quality and compare that with the iForest scheme. In the experiment,

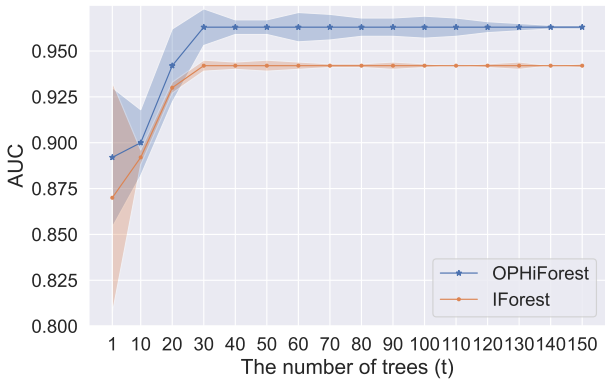
we detect the AUC results of two methods under the same number of isolation trees (1-150) in datasets "Breastw", "Sensor" and "Mfeat" (from UCI Machine Learning Repository), since the detection accuracy of two methods is similar in these datasets. Each experiment is run 20 times to obtain the average AUC and the standard deviation. The results are shown in Figure 3.4.

Figure 3.4 shows the effects of the number of trees on iForest and OPHiForest under three datasets. It can be observed from Figure 3.4 that the AUC result has a sharp growth with the number of isolation trees increasing in both methods. In Figure 3.4(a), the AUC of iForest and OPHiForest reach the peak when the number of trees is around 30 and remains unchanged after the number of trees is larger than 30. Besides, we can find that the AUC result is stable when the number of trees exceeds 30, with a small standard deviation (Std$<$ 0.02). In Figure 3.4(b), the AUC of both methods remains stable with $t = 60$, while the AUC of both methods reaches the peak with $t = 70$ in Figure 3.4(c). We can conclude that the effect of the number of trees on iForest is similar to OPHiForest. Fewer isolation trees in constructing the iForest can reduce the consumption of resources and time, so we always use the least number of trees corresponding to the peak AUC to build iForest.

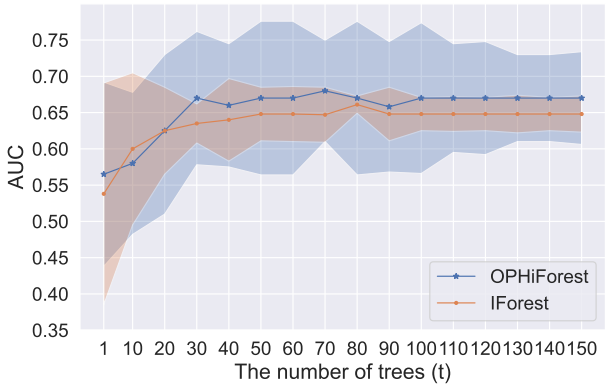
3.5.2 Detection of Local Anomalies

Local anomalies are more difficult to detect than global anomalies since local anomalies are very close to normal objects. Thus, we make a feasibility analysis of detecting Local Anomalies for all compared methods. Specifically, we compare the ability to detect local anomalies of iNNE, EnKNN, EnLOF, iForest, SCiForest, LSHiForest and our approach. There are many instantiations in LSHiForest. Our method focuses on the projection from Euclidean space to Hamming space, so we compare the corresponding instantiation (L2SH) in LSHiForest, which realises the hashing map scheme of Euclidean space. Besides, the instantiation (KLSH) in LSHiForest is specialised in detecting local anomalies, so we also make a comparison with our approach.

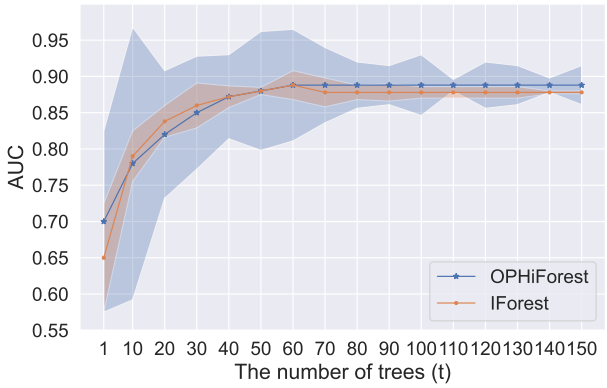
The single-anomaly synthetic dataset in [13], as shown in Figure 3.5(a), is reused in



(a) Breastw



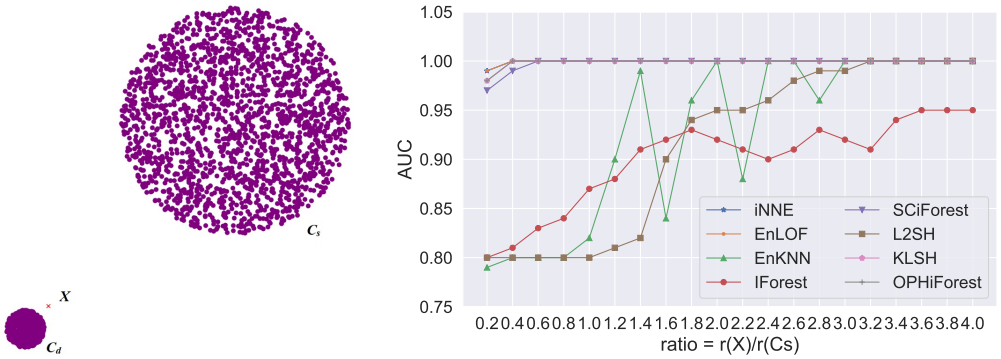
(b) Sensor



(c) Mfeat

Figure 3.4: The effects of the number of trees. It can be observed that the AUC results have a sharp growth with the number of isolation trees increasing and keep stable when the number of trees exceeds 30.

the comparison. The dataset contains 2500 normal data instances and 1 anomaly. The normal instances are separated into two clusters $\boldsymbol{C_s}$ and $\boldsymbol{C_d}$, consisting of 2000 and 500 data instances, respectively. Let $r(\boldsymbol{C_s})$ denote the maximum nearest neighbour distance in $\boldsymbol{C_s}$, and $r(\boldsymbol{X})$ denote the distance from anomaly $\boldsymbol{X}$ to the boundary of $r(\boldsymbol{C_s})$. To signify the locality of an anomaly, the ratio= $r(\boldsymbol{X})/r(\boldsymbol{C_s})$ is introduced to indicate that $\boldsymbol{X}$ is a local anomaly if ratio<= 1. The AUC of different methods is conducted in different ratios (from 0.2-4), as shown in Figure 3.5(b).



(a) Single-anomaly dataset (b) AUC w.r.t. ratio

FIGURE 3.5: Capability of detecting local anomalies. Five methods, including iNNE, EnLOF, SCiForest, KLSH and OPHiForest, can detect local anomalies among all eight methods.

From Figure 3.5(b), it can be observed that five methods, including iNNE, EnLOF, SCiForest, KLSH and OPHiForest, can detect local anomaly (ratio $<$ 1). Besides, the AUC of all methods except for iForest reaches 1.00 with the ratio increasing. Considering the efficiency of the above methods, iNNE, EnLOF and KLSH have very high computational costs. SCiForest and our method use a learning-based hashing scheme, producing more computational cost than iForest, but much less than iNNE, EnLOF and KLSH. In the next section, we will make extensive experiments on real-world datasets to illustrate the accuracy and execution time of all the methods.

3.5.3 Comparison with General Ensemble-Based Methods

In this part, we evaluate the method based on a variety of real-world datasets, which have been widely used in the evaluation of state-of-the-art methods. All the datasets are available online at the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets.php). It is necessary to process the datasets gathered from the UCI Machine Learning Repository and remove the missing data. Then, the description of real-world datasets is shown in Table 3.2.

Table 3.2: Real-world datasets used in experiments

#	Name	n	m	Outlier vs. Inlier Labels	Rate
1	Glass	214	9	class 6 vs. others	4.2%
2	Ionosphere	351	34	"bad" vs. "good"	35.9%
3	Breastw	683	9	"malignant" vs. "benign"	35.0%
4	Pima	768	8	"positive" vs. "negative"	34.9%
5	Vowel	1,456	12	speaker 1 vs. 6, 7, 8	3.43%
6	Waveform	3,505	21	class 0 vs. others	4.62%
7	Wine	5,318	11	class 3, 9, 4 vs. others	4.53%
8	Occupancy	6,841	5	"1" vs. "0"	6.24%
9	Skin	9,521	3	"1" vs. "2"	3.67%
10	Satellite	6,435	36	class 2, 4, 5 vs. others	31.6%
11	Shuttle	49,097	9	class 2, 3, 5, 6, 7 vs. class 1	7.15%
12	Arrhythmia	452	274	class 3, 4, 5, 7, 8, 9, 14, 15 vs. others	14.6%
13	Mfeat	666	649	class 4, 8, 9 vs. others	9.91%
14	Sensor	5,456	24	"Slight-Left-Turn" vs. others	6.01%

In Table 3.2, there are 14 datasets collected from different application areas. n denotes the size of the dataset, m denotes the dimension of the dataset, and 'Rate' represents the proportion of anomalous data in total data.

In the implementation, we initialise projection matrix $\boldsymbol{W}$ of the hash functions in coarse-learning to a random variable, which is sampled from a standard Gaussian

distribution $\mathbb{R}(0,1)$ and normalizing the projection vector to have unit length. When using the stochastic gradient descent, we need to initialise the parameters by $\eta = 10$, γ is a random variable sampled from a standard Gaussian distribution $R(0,1)$, α is selected according to the range of $\boldsymbol{x}$. Specifically, if $\max_{\boldsymbol{x}\in \boldsymbol{S^E_{Ni}}}\{|\boldsymbol{x}|\} = \xi$, $\alpha = 2log(\varepsilon^{-1})/\xi$ ($\varepsilon = 10^{-6}$). Firstly, the dataset is equally mapped into $M+1$ categories in Euclidean space. But we partition the datasets into a small number of binary super-categories $\boldsymbol{S^H_{Ni}}$ to replace the $M+1$ binary super-categories, since it is impractical to optimise the super-categories as M is huge.

Next, we compare the AUC of all the methods, mentioned in Section 3.5.2. We use 'SCi' to denote SCiForest, and 'OPHi' to denote OPHiForest. The baseline is the simple combination of the OPH scheme and isolation forest without our proposed learning scheme. For each dataset, the top 3 leading values are highlighted, as shown in Table 3.3.

Table 3.3: Comparing AUC (%) of ensemble-based methods

#	iNNE	EnKNN	EnLOF	iForest	SCi	L2SH	KLSH	baseline	**OPHi**
1	69.61	77.50	68.22	69.14	79.14	**79.68**	**83.78**	75.50	**93.66**
2	29.10	**93.66**	90.97	83.55	91.28	**91.30**	89.92	85.77	**92.81**
3	39.11	**97.22**	39.20	94.10	**98.10**	**97.12**	97.21	94.01	96.01
4	55.41	**70.88**	55.95	66.51	65.10	70.20	**70.37**	60.05	**71.50**
5	87.95	**94.76**	**94.88**	79.50	88.98	90.04	91.01	86.31	**93.46**
6	62.12	**73.88**	73.20	72.26	**74.01**	71.89	73.58	52.11	**74.20**
7	67.13	67.69	**67.98**	64.40	63.57	67.01	**67.86**	57.44	**68.08**
8	46.35	**97.57**	58.87	97.12	**98.12**	97.42	97.22	90.02	**98.52**
9	47.88	**78.69**	55.13	62.02	73.44	72.63	**77.61**	70.68	**79.93**
10	52.50	**76.01**	60.98	70.65	70.17	**77.09**	74.66	68.62	**76.62**
11	63.45	99.15	98.12	**99.28**	**99.17**	97.14	89.55	94.12	**98.98**
12	70.15	76.98	75.61	**79.89**	72.66	77.10	**79.98**	71.83	**77.83**
13	65.52	**93.35**	**93.24**	89.61	91.02	85.23	**91.79**	82.63	89.63
14	42.10	**63.40**	54.70	**65.88**	62.10	62.47	57.92	60.38	**67.68**

We can observe from Table 3.3 that our method has good performance in all

datasets, especially in half of the datasets, our method performs best. The baseline of our method is weak with low detection accuracy in all datasets. Besides, EnKNN has good AUC results in most datasets, but it is time-consuming in the evaluation stage. For real-life applications, we care about not only the accuracy but also the execution time. So we evaluate the execution time (analysed in Section 3.4.4) of all datasets, and the results are shown in Table 3.4.

Table 3.4: Comparing execution time (s) of all methods

#	iNNE	EnKNN	EnLOF	iForest	SCi	L2SH	KLSH	baseline	**OPHi**
1	36	35	37	0.2	20	3	13	3	3
2	80	72	105	0.4	37	3	39	7	4
3	101	179	198	0.4	51	6	50	16	6
4	205	239	235	0.4	59	7	57	25	9
5	420	606	496	0.4	70	12	79	37	18
6	1,068	1,187	1,106	0.8	98	16	196	91	45
7	1,499	1,650	1,679	0.8	140	21	235	118	48
8	1,984	2,008	2,380	0.9	151	38	288	158	62
9	3,367	3,561	3,415	2	152	56	299	183	84
10	2,115	2,278	2,023	1	115	31	620	179	54
11	15,362	19,845	14,673	18	712	218	2,801	825	612
12	126	144	131	0.7	52	3	296	39	24
13	261	246	245	1	94	6	955	26	17
14	1,682	1,799	1,760	0.8	142	32	197	81	65

In Table 3.4, we can find that iForest is the fastest method among all ensemble-based methods in our experiment, on the contrary, EnKNN is the slowest method. The efficiency in OPHiForest has a significant improvement to the baseline. Compared with EnKNN, our method is around 30 times faster, while L2SH is around 50 times faster. Combining the result of AUC, we can conclude that our method has better robustness and accuracy than ensemble-based anomaly detection methods in general real-world applications.

3.5.4 Comparison with SCiForest

To further illustrate the superiority of our method, we compare OPHiForest with SCiForest, which also utilises the learning scheme to learn the best hash functions for ensemble-based anomaly detection. SCiForest has shown to be superior in detecting local clustered anomalies. Herein, we use three datasets, only containing clustered anomalies, to verify the effectiveness and efficiency of our method. Besides, the parameters of SCiForest are all selected according to the best performance in experiments [111].

Three anomaly detection datasets are from UCI repository, including Kddcup99, Dermatology and Thyroid. These datasets are all used in the empirical evaluation of SCiForest [111]. Kddcup99 is the network intrusion data, used for The Third International Knowledge Discovery and Data Mining Tools Competition. The attack data instances are treated as anomalies in Kddcup99. Dermatology is used to determine the type of Eryhemato-Squamous Disease, so the smallest class is defined as anomalies. Thyroid is the Disease dataset, which contains two anomalous classes and one normal class. All the datasets have been preprocessed. Then, the description of three clustered datasets are shown in Table 3.5.

Table 3.5: Three clustered datasets used in experiments

#	Name	n	m	Outlier vs. Inlier Labels	Rate
1	Kddcup99	494,021	38	others vs. "neptune", "smurf" & "normal"	1.77%
2	Dermatology	366	34	class 6 vs. class 1, 2, 3, 4 & 5	5.46%
3	Thyroid	7,200	21	class 1 & 2 vs. class 3	8%

The parameter in the first line of Table 3.5 has the same meaning as that in Table 3.2. For the datasets in Table 3.5, we make a comparison and analysis between our method and SCiForest in AUC. The detailed results are shown in Table 3.6.

In Table 3.6, the top leading value is highlighted. We can find that our method has better AUC results in all three datasets, compared to that of SCiForest. Specifically,

Table 3.6: Comparing AUC (%) of clustered datasets

#	Dataset	SCi	**OPHi**
1	Kddcup99	99.29	**99.43**
2	Dermatology	90.57	**98.03**
3	Thyroid	63.77	**64.67**

our method is significantly more effective in detecting anomalies in the Dermatology dataset, which is 8% better in AUC compared with SCiForest.

The processing time of compared methods is another important feature of our experiment. Given the real-life application, we just need to record the execution time, since the applications do not train a new model each time. Herein, we will evaluate the execution time of three datasets, as shown in Table 3.7.

Table 3.7: Comparing execution time (s) of clustered datasets

#	Dataset	SCi	**OPHi**
1	Kddcup99	6,145	1,537
2	Dermatology	1	2
3	Thyroid	48	52

We can find from Table 3.7 that two methods have similar execution times in datasets Dermatology and Thyroid. But in dataset Kddcup99, our method is around 4 times faster than SCiForest in detecting anomalies. So our method has superiority for anomaly detection in big data environments. Combined with the previous analysis of AUC, we can conclude that our method has better effectiveness and efficiency compared with SCiForest.

Through the analysis of Section 3.5.1 and 3.5.2, we find that our method has better robustness and scalability for anomaly detection, compared with other ensemble-based anomaly detection methods. Although the training process of OPHiForest is time-consuming, we just consider the execution time to find anomalies by using a model

obtained after training in real-life applications.

3.6 Conclusion

To cope with the challenge of big data, there is a need for fast and accurate anomaly detection methods. In this chapter, we make a detailed analysis of the recently developed anomaly detection methods. We also investigate the hashing scheme and how to integrate it into fast ensemble-based anomaly detection. On this basis, we propose a novel isolation forest based anomaly detection method called OPHiForest, which generates isolation trees by learning hash functions with order preservation. OPHiForest centres on the learning theory to minimise the penalizing misalignment rate of similarity orders between the original space and Hamming space. There are three stages in our method, pre-training, training, and evaluation stage. Although the training stage in our approach is time-consuming, it does not affect the execution time of the trained model when applied to find anomalies in a dataset. In the experimental evaluation, we analyse how the number of trees influences the effectiveness and demonstrate the ability to detect local anomalies by using our methods. Then, the extensive experimental results illustrate the effectiveness and efficiency of the proposed method in detecting anomalies in real-world datasets. Specifically, our method shows outstanding accuracy compared with both general ensemble-based methods and SCiForest. The execution time of our method is comparable to or even better than other ensemble-based methods.

In the next chapter, we aim to explore the influence of isolation tree structures on the detection performance and design a suitable method to learn this tree structure.

4

Optimal Isolation Forest for Anomaly Detection

In this chapter, we mainly focus on discovering the optimal tree structure for an isolation forest with respect to the branching factor since there is no theoretical work answering this fundamentally and practically important question. This chapter corresponds to the second problem of our thesis for the previous chapter has resolved the feature learning problem. Specifically, we establish a theory on isolation efficiency to answer the question and determine the optimal branching factor for an isolation tree. Based on the theoretical underpinning, we design a practical optimal isolation forest OptIForest incorporating clustering based learning to hash which enables more information to be learned from data for better isolation quality. The rationale of our

approach relies on a better bias-variance trade-off achieved by bias reduction in OptIForest. Extensive experiments on a series of benchmarking datasets for comparative and ablation studies demonstrate that our approach can efficiently and robustly achieve better detection performance in general than the state-of-the-arts including the deep learning based methods.

4.1 Overview

Detection of anomalies (also known as outliers) is an important machine learning task to capture abnormal patterns or sparse observations in data that collide with the majority showing the expected behaviours [146], and has been deployed in a broad range of fields for critical applications such as intrusion detection in cybersecurity, financial risk detection, and human or device health monitoring [6, 30, 43]. While anomalies often take a very small portion of the data or appear infrequently, failing to catch them in a timely manner for further actions can result in severe consequences such as cascading failures in manufacture and deaths in healthcare. Therefore, a variety of unsupervised anomaly detection methods [164], from shallow to deep, have been proposed for different types of anomalies and data types. Note that since collecting labels for data is usually difficult and expensive, especially for anomalies, we herein focus on unsupervised anomaly detection which is more commonly-used in practice. Recently, several deep learning based detection methods, e.g., RDP [207] and REPEN [143], have shown the advantages of having feature representation learning for anomaly detection [146]. But this does not mean that the traditional (shallow) detection methods will be obsolete because deep neural networks have their own intrinsic limitations such as high computational cost, poor explainability, and difficulty in hyperparameter tuning [103], or a shallow model can have comparable performance but cost much less, e.g., a very recent work ECOD [103] just use the statistic analysis of the tail event of a distribution to achieve the good performance over various benchmark datasets. Instead, traditional models can be preferred in specific scenarios, e.g., edge computing where computational resources are limited and medical research where high explainability is

in demand. Moreover, these methods can work together with the feature representations learned from deep learning for better performance, e.g., a recent work [230] has shown such an example that iForest is successfully used together with deep learning.

Benefiting from ensemble learning [5], a category of detection methods based on the isolation forest mechanism [63] stands out of the shallow models due to its excellent simplicity, effectiveness, and efficiency, therefore being really promising for big data anomaly detection. The basic idea is to randomly and recursively partition relatively small samples drawn from a dataset until all data instances are isolated, which produces a forest of isolation trees. Anomaly scores can be derived from an isolation forest based on an observation that anomalies often have shorter path lengths due to the ease of isolating them from others. As the first instance, iForest [110] has been widely recognised in academia and deployed in real applications, e.g., it has been included in *scikit-learn*, a commonly-used machine learning library in Python. While most of isolation forests following iForest use the binary tree structure for data isolation, a framework LSHiForest [249] producing multi-fork isolation trees with the use of similarity hash functions has demonstrated better detection performance. An interesting question arises naturally: What is the optimal branching factor for an isolation tree? However, there is little theoretical work answering this fundamentally important question, and practically this wide gap can hamper the further development of isolation forest based anomaly detection methods.

In this chapter, we investigate this interesting problem and establish a theory on the structure optimality of an isolation tree with respect to the branching factor by introducing the notion of isolation efficiency. A constrained optimisation problem is formulated to solve the optimality problem, and an interesting finding is that the optimal branching factor is e (Euler's number), rather than 2 which is commonly-used in existing methods. Based on the theoretical foundation, a practical optimal isolation forest named OptIForest is proposed for efficient and robust anomaly detection. Specifically, we adapt clustering based learning to hash in OptIForest to let the isolation process optimised with more information learned from data. With two key observations on ensemble learning and anomaly distribution in an isolation tree, we design

controllable initialisation for agglomerative hierarchical clustering, enabling OptIForest to achieve a better bias-variance trade-off via bias reduction from learning and improve the computation efficiency. Extensive experiments are performed on a suite of benchmarking datasets in our ablation and comparative studies. The results validate our proposed theory on optimal isolation forest, and also show that with respect to detection performance, OptIForest can generally outperform the state-of-the-arts including the deep anomaly detection methods while maintaining a high computation efficiency. The source code is available at https://github.com/xiagll/OptIForest.

The main contributions of this work are threefold, summarised as follows: (1) We are the first to formally investigate the optimality problem of isolation tree structure with respect to the branching factor and establish a theory on the optimal isolation forest, offering a theoretical understanding for the effectiveness of the isolation forest mechanism. (2) We innovatively propose a practical optimal isolation forest OptIForest that can enhance both detection performance and computational efficiency by designing a tailored clustering based learning to hash for a good bias-variance trade-off. (3) Results from extensive experiments support our theory well and validate the effectiveness and efficiency of our approach, as well as the advantages over the state-of-the-arts.

4.2 Related Work

Various methods for unsupervised anomaly detection have emerged, including distance-based, density-based, statistical, ensemble-based, and deep learning-based methods [27, 146]. This section delves into the crucial methods related to our study.

Deep Learning-based Methods. Recently, deep neural networks are widely explored for anomaly detection, particularly on complex data types [267, 242]. Techniques like generative adversarial networks (GANs) [118], AutoEncoders [29], and reinforcement learning have been leveraged to enhance the detection performance [147]. For example, a random distance-based anomaly detection method called REPEN is proposed in [143], where learning low-dimensional representation in random subsample is optimised. While these deep methods can have high accuracy with learning feature

representation [245, 247], they often suffer from issues like expensive computations, complex hyperparameter tuning, etc.

Ensemble-based Methods. To achieve robust detection, classical anomaly detection methods are integrated with ensemble learning, e.g., ensemble LOF [266], isolation using Nearest Neighbour Ensemble (iNNE) [13], and average k-NN distance ensemble [5]. These methods suffer from heavy computational cost when handling big data. A simpler but more effective method is iForest [112] which leverages the principle that anomalies are more likely to be isolated from others. This pioneering work shows the strong ability of the isolation forest mechanism and has been widely adopted in real applications. A sequence of work on isolation forests have been proposed to mitigate the shortcomings in iForest, e.g., SCiForest [111] addresses the failure of detecting axis-parallel anomalies and local anomalies by using a optimsation strategy. Another interesting work is LSHiForest which produces multi-fork isolation trees with the use of LSH (locality-sensitive hashing) functions that can hash similar data into the same hash value. The work formally understands the distance metric underlying iForest and enables the isolation forest mechanism widely applicable to any data types where an LSH family can be defined. But why a multi-fork isolation forest performs better has not been tackled in LSHiForest. Deep isolation forest [230] has been recently proposed to incorporate isolation forest with random feature representation, extending iForest to complex data types. However, these works still fail to learn information properly from data to reduce the bias of a base detector for a better bias-variance trade-off.

4.3 Preliminaries and Problem Statement

LSHiForest Framework [249]. Our approach is built on top of the LSHiForest Framework which is an effective and generic anomaly detection framework with the forest isolation mechanism. While the algorithmic procedural and the way of deriving anomaly scores in LSHiForest are quite similar to that in iForest, a key difference is that LSHiForest makes use of a hash function to determine the branching when isolating data in the recursive tree construction process, i.e., data instances with the

same hash value go into the same branch. As a result, the isolation trees can be multi-fork, which is significantly different from the binary case in iForest. Since the hash functions are drawn from an LSH (Locality-Sensitive Hashing) family [16], the data instances falling into the same branch are similar to each other with provably high probability and the isolation process is more natural than binary splitting. Moreover, the LSHiForest framework has high applicability to work with any distance metric with an LSH family. In fact, iForest and its variant have been proved to be two specific instances of the framework with less commonly-used distance metrics. Besides, other instances of the framework with specific distance metrics like Angular distance, Manhattan (ℓ_1) distance, and Euclidean (ℓ_2) distance have been implemented as well, and the instance with Euclidean distance (denoted as L2SH) shows a very efficient, robust, and accurate detection performance. Thus, we are motivated to use LSHiForest and its L2SH instance as the foundation of our approach given their prominent features and excellent performance.

Problem Statement. Although multi-fork isolation trees can be elegantly constructed in LSHiForest and better detection performance can be achieved, the theoretical understanding of this phenomenon is still missing. Accordingly, an interesting question arises naturally: What is the optimal branching factor for an isolation tree? Since the branching factor influences the tree structure which can be regarded as a parameter of the detection model, answering this question is fundamentally important to understand how the branching factor affects the performance of isolation forest based anomaly detection methods, but intrinsically a challenging task. Fig. 4.1 shows an example of isolating nine data instances with three different tree structures, including 9-fork tree, binary tree, and ternary tree. It is hard to intuitively tell which structure is the best for isolation, so how to address this problem is non-trivial.

Besides the tree structure, another challenge imposed on existing isolation forest based anomaly detection methods is how much information they should learn from the data to facilitate data partition at each internal node. For instance, iForest learns the minimum and maximum of the selected feature to determine the random splitting

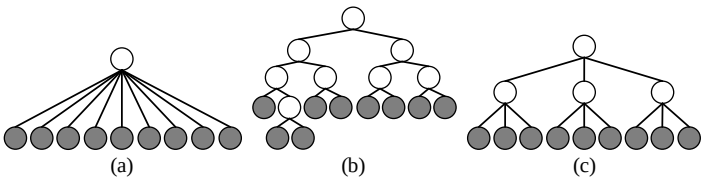


Figure 4.1: Isolating 9 data instances with different tree structures.

value, and its variant SCiForest[111] learns more information to determine the splitting hyperplane and gains better performance. But it is worth noting that more learning does not means better detection performance while the bias of a single base detector can be reduced, according to the bias-variance trade-off theory in ensemble learning [5]. For example, the recent work [230] actually reports that full learning for the neural networks has poorer detection performance than the random weight initiation. Because LSH hash functions used in LSHiForest are data-independent, no learning is incorporated in LSHiForest for data isolation. Thus, it is conjectured that the detection performance could be further improved if an isolation tree is constructed with the learning to hash technique [209] which produces hash values based on learning information from the data. But learning to hash usually incurs higher computational cost than LSH. Therefore, non-trivial effort is required for the design of an anomaly detector based on isolation forest with learning to hash, to target both high computational efficiency and a good bias-variance trade-off.

4.4 Methodology

4.4.1 Optimal Isolation Forest

To answer the research question stated above about the optimal branching factor of an isolation tree, this section formulates the problem to derive the solution and discusses its practical implementation. Let T represents an isolation tree, with the branching factor v, the tree depth d, and the number of leaf nodes ψ. For the purpose of theoretical analysis, we can reasonably allow v, d, and ψ to take real values rather than just integers. Actually, v and d can be regarded as average branching factor of the

internal nodes and the average height of the leaf nodes, respectively. To study how the branching factor influences the isolation performance of an isolation tree, we assume that T is a perfect tree where all internal nodes have v children and all leaves have the same depth d. Note that the isolation tree structure may be totally different in practice, but the assumption is the average case which is reasonable in the ensemble learning setting. Given T is perfect, we can have the following relationship:

$$\psi = v^d. \tag{4.1}$$

Definition 1 (Isolation Capacity)**.** *The maximum number of data instances an isolation tree can isolate is defined as the isolation capacity of the tree, denoted as ψ, which is also the number of leaf nodes.*

Given an isolation capacity, it can be interestingly observed that the width (controlled by v) and the depth (controlled by d) of the tree compete with each other, i.e., if the branching factor is smaller, the tree has to be deeper, or vice versa. So, we can have the following definition to capture the overall effect of branching factor and tree depth.

Definition 2 (Isolation Area)**.** *The isolation area of an isolation tree, denoted as ϕ, is defined as the product of the branching factor v which controls the width of a tree and the depth d, i.e.,*

$$\phi = v{\cdot}d. \tag{4.2}$$

Note that isolation trees of the same isolation area can achieve different isolation capacities. For example, given an isolation area $\phi = 6$, a perfect binary tree (i.e., $v = 2$ and $d = 3$) has the isolation capacity $\psi = 2^3 = 8$, but a perfect ternary tree (i.e., $v = 3$ and $d = 2$) has the isolation capacity $\psi = 3^2 = 9$. The latter case seems more efficient, and we can further define the concept of isolation efficiency as follows.

Definition 3 (Isolation Efficiency)**.** *The isolation efficiency of an isolation tree, denoted as η, is defined as the quotient of the isolation capacity divided by the isolation area, i.e.,*

$$\eta = \frac{\psi}{\phi}. \tag{4.3}$$

Isolation efficiency fundamentally affects the anomaly detection performance of isolation forests because it is associated with the hardness of distinguishing data instances isolated by a tree. Specifically, higher isolation efficiency leads to stronger distinguishability. Continuing the example above, the isolation efficiency of the binary tree is $\eta = 8/6 \simeq 1.33$ and that of the ternary tree is $\eta = 9/6 = 1.5$. Fig. 4.1(c) shows that the ternary tree can isolate all the data, while Fig. 4.1(b) shows that a binary tree with only 3 layers fails to do so.

As the branching factor indeed affects the detection performance in terms of the analysis presented above, it is of both theoretical and practical importance to study the problem of the optimal branching factor. With the concepts introduced above, we can formulate the problem of identifying the optimal branching factor v^* as a constrained optimisation problem as follows.

$$\begin{aligned} v^* = \quad & \underset{v}{\operatorname{argmax}} \quad \eta(v,d) = \frac{v^d}{vd}, \\ & \text{s.t.} \quad vd = \Phi, \end{aligned} \tag{4.4}$$

where $\eta(v,d)$ is the function of isolation efficiency with respect to branching factor v and tree depth d, and Φ is a constant number representing the fixed isolation area. By solving the optimisation problem, we can have the following theorem.

Theorem 1. *An isolation tree T has the highest isolation efficiency when its branching factor $v = e$, where e is the Euler's number with numerical values around* 2.718*.*

The theorem can be proved with solving the optimisation problem in Equ. (4.4).

Proof. According to the Definition of the isolation efficiency, to obtain the optimal isolation tree is to maximise the isolation efficiency, which is formulated as:

$$\eta(v,d) = \frac{\psi}{\phi} = \frac{v^d}{vd}, \tag{4.5}$$

where v represents the branching factor, d represents the tree depth. After fixing the isolation area by a constraint number Φ, we can obtain the function of isolation efficiency with respect to v:

$$\eta(v) = \frac{1}{\Phi} v^{\frac{\Phi}{v}}. \tag{4.6}$$

The derivative of $\eta(v)$ can be derived by:

$$\eta'(v) = v^{(\frac{\Phi}{v}-2)}(1 - \ln v). \tag{4.7}$$

Because $v^{(\frac{\Phi}{v}-2)} > 0$, if $1 - \ln v > 0$, we can have $\eta'(v) > 0$, and if $1 - \ln v < 0$, we can have $\eta'(v) < 0$. Thus, $\eta(v)$ is a convex function and has a maximum when $1 - \ln v = 0$, i.e., the optimal branching faction $v^* = e$.

Also, we can visualise the relationship between v and $\eta(v)$. Without loss of generality, Fig. 4.2 demonstrates an example where Φ is fixed at 6 (other values should show the same trend of $\eta(v)$ with respect to v). It can be seen that the highest isolation efficiency is attained when the branch factor is equal to e.

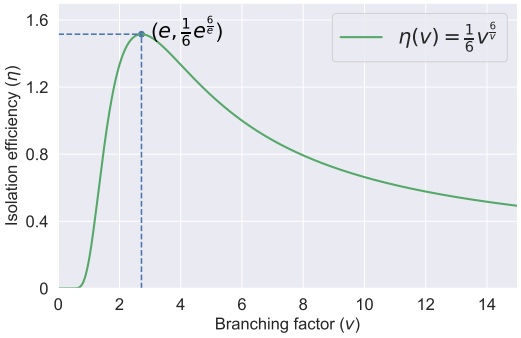


FIGURE 4.2: The relationship between isolation efficiency $\eta(v)$ and branching factor v. We use $\Phi = 6$ as an example for an illustration without loss of generality.

□

The above proof shows how the isolation efficiency changes with respect to branching factor with a fixed isolation area is also illustrated. This interesting result reveals that the commonly-used binary tree is not the optimal for the forest isolation mechanism, and explains the better detection accuracy and robustness gained by the multi-fork LSHiForest instances like L2SH.

Definition 4 (Optimal Isolation Tree)**.** *An isolation tree with branching factor $v = e$ is defined as an optimal isolation tree.*

Definition 5 (Optimal Isolation Forest)**.** *A forest consisting of a set of optimal isolation trees is defined as an optimal isolation forest.*

Although the optimal isolation tree is theoretically promising, unfortunately there is no real e-fork branching can be constructed directly in reality. As a practically viable way, we can build an isolation tree with the average branching factor equal to e. To achieve this, we need to generate the branching factors during the tree construction. Let a random variable V denote the branching factor for an isolation tree T, with the sample space $\{v \mid v \in \mathbb{Z} \ \& \ v \geqslant 2\}$. Let $\mathcal{D}$ be a distribution with the probability of taking value v being: $\Pr(V = v) = p_v$. We can generate the branching factors from the distribution $\mathcal{D}$ if it satisfies the following condition:

$$\mathbb{E}(V) = \sum_{v=2}^{+\infty} v \cdot p_v = e. \tag{4.8}$$

As the branching factor 2 is the only one less than e, there should be a lower bound for p_2 and upper bounds for p_v, $v > 2$, to satisfy the condition mentioned above.

Theorem 2. *To satisfy the condition in Eq.(4.8), the probability of having a branching factor $V \geqslant v$, $v > 2$, should have the following upper bound:*

$$Pr(V \geqslant v) \leqslant \frac{(e-2)}{(v-2)}. \tag{4.9}$$

The theorem can be proved by letting $p_i = 0$ for $2 < i < v$ and leveraging the inequality $\sum_{i=v}^{+\infty} i p_i \geqslant v \sum_{i=v}^{+\infty} p_i$.

Proof. When the average branching factor is equal to e, we can formalise the branching factor and the corresponding probability by:

$$\begin{cases} \sum_{i=2}^{+\infty} i \cdot p_i & = e, \\ \sum_{i=2}^{+\infty} p_i & = 1, \end{cases} \tag{4.10}$$

where p_i corresponds to the probability to produce i branches and the sum of the probabilities of producing different branches is equal to 1. To calculate the upper bond of $\Pr(V \geqslant v)$, we should make $p_i = 0$ for $2 < i < v$, and then leverage the inequality:

$$\sum_{i=v}^{+\infty} i p_i \geqslant v \sum_{i=v}^{+\infty} p_i. \tag{4.11}$$

Combining the condition that $p_i = 0$ for $2 < i < v$, we can rewrite Eq.(4.10) as:

$$\begin{cases} 2p_2 + \sum_{i=v}^{+\infty} i \cdot p_i & = e, \\ p_2 + \sum_{i=v}^{+\infty} p_i & = 1. \end{cases} \tag{4.12}$$

Then, we can have the following equation:

$$e - 2 = \sum_{i=v}^{+\infty} ip_i - 2\sum_{i=v}^{+\infty} p_i, v > 2. \tag{4.13}$$

By substituting Eq.(4.11) into Eq.(4.13), we can obtain:

$$e - 2 = \sum_{i=v}^{+\infty} ip_i - 2\sum_{i=v}^{+\infty} p_i \geqslant (v-2)\sum_{i=v}^{+\infty} p_i, v > 2. \tag{4.14}$$

Finally, we can have the following inequality relationship:

$$\sum_{i=v}^{+\infty} p_i \leqslant \frac{e-2}{v-2}, v > 2. \tag{4.15}$$

Thus, the upper bound of $\Pr(V \geqslant v)$ is $\frac{e-2}{v-2}$, $v > 2$.

□

The upper bound result can give us an intuition on how the probability decreases when the branching factor increases, e.g., the probability is $e - 2 \simeq 0.718$ for $V \geqslant 3$, $\frac{(e-2)}{2} \simeq 0.359$ for $V \geqslant 4$, and $\frac{(e-2)}{3} \simeq 0.239$ for $V \geqslant 5$.

Corollary 1. *To satisfy the condition in Eq.(4.8), the probability of sampling the branching factor $V = 2$ should follow $p_2 \geqslant 3 - e \simeq 0.282$.*

This corollary can be simply derived from Theorem 2 when $V \geqslant 3$, i.e., $p_2 = 1 - \Pr(V \geqslant 3) \leqslant 1 - (e-2) = 3 - e$.

There are many possible concrete distributions for $\mathcal{D}$ to be specified, either finite or infinite. For example, by just using binary and ternary branches, we can have a simple finite distribution as follows: $p_2 = 3 - e$, $p_3 = e - 2$, and $p_i = 0$, $i \geqslant 4$. For the infinite case, the probability is often a function of the branching factor. The work in [165] derives a distribution $p_v = \frac{(v-1)}{(v!)}$, $v \geqslant 2$, from a stochastic technique. Besides, we can have another infinite distribution $p_v = \frac{(e-1)^2}{2e-1} e^{2-v}$, $v \geqslant 2$, to satisfy the condition in Eq.(4.8).

Let a random variable V denote the branching factor for an isolation tree T with the sample space $\{v \mid v \in \mathbb{Z} \ \& \ v \geqslant 2\}$. Let $\mathcal{D}$ be a distribution with the probability of taking value v being: $\Pr(V = v) = p_v$. We can generate the branching factors from the distribution $\mathcal{D}$. Then, we can have the following theorem with a proof.

Proposition 1. *If the distribution $\mathcal{D}$ is instantiated by assigning the probability of taking the branching factor v as $p_v = \frac{(e-1)^2}{2e-1}e^{2-v}$, $v \geqslant 2$, the resultant distribution satisfies the condition: $\mathbb{E}(V) = \sum_{v=2}^{+\infty} v \cdot p_v = e$.*

Proof.

$$\mathbb{E}(V) = \sum_{v=2}^{+\infty} v \cdot P_v = \sum_{v=2}^{+\infty} v \cdot \frac{e(e-1)^2}{2e-1} \cdot e^{1-v} = \frac{e(e-1)^2}{2e-1} \sum_{v=2}^{+\infty} v \cdot e^{1-v}. \tag{4.16}$$

For convenience, let Z denote the quantity $\sum_{v=2}^{+\infty} v \cdot e^{1-v}$, i.e.,

$$Z \triangleq \sum_{v=2}^{+\infty} v \cdot e^{1-v}. \tag{4.17}$$

Multiplying both sides of the Eq.(4.17) by e^{-1} can result in:

$$e^{-1} \cdot Z = \sum_{v=2}^{+\infty} v \cdot e^{-v}. \tag{4.18}$$

Subtracting Eq.(4.18) from Eq.(4.17), we can derive the following result by using the formula of computing the sum of a geometric series with a common ratio $e^{-1} < 1$.

$$\begin{aligned} Z - e^{-1}Z &= 2e^{-1} + e^{-2} + e^{-3} + ... + e^{-v} + ... \\ &= e^{-1} + (e^{-1} + e^{-2} + e^{-3} + ... + e^{-v} + ...) \\ &= e^{-1} + \frac{e^{-1}}{1 - e^{-1}} \\ &= e^{-1} + \frac{1}{e-1}. \end{aligned} \tag{4.19}$$

According to Eq.(4.19), the value of Z can be calculated by:

$$Z = \frac{e^{-1} + \frac{1}{e-1}}{1 - e^{-1}} = \frac{1 + \frac{e}{e-1}}{e-1} = \frac{2e-1}{(e-1)^2}. \tag{4.20}$$

Finally, the expectation $\mathbb{E}(V)$ can be calculated by Eq.(4.16), (4.17) and (4.20):

$$\mathbb{E}(V) = \frac{e(e-1)^2}{2e-1} \cdot Z = \frac{e(e-1)^2}{2e-1} \cdot \frac{2e-1}{(e-1)^2} = e. \tag{4.21}$$

□

Understanding the distributions of branching factors and the probability bounds can facilitate the decision-makings when one tries to design a more practical optimal isolation forest, as shown in the following section.

4.4.2 OptIForest: Practical Detector Design with Clustering based Learning to Hash

In this section, we investigate how to implement a practical optimal isolation tree. Since the data-independent LSH hash functions cannot exactly produce a specified number of hash values, we have to leverage learning to hash to achieve this for a specified branching factor. Moreover, learning to hash can capture more information from the data to benefit anomaly detection potentially. There are many types of learning to hash [209]. Given the higher accuracy and less quantification loss, the non-parametric hash function $h(\cdot)$ based on nearest vector assignment is adopted herein, i.e.,

$$h(\boldsymbol{x}) = \underset{k\in\{1,\cdots,v\}}{\operatorname{argmin}} ||\boldsymbol{x} - \boldsymbol{c}_k||, \tag{4.22}$$

where $\{\boldsymbol{c}_1, \cdots, \boldsymbol{c}_v\}$ is a set of centres of data partitions. Clustering algorithms are employed to compute the centres.

As our goal is to arrange the clusters into a natural hierarchy to form an isolation tree, agglomerative hierarchical clustering is adopted in our approach. The clustering method treats each data instance as a cluster initially and merges similar clusters sequentially until a single cluster is left, forming a hierarchical tree in a bottom-up fashion. While good clustering quality with less data distortion can be obtained, the high computational complexities are often regarded as a downside. With a cubic time complexity, clustering on a small dataset of a fixed size (e.g., 256 in iForest, and 1024 in LSHiForest) still seems too time-consuming. Straightforward adoption of the clustering technique is neither effective nor efficient.

Fortunately, we find that the clustering technique can be optimised based on two key observations in an isolation forest and design a more efficient clustering method. One observation is that anomalies are often located at upper levels of an isolation tree while normal data at lower levels. The quality of clustering at upper levels is

more sensitive to anomaly detection than that at lower levels. The other observation is that no learning or full learning in existing methods can lead to poorer detection performance. Therefore, we can let the upper levels of an isolation tree learn more from data to have better clustering quality, while letting the lower levels learn less, aiming to improve detection performance and save computational cost simultaneously.

The basic idea of our approach is to use an isolation tree efficiently produced in LSHiForest to initialise the clusters for agglomerative hierarchical clustering so that the clustering process begins with bigger initial clusters rather than the ones with a single data instance. This can maintain the high clustering quality for upper levels of the hierarchy and save much computational cost because the majority of computation occurs at lower levels. The specific steps for constructing an optimal tree T_{Opt}, are outlined in Algorithm 5, and details are discussed subsequently.

Let T_{LSH} denote an isolation tree in LSHiForest. It can be horizontally partitioned into two disjoint parts by a cut which can be represented as a set of node. Let Γ denote a cut, and $\Gamma = \{N_1, \cdots, N_{n_0}\}$, where N_i, $1 \leqslant i \leqslant n_0$ represents a node, and n_0 is the number of the nodes in Γ. The node N_i is either a subtree of T_{LSH} or a leaf (a trivial subtree), and contains a dataset which can be regarded as a cluster denoted as $\mathcal{C}_i$. Thus, we can obtain a set of clusters $\mathbb{C}_\Gamma = \{\mathcal{C}_1, \cdots, \mathcal{C}_{n_0}\}$ from a cut Γ. Then, we can define the concept of ϵ-cut below.

Definition 6 (ϵ-Cut)**.** *Given a cut Γ and its associated clusters $\mathbb{C}_\Gamma = \{\mathcal{C}_1, \cdots, \mathcal{C}_{n_0}\}$, for any node $N_i \in \Gamma$, if its cluster size satisfies $|\mathcal{C}_i| \leqslant \epsilon < |\mathcal{C}_p|$, where ϵ is a threshold and $\mathcal{C}_p$ is the cluster associated with its parent node, Γ is called ϵ-cut and denoted as $\Gamma(\epsilon)$.*

We can construct $\Gamma(\epsilon)$ by simply traversing the tree, and use the associated clusters $\mathbb{C}_{\Gamma(\epsilon)}$ as the initial ones for agglomerative clustering. By tuning $\epsilon \in [1, \psi]$, where ψ is the sample size in isolation forests, we can control the number and size of the clusters to further, and further manipulate the degree of learning. A higher ϵ leads to fewer initial clusters and implies less learning. The initialisation degenerates into the conventional case if $\epsilon = 1$, while if $\epsilon = \psi$, the resultant isolation forest is still T_{LSH} without learning

Algorithm 5 Constructing an Optimal Isolation Tree

Input: A dataset (sample) D of size ψ, and cut threshold ϵ.

Output: T_{Opt}-an optimal isolation tree.

Train LSHiForest to get T_{LSH}; ▷ Pre-training
Traverse T_{LSH} to get $\Gamma(\epsilon)$;
$\mathbb{C}_0 \leftarrow \mathbb{C}_{\Gamma(\epsilon)}$; $\Gamma_0 \leftarrow \Gamma(\epsilon)$ ▷ Initialisation
while $|\mathbb{C}_i| > 1$ **do**
    Generate a branching factor v from $\mathcal{D}$;
    if $|\mathbb{C}_i| \leqslant v$ **then**
        return T_{Opt} with $(N_{\text{root}}, \mathcal{C}') \leftarrow \text{merge}(\mathbb{C}_i)$;
    end if
    $J_{\text{cur}} \leftarrow +\infty, \mathbb{C}_{\text{cur}} \leftarrow NULL$
    for all $\mathbb{C}_v = \{\mathcal{C}_{i1}, \cdots, \mathcal{C}_{iv}\} \subset \mathbb{C}_i$ **do**
        if $\text{dist}(\mathbb{C}_v) < J_{\text{cur}}$ **then**
            $J_{\text{cur}} \leftarrow \text{dist}(\mathbb{C}_v)$, $\mathbb{C}_{\text{cur}} \leftarrow \mathbb{C}_v$
        end if
    end for
    $(N', \mathcal{C}') \leftarrow \text{merge}(\mathbb{C}_{\text{cur}})$, ; ▷ Optimal identified
    $\mathbb{C}_{i+1} \leftarrow \mathbb{C}_i \backslash \mathbb{C}_{\text{cur}}$, $\mathbb{C}_{i+1} \leftarrow \mathbb{C}_{i+1} \cup \{\mathcal{C}'\}$;
    $\Gamma_{i+1} \leftarrow \Gamma_i \backslash \{N_{i1}, \cdots, N_{iv}\}$, $\Gamma_{i+1} \leftarrow \Gamma_{i+1} \cup \{N'\}$;
end while

anything.

Unlike traditional agglomerative hierarchical clustering where a binary tree is generated, we need to achieve multi-fork branches for an optimal isolation tree. The distortion measure is employed to capture the learning loss caused by merging multiple clusters. Let $\boldsymbol{\mu}_{\mathbb{C}}$ denote the merged centre (mean) of a set of clusters $\mathbb{C} = \{\mathcal{C}_1, \cdots, \mathcal{C}_v\}$, calculated by

$$\boldsymbol{\mu}_{\mathbb{C}} = \frac{\sum_{i=1}^{v} \boldsymbol{\mu}_{\mathcal{C}_i} \cdot n_i}{\sum_{i=1}^{v} n_i}, \tag{4.23}$$

where $\boldsymbol{\mu}_{\mathcal{C}}$ is the centre of cluster $\mathcal{C} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}$, calculated by $\boldsymbol{\mu}_{\mathcal{C}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i$. Then, the distortion of merging the clusters in $\mathbb{C}$ can be calculated as

$$\text{dist}(\mathbb{C}) = \sum_{i=1}^{v} ||\boldsymbol{\mu}_{\mathcal{C}_i} - \boldsymbol{\mu}_{\mathbb{C}}|| \cdot n_i. \tag{4.24}$$

After generating the branching factor v (Line 5 in Algorithm 5), Lines 9-14 show the details of performing cluster merging. We need to try all the combinations of v clusters and evaluate their distortions. The one with minimum distortion is selected for merging, and a new node is created for the resultant cluster. This process repeats until the number of candidate clusters is not greater than the branching factor. Finally, all the remaining clusters are merged as the root node of the resultant isolation tree.

It is worth noting the produced tree is not an exact optimal isolation tree if ϵ is not 1, because this practical algorithm takes the trade-off between accuracy and efficiency into account. The most time-consuming part is the evaluation of all the combinations of v clusters, whose time complexity grows exponentially with respect to v. Thus, to be efficient, our approach only leverages binary and ternary branches for the tree construction, i.e., we adopt a finite distribution, which is technically reasonable because the probability drops considerably when the branching factor increases. But note that other distributions can be used if other clustering techniques like K-means are used.

Once a forest of optimal isolation trees are built as an optimal isolation forest for OptIForest, it can perform anomaly detection with the same way to derive anomaly scores as LSHiForest. Although a practical isolation tree can consists of two parts with the upper layers resulting from clustering and the lower layers from LSHiForest, the hash function interface makes the two implementations (LSH functions and the non-parametric learning to hash function in Eq.(4.22)) no difference as to anomaly score computation.

4.5 Experiments

4.5.1 Experiment Setting

Baselines and Datasets. Our method OptIForest is compared with six state-of-the-art anomaly detection methods: iForest, LSHiForest, ECOD, REPEN, RDP, and DIF, which are conventionally used for empirical study in many prior works. While there are many other anomaly detection methods, these six state-of-the-arts are chosen with the rationale that they are either very fresh or have shown superior performance in most cases. These methods are categorised into shallow anomaly detection methods and deep anomaly detection methods, briefly described as follows:

- **Shallow Anomaly Detection Methods.** iForest [110] is a seminal anomaly detection method, which can isolate data instances very efficiently with an ensemble of binary trees and usually achieves good performance. iForest has been widely recognised in academia and deployed in real applications, e.g., it has been included in *scikit-learn*[1], a commonly-used machine learning library in Python. LSHiForest [249] is a generic framework, which generalise the forest isolation mechanism with the multi-fork tree structure and achieves higher performance and applicability. We select its instance L2SH which in general has the best performance with respect to accuracy and robustness for the comparison study. The source code is available in a public GitHub repository[2]. ECOD [103] is a very fresh anomaly detection method, which is parameter-free and easy to interpret. ECOD uses the empirical distributions of the input data to estimate tail probabilities per dimension for each data point. This method is simple but effective on many benchmark datasets. The source code is available in a public GitHub repository[3].

- **Deep Anomaly Detection Methods.** REPEN [143] is a random distance-based anomaly detection method, which utilises deep representation learning and

[1] https://scikit-learn.org
[2] https://github.com/xuyun-zhang/LSHiForest
[3] https://github.com/yzhao062/pyod

distance calculation to learn low-dimensional representations. REPEN has been widely accepted as a benchmark for deep anomaly detection and its source code is available in a public GitHub repository[4]. RDP [207] trains neural networks to predict the abnormalities of data distances in a randomly projected space, where the genuine class structures are learned and implicitly embedded in the randomly projected space. RDP is a state-of-the-art deep anomaly detection method and achieves good performance on many datasets. The source code can be found in a public GitHub repository[5]. DIF [230] is a very recent anomaly detection approach that utilises neural networks to learn representations and these representations are then used to construct isolation forests for anomaly detection. The source code is available in a public GitHub repository[6].

We conduct experiments on 20 real-world datasets from different fields including finance, healthcare, network, etc [144, 61, 103]. All datasets are available in public repositories like UCI Machine Learning Repository[7], Kaggle Repository[8], and ADRepository[9]. The basic information about the datasets is summarised in Table 4.1.

Metrics and Parameter Settings. We conventionally use the Area Under Receiver Operating Characteristic Curve (AUC-ROC) and Area Under the Precision-Recall Curve (AUC-PR) as the performance metrics [92, 207]. The value of AUC ranges from 0 to 1, where a larger AUC result indicates better performance. The result of AUC has been extensively adopted in many anomaly detection works and has become an essential measure of accuracy in correlational research. Furthermore, the execution time is used as the efficiency evaluation criterion. All experiments are run 15 times and averaged results are reported.

Our method uses 100 isolation trees as the base detector for the isolation forest, the same as the existing isolation forest based methods. In section 4.5.3, we study the

[4] https://github.com/Minqi824/ADBench/tree/main/baseline
[5] https://git.io/RDP
[6] https://github.com/xuhongzuo/deep-iforest
[7] https://archive.ics.uci.edu/ml/datasets.php
[8] https://www.kaggle.com/datasets
[9] https://github.com/GuansongPang/ADRepository-Anomaly-detection-datasets

TABLE 4.1: A summary of datasets used in the experiments. Here, "n" in the table denotes the size of the dataset, "m" denotes the dimension of the dataset, and "Rate" represents the proportion of anomalous data in total data instances.

Dataset	n	m	Rate(%)	Category
AD	3,279	1,555	13.79	Finance
campaign	41,188	62	11.27	Finance
Arrhythmia	452	274	14.60	Healthcare
cardio	1,831	21	9.61	Healthcare
backdoor	95,329	196	2.44	Network
KDDCup99	494,021	38	1.77	Network
Celeba	202,599	39	2.24	Image
mnist	7,603	100	9.21	Image
Census	299,285	500	6.20	Sociology
Donors	619,326	10	5.92	Sociology
Cover	286,048	10	0.96	botany
http	567,498	3	0.39	Web
smtp	95,156	3	0.03	Web
Ionosphere	351	34	35.90	Oryctognosy
Satellite	6,435	36	31.60	Astronautics
Shuttle	49,097	9	7.15	Astronautics
Spam	4,207	57	39.91	Document
Vowel	1,456	12	3.43	Linguistics
Waveform	3,505	21	4.62	Physics
Wine	5,318	11	4.53	Chemistry

relationship between the sampling size and the detection performance and observe the performance of our method is saturated when the sampling size reaches a threshold ($2^9 = 512$ herein). Thus, the size of the sample used for constructing each tree is 512 in our method. Besides, we study how the cut threshold influence the detection performance in section 4.5.3 and observe that the selection of cut threshold ϵ is influenced by the size of dataset, i.e., the large datasets perform well with a big cut threshold (e.g., $\epsilon = 403$), while the small datasets perform well with a small cut threshold (e.g., $\epsilon = 55$). To make a fair comparison, our method will use the above optimal settings

to get the results. In the compared methods, all the parameters are set as the optimal settings in the original papers.

4.5.2 Comparison Study Results and Discussion

AUC-ROC Results. Table 4.2 illustrates that OptIForest is the most robust method that achieves the best performance on most datasets. Specifically, OptIForest has the highest average AUC-ROC score among all the compared methods, surpassing the second-best method by 2%. Additionally, RDP and LSHiForest perform well and are stable on most datasets. ECOD produces exceptional results on certain datasets, but performs poorly on others like "Vowel" and "CD" due to its requirement on the data distribution. Note that ECOD has zero standard deviation because it is a deterministic method. The rest methods iForest, REPEN, and DIF only perform well on a few datasets, indicating their lack of robustness across all datasets. Thus, it can be seen that no learning in iForest or DIF fails to achieve a robust performance due to the lack of knowledge learned from data. LSHiForest and OptIForest with better tree structures can achieve bias reduction, and learning from data further makes OptIForest perform better and even outperform the deep detector.

AUC-PR Results. The results in Table 4.2 show that OptIForest consistently performs best across various datasets with the highest average AUC-PR score, showing the superiority of our method. Specifically, our method outperforms the second-best one by about 0.6%. The top two results of each dataset are highlighted in bold. Although iForest and ECOD have exceptional results on a few datasets, they perform poorly on others. REPEN has the lowest average AUC-PR score and performs poorly on most datasets. The AUC-PR results show the same trend as the AUC-ROC results, supporting that our OptIForest performs the best performance in a robust way.

Execution Time. We use execution time as the efficiency metric to compare OptIForest with other methods on 20 real-world datasets , and the results are reported in Table 4.3. It can be seen in Table 4.3 that our method OptIForest has a much shorter execution time than the deep anomaly detection methods of REPEN and RDP for

TABLE 4.2: AUC-ROC and AUC-PR performance (mean ± standard deviation) of all methods. Our **OptIForest** method outperforms others.

	AUC-ROC (%)							AUC-PR (%)						
Datasets	iForest	LSHiForest	ECOD	REPEN	RDP	DIF	**OptIForest**	iForest	LSHiForest	ECOD	REPEN	RDP	DIF	**OptIForest**
AD	69.3 ± 1.9	**77.9 ± 0.4**	69.8 ± 0	70.0 ± 2.2	**88.7 ± 0.3**	76.8 ± 0.7	77.4 ± 0.7	40.1 ± 4.9	46.2 ± 0.6	48.3 ± 0	37.7 ± 4.0	**72.6 ± 0.7**	**51.8 ± 2.7**	43.8 ± 3.1
campaign	70.9 ± 1.0	67.7 ± 0.6	**77.5 ± 0**	61.1 ± 4.4	**76.3 ± 0.8**	69.3 ± 0.9	74.7 ± 0.4	28.5 ± 1.3	24.7 ± 1.0	**35.6 ± 0**	17.8 ± 3.9	**37.2 ± 0.9**	27.5 ± 1.3	32.0 ± 0.5
Arrhythmia	**79.7 ± 1.0**	77.5 ± 0.5	**82.3 ± 0**	73.7 ± 3.6	75.5 ± 0.5	76.3 ± 1.1	79.6 ± 0.8	**47.5 ± 1.4**	38.6 ± 0.7	**49.4 ± 0**	37.4 ± 3.5	32.0 ± 0.6	38.2 ± 1.2	45.1 ± 1.2
cardio	**93.0 ± 0.7**	90.4 ± 0.6	**95.0 ± 0**	91.5 ± 2.9	88.1 ± 0.6	**93.0 ± 0.5**	92.8 ± 1.3	57.0 ± 2.8	48.9 ± 1.0	**67.6 ± 0**	53.3 ± 10.6	53.9 ± 1.5	58.7 ± 1.9	**58.9 ± 3.7**
backdoor	72.7 ± 2.9	89.2 ± 0.9	84.9 ± 0	86.8 ± 1.6	91.0 ± 2.1	**92.0 ± 0.5**	**92.7 ± 0.5**	4.5 ± 0.7	27.3 ± 2.6	9.6 ± 0	12.5 ± 1.7	3.5 ± 0.8	**39.4 ± 3.3**	**51.7 ± 8.7**
KDDCup99	**97.0 ± 0.6**	96.4 ± 0.2	91.1 ± 0	95.9 ± 0.6	41.0 ± 3.1	88.5 ± 0.7	**97.4 ± 0.1**	**48.6 ± 7.0**	32.6 ± 1.1	**48.5 ± 0**	44.1 ± 1.9	15.4 ± 0.9	16.7 ± 0.5	43.0 ± 0.1
Celeba	69.4 ± 2.4	72.5 ± 0.7	72.3 ± 0	**84.3 ± 2.2**	**86.0 ± 0.6**	67.9 ± 1.7	79.2 ± 1.9	6.3 ± 0.9	6.8 ± 0.3	8.5 ± 0	**10.7 ± 1.8**	**10.4 ± 0.6**	5.5 ± 0.6	8.1 ± 1.0
mnist	80.2 ± 1.8	**85.3 ± 0.6**	83.8 ± 0	67.6 ± 10.6	85.1 ± 1.6	83.7 ± 1.3	**85.5 ± 1.0**	27.7 ± 3.2	**38.3 ± 1.0**	30.5 ± 0	20.4 ± 10.2	36.7 ± 2.4	33.0 ± 2.1	**40.7 ± 1.9**
Census	60.1 ± 1.8	62.6 ± 0.4	**66.8 ± 0**	62.7 ± 1.5	65.3 ± 0.4	59.4 ± 1.1	**67.8 ± 1.0**	7.1 ± 0.3	7.5 ± 0.1	**8.6 ± 0**	7.7 ± 0.2	**8.6 ± 0.1**	6.9 ± 0.2	**8.8 ± 0.2**
Donors	76.6 ± 1.0	74.5 ± 0.7	74.0 ± 0	**83.2 ± 1.7**	**96.2 ± 1.1**	67.8 ± 1.2	77.1 ± 3.2	11.9 ± 0.8	10.5 ± 0.6	13.6 ± 0	**15.5 ± 1.3**	**43.2 ± 6.1**	8.0 ± 0.3	11.3 ± 1.6
Cover	88.0 ± 2.1	**93.6 ± 0.5**	**93.3 ± 0**	86.6 ± 5.7	51.2 ± 1.3	76.4 ± 4.0	90.9 ± 0.8	6.4 ± 0.9	**9.0 ± 0.8**	**11.6 ± 0**	5.3 ± 2.0	2.0 ± 1.1	4.0 ± 0.8	6.5 ± 0.7
http	**99.9 ± 0**	93.3 ± 0	97.9 ± 0	**99.4 ± 0.1**	99.3 ± 0.1	99.3 ± 0.1	**99.4 ± 0.1**	**90.2 ± 7.9**	34.2 ± 0.6	14.5 ± 0	**39.5 ± 2.4**	36.2 ± 0.8	35.1 ± 0.4	35.4 ± 1.5
smtp	90.5 ± 0.8	86.9 ± 0.9	88.0 ± 0	**90.9 ± 1.3**	69.8 ± 1.1	84.6 ± 0.5	**92.4 ± 0.6**	0.4 ± 0	**56.9 ± 3.1**	50.7 ± 0	26.9 ± 10.8	21.7 ± 3.9	**55.2 ± 7.0**	36.9 ± 13.7
Ionosphere	84.5 ± 0.5	91.2 ± 0.2	76.8 ± 0	86.5 ± 2.3	82.7 ± 0.6	**94.3 ± 0.5**	**93.4 ± 0.3**	80.8 ± 0.5	89.6 ± 0.4	66.3 ± 0	81.0 ± 3.6	80.7 ± 0.6	**93.0 ± 0.6**	**92.3 ± 0.5**
Satellite	70.3 ± 1.8	**76.7 ± 0.5**	74.6 ± 0	74.0 ± 3.0	60.9 ± 0.6	69.2 ± 1.3	**78.6 ± 0.7**	65.0 ± 2.1	63.5 ± 0.6	66.1 ± 0	**70.3 ± 3.9**	61.3 ± 0.7	47.5 ± 1.3	**71.5 ± 0.9**
Shuttle	**99.7 ± 0.1**	97.2 ± 0.8	**99.7 ± 0**	99.4 ± 0.1	99.6 ± 0.9	96.3 ± 1.2	98.0 ± 0.4	**97.6 ± 0.5**	40.1 ± 2.7	**95.0 ± 0**	91.9 ± 0.6	89.7 ± 1.1	56.5 ± 6.5	64.0 ± 4.3
Spam	61.7 ± 2.9	70.7 ± 0.2	65.6 ± 0	**73.4 ± 0.3**	**74.7 ± 0.5**	65.0 ± 0.9	71.1 ± 0.2	46.8 ± 3.0	59.1 ± 0.5	51.8 ± 0	**62.7 ± 0.5**	**63.0 ± 0.2**	59.6 ± 0.5	58.1 ± 0.5
Vowel	75.3 ± 1.2	**90.8 ± 0.7**	40.8 ± 0	77.9 ± 5.5	67.2 ± 1.2	**94.4 ± 1.4**	90.0 ± 1.2	12.6 ± 1.7	29.5 ± 2.1	2.8 ± 0	14.5 ± 6.5	7.6 ± 3.2	**41.1 ± 9.0**	**32.4 ± 4.3**
Waveform	69.6 ± 2.3	70.7 ± 0.8	**71.5 ± 0**	63.2 ± 9.5	66.9 ± 0.9	63.2 ± 3.4	**74.4 ± 1.3**	9.7 ± 0.9	9.8 ± 0.4	9.4 ± 0	7.6 ± 2.4	8.7 ± 1.5	**11.2 ± 3.3**	**11.3 ± 1.0**
Wine	63.9 ± 0.7	**67.0 ± 0.3**	62.5 ± 0	64.6 ± 2.2	61.7 ± 0.4	65.9 ± 1.2	**66.4 ± 0.5**	7.8 ± 0.4	**8.9 ± 0.2**	8.1 ± 0	8.1 ± 0.7	8.1 ± 0.2	**8.5 ± 0.5**	8.1 ± 0.2
Average	74.3 ± 1.4	81.9 ± 0.5	78.8 ± 0	79.6 ± 3.1	81.2 ± 0.8	79.2 ± 1.2	**83.9 ± 0.9**	34.8 ± 2.5	34.1 ± 1.0	34.8 ± 0	33.3 ± 3.6	37.4 ± 1.4	34.9 ± 2.2	**38.0 ± 3.7**

most datasets. It is worth noting that the more our method learns, the more time it will spend. As previously discussed in the AUC comparison, achieving optimal performance on small datasets necessitates more learning, resulting in a longer execution time compared to REPEN on some small datasets. As expected, OptIForest reasonably has a longer execution time than other isolation forests where no learning is conducted. So, it can be concluded that our approach strikes a good balance between execution efficiency and detection performance. It is appealing that OptIForest can achieve better performance than deep learning methods but takes much less execution cost.

4.5.3 Ablation Study Results and Discussion

To understand how the branching factor, the cut threshold, and the sampling size influence the performance of OptIForest, we performed detailed ablation studies on eight datasets with a range of data types and sizes.

Table 4.3: Comparing execution time (s) of all methods. It is worth noting that OptIForest has a much shorter execution time than the deep learning methods of REPEN and RDP for most datasets.

Dataset	iForest	LSHiForest	ECOD	REPEN	RDP	DIF	**OptIForest**
AD	10	26	4	23	7,327	14	83
campaign	7	294	3	584	7,384	111	637
Arrhythmia	0.3	6	0.2	16	6,037	2	58
cardio	0.3	26	0.1	9	6,508	6	68
backdoor	26	758	18	2,647	5,267	245	1,289
KDDCup99	40	2,409	18	38,561	6,991	634	5,652
celeba	22	1,055	7	14,334	8,026	579	2,622
mnist	2	35	1	43	7,482	22	132
census	205	1,911	185	30,794	8,991	883	4,773
donors	24	3,901	7	51,496	7,655	1,367	6,784
Cover	14	1,474	6	27,930	6,472	732	3,637
http	19	3,557	4	91,980	6,098	1,322	7,083
smtp	4	501	1	7,115	6,422	203	1,262
Ionosphere	0.3	6	0.1	3	3,787	2	69
Satellite	1	36	0.4	213	4,649	17	209
Shuttle	2	313	1	1,060	4,928	124	662
Spam	1	15	0.3	123	6,316	10	83
Vowel	0.3	12	0.1	60	3,743	5	50
Waveform	0.5	17	0.2	112	3,839	10	109
Wine	0.4	29	0.1	156	4,613	15	126

Branching Factor. To eliminate the effects of feature learning, we use a data-independent baseline (i.e., $\epsilon = \psi$) to analyse the impact of the branching factor. Besides, it is difficult to implement the average branch of e in a practical experiment because many of the underlying branches just produce 2 branch forks. However, we can still analyse the AUC-ROC results for branches that are either near or far from e. The AUC-ROC results for different branching factors are presented in Fig. 4.3. When the branching factor is close to e, the AUC-ROC results are the best on almost all datasets. This result means that the best anomaly detection accuracy can be achieved

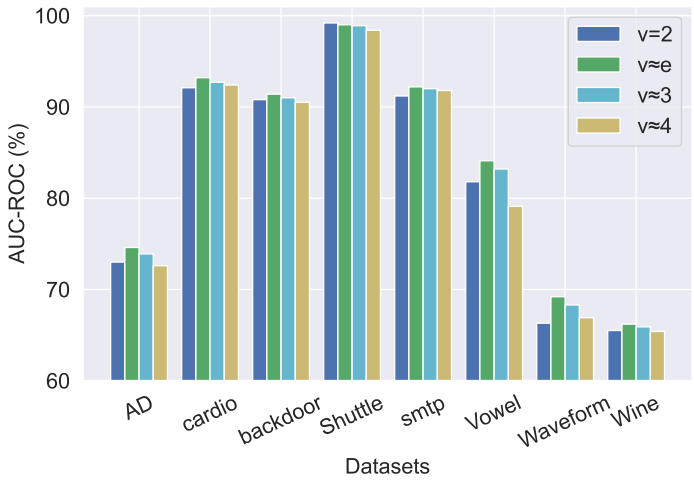


FIGURE 4.3: Detection performance changes w.r.t. branching factor v. A branching factor closer to e leads to a better performance.

if the branching factor satisfies the condition: $v \approx e$. These experimental results validate Theorem 1 in Section 4.4.1.

Cut Threshold ϵ. The cut threshold is studied by raising the branching factor to the power of e, as small changes of the threshold do not greatly affect detection accuracy. We also study the boundary condition of the cut threshold ($\epsilon = 512$). Fig. 4.4 displays AUC-ROC results and standard deviations for different ϵ. It can be seen that the curve increases as the threshold increases in four large datasets with data sizes larger than 10,000 or dimension sizes larger than 1,000, indicating little learning is required for large datasets (with $\epsilon = e^6$ as a reference). Conversely, more learning is necessary in four small datasets, and $\epsilon = e^4$ serves as a good reference point to balance accuracy and time efficiency. A comparison between the results of $\epsilon = 512$ with others illustrates that appropriate learning results in better outcomes than not learning on most datasets. But it is exceptional in the "AD" and "vowel" datasets, where not learning yields better results. This could be attributed to the fact that the isolation forest without learning exactly has the average branch of e, which achieves a favorable bias-variance trade-off.

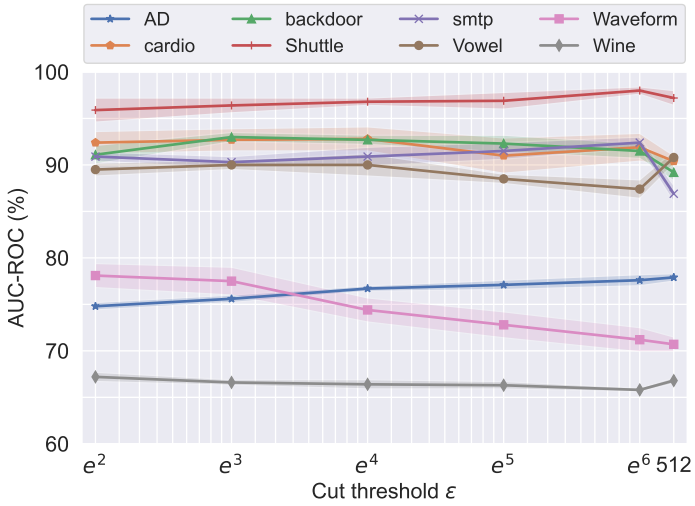


Figure 4.4: Detection performance changes w.r.t. cut threshold ϵ. In general, a certain level of learning outperforms the case of no learning ($\epsilon = 512$ means no learning).

Sampling Size. To facilitate the observation of the results, the sampling size is set from 2^6 to 2^{11} with exponential increase, as the results in this range can display significant change. The cut threshold of the same sampling size will impact the AUC-ROC results, so we determine the best results for each sampling group as the final outcomes. It can be concluded from Fig. 4.5 that the AUC-ROC results improve with the increase of the sampling size on most datasets. The AUC-ROC results remain stable once the sampling size exceed 2^9, except for the "cardio" and "Shuttle" datasets, which keep stable on any sampling size.

4.6 Conclusion and Future Work

Isolation Forest is an appealing anomaly detection method given its salient characteristics, but it lacks a theoretical foundation about the structure optimality of an isolation tree. In this chapter, we have introduced the concept of isolation efficiency and formulated a constrained optimisation problem to derive the optimal branching factor for an isolation tree. We have shown that an optimal isolation tree or forest is theoretically

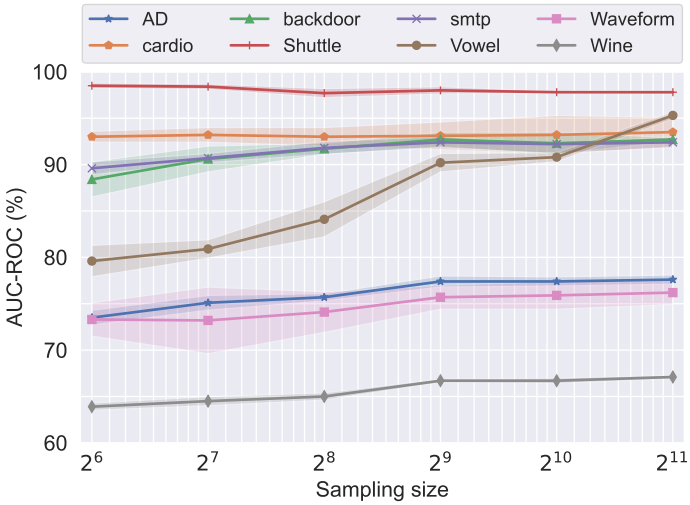


FIGURE 4.5: Detection performance changes w.r.t. sampling size ψ. In general, the performance is saturated when ψ reaches a threshold ($2^9 = 512$ herein).

with the branching factor e. Furthermore, we have developed a practical optimal isolation forest OptIForest which can achieve both high computational efficiency and a good bias-variance trade-off by designing a novel clustering based learning to hash for data isolation. We have conducted extensive experiments on a variety of benchmarking datasets for both ablation and comparative studies, and the results have confirmed the effectiveness and efficiency of OptIForest.

In the future, we plan to design a version in the context of federated learning where data are scattered across multiple clients. Besides, this algorithm finds the approximate optimal solution of the tree structure. In the following chapter, we will design a deep model to search for the optimal isolation tree structure.

5

Deep Isolation Forest for Anomaly Detection

Deep neural networks based methods often achieve good accuracy in anomaly detection, but it is troubled with long execution time and high memory consumption. These problems are associated with the inherent drawbacks of deep learning, such as too many parameters and deep fixed training layers. To remedy the above drawbacks, we try to explore an unsupervised non-neural network deep model for anomaly detection based on the experience of deep forest. In this chapter, a deep isolation tree ensemble approach, named DeepiForest, is proposed to do robust anomaly detection by enhanced representation learning and multi-layer cascaded architecture. Besides, the proposed method can solve the third problem in this thesis. Specifically, DeepiForest utilises

locality-sensitive hashing (LSH)-based isolation forest to produce unsupervised representation learning and tree-embedding scheme to obtain more enhanced features. Our method inherits the advantages of deep forests, such as fewer hyper-parameters and smaller model complexity than DNNs, simultaneously obtaining robust accuracy on anomaly detection. Extensive experiments on different-scale datasets illustrate the efficiency of DeepiForest and its comparable effectiveness to the state-of-the-art deep anomaly detection methods. Finally, we conduct a series of ablation studies to show the impact of different segments in DeepiForest.

5.1 Overview

Anomaly detection is committed to identifying abnormal patterns or sparse objects that do not meet the expected behavior or deviate from most dense objects [247]. Currently, anomaly detection has been broadly explored in various range of fields, such as risk management, financial surveillance, health and medical risk, and network security [146, 93]. With the increase of anomaly detection applications, the explosion of data and the diversified requirements of users present a growing challenge for the performance on time efficiency and detection accuracy of anomaly detection methods. So, deep learning has become a research hotspot in anomaly detection by its tremendous capabilities in learning expressive representations of complex and high-dimensional data [147, 245]. Almost all current deep learning applications apply neural network (NN) models as the core mechanics, which use the backpropagation algorithm to train parameters in their nonlinear modules [97].

Although deep neural networks (DNNs) have a strong ability to train high-accuracy models, it is subject to complicated parameter tuning, the very deep layers, and the large-scale datasets [261]. Specifically, there are too many parameters for tuning in DNNs, which are time-consuming and cause huge memory consumption [70]. Ascribed to the inherent attribute of black-box models, DNNs are also difficult to understand the decision processes. Moreover, the deep model is always more complicated than it should be since the neural network architecture has to be determined before the

model training [164]. Thus, deep models are time-consuming to train a complex and multi-layered structure. Compared with DNNs, there are still some occasions where shallow anomaly detection methods, such as iForest, SciForest, or LSHiForest, can obtain better efficiency as well as accuracy [112, 249].

Exploration as another deep learning model, multi-Grained Cascade Forest (gcForest) is proposed as a non-NN style deep model, which integrates decision forests into layer-by-layer feature learning, abandoning the original backpropagation scheme in NN [261]. Compared with DNN models, gcForest has much fewer hyper-parameters, less CPU usage, and faster elapsed Time [262]. Since the cascade levels can be automatically determined in model training rather than pre-establishing, gcForest facilitates better performance on small-scale datasets and achieves comparable accuracy on large-scale datasets. Furthermore, gcForest has been widely explored in abundant classification experiments, which showed robust and excellent performance on efficiency and accuracy across different datasets. Correspondingly, we propose a question: Can deep forest modules be adopted into anomaly detection rather than NN-based deep anomaly detection (DAD) methods? This is what we are keen to solve in this dissertation.

The biggest gap in applying cascaded forest to anomaly detection lies in the training forest of each layer. The core part of gcForest is the multi-layer training of ensemble decision forest, which is a supervised paradigm for classification [34, 253]. Specifically, through adding the enhanced features from the previous layer to the input data of the next layer, gcForest reinforces the results of each layer until the accuracy no longer increases. However, the labels of data are always missing or undiscovered in anomaly detection applications, and isolation forest based methods for anomaly detection are also unsupervised learning [15]. The features learned from isolation forests are not as accurate as those obtained by supervised learning. What is more, anomaly detection only produces two results (normal or abnormal), which do not meet the requirement that each layer of deep forest produces multiple enhanced features. If the isolation forests are directly applied to the deep forest, it will affect the accuracy of anomaly detection.

To address these issues, we proposed a deep isolation forest (DeepiForest) for

anomaly detection, which applies locality-sensitive hashing forest (LSHiForest) [249] to construct the cascaded structure and embedding learning to provide more enhanced features. In this novel isolation tree ensemble, we apply ALSH, L1SH, and L2SH (three instances of LSHiForest) as the training forest in each layer. Different isolation forests are applicable in our cascade layer, but the more robust forests will produce better results as the number of cascaded layers increases. LSHiForest has been approved to be superior and robust in most datasets compared with other isolation forests, so it is applied in our DeepiForest to produce features. These processes can reduce the impact of weak features on the whole dataset as well as move the supervised learning paradigm to unsupervised learning. Then, the tree embedding technique is adopted to provide more enhanced features for the input data on the cascaded structure. DeepiForest is a new attempt at the non-NN DAD model, which inherits the advantages of gcForest and deepens the isolation forest for anomaly detection. The experiments show the performance of our method is quite robust and stable in all datasets. DeepiForest holds good efficiency as well as comparable accuracy to the state-of-the-art DAD methods. The main contributions of this work are three-fold:

- It is the first work to apply deep forest for deep anomaly detection and deepen the isolation forests into the cascaded forest. Herein, we proposed an unsupervised DAD method, called DeepiForest, which holds the characteristics of very few parameters, short training time and robust detection accuracy.

- We improve the representation learning by combining the features of label representation and tree-embedding representation. This process enriches the diversity of features and adds more enhanced features to the original data instances.

- Extensive experiments illustrate the efficiency and robustness of our method. Besides, we conduct a series of ablation studies and make an analysis on how the tree-embeddings, the number of trees in a forest, and the number of cascaded layers influence experimental results. Our source code that implements the proposed method is available at https://github.com/xiagll/DOIForest.

5.2 Related Work

In this section, we introduce the research background and review related work of the latest unsupervised DAD methods, iForest-based anomaly detection methods and deep forest.

5.2.1 Deep Learning Methods for Anomaly Detection

Deep anomaly detection aims to learn a feature representation mapping function or an anomaly score learning function that can easily distinguish the anomalies from the normal data in the representation space [258, 39, 25, 210]. These mapping functions are neural network-enabled. Currently, there have been many unsupervised DAD methods, where autoencoder (AE) networks and their variations are the classic cases [71, 126]. AE methods are easy to implement and have straightforward intuitions in detecting anomalies, but it is originally designed for dimension reduction rather than anomaly detection. This will more or less produce some acclimatization phenomenon in anomaly detection applications. Besides, Zong et al. [267] utilised a deep autoencoder to generate low-dimensional representations and reconstruction errors for each input data instance, which is further fed into a Gaussian mixture model (GMM). Then, the parameters of both the deep autoencoder and GMM are jointly optimised to facilitate parameter learning of the proposed method. Furthermore, Pang et al. [143] proposed a random distance-based anomaly detection approach, called REPEN, which drives the learning of low-dimensional features out of ultrahigh-dimensional data instances. The key point of REPEN is to optimise the representations so that the nearest neighbour distance of the pseudo-label anomalies in the random subsample is much larger than that of the normal instances. But the computational overhead of REPEN is very high and their capabilities are limited by the inherent weaknesses of the distance-based characteristic. Recently, Wang et al. [207] proposed an unsupervised deep anomaly detection method, which trains neural networks to predict data distances in a randomly projected space. These methods are all based on DNNs, which can achieve decent accuracy along with complex parameter learning and the loss of efficiency.

Considering the defects of DAD methods such as too many training parameters, complex models and the static layer scheme, we aim to design a more portable deep learning mechanism for anomaly detection. Compared with the original DAD method, our new approach will spend less execution time to achieve comparable accuracy.

5.2.2 Isolation Forest based Methods for Anomaly Detection

There is another type of isolation forest based method, which is very fast and explicable for anomaly detection [222]. The most classical method is iForest, which has a very steady performance on many anomaly detection fields and detects anomalies very quickly [112]. Specifically, iForest randomly selects one dimension of data instances to partition anomalous and normal data each time until all data instances are partitioned in a leaf branch of a tree. Besides, iForest applies the sampling technique to tremendously decrease the training set of large-scale datasets and averages the sample results to improve the accuracy [16]. SciForest is a follower to iForest, which divides data instances accurately by learning data information in each layer of a tree [111]. This method greatly improves the accuracy of detecting clustered anomalies at the cost of time. Then, Hariri et al. [63] made a major improvement to iForest, by defining the interval of each data slice to replace the random partitioning. This step greatly improved the detecting accuracy of iForest, while it took some more time. Besides, Zhang et al. [249] integrated the locality-sensitive hashing (LSH) scheme with iForest technique to shape a generic anomaly detection framework, named LSHiForest. Compared with most isolation forest based anomaly detection methods, LSHiForest demonstrates to be faster and more accurate in the big data environment, and not limited to some specific data types. These methods have shown certain advantages in shallow anomaly detection, but they lack exploration in the DAD paradigm due to unsupervised characteristics.

In our method, multiple isolation forests are ensembled and constitute the cascaded forest through layer-by-layer processing. Besides, the development of isolation forest based methods will act on our method by providing more alternative forest baselines.

5.2.3 Deep Forest

Deep forest is the first work that utilises cascaded random forest to do deep learning rather than the back propagation-based neural networks [261]. Deep forest is a supervised model, that applies random forests to learn the labels of the data instances, and concatenates these labels with the original data to the next level. In the deep model, it will repeat the above process until the accuracy of the whole model stops increasing. Compared with traditional DNNs methods, deep forest not only provides comparable accuracy but also saves lots of time and resource costs. Besides, deep forest requires fewer parameters and has a good model interpretation. Then, Ming et al. [149] proposed a simple yet effective approach to improve the efficiency of deep forest. The instances with high confidence are directly passed to the final stage rather than passing through all the layers, which significantly reduces time cost and memory requirements.

Deep forest utilises the label vectors produced by random forests and enhances the accuracy of the model by constantly reinforcing the representation learning of each layer. However, the datasets in anomaly detection always lack valid labels or even contain no labels, in which random forests will not work. So, how to efficiently apply deep forest in anomaly detection is still an unexplored field.

Our method tries to solve the above problem by deepening the isolation forests with the cascaded structure. Specifically, our method differs from the deep forest in three aspects: 1) Remove the multi-grained scanning scheme, since this process added more noise into the data under an unsupervised paradigm. 2) Introduce the isolation forest into the cascaded layer rather than the random forest, since the isolation forest depends on unsupervised learning and ensemble learning. 3) Design a tree-embedding scheme to produce more enhanced features, since characteristic diversity plays a big role in deep forest while our approach can only produce a one-dimensional class vector.

5.3 Proposed Approach DeepiForest

In this part, a novel deep isolation forest approach is proposed for anomaly detection. First, the overall structure is introduced and the individual components are analysed. Then, a tree-embedding scheme is designed to enhance the input features, followed by time complexity analysis and comparison.

5.3.1 Cascaded Isolation Forest

In deep neural networks, two processing mechanisms are crucial to the whole model training, including the layer-by-layer training of raw features and powerful feature extraction ability [90, 121]. Inspired by deep forest models, our proposed method also employs cascaded layers of different forests to train the layer-by-layer structure and includes powerful low-dimensional feature representation. Besides, in the new Deep Forest version (DF21), the multi-grained scanning scheme has been removed for tabular data[1]. And the multi-grained scanning also does not fit our method, for producing too much noise to the whole data. To adapt to the applications of anomaly detection, we make other positive improvements to the deep forest: 1) Use three instances (ALSH, L1SH and L2SH) of LSHiForest rather than random forests, changing the supervised paradigm to the unsupervised paradigm for anomaly detection. 2) Combine the tree-embeddings with the class vectors (or anomaly scores), replenishing the one-dimensional output of anomaly detection. The whole structure of DeepiForest is illustrated in Fig. 5.1.

In ensemble methods, diversity is a crucial component, for which different isolation forests and feature representations are applied in each layer. Suppose there are m layers, and each layer contains n different types of forests $\{p_1, ..., p_i, ..., p_n\}$, where p_i is one kind of isolation forest such as L1SH. Initially, the original feature representation "$\boldsymbol{X}$" is used as input. The cascaded layer applies L1SH, L2SH and ALSH to create enhanced features, corresponding to the parameters $\boldsymbol{F_{L1}}$, $\boldsymbol{F_{L2}}$ and $\boldsymbol{F_{AL}}$, respectively.

[1] https://github.com/LAMDA-NJU/Deep-Forest

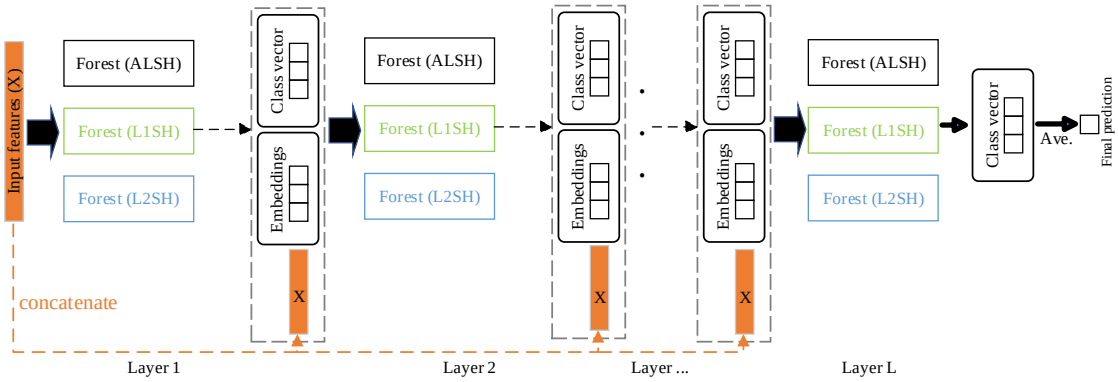


Figure 5.1: The structure of DeepiForest. Suppose each layer consists of three different types of isolation forests (ALSH, L1SH and L2SH). Each forest produces a class vector by anomaly score and an embedding feature by tree-embedding learning, thus, each layer will output a six-dimensional feature vector. Finally, this feature vector is concatenated with the input vector for next layer training.

Besides, tree-embedding scheme is applied in each forest to create a group of embedded features, corresponding to the parameters $\boldsymbol{E_{L1}}$, $\boldsymbol{E_{L2}}$ and $\boldsymbol{E_{AL}}$, respectively. Then, the original feature representation is concatenated with the generated representations as the input of the next layer. Thus, cascaded forests in each layer, except for the first layer, are trained by $\boldsymbol{X} + \boldsymbol{F_{L1}} + \boldsymbol{F_{L2}} + \boldsymbol{F_{AL}} + \boldsymbol{E_{L1}} + \boldsymbol{E_{L2}} + \boldsymbol{E_{AL}}$, where $+$ represents the concatenation of two vectors. The training process is repeated until it reaches the stable layer, shown in Fig. 5.1. Finally, our method employs all the forests $\{p_1, ..., p_i, ..., p_n\}$ in the last layer and averages the final results to predict the anomaly score of data.

It is worth noting that DeepiForest can not produce multi-dimensional class vectors like deep forest, but can only produce one anomalous probability feature. Such a small number of augmented features can provide very limited information enhancement, which is easily overwhelmed by the high-dimensional input features. But such simple augmented features also benefit and it is possible to add more associated features, such as the anomalous probability of the sibling nodes, the number of data instances in the parent node, etc. In our method, we apply a tree-embedding scheme to train more

features, which provide the augmented features as strong as the anomalous probability feature. This is discussed in the next section "Feature Extraction".

Although different isolation forests will work in our framework, the more stable and accurate methods in the cascaded layer will produce more useful features. This is different from the supervised paradigm of deep forest, where strong features can be learned through any decision forest. In DeepiForest, some unsupervised isolation forests will backfire when generating inaccurate features. LSHiForest holds four stable instances (ALSH, L1SH, L2SH, KLSH) and it has been proven to be a generic and robust framework compared with other isolation forest based anomaly detection methods [249]. Thus, we applied the instances of LSHiForest in the cascaded layer. Given the high time complexity of KLSH and the characteristic of a lightweight deep model on DeepiForest, KLSH is not adopted for our cascading layer.

Through the above understanding of our framework, the DeepiForest can be formalised as $(L, P, \mathcal{F})$, shown as follows:

1. $L = \{l_0, ..., l_i, ..., l_m\}$, where l_i is a layer in DeepiForest, containing the cascade of ensemble forests.

2. $P = \{p_1, ..., p_i, ..., p_n\}$, where p_i is one kind of ensemble trees at one layer.

3. $\mathcal{F} = \{\mathcal{F}_1, ..., \mathcal{F}_i, ..., \mathcal{F}_n\}$, where $\mathcal{F}_i$ is a family of functions that can be used in p_i.

In our framework, the hash family $\mathcal{F}_i$ of L1SH and L2SH can be achieved by the randomised hash functions [249]:

$$f_{\boldsymbol{a},b}(\boldsymbol{x}) = \lfloor \frac{\boldsymbol{a} \cdot \boldsymbol{x} + b}{A} \rfloor, \tag{5.1}$$

where $\boldsymbol{a}$ is a random vector with components extracted from the p-stable distribution independently, b is a random constant uniformly drawn from $[0, A]$, and A represents the hashing bucket size defined by users.

Similarly, the corresponding hash family of ALSH can be achieved by [249]:

$$f_{\boldsymbol{a}}(\boldsymbol{x}) = sgn(\boldsymbol{a}^T \cdot \boldsymbol{x}), \tag{5.2}$$

where $\boldsymbol{a}$ is selected from the standard normal distribution $\mathcal{N}(0, 1)$.

Algorithm 6 DeepiForest

Input: $\boldsymbol{X}$-training set, l-the number of cascade levels, $|\mathcal{F}|$-forest types used in the framework.

Output: ud-the DeepiForest model.

1: $\boldsymbol{S} \leftarrow \boldsymbol{X}$ ▷ data sampling
2: **for** i to $l-1$ **do**
3: $\boldsymbol{W} \leftarrow \mathcal{F}_i$ ▷ Initial all the parameters
4: $\mathcal{F}_i \leftarrow$ Calculate Eq. 5.1 and Eq. 5.2 ▷ Train different forest family
5: Compute $\boldsymbol{F_i}$ ▷ Compute the anomaly score features
6: Compute $\boldsymbol{E_i} \leftarrow$ Algorithm 7 ▷ Traverse all trees to calculate the tree-embeddings
7: Cascade $\boldsymbol{X} + \boldsymbol{F_i} + \boldsymbol{E_i}$ ▷ Cascade the input data with enhanced features
8: **end for**
9: Train $\mathcal{F}_l \leftarrow l_l$ ▷ Train different forest families in last layer
10: **return** ud ▷ Return the whole DeepiForest model

Finally, the concrete processes of the proposed DeepiForest are summarised in Algorithm 6.

5.3.2 Feature Extraction

Unlike the classification problem in deep forest, the estimated classes of anomaly detection are only two types: normal or anomalous. So we need to increase the diversity of feature representations in each layer. In our framework, three different isolation forests are applied to generate three-dimensional features and tree embedding is adopted to learn more enhanced features to reinforce representation learning.

Firstly, the example is used to calculate the anomaly score features, which are concatenated with the original feature vector and tree-embeddings as the input to the next layer. As can be seen in Fig. 5.2, each of the three forests will produce a two-dimensional class vector, representing the probability of being predicted as normal and

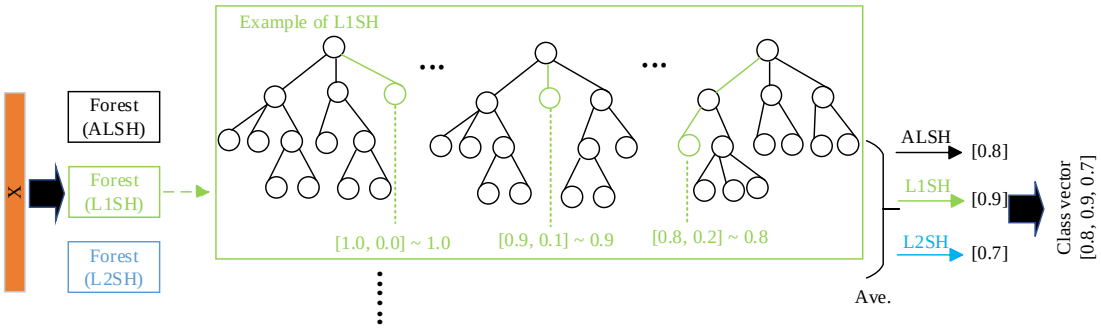


FIGURE 5.2: An example to illustrate the generation of class vectors. Suppose each isolation forest contains multiple isolation trees and each isolation tree will produce two-dimensional feature vector (the probability of anomaly or normal). Average the anomalous probability in all trees to obtain one-dimensional enhanced features in a forest. Then, concatenate all the features to get a three-dimensional label vector.

abnormal, respectively. The weights of these two probabilities add up to 1, so only the abnormal probability works on feature concatenation. Through the above feature learning, the next layer of the cascade will receive $3(= 1 \times 3)$ enhanced features.

Then, tree-embedding scheme is used to generate more features in the trained forest models. There are three steps to acquire the enhanced features [154]. The first step is to convert the tree node into a binary vector, which represents whether one data instance lies in this node or not. Then, these vectors in ensemble trees are concatenated to be a long sequence vector, denoted as $\boldsymbol{V} = [v_1, ..., v_i, ..., v_{|v|}]$ ($|v|$ is the number of nodes in all isolation trees). $\boldsymbol{V}$ is regarded as a new representation of a data instance, where v_i equals 1 if this data instance belongs to node i, otherwise, its value equals 0. In Fig. 5.3, a concrete example is given to show how the vector representation is shaped when a data instance is classified into leaf nodes 5 and 8 in two randomised isolation trees.

However, the above representation learning is very weak and these features contain lots of useless information or inaccurate information. In the second step, we remove non-meaningful nodes and assign weights to different nodes to enhance the features.

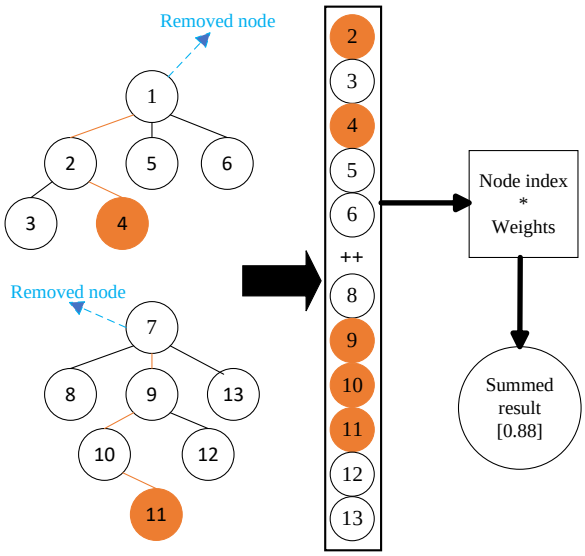


Figure 5.3: An example to illustrate the generation of tree-embeddings. Suppose one isolation tree has six nodes and another isolation tree has seven nodes. All nodes are concatenated to be a vector by removing the useless nodes. Then, assign weights to nodes and add the related nodes to obtain the tree-embedding representation in each forest.

It is obvious that nodes closer to the root contain less information, so these nodes, containing too many data instances (such as more than 90% of all instances), are discarded. Then, different weights are assigned to the remaining nodes according to the number of data instances they hold since the more data a node contains, the less important it is (corresponding to a lower weight). The weight w_i can be calculated as [134]:

$$w_i = \frac{1}{\log(|v_i|) + e}, \tag{5.3}$$

where $|v_i|$ represents the number of data instances that traverse node i, and e is a small positive constant.

The last step is to optimise the tree-embeddings to get the enhanced features. The node vectors are multiplied by their weights, and then new node vectors are added up to be one enhanced feature of the data instance. This method can compress global

TABLE 5.1: Average time complexity of shallow and deep models.

Method	Training	Predicting
L1SH L2SH	$\theta(t\psi \log_k(\psi)m)$	$\theta(tn \log_k(\psi)m)$
ALSH	$\theta(t\psi \log_2(\psi)m)$	$\theta(tn \log_2(\psi)m)$
KLSH	$\theta(t(\lambda^2 m + \lambda^3 + \lambda\psi(m + \log_k(\psi))))$	$\theta(tn\lambda(m + \log_k(\psi)))$
DeepiForest	$\theta(l \cdot (2t\psi \log_k(\psi)m + t\psi \log_2(\psi)m$ $+2tn \log_k(\psi)m + tn \log_2(\psi)m + 3\|v\|))$	$\theta(l \cdot (2tn \log_k(\psi)m + tn \log_2(\psi)m + \|v\|))$

information to a one-dimensional feature representation, corresponding to the first anomaly score feature. Finally, the next layer of the cascade will receive $3(= 1 \times 3)$ tree-embeddings. The concrete steps of tree-embedding learning are summarised in Algorithm 7.

Algorithm 7 Generation of the tree-embeddings

Input: $\boldsymbol{X}$-training set, $\mathcal{F}$-trained forest families, T-the types of forest.

Output: $\boldsymbol{E}$-the tree-embeddings.

1: **for** i to T **do**
2: $\boldsymbol{V} \leftarrow \mathcal{F}$ ▷ Traverse all the trees to get a vector of all nodes
3: $\boldsymbol{V'} \leftarrow \boldsymbol{V}$ ▷ Remove the useless nodes
4: $\boldsymbol{W} \leftarrow$ Eq. 5.3 ▷ Calculate the weight of associated nodes
5: $Sum_w \leftarrow \boldsymbol{W}$ ▷ Sum all the weights of nodes and average it according to the number of trees
6: **end for**
7: Cascade all the tree-embeddings.
8: **return** $\boldsymbol{E}$ ▷ Return a vector of tree-embeddings

Why does the above feature extraction work in our framework? To prevent overfitting, the ensembles and multiple isolation forests are adopted in the models. But isolation forests are different from decision forests, which are supervised learning and

can learn more precise features as the layers get deeper. On the contrary, inaccurate isolation forests will acquire fewer precise features as the layers get deeper since more and more random features join. So, we must add more enhanced features by anomaly score or integrate the tree-embedding result to learn presentations. In this way, the learned features will have stronger and stronger information with increasing layers. This deep learning process tends to be an upper limit, which can be reached with a small number of layers under our unsupervised non-NN framework.

5.3.3 Time Complexity Analysis and Comparison

The time complexity of our model is determined by the forest training process and representation learning, so we have to calculate the time complexity of obtaining the class vectors and tree-embeddings. Then, the time complexity of DeepiForest can be calculated by its layers.

Firstly, the time complexity of producing the class vector is associated with the forest training process and node traverse process. Specifically, the training time of L1SH and L2SH can be calculated as $\theta(t\psi \log_k(\psi)m)$, where t is the number of trees in an isolation forest, k is the branching factor, ψ is the number of sampled data and m is the dimensions of data instances. The training time of ALSH is $\theta(t\psi \log_2(\psi)m)$, since there are only two branches in the ALSH tree. Besides, the node traverse time of L1SH and L2SH can be determined as $\theta(tn \log_k(\psi)m)$, where n represents the number of input data. Correspondingly, the node traverse time of ALSH is $\theta(tn \log_2(\psi)m)$.

The time complexity of obtaining the tree-embeddings is similar to that in the class vector, and they will share the forest training process and node traverse process. So the time complexity of this process contains the forest training time, node traverse time and weights integration time. The last time complexity is associated with the number of nodes $|v|$. Thus, the total complexity of calculating tree-embeddings is $\theta(t\psi \log_k(\psi)m + tn \log_k(\psi)m + |v|)$ in L1SH/L2SH, and $\theta(t\psi \log_2(\psi)m + tn \log_2(\psi)m + |v|)$ in ALSH, respectively.

In our framework, three isolation forests (L1SH, L2SH and ALSH) are used to produce features, and l layers are applied to determine the stable results. In the course

of model training, the former $l-1$ layers all need training and feature learning, but only the last layer performs the training process without feature learning. According to the above analyses, the time complexity of training the whole model can be achieved by:

$$T_{tra} = \theta(l \cdot (2t\psi \log_k(\psi)m + t\psi \log_2(\psi)m + 2tn \log_k(\psi)m + tn \log_2(\psi)m + 3|v|)). \quad (5.4)$$

Then, the time complexity of the whole dataset testing can be achieved by:

$$T_{tes} = \theta(l \cdot (2tn \log_k(\psi)m + tn \log_2(\psi)m + |v|)). \quad (5.5)$$

Through the above analysis, we can make a comparison on the time complexity between our deep method with the shallow method of LSHiForest, so as to understand the gap on these different paradigms. The average time complexity of shallow and deep models is shown in Table 5.1 (λ is a parameter in kernel matrix, such as $\lambda = \sqrt{n}$).

To simplify the analysis, the two time complexities of $\log_k(\psi)$ $\log_2(\psi)$ can be considered close in our applications. Then, the training time complexity of DeepiForest can be simplified to:

$$T_{Dtra} = \theta(3l \cdot (t\psi \log_k(\psi)m + (tn_1 \log_k(\psi)m) + 3|v|)), \quad (5.6)$$

where n_1 is the number of a training dataset. If n_1 is as small as ψ, the time complexity of our model is around $6l$ times slower than that of LSHiForest. If n_1 is very large, the time complexity of our model is around $(3l + 3(l-1)n_1/\psi)$ times slower than that of LSHiForest. Correspondingly, the testing time complexity of the DeepiForest can be achieved by:

$$T_{Dtes} = \theta(3ltn_2 \log_k(\psi)m + (l-1) \cdot |v|), \quad (5.7)$$

where n_2 is the number of a testing dataset. Since the traverse of a trained tree is very fast, the traverse time $(l-1) \cdot |v|$ can be neglected in a big size dataset. So, the time complexity of our model is around $3l$ times slower than that of LSHiForest in the testing process.

5.4 Empirical Evaluation

In this section, we first introduce all the datasets and the basic settings of the experiment. Then, the compared methods are analysed and the evaluation metrics are discussed. To illustrate the performance of our method, extensive experiments and analyses are conducted, and a group of ablation studies is explored.

5.4.1 Datasets and Experiment Setup

Datasets. In the experiments, we collect 20 datasets from public UCI Machine Learning Repository[2] and Kaggle Repository[3]. All datasets are widely used in anomaly detection methods, such as iForest, LSHiForest, RDP, etc. So we apply the same process method as used in previous work to preprocess the datasets and remove the missing values[110, 249, 207]. Besides, these datasets range from a small-scale size of 214 to a large-scale size of 619,326 and hold different dimensions from 3 to 649. The scale of data can be distinguished by the product of data volume and dimension. Then, the description of real-world datasets is shown in Table 5.2.

Experiment Setup. Our framework DeepiForest uses the same cascade structure as gcForest, which contains multiple forests in each layer. Specifically, each layer of DeepiForest consists of L1SH, L2SH and ALSH. The feature vectors of each forest are generated by anomaly score prediction and tree-embedding learning. Besides, there are 100 trees in each forest as recommended in LSHiForest [249]. We also conduct the ablation study on the number of trees in this section, which is set to 10, 40, 70, 100, 130, 160 and 190, respectively. To simplify the calculation, we use the subsampling technique discussed in LSHiForest for all methods and the sample size varies within the range [64, 1024]. In tree-embedding learning, the parameter $p\%$ of removed nodes is set as 90% [134]. Furthermore, the number of cascade levels is set to 4, since more levels do not improve the detection accuracy and waste lots of computing resources.

[2] https://archive.ics.uci.edu/ml/datasets.php

[3] https://www.kaggle.com/datasets

TABLE 5.2: A summary of datasets used in the experiments. Here, "n" in the table denotes the size of the dataset, "m" denotes the dimension of the dataset, and "Rate" represents the proportion of anomalous data in total data instances.

Dataset	n	m	Outlier vs. Inlier Labels	Rate(%)
Glass	214	9	class 6 vs. others	4.20
Ionosphere	351	34	"bad" vs. "good"	35.90
Breastw	683	9	"malignant" vs. "benign"	35.00
Pima	768	8	"positive" vs. "negative"	34.90
Vowel	1,456	12	speaker 1 vs. 6, 7, 8	3.43
Occupancy	6,841	5	"1" vs. "0"	6.24
Skin	9,521	3	"1" vs. "2"	3.67
Wine	5,318	11	class 3, 9, 4 vs. others	4.53
Waveform	3,505	21	class 0 vs. others	4.62
Arrhythmia	452	274	class 3, 4, 5, 7, 8, 9, 14, 15 vs. others	14.60
Mfeat	666	649	class 4, 8, 9 vs. others	9.91
Satellite	6,435	36	class 2, 4, 5 vs. others	31.60
Shuttle	49,097	9	class 2, 3, 5, 6, 7 vs. class 1	7.15
Sensor	5,456	24	"Slight-Left-Turn" vs. others	6.01
Creditcard	284,807	29	"1" vs. "0"	0.17
U2R	60,821	34	"1" vs. "0"	0.37
Census	299,285	500	"1" vs. "0"	6.20
Covertype	581,012	44	class 4 vs. others	0.47
Celeba	202,599	39	"1" vs. "0"	2.24
Donors	619,326	10	"1" vs. "0"	5.92

5.4.2 Baselines and Evaluation Metrics

Baselines. To verify the performance of our method, we compared it with iForest, LSHiForest, RDP [110, 207]. The concrete description of these methods is listed as follows:

- iForest: This is the most classic tree-based anomaly detection method, which is very fast and accurate in many applications.
- LSHiForest: An ensemble framework that is robust and proved to include iForest

and SciForest approaches. There are four instantiations in LSHiForest, including L1SH, L2SH, ALSH and KLSH.

- RDP: A very recent DAD method, which trains neural networks to predict data distances in a randomly projected space and has been confirmed superior to other DAD approaches in extensive datasets.

Evaluation Metrics. We use the Area Under receiver operating characteristic Curve (AUC) as the accuracy evaluation criterion and execution time as the efficiency evaluation criterion. AUC has been extensively adopted in much anomaly detection work and has become an essential measure of accuracy in correlational research [92, 229, 242]. A larger AUC result indicates better performance. To exclude the contingency, all the experiments are run 20 times and take the arithmetic average as the final result.

5.4.3 Results and Discussion

In this part, we evaluate the methods based on a broad range of tasks, which have been widely used in the evaluation of state-of-the-art approaches. In the implementation, we use the same parameter settings in tree-ensemble methods, including the sampling size, the number of trees in a forest, and the height limit of a tree. RDP applies the optimal parameter settings mentioned in its paper.

For the experimental results, we first analyse the average results of all the datasets to find out which method has the best robustness. Then, we did a detailed analysis based on the size of the datasets. We record the AUC results and their standard deviation of DeepiForest and its six competing methods on all datasets. Finally, the test results are summarised in Table 5.3.

It is observed that our method has the biggest average AUC score, which illustrates the excellent robustness on different types of datasets. Besides, L2SH also has better robustness than other shallow anomaly detection methods, which is consistent with the content in the previous paper [249]. To facilitate the analysis of different datasets, we divide datasets into three categories according to the different sizes of the datasets.

TABLE 5.3: AUC (%) performance of different methods on various datasets. Here, the top two AUC results of each dataset and the biggest average result are bold. As we can see, DeepiForest has the best average AUC result and is the most robust method over various datasets. Our method has better accuracy than the shallow anomaly detection methods in most datasets and is comparable to the DAD methods in large-scale datasets.

Dataset	iForest	ALSH	L1SH	L2SH	KLSH	RDP	DeepiForest (ours)
Glass	69.14 ± 0.25	**89.94** ± 0.70	78.40 ± 0.15	79.68 ± 0.13	82.66 ± 1.01	60.98 ± 0.64	**82.75** ± 0.55
Ionosphere	83.55 ± 0.35	**92.22** ± 0.50	84.84 ± 0.17	91.26 ± 0.15	89.41 ± 0.36	82.70 ± 0.99	**93.07** ± 0.80
Breastw	94.10 ± 0.22	88.27 ± 0.64	**97.96** ± 0.19	**97.82** ± 0.28	95.28 ± 1.52	96.99 ± 0.61	95.24 ± 0.88
Pima	66.51 ± 0.42	61.84 ± 1.81	69.33 ± 0.67	**70.37** ± 0.46	69.61 ± 0.62	67.17 ± 0.85	**70.47** ± 0.76
Vowel	79.50 ± 0.16	**95.14** ± 0.79	81.32 ± 0.82	90.65 ± 0.66	92.17 ± 2.10	67.23 ± 1.16	**96.28** ± 1.06
Occupancy	96.12 ± 0.36	91.09 ± 0.78	**96.80** ± 0.55	96.03 ± 0.57	96.12 ± 0.82	**98.13** ± 0.77	96.37 ± 0.63
Skin	62.02 ± 0.68	72.66 ± 0.62	**78.74** ± 0.83	72.28 ± 0.78	**77.52** ± 1.56	45.75 ± 0.81	76.85 ± 0.85
Wine	64.40 ± 0.11	59.66 ± 0.64	64.17 ± 0.38	66.27 ± 0.10	**67.07** ± 0.96	61.66 ± 0.35	**68.41** ± 0.49
Waveform	**72.26** ± 0.20	51.41 ± 1.08	67.28 ± 0.71	70.30 ± 1.02	**73.56** ± 1.21	66.87 ± 0.91	68.86 ± 0.98
Arrhythmia	**79.89** ± 0.55	65.22 ± 0.93	78.94 ± 0.85	77.08 ± 0.87	**80.06** ± 0.41	75.52 ± 0.53	77.56 ± 0.89
Mfeat	89.11 ± 0.88	84.97 ± 0.87	69.57 ± 0.96	82.62 ± 0.84	**92.19** ± 1.02	89.23 ± 0.97	**92.28** ± 0.86
Satellite	70.65 ± 0.81	62.15 ± 0.86	**76.10** ± 1.68	**76.11** ± 0.83	72.83 ± 2.12	60.85 ± 0.56	72.42 ± 0.44
Shuttle	**99.28** ± 0.13	74.29 ± 0.19	94.75 ± 0.62	**96.81** ± 0.83	91.27 ± 1.16	99.20 ± 0.93	96.16 ± 0.60
Sensor	**65.88** ± 0.43	47.21 ± 1.06	60.08 ± 0.95	**61.91** ± 0.62	52.27 ± 1.37	56.85 ± 1.03	60.77 ± 0.97
Creditcard	**94.74** ± 0.54	71.37 ± 1.06	83.54 ± 0.95	80.83 ± 0.91	50.09 ± 1.82	**95.42** ± 0.58	90.04 ± 0.83
U2R	**98.74** ± 0.21	98.69 ± 0.34	98.58 ± 0.23	98.70 ± 0.42	98.25 ± 0.53	98.57 ± 0.26	**99.02** ± 0.45
Census	58.82 ± 1.86	57.40 ± 1.21	**63.38** ± 0.90	62.78 ± 0.97	59.41 ± 1.36	**65.12** ± 0.42	61.96 ± 1.47
Covertype	**95.86** ± 0.76	93.27 ± 1.19	92.41 ± 0.45	92.27 ± 1.03	93.10 ± 0.89	51.15 ± 1.33	**93.77** ± 1.56
Celeba	69.24 ± 1.26	64.91 ± 0.98	72.48 ± 1.52	71.34 ± 1.41	69.02 ± 1.06	**85.76** ± 0.68	**72.96** ± 1.63
Donors	77.32 ± 0.91	69.91 ± 0.86	**78.88** ± 1.16	74.13 ± 1.07	75.24 ± 1.68	**95.64** ± 1.20	77.44 ± 1.13
Ave AUC	79.43 ± 0.55	74.52 ± 0.86	79.38 ± 0.74	80.46 ± 0.70	78.91 ± 1.18	76.04 ± 0.78	**82.14** ± 0.89

There are two lines dividing the datasets, shown in Table 5.3. Firstly, the product of data volume and dimension is less than 50,000 on the small-scale datasets. We can observe that our method DeepiForest has good AUC results in most datasets, especially in three datasets DeepiForest ranks the top one, while other shallow methods behave similarly. Although RDP performs poorly on small-scale datasets, it has not been the focus of DAD methods, and it is known to all that the traditional deep learning methods will cause overfitting on small-scale datasets. Herein, we just want to show

that our method performs well on small-scale datasets, compared with the shallow method. Secondly, the product of data volume and dimension on seven datasets ranges from 50,000 to 500,000, which is regarded as the medium-scale datasets. All the AUC results of DeepiForest are pretty close to the maximum values in these seven datasets. Besides, other methods perform similarly on these datasets except for ALSH. The key point why DeepiForest can not rank the top one in these six datasets is that ALSH is one forest in the cascaded layer of DeepiForest. The features produced by ALSH forest will influence the detecting results of DeepiForest. Finally, the rest six datasets hold more than 50,000 records and 10 dimensions. In these datasets, DAD methods are more competitive since RDP ranks in the top two in four datasets. But DeepiForest is comparable in large-scale datasets since DeepiForest ranks in the top two in three datasets and it does not perform poorly in any datasets.

The experimental results have achieved our expected goal, that is, the accuracy is comparable to the traditional DAD methods and more robust than the state-of-the-art methods. Because the deep anomaly detection method with the non-neural network is only a trial, simultaneously the accuracy can be further improved.

Next, experiments are conducted on the execution time, which is also an important indicator for anomaly detection. In real applications, it will cause huge losses if the anomaly detection is not timely, such as in network intrusion detection and bank fraud detection. Herein, we make a time comparison on all datasets, especially for the comparison between our method and the DAD method (RPD). The execution time of all datasets is summarised in Table 5.4.

Table 5.4 illustrates most shallow methods have less execution time, except for KLSH, than the deep anomaly detection methods. These results are consistent with the above time complexity analysis. When the size of the dataset is not very large, DeepiForest is more advantageous than RDP, since the ensemble isolation forest utilises the sampling technique in the training stage and has a fast retrieval capability. However, tree construction will become more complex with the increase of data dimension, or testing time will greatly increase with the larger data volume. So, the execution

TABLE 5.4: Comparing execution time (s) of all methods. As we can see, udForst gets a comparable execution time to the shallow method of KLSH and outperforms RDP in most datasets.

Dataset	iForest	ALSH	L1SH	L2SH	KLSH	RDP	**DeepiForest**
Glass	0.2	3	2	3	13	14,745	32
Ionosphere	0.4	4	2	3	39	15,145	36
Breastw	0.4	4	4	6	50	13,662	46
Pima	0.4	4	3	7	57	13,664	51
Vowel	0.4	11	8	12	79	14,971	108
Occupancy	1	23	49	39	620	17,856	705
Skin	11	96	82	186	1,401	18,182	2,184
Wine	0.9	20	42	38	588	18,451	673
Waveform	0.8	13	10	16	196	15,356	137
Arrhythmia	0.8	12	8	8	662	24,148	101
Mfeat	1	26	10	12	955	26,957	169
Satellite	0.7	7	5	5	296	18,597	156
Shuttle	63	139	144	196	2,031	19,710	2,814
Sensor	2	9	8	8	330	17,951	183
Creditcard	24	1,332	751	832	29,644	34,549	26,002
U2R	6	269	336	358	5,214	20,740	6,425
Census	359	1,872	1,043	1,479	48,261	35,961	40,354
Covertype	70	4,045	3,450	5,275	70,435	25,886	49,584
Celeba	23	1,283	735	1,283	24,215	32,106	32,815
Donors	35	4,146	4,236	5,962	48,414	30,620	61,253

time of DeepiForest increases greatly in the last six datasets, where RDP even takes a similar execution time.

Through the above experiments, it can be summarised that DeepiForest is much more robust across all datasets and obtains comparable AUC results with the state-of-the-art DAD method (RDP) in large-scale datasets. Besides, the execution time of DeepiForest is still pretty good and faster than RDP in most datasets.

5.4.4 Ablation Study

To better demonstrate the impact of segmented parts on the whole method, we compare the effects of different parameters on the following three components: the tree-embeddings, the number of trees in a forest and the number of cascaded layers. Three datasets ("Glass", "Wine" and "Creditcard") of different sizes are selected to show the results of DeepiForest. Specifically, "Glass" dataset only contains 214 data instances, which represents the small-scale datasets. "Creditcard" dataset contains 284,807 data instances, which represents the large-scale datasets. "Wine" represents a compromise in all datasets. The results are the mean value of 20 folds measured in the comparison experiments.

Compared with gcForest, our method can not produce many enhanced features through label learning of forests. So, we apply the tree-embedding scheme to learn more features in each forest. To validate the effectiveness of the tree-embeddings, we make a comparison between DeepiForest and its variant. We name the model that just produces the class vector as DeepiForest_1, and generates two types of enhanced features as DeepiForest. The experimental results are shown in Table 5.5.

Table 5.5: The influence of cascaded features on DeepiForest.

#	DeepiForest_1	**DeepiForest**
Glass	81.85 ±0.52	82.75 ±0.55
Wine	68.26 ±0.35	68.41 ±0.44
Creditcard	89.28 ±0.46	90.04 ±0.59

The results in Table 5.5 show that DeepiForest has better accuracy than that of DeepiForest_1. Since more enhanced features will strengthen the capability on representation learning of our forests, which has been explored in the cascaded forest. DeepiForest applies both label learning and tree-embedding learning to obtain various enhanced features while DeepiForest_1 only adopts the class vector for model training.

Then, we will explore the influence of the number of trees in each forest. The experiments are conducted on three different sizes of datasets, including "Glass", "Wine" and "Creditcard". The parameters are set as 10, 40, 70, 100, 130, 160, 190, which produce different AUC results, shown in Fig. 5.4.

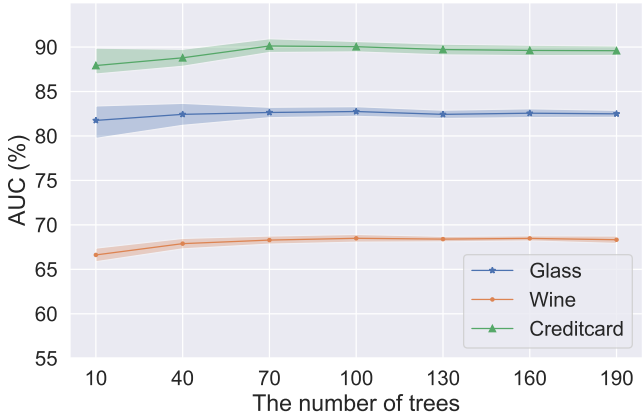


FIGURE 5.4: The influence of the number of trees in each forest.

From Fig. 5.4, we can find that the number of trees will affect the AUC results and fluctuation in all datasets. The results fluctuate greatly as the number of trees is small, while the results are very stable as the number of trees is large (> 70). Besides, the AUC results are not so good when the number of trees is 10 and 40. Thus, we set the tree number as 100, which is adopted in most isolation forest based methods.

The layers in DeepiForest are another important impact factor, which can not just add depth to obtain better results. As an unsupervised learning model, DeepiForest does not know when to stop training like gcForest. Besides, the training results tend to be stable or even get worse as the number of layers continues to increase. So, we try to find the critical point that most datasets can perform well on such layers. The layers in our experiment include both training and final testing layers and the experiments are conducted on "Glass", "Wine" and "Creditcard" datasets. Then, the relationship between AUC results and the number of layers in DeepiForest is shown in Fig. 5.5.

Fig. 5.5 presents the AUC results of different datasets get better as the number of cascaded layers increases and gradually becomes stable. Specifically, "Glass" and

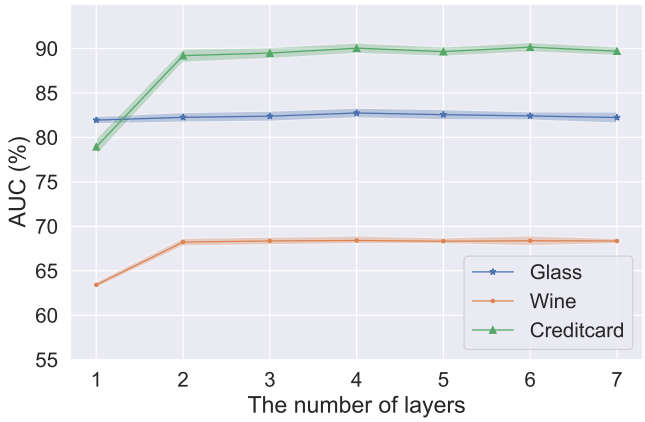


Figure 5.5: The influence of the number of layers in DeepiForest.

"Wine" datasets have poor results when DeepiForest just has one layer and the AUC results reach the peak when the number of layers is 4 in "Glass" and "Creditcard" datasets. Since the first layer does not produce the enhanced features for training, the results with just one layer are always poor. In our experiments, 4 layers are adopted to produce stable and high performance.

5.5 Conclusion

To solve the problems of traditional DNNs methods in anomaly detection, such as too many parameters, long training time and fixed number of layers, we first introduce the structure of deep forest into anomaly detection. Then, an unsupervised deep isolation forest is proposed to provide another efficient and effective DAD paradigm. Specifically, DeepiForest draws the merits of ensemble learning and high efficiency in LSHiForest to construct cascaded layers, and tree-embedding scheme to obtain more enhanced features. These two processes not only make DeepiForest keep the advantages of the deep forest but also enable it to gain high efficiency and accuracy in anomaly detection. Finally, extensive experiments are conducted on different scales of datasets. It is validated that DeepiForest outperforms the state-of-the-are DAD methods in efficiency and obtains comparable effectiveness to the state-of-the-are DAD methods on large-scale datasets. Moreover, we analyse the impact of segmented parts on the whole approach

through different ablation studies. Three components of comparative learning on the tree-embeddings, the number of trees in a forest and the number of cascaded layers provide us with a comprehensive view of DeepiForest.

In the future, we aim to obtain more effective features to enhance the robustness of the model by diverse representation learning. Besides, model efficiency and memory consumption can also be further reduced through layer control, parallelization technique or algorithm optimization.

6

Deep Optimal Isolation Forest with Genetic Algorithm for Anomaly Detection

Deep anomaly detection methods are considered as the effective approaches for addressing complex anomaly detection problems. Among them, the deep isolation forest methods have gained rapid development recently due to their simplicity in parameter turning and efficiency in model training. The existing deep isolation forest approaches are all based on representation learning, while OptiForest theoretically proves the crucial role of the tree structure in isolation forest based methods. In this chapter, we analyse the search space of isolation trees under specific data instances and address the challenges in finding optimal isolation forest, corresponding to the fourth problem in this thesis. Based on the theoretical underpinning and genetic algorithm, we design a deep model DOIForest with two mutation schemes and solution selection, which

learns the optimal isolation forest and optimises the parameters in data partitioning. Extensive experiments on both synthetic dataset and a series of real-world datasets demonstrate that our approach can achieve better detection accuracy and robustness than the state-of-the-arts.

6.1 Overview

Anomaly detection, also known as outlier detection, aims to identify abnormal behaviours or sparse objects in data that deviate from the expected behaviors or the majority of objects, which is currently a significant task in machine learning [195]. This technique has been widely applied in various fields, including intrusion detection in cybersecurity [6], financial fraud detection [30], and health or medical risk detection [43]. Although anomalies typically make up the rare part of the whole dataset or occur infrequently, the failure to detect them in a timely manner can have severe consequences such as enormous economic loss in finance and disclosure of military secrets. Therefore, a diverse range of unsupervised anomaly detection methods [164], including shallow and deep approaches, have been proposed to address the complex data formats and different types of anomalies.

Constrained by the limited availability of labels and the high cost associated with manual annotation, we herein focus on the unsupervised anomaly detection that is more commonly-used in real applications. Some typical deep anomaly detection (DAD) methods, like REPEN [143] and RDP [207], have demonstrated a powerful ability in representation learning and achieved good detection accuracy in anomaly detection. But there are still some limitations for these DAD methods, such as high computational cost, poor explainability, and difficulty in hyperparameter tuning [103]. A recent study [231] has attempted to simplify the complexity of the deep learning process by introducing isolation forest into deep learning models and employing randomised neural networks to learn features. Subsequently, a deep detector based on isolation forest was developed for anomaly detection. This method achieves good performance both in efficiency and effectiveness on a variety of datasets compared with many traditional

neural network based DAD methods.

The above deep isolation forest provides us with inspiration to move beyond DAD methods to isolation forest based paradigm. The proposed deep isolation forest applies randomly projected representations rather than the fine trained one for prediction [231], although we intuitively think more learning will strengthen the detection model. Another isolation forest based deep learning approach is DeepiForest [224], which extends deep forest to anomaly detection and applies deep cascaded layers to enhance the feature representation to achieve higher detection accuracy. These two methods deviate from the traditional approaches of using hash mapping to learn representations in isolation forest methods, while they employ deep isolation forest models to enhance the representation learning. However, they have not take the optimization of the isolation tree structures into consideration and the detection performance of isolation forest methods is determined by both feature representations and isolation tree structures. Instead, the powerful feature representations can be achieved to some extent by traditional deep neural networks as well.

Given the nature of the isolation tree structures, we need to review the shallow methods of isolation forest. The most classic isolation forest approach is iForest [110], which is quite fast and robust for anomaly detection and has been widely applied in industry, e.g., it has been included in scikit-learn, a commonly-used machine learning library in Python. Benefiting from ensemble learning [5], isolation forest based methods stand out of the shallow models and get the rapid development. Some extensions to iForest, like SciForest [111] and extended isolation forest [63], improved the hash learning scheme or anomaly score calculation to achieve better detection accuracy. While most extensions following iForest use the binary tree structure for data partitioning, a framework LSHiForest [249] changes the isolation tree structure by producing multi-fork isolation trees and indicates that iForest is one of its prominent instances. Based on the observation and analysis to the influence of different branching factors, a very recent work OptIForest [225] is proposed to determine the optimal branching factor with e and designs an applicable detector to learn the optimal isolation forest by learning to hash scheme. However, the produced tree in OptIForest is not an exact optimal

isolation tree since it is very hard to obtain the optimal isolation forest, limited by the huge search space of isolation trees and the discrete solution space.

In this work, we investigate the above complex optimization problem and propose a deep optimal forest named DOIForest with improved genetic algorithm [114] to search the optimal solution. Specifically, two mutation schemes are designed according to the observations that the isolation tree structure will be influenced by the sampling data and the inner data partitioning of an isolation tree. Besides, the selection scheme are determined by the isolation efficiency, which is associated with the hardness of distinguishing data instances isolated by a tree. Extensive experiments are conducted on a variety of benchmarking datasets in our ablation and comparative studies. The experimental results validate DOIForest outperforms the state-of-the-arts and achieves the robust effectiveness. The main contributions of this research are three-fold:

- It is the first work to optimise the isolation tree structure through the deep learning model. We make theoretical analysis for the search space in our problem and design a genetic algorithm based deep model to solve the discrete optimization problem.
- We design two mutation schemes for isolation tree evolution and use the isolation efficiency to select the optimal isolation forest in our approach.
- Extensive experiments illustrate the effectiveness and robustness of our method. We conduct a series of ablation studies and make analysis on how the number of layers and the number of trees in a forest influence the experimental results. Our source code is available at https://github.com/xiagll/DOIForest.

6.2 Related Work

There have been a great number of unsupervised anomaly detection methods proposed, including statistical methods, similarity-based, ensemble-based, and deep learning-based methods [27, 146]. In this section, we review the most relevant methods to our research.

Deep Learning-based Methods There has been significant exploration of deep neural networks for anomaly detection, especially when dealing with complex data types [242, 30]. The widely-used methods, including generative adversarial networks (GANs) [62], AutoEncoders [267], and reinforcement learning [245], have been harnessed to improve the performance of anomaly detection. These methods offer promising capabilities to better identify and classify anomalies in diverse datasets. For example, a random distance-based anomaly detection approach, named REPEN [143], is proposed by learning low-dimensional features from ultrahigh-dimensional data. The key point of REPEN is to optimise representations by significantly increasing the nearest neighbor distance between pseudo-label anomalies rather than the normal instances. Besides, Wang et al. [207] presented a deep anomaly detection approach, which uses random mapping to map data distances within a randomly projected space and trains neural networks to predict data distances in this new space. Although these methods can achieve high detection accuracy in anomaly detection [148, 60], they often suffer from issues like complex parameter learning, expensive computations, etc. To explore new non-neural network based deep models, Xiang et al. [224] proposed a deep isolation forest method for anomaly detection, which incorporates the concept of deep forest and utilises the cascaded structure to reduce the layers in deep model. A recent work [231], named DIF, extended the isolation forest to deep paradigm, which applies random neural networks to learn powerful representations in each layer of the isolation trees and achieves fast detection with high accuracy. These works explore new deep models based on isolation forest, and leverage the power of deep isolation forest to learn stronger representations for anomaly detection. However, they have not learned deep models based on the tree structure while the tree structure of isolation forest plays a crucial role on the effectiveness for anomaly detection.

Ensemble-based Methods Benefiting from ensemble learning scheme, many classical anomaly detection methods are integrated with it, including ensemble LOF (EnLOF) [266], isolation using Nearest Neighbour Ensemble (iNNE) [13], and average k-NN distance ensemble (EnKNN) [5]. These methods typically have high time complexity when

dealing with large-scale datasets. Isolation forest based methods stand out of the above ensemble methods, e.g., iForest [110] and SciForest [111]. These methods can provide robust detection accuracy with better efficiency based on the principle that anomalies are more likely to be isolated from others, i.e., anomalies often obtain shorter path length in the isolation trees. Then, a generic framework LSHiForest [249] is proposed through applying locality-sensitive hashing (LSH) and extending the binary tree structure in iForest to multi-fork isolation trees. There is an interesting observation that iForest and SciForest can be identified as a significant instance of LSHiForest. But these works have not investigated which is the optimal tree structure for anomaly detection. Xiang et al. [225] proposed a theoretical work that proves the optimal isolation tree structure with the branching factor e. They also design a practical optimal isolation forest, named OptIForest, which can achieve both high computational efficiency and robust detection accuracy in a variety of datasets. However, it is highly challenging to implement the real optimal isolation forest in practical algorithms, as OptIForest fails to provide effective optimization solutions.

Unlike neural network-based techniques, optimizing the structure of isolation trees is a complex discrete optimization problem, which inherently makes it more challenging. There have been many discrete optimization techniques [264, 114], such as integer linear programming, genetic algorithms, and simulated annealing, aiming to find the optimal solutions within discrete solution spaces. To achieve a complete deep isolation forest, it is essential to optimise the structure of the isolation trees, which can simultaneously optimise the representations in each isolation tree.

6.3 Preliminary and Problem Statement

6.3.1 The Optimal Isolation Forest

Through the observation and theoretical analysis to the branching factor of the isolation trees, Xiang et al. [225] proposed an optimal isolation forest (OptIForest) for anomaly detection. OptIForest determines the optimal branching factor with Euler's number

e and incorporates clustering based learning to hash to learn more data information for better isolation quality. Specifically, OptIForest defines the isolation efficiency to represent the hardness of distinguishing data instances isolated by a tree, where the higher isolation efficiency leads to stronger distinguishability, i.e., better anomaly detection performance. Given ψ data instances, the isolation efficiency η can be obtained by [225]:

$$\eta = \frac{\psi}{vd}, \tag{6.1}$$

where v can be regarded as the average branching factor of the internal nodes in an isolation tree and d is the average height of the leaf nodes. Besides, OptIForest designs a learning detector with agglomerative hierarchical clustering based learning to hash, but the practically produced tree is not an exact optimal isolation tree since the OptIForest method takes the trade-off between accuracy and efficiency. Compared with iForest and LSHiForest, OptIForest is more natural with the optimal tree structure by discarding the original binary splitting or multi-fork splitting. It is noting that the isolation forest based methods can be applied in OptIForest as the pre-trained tree structure for further optimization.

6.3.2 Problem Statement

When constructing all possible isolation tree structures using ψ data points and random branching factors, we find that the search space $S(\psi)$ of isolation trees is limited but extremely huge if ψ is a big value. Thus, it is impossible to list all possible isolation trees and to find some optimal or near optimal trees. For example, if $\psi = 0$, the search space $S(\psi) = \varnothing$; if $\psi = 1$, $S(\psi) = \{t_i\}$, where t_i represents an isolation tree under the corresponding search space; if $\psi = 2$, $S(\psi) = \{t_i\}$; if $\psi = 3$, $S(\psi) = \{t_i, t_j\}$, it is noting that the symmetrical tree structures are regarded as the same tree structure since the distinguishability of data is same in the symmetrical tree structures. To illustrate the search space in our problem, we use a simple example of three data instances to construct the different tree structures, shown in Fig. 6.1. We use different parameters to construct different isolation trees t_1, t_2, t_3 and t_4, corresponding to the sub-figures

(a), (b), (c) and (d) in Fig. 6.1, respectively. Although data instances are separated into the different leaf nodes in Fig. 6.1 (a) and (b), the distinguishability of two isolation trees are the same and these two structures are regarded as the same structure, i.e., with the same average depth and average branching factor, in our search space. The symmetrical tree structures in Fig. 6.1 (b) and (c) also have the same distinguishability for isolated data, which decides the symmetrical tree structures to be the same one. Only the tree structure in Fig. 6.1 (d) is different to others. Thus, the size of the search space is two when isolating three data instances with different isolation tree structures. Keep increasing the number of data instances, $|S(\psi)|$ increases to 5, 12 and more. Given ψ data instances, the isolation tree structure can be binary tree, ternary tree to ψ-fork tree, so we can derive a relationship between the size of search space and the number of data instances as:

$$\begin{aligned} |S(\psi)| &> \sum_{i=1}^{\psi-3} i|S(\psi-i)| > \sum_{i=1}^{\psi-4} i|S(\psi-i-1)| + \sum_{i=2}^{\psi-3} i|S(\psi-i)| \\ &> 2^1 \sum_{i=1}^{\psi-3} i|S(\psi-i-1)| > ... > 2^{\psi-5}(|S(4)| + 2|S(3)|), \end{aligned} \tag{6.2}$$

using the results in the above analysis that $|S(4)| > 2|S(3)|$, the Equ. (6.2) can be calculate by:

$$|S(\psi)| > 2^{\psi-5}(4|S(3)|) = 2^{\psi-3}|S(3)|, (\psi > 3) \tag{6.3}$$

the search space grows exponentially relative to the size of the data, so it is impossible to search for all possible isolation trees to obtain the optimal one. Furthermore, finding the optimal isolation tree is a discrete optimization problem and many existing methods, like the gradient decent and Newton's method, cannot solve this complex problem. Therefore, it is still a challenging and pressing problem on how to search the optimal isolation trees in practical algorithm.

6.4 Methodology

In this part, we present the theoretical analysis and propose a novel method for the above problem. Specifically, we propose a genetic algorithm-based approach to address

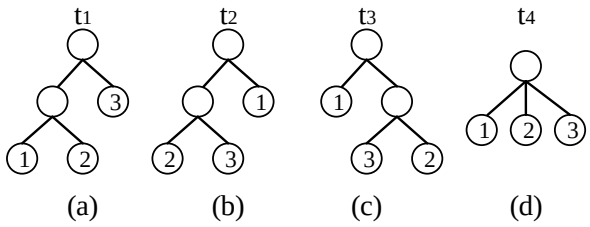


FIGURE 6.1: Isolating three data instances with different data partitioning.

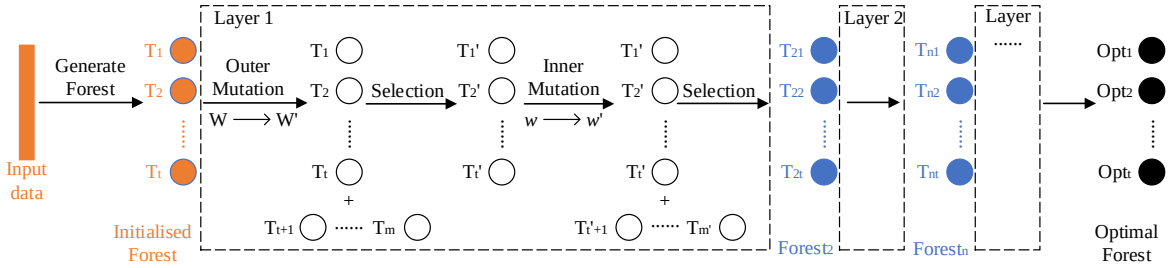


FIGURE 6.2: The structure of DOIForest. The initialised isolation forest is generated by the input data and used as the beginning population for mutation in the first layer of our deep model. Two types of mutation schemes are designed for isolation forest and selection scheme is performed by the objective function. Finally, the optimal forest is output through iterative evolution.

the aforementioned problem, and then analyze its corresponding time complexity.

6.4.1 Theoretical Analysis

In real algorithm implementation, the tree structure is determined by the parameters in data partitioning. In an isolation tree, we can define the relationship between the average branching factor and data partitioning parameter w as:

$$\overline{v} = \frac{\sum_{i=1}^{In} f_i(\boldsymbol{w_i})}{In} = E(F(\boldsymbol{W})), \tag{6.4}$$

where In represents the number of internal node of the isolation tree, $\boldsymbol{W} \sim (\boldsymbol{w_1}; \boldsymbol{w_2}; ...; \boldsymbol{w_\psi})$ represents a group of discrete variables, $f_i(\cdot)$ is a nonlinear function, and $v_i = f_i(\boldsymbol{w_i})$, i.e., the branching factor of an internal node is determined after data instances are divided into different nodes.

The average depth is determined by the depths of all data instances, which is also fixed after data partitioning. Thus, the average depth can be calculated by:

$$\overline{d} = \frac{\sum_{i=1}^{\psi} h_i(\boldsymbol{w_i})}{\psi} = E(H(\boldsymbol{W})), \tag{6.5}$$

where $d_i = h_i(\boldsymbol{w_i})$ and $h_i(\cdot)$ represents a nonlinear function relationship of $\boldsymbol{w_i}$.

Since the isolation tree structure is associated with the branching factor and the depth, we can infer that the isolation tree structure is determined by the data partitioning parameters (or the parameters in hash function of the hash-based isolation forest methods). Next, the isolation efficiency can be calculated by:

$$\eta = \frac{\psi}{\overline{v} \cdot \overline{d}} = \frac{\psi}{E(F(\boldsymbol{W})) \cdot E(H(\boldsymbol{W}))}, \tag{6.6}$$

The optimal isolation tree has the biggest isolation efficiency, i.e., the branching factor is equal to e, as analysed in OptIForest [225]. However, the optimization to isolation efficiency is a complex discrete optimization problem according to Equ. (6.6). The backpropagation-based neural network is not suitable to solve our problem. Thus, we design a genetic algorithm with two designed mutations to evolve the isolation tree and extend the traditional optimal isolation tree into the deep paradigm.

In our method, the final object is to produce a group of optimal isolation trees that hold the optimal isolation efficiency. Given an isolation forest that contains t isolation trees, the whole isolation efficiency of the isolation forest is calculated by:

$$\hat{\eta} = \sum_{i=1}^{t} \eta. \tag{6.7}$$

In each iteration of our deep model, we will mutate the parameters $\boldsymbol{W} \sim (\boldsymbol{w_1}; \boldsymbol{w_2}; ...; \boldsymbol{w_\psi})$ to produce new tree structures, and then the corresponding isolation efficiency is calculated and selected. With the above analysing, we can formulate the objective function by optimizing the data partitioning parameters $\boldsymbol{W}$ as follows.

$$\boldsymbol{W}^* = \max \hat{\eta}(v, d) = \max \hat{\eta}(\boldsymbol{W}) = \max \sum_{i=1}^{t} \frac{\psi}{E(F(\boldsymbol{W})) \cdot E(H(\boldsymbol{W}))}, \tag{6.8}$$

where $\hat{\eta}(\boldsymbol{W})$ is the function of isolation efficiency with respect to the data partitioning parameters and $\boldsymbol{W}$ is constitute with $\boldsymbol{w_i}$ in each level of an isolation tree. The parameter $\boldsymbol{w_i}$ is randomly produced for data partitioning and the search space of isolation

trees is huge according to the analysis in problem statement. Besides, all parameters of $\boldsymbol{W}$ are optimised in one process, corresponding to construct a complete isolation tree, which can not be solved by gradient descent. In the following content, we propose a method to maximise the objective function and solve the discrete optimization problem.

6.4.2 DOIForest: Deep Detector Design with Genetic Algorithm

In this part, we investigate how to implement a practical deep optimal isolation forest. The first step is to initialise an isolation forest with the traditional isolation forest based methods, like iForest and LSHiForest. It does not matter which kind of isolation forest is used in the initialization since we will evolve the forest to be the optimal one. Then, we will use the mutation schemes to change the parameters of the tree structure and select the optimal or near-optimal ones according to the objective function. The whole structure of our method is shown in Fig. 6.2.

The input data of our method is tabular data but can be any fields, including finance, medicare, government, etc. Because deep neural networks can provide powerful representation learning and transfer pictures or text data into multi-dimensional features (i.e., the tabular data), our method can be connected with the produced features for deep anomaly detection. DOIForest can be separated into three stages: The first stage is the pre-training stage, which initialises the isolation forest through the input data and these initialised trees in the forest can be regarded as the populations. The second stage is the training stage, which applies the improved genetic algorithm to create a deep model for anomaly detection. The key point in each layer is to apply two mutation schemes to search the space of different isolation tree structures and constantly approach the optimal or near-optimal tree structures. The last stage is the testing stage, which adopts the trained optimal forest to obtain the anomaly score for data instances and predict the anomalies.

Finally, the concrete processes of DOIForest are summarised in Algorithm 8.

Algorithm 8 Constructing the DOIForest.

Input: A dataset (sample) $\boldsymbol{D}$ of size ψ, outer mutation rate r_1, inner mutation rate r_2, the number of trees t and the number of layers l.

Output: F_{Opt}-an optimal isolation forest.

Train LSHiForest to get the initialised forest F_{LSH}; ▷ The pre-training stage
for i to l **do** ▷ The training stage
  for j to t **do**
    $r' \leftarrow$ generate a random rate from $[0, 1]$;
    if $r' \leqslant r_1$ **then** ▷ The outer mutation
      $\boldsymbol{W}' \leftarrow \boldsymbol{W}$, mutate a new isolation tree T_j by a new sampling;
      $F'_{Opt} \leftarrow F_{Opt} \cup T_j$;
    end if
  end for
  $F^i_{Opt} \leftarrow$ Select t near-optimal trees from F'_{Opt};
  for g to t **do**
    Traverse all nodes in T_g from a root node r_d;
    initialise the current node $c_d = r_d$;
    while c_d has children **do**
      $r' \leftarrow$ generate a random rate from $[0, 1]$;
      if $r' \leqslant r_2$ **then** ▷ The inner mutation
        $\boldsymbol{w}' \leftarrow \boldsymbol{w}$, mutate c_d to c'_d by $\boldsymbol{w}'$;
      else
        $c_d \leftarrow$ get its children c_{dr};
      end if
      $F^{i\prime}_{Opt} \leftarrow F^i_{Opt} \cup T'_g$;
    end while
  end for
  $F^i_{Opt} \leftarrow$ Select t near-optimal trees from $F^{i\prime}_{Opt}$;
end for
return The optimal isolation forest F_{Opt}.

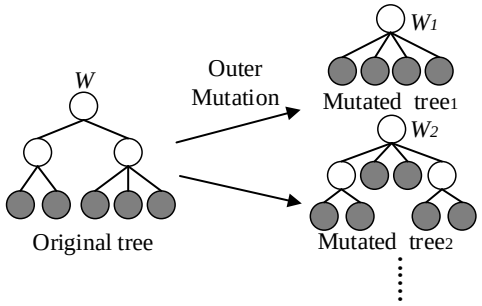


Figure 6.3: An example to illustrate the process of the outer mutation. The parameters in data partitioning are completely changed and new tree structures are produced by the new sampling datasets.

6.4.3 Mutation Scheme

As analysed in problem statement, the search space of different tree structures is so huge that we can not list all possible cases to find the optimal one. Besides, the optimization to tree structure is discrete, so deep neural networks (DNNs) are not directly applicable for our problem. By using genetic algorithm, the problem can be more effective to escape from the local optimal solution compared with the gradient descent algorithm used by neural networks. In our approach, the mutation and selection to the isolation tree structure correspond to the gradient decent process in neural networks and one layer in our approach corresponds to one layer network in DNNs. The difference is DNNs aim to learn powerful representations of original data, while our approach aims to learn the optimal solutions (or the optimal isolation forest) for data prediction.

The isolation tree structure is fundamentally influenced by two input features: the sampling data and the branching factor. So our two mutation schemes are designed according to the change of these two features. The first mutation is the outer mutation, which aims to add new populations for selection. Because the mutation types or search space of isolation trees are limited under the fixed sampling data. To escape from the local optimum, we need to add new sampling data to create outer mutation for a bigger search space. In Fig. 6.3, a concrete example is given to show how the outer mutation happens.

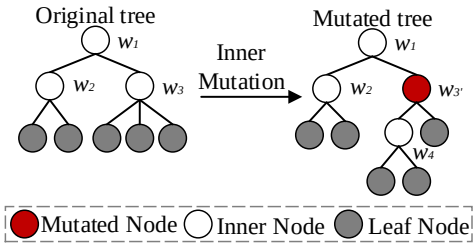


Figure 6.4: An example to illustrate the process of the inner mutation. The parameters in data partitioning are partially changed and new tree structures are generated by reconstructing subtrees.

In the outer mutation, the new samplings with different sample size will be added to mutate new tree structures since we can expand the search space of isolation trees and escape from the local optimum by new samplings. The detailed process is shown in lines 5-8 in Algorithm 8. In this case, the parameters $\boldsymbol{W} \sim (\boldsymbol{w_1}; \boldsymbol{w_2}; ...; \boldsymbol{w_\psi})$ will be changed to $\boldsymbol{W'} \sim (\boldsymbol{w'_1}; \boldsymbol{w'_2}; ...; \boldsymbol{w'_{\psi'}})$ and we can get the isolation efficiency of the new isolation tree by:

$$\eta(\boldsymbol{W'}) = \frac{\psi'}{E(F(\boldsymbol{W'})) \cdot E(H(\boldsymbol{W'}))}, \tag{6.9}$$

only the isolation efficiency of the mutated isolation tree satisfies the following condition can be selected,

$$\eta(\boldsymbol{W'}) > \min \eta(\boldsymbol{W_i} | 1 \leqslant i \leqslant t, i \in \mathbb{N}). \tag{6.10}$$

The second feature that will influence the isolation tree structure is the branching factor of each internal node. The branching factor is fundamentally determined by the hash parameters that used to map and split the data instances. The inner mutation just change the subtree of the current isolation tree, which can help rapidly converge to the optimal tree structure among all the mutations existing under the current isolation tree. In Fig. 6.4, a simple example is given to show how the inner mutation happens.

In the inner mutation, we will traverse the whole isolation tree from the root node and give a random mutation rate to each internal node. The internal node will mutate when the mutation rate are smaller than the threshold, shown in lines 16-20 in Algorithm 8. Because the mutation in the parent node will change the whole subtree structure, its children nodes do not need for repeated mutation. In the above example,

Table 6.1: Average time complexity of L2SH and DOIForest.

Method	Training	Predicting
L2SH	$\Theta(t\psi \log_k(\psi)m)$	$\Theta(tn \log_k(\psi)m)$
DOIForest	$\Theta(t\psi \log_e(\psi)m+$ $l(r_1 t\psi \log_e(\psi)m + t(edr_2)\psi \log_e(\frac{\psi}{e})m))$	$\Theta(tn \log_e(\psi)m)$

the parameter $\boldsymbol{m_3}$ mutates to $\boldsymbol{m'_3}$ and $\boldsymbol{m_4}$, which will change the current branching factor and the depth as:

$$\begin{cases} v_3 = f_3(\boldsymbol{w_3}) \rightarrow v'_3 + v_4 = f'_3(\boldsymbol{w'_3}) + f_4(\boldsymbol{w_4}), \\ d_3 = h_3(\boldsymbol{w_3}) \rightarrow d'_3 + d_4 = h'_3(\boldsymbol{w'_3}) + h_4(\boldsymbol{w_4}). \end{cases} \tag{6.11}$$

The selection condition is same as Equ. (6.10) and only the mutated isolation tree with higher isolation efficiency can be selected as a new solution.

6.4.4 Time Complexity Analysis

The time complexity of our method consists of three parts: the pre-training stage, the training stage, and the testing stage. The time of pre-training stage and training stage can be added together since each layer produces the similar training time as the pre-training stage. As mentioned before, the pre-training stage can apply any isolation forest scheme, including iForest and LSHiForest. Besides, iForest and its variant have been proved to be two specific instances of the framework with less commonly-used distance metrics. In LSHiForest, many distance metrics like Angular distance, Manhattan (ℓ_1) distance, and Euclidean (ℓ_2) distance have been adopted to realise corresponding instances, and the instance with Euclidean distance (denoted as L2SH) shows a very effective, efficient, and robust detection performance. Thus, we take L2SH instance as an example case for time complexity analysis and experimental evaluation given their prominent features and excellent performance. The detailed time complexity is shown in Table 6.1.

The training time of L2SH is $\Theta(t\psi \log_k(\psi)m)$, where t is the number of trees in an isolation forest, k is the branch factor, ψ is the number of sampled data and m is the dimensions of data instances. Since we use L2SH to initialise the isolation forest, the time of pre-training stage in our method is same as that of L2SH.

In the training stage, the outer mutation will produce $r_1 \cdot t$ new isolation trees, so the time complexity is $\Theta(r_1 t\psi \log_k(\psi)m)$. The inner mutation happens in each internal node, so the time complexity can be calculated by:

$$T_{in} = \Theta(t(kdr_2)\psi \log_k(\frac{\psi}{k})m), \tag{6.12}$$

where d represents the average depth of an isolation tree and kdr_2 is the average number of mutation. Given l layers in our deep model, the time complexity in pre-training stage and training process can be obtained by:

$$T_{train} = \Theta(t\psi \log_k(\psi)m) + \Theta(l(r_1 t\psi \log_k(\psi)m + t(kdr_2)\psi \log_k(\frac{\psi}{k})m)). \tag{6.13}$$

The testing stage applies the trained optimal isolation forest to detect anomalies, which produces the similar time complexity to L2SH. In fact, the average branching factor in our method is very close to e, so the branching factor k in each stage of DOIForest should be e, shown in Table 6.1. Finally, we can calculate the time complexity of the testing stage by:

$$T_{test} = \Theta(tn \log_e(\psi)m), \tag{6.14}$$

where n represents the size of testing dataset. The whole time complexity of our method is the sum of T_{train} and T_{test}.

In practical applications, the training process can be conducted offline, so we only need to focus on the testing time of the model. In our deep learning approach, the testing time is similar to the traditional methods, allowing our method to achieve good efficiency in real-world applications.

6.5 Experiments

In this part, we introduce the datasets, basic settings, comparative methods, and evaluation metrics of the experiment. Then, extensive experiments on both synthetic and

real-world benchmark datasets are conducted, and a group of ablation studies are explored.

6.5.1 Datasets and Experiment Setting

Datasets In the experiments, we firstly evaluate our method on a synthetic Twospirals dataset, and then use 20 datasets from public UCI Machine Learning Repository[1] and ADRepository[2], which have been widely applied in anomaly detection methods, such as iForest, RDP, OptIForest, etc. Besides, we apply the same process method as that in previous work to preprocess the datasets and remove the missing values[110, 249, 207]. These datasets cover different sizes and application fields, aiming to validate the robustness of our method. The scale of data can be distinguished by the product of data volume and dimension. The description of 20 real-world datasets is shown in Table 6.2.

Experiment Setting In the pretraining stage, we apply the same parameter setting as L2SH. The layers of our deep model is 80 and each layer contains 100 initial isolation trees. Besides, we use the subsampling technique discussed in LSHiForest for our methods and the sample size varies within the range $[64, 1024]$ in small datasets while the sample size varies within the range $[256, 4096]$ in big datasets. The outer mutation rate is 0.2 and the inner mutation rate is 0.5 since we need faster mutation inside the tree structure. Other comparable methods apply the optimal parameter settings mentioned in their original implementations.

6.5.2 Baselines and Evaluation Metrics

Baselines Our method DOIForest is compared with six state-of-the-art anomaly detection methods, including iForest [110], L2SH [249], OptIForest [225] and REPEN [143], RDP [207], DIF [231]. The description of all methods is as follows:

[1] https://archive.ics.uci.edu/ml/datasets.php

[2] https://github.com/GuansongPang/ADRepository-Anomaly-detection-datasets

Table 6.2: A summary of real-world datasets used in the experiments. Specifically, "n" in the table denotes the size of the dataset, "m" denotes the dimension of the dataset, and "Rate" represents the proportion of anomalous data in total data instances.

Dataset	n	m	Rate(%)	Category
Arrhythmia	452	274	14.60	Healthcare
Cardio	1,831	21	9.61	Healthcare
Breastw	683	9	35.00	Healthcare
Pima	768	8	34.90	Healthcare
Backdoor	95,329	196	2.44	Network
KDDCup99	494,021	38	1.77	Network
Campaign	41,188	62	11.27	Finance
Creditcard	284,807	29	0.17	Finance
Celeba	202,599	39	2.24	Image
Mnist	7,603	100	9.21	Image
Skin	9,521	3	3.67	Image
Census	299,285	500	6.20	Sociology
Donors	619,326	10	5.92	Sociology
Cover	286,048	10	0.96	botany
Http	567,498	3	0.39	Web
Smtp	95,156	3	0.03	Web
Satellite	6,435	36	31.60	Astronautics
Vowel	1,456	12	3.43	Linguistics
Waveform	3,505	21	4.62	Physics
Wine	5,318	11	4.53	Chemistry

- iForest[3]: This is the most classic tree-based anomaly detection method, which is fast and robust for anomaly detection and has been widely applied in industria.

- L2SH[4]: A robust instantiated method of LSHiForest framework, which extends iForest into the multi-branch tree and achieves more effective performance.

[3]https://scikit-learn.org

[4]https://github.com/xuyun-zhang/LSHiForest

- OptIForest[5]: An optimal isolation forest with the branching factor e and learning to hash detector design.

- REPEN[6]: A random distance-based anomaly detection method, which applies deep representation learning and distance calculation to learn low-dimensional features.

- RDP[7]: A recent DAD method, which trains neural networks to predict data distances in a randomly projected space and has been confirmed superior to other DAD approaches in extensive datasets.

- DIF[8]: It utilises neural networks to learn representations and builds isolation forest in new projected space for anomaly detection.

Evaluation Metrics We use the Area Under Receiver Operating Characteristic Curve (AUC-ROC) and Area Under the Precision-Recall Curve (AUC-PR) as the accuracy evaluation criterion. AUC has been extensively adopted in many anomaly detection works, and has become an essential measure of accuracy in correlational researches [92, 229, 242]. The value of AUC ranges from 0 to 1 and a larger AUC result indicates better performance. To exclude the contingency, all the experiments are run 20 times and take the arithmetic average as the final result.

6.5.3 Results and Discussion

Our experiments are conducted on both synthetic and real-world datasets. We first evaluate how the isolation efficiency influences the detection quality on synthetic dataset. Then, a variety of benchmark datasets are used to evaluate effectiveness and robustness of all the aforementioned methods. Finally, ablation studies are conducted to evaluate the influence of different parts in our method.

[5] https://github.com/xiagll/OptIForest
[6] https://github.com/Minqi824/ADBench/tree/main/baseline
[7] https://git.io/RDP
[8] https://github.com/xuhongzuo/deep-iforest

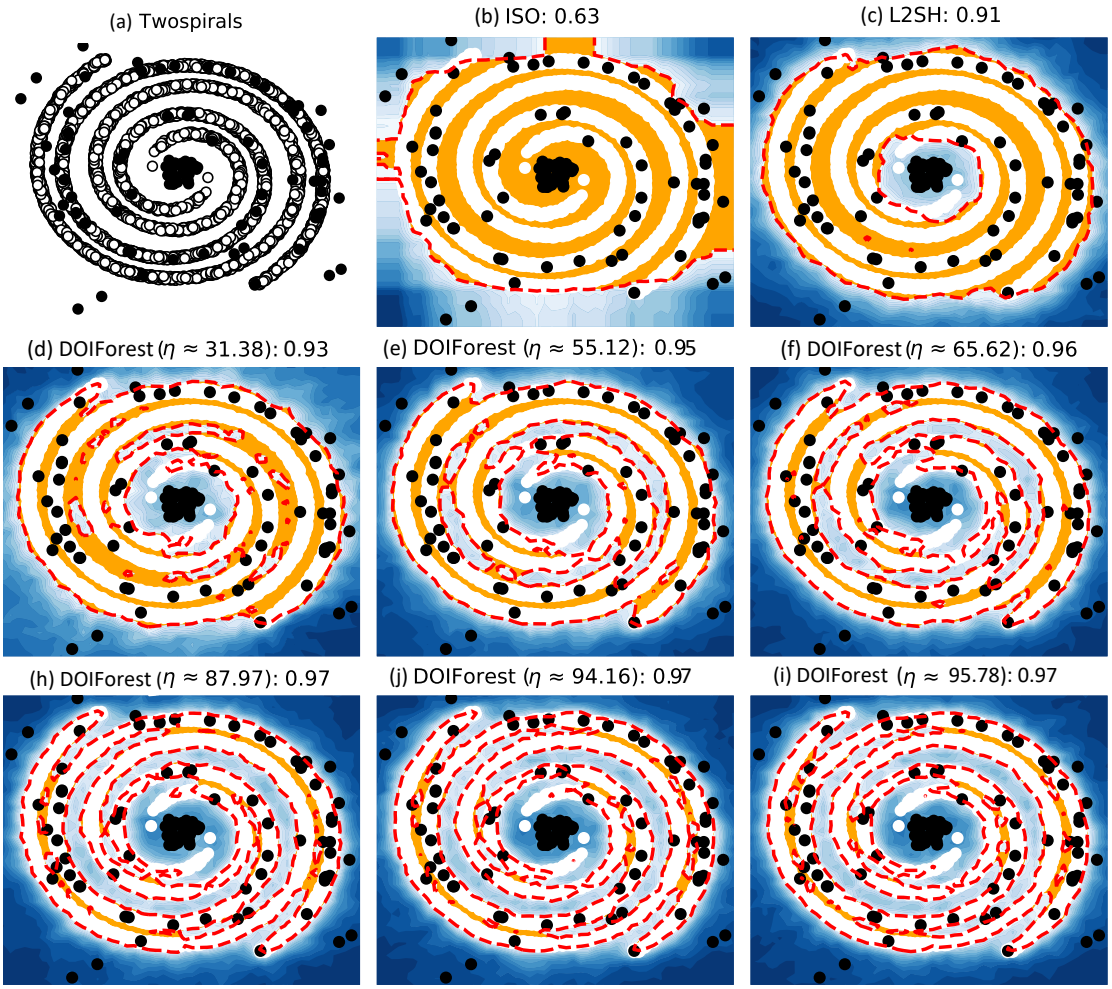


Figure 6.5: The effects of isolation efficiency on DOIForest.

Effects of Isolation Efficiency To explore how the core selection scheme works in our deep model, we use a synthetic dataset to visually display the detection performance of different isolation efficiency. We use the fundamental iForest and L2SH as the baseline methods, and set different layers (10, 50, 100, 150, 200, 250) to obtain different isolation efficiency for our approach. Similar to [249], a synthetic Twospirals dataset is generated with 5,000 2-dimensional data instances, which are composed of two normal spiral patterns, 1% uniformly distributed anomalies and 1% clustered anomalies following the standard Gaussian distribution $\mathcal{N}(0, 1)$. This synthetic dataset is shown in Fig. 6.5(a) and the experimental results are illustrated in Fig. 6.5(b)-(i).

We plot the contour maps of anomaly scores and the detected boundaries separating

the reported anomalies from normal data instances, shown in Fig. 6.5. The AUC results are reported for the baselines iForest and L2SH for contrast reference, shown in Fig. 6.5(b) and Fig. 6.5(c), respectively. We set different numbers of iterations to obtain different isolation efficiency, shown in Fig. 6.5(d)-(i). It is noting that if the iteration is equal to 0, the model will degrade to L2SH in this case. The isolation efficiency and AUC results are reported in the subtitles with the following format: *method: (isolation efficiency, AUC)*. It can be seen from Fig. 6.5 that our baselines iForest and L2SH get low detection accuracy in this synthetic dataset and are hard to detect the surrounded anomalies. DOIForest is effective to detect the surrounded anomalies and the detection accuracy get higher with the increase of isolation efficiency. The detection performance of DOIForest will get stable until the isolation efficiency is sufficiently large. Therefore, our deep model that applies two mutation schemes and isolation efficiency for selection is effective for anomaly detection.

Evaluations on real-world datasets Except from the synthetic dataset, we also evaluate our method on extensive real-world benchmark datasets. Firstly, the AUC-ROC results are recorded in Table 6.3. For each dataset, the leading values are highlighted. It can be seen from Table 6.3 that our method DOIForest is the most effective and robust method that achieves the best performance on most datasets (16 datasets among 20 datasets). Specifically, DOIForest produces the highest average AUC-ROC score among all other compared methods, surpassing the baseline L2SH by 3% and iForest 6%. The second method is OptIForest, which is also based on the theory of the optimal isolation forest. However, it fails to deepen the model to learn the real optimal isolation forest in the practical detector design. RDP has the highest AUC-ROC score in a few datasets, like "Creditcard", "Celeba" and "Campaign", while it performs the worst in some datasets. Other methods iForest, L2SH, REPEN and DIF only perform well on a few datasets, indicating their lack of robustness across all datasets. Therefore, it is evident that most original isolation forest based methods, including iForest, L2SH and DIF, fail to achieve robust performance because they lack the knowledge learning from data. OptIForest and our method produce the better performance than other

detectors by learning the optimal isolation tree structures for bias reduction.

Table 6.3: AUC-ROC (%) performance (mean ± standard deviation) of all methods. Our method DOIForest has the highest AUC-ROC score and outperforms others on most datasets.

Dataset	iForest	L2SH	OptIForest	REPEN	RDP	DIF	DOIForest (ours)
Arrhythmia	**79.7 ± 1.0**	77.5 ± 0.5	**79.6 ± 0.8**	73.7 ± 3.6	75.5 ± 0.5	76.3 ± 1.1	77.8 ± 0.9
Cardio	**93.0 ± 0.7**	90.4 ± 0.6	92.8 ± 1.3	91.5 ± 2.9	88.1 ± 0.6	**93.0 ± 0.5**	**94.1 ± 1.1**
Breastw	95.3 ± 0.1	97.7 ± 0.2	98.2 ± 0.1	97.0 ± 0.1	**98.6 ± 0.3**	92.1 ± 0.2	**98.5 ± 0.1**
Pima	66.5 ± 0.7	**70.2 ± 0.1**	69.8 ± 0.3	67.9 ± 4.7	67.2 ± 0.9	69.6 ± 1.9	**71.4 ± 0.8**
Backdoor	72.7 ± 2.9	89.2 ± 0.9	**92.7 ± 0.5**	86.8 ± 1.6	91.0 ± 2.1	**92.0 ± 0.5**	88.9 ± 1.2
KDDCup99	97.0 ± 0.6	96.4 ± 0.2	**97.4 ± 0.1**	95.9 ± 0.6	41.0 ± 3.1	88.5 ± 0.7	**97.6 ± 0.3**
Campaign	70.9 ± 1.0	67.7 ± 0.6	74.7 ± 0.4	61.1 ± 4.4	**76.3 ± 0.8**	69.3 ± 0.9	**74.8 ± 0.4**
Creditcard	94.6 ± 0.5	95.2 ± 0.2	84.8 ± 2.4	95.0 ± 0.1	**95.7 ± 0.5**	94.5 ± 0.7	**95.6 ± 0.3**
Celeba	69.4 ± 2.4	72.5 ± 0.7	79.2 ± 1.9	**84.3 ± 2.2**	**86.0 ± 0.6**	67.9 ± 1.7	80.3 ± 1.0
Mnist	80.2 ± 1.8	85.3 ± 0.6	**85.5 ± 1.0**	67.6 ± 10.6	85.1 ± 1.6	83.7 ± 1.3	**86.0 ± 0.9**
Skin	61.6 ± 1.2	72.2 ± 1.8	**74.7 ± 1.5**	69.8 ± 6.3	55.5 ± 3.1	72.3 ± 2.2	**78.9 ± 2.1**
Census	60.1 ± 1.8	62.6 ± 0.4	**67.8 ± 1.0**	62.7 ± 1.5	65.3 ± 0.4	59.4 ± 1.1	**67.2 ± 1.1**
Donors	76.6 ± 1.0	74.5 ± 0.7	77.1 ± 3.2	**83.2 ± 1.7**	**96.2 ± 1.1**	67.8 ± 1.2	77.2 ± 1.8
Cover	88.0 ± 2.1	**93.6 ± 0.5**	90.9 ± 0.8	86.6 ± 5.7	51.2 ± 1.3	76.4 ± 4.0	**94.4 ± 0.6**
Http	**99.9 ± 0**	93.3 ± 0	99.4 ± 0.1	99.4 ± 0.1	99.3 ± 0.1	99.3 ± 0.1	**99.6 ± 0.1**
Smtp	90.5 ± 0.8	86.9 ± 0.9	**92.4 ± 0.6**	90.9 ± 1.3	69.8 ± 1.1	84.6 ± 0.5	**93.3 ± 0.9**
Satellite	70.3 ± 1.8	76.7 ± 0.5	**78.6 ± 0.7**	74.0 ± 3.0	60.9 ± 0.6	69.2 ± 1.3	**77.8 ± 0.7**
Vowel	75.3 ± 1.2	90.8 ± 0.7	90.0 ± 1.2	77.9 ± 5.5	67.2 ± 1.2	**94.4 ± 1.4**	**94.8 ± 0.9**
Waveform	69.6 ± 2.3	70.7 ± 0.8	**74.2 ± 1.3**	63.2 ± 9.5	66.9 ± 0.9	63.2 ± 3.4	**76.8 ± 1.0**
Wine	63.9 ± 0.7	67.0 ± 0.3	**66.4 ± 0.5**	64.6 ± 2.2	61.7 ± 0.4	65.9 ± 1.2	**68.1 ± 0.5**
Ave AUC	78.8 ± 1.2	81.5 ± 0.6	84.4 ± 1.0	79.7 ± 3.4	74.9 ± 1.1	79.0 ± 1.3	**84.7 ± 0.8**

We also use the AUC-PR to illustrate the performance of different methods, which is reported in Table 6.4. It can be concluded that our method DOIForest is the most effective and robust method that achieves the best performance on most datasets and produces the highest average AUC-PR score among all other compared methods. Specifically, DOIForest even surpasses the second-ranked method by 3% and the third-ranked method by 7%. While REPEN and RDP demonstrate exceptional results on some datasets, their performance is poor on others. Isolation Forest based methods

Table 6.4: AUC-PR (%) performance (mean ± standard deviation) of all methods. Our method DOIForest outperforms others.

Dataset	iForest	L2SH	OptIForest	REPEN	RDP	DIF	DOIForest (ours)
Arrhythmia	**47.5 ± 1.4**	38.6 ± 0.7	**45.1 ± 1.2**	37.4 ± 3.5	32.0 ± 0.6	38.2 ± 1.2	40.9 ± 1.8
Cardio	57.0 ± 2.8	48.9 ± 1.0	**58.9 ± 3.7**	53.3 ± 10.6	53.9 ± 1.5	58.7 ± 1.9	**59.2 ± 2.3**
Breastw	93.5 ± 0.3	92.9 ± 0.3	94.8 ± 0.4	**96.4 ± 1.0**	94.9 ± 0.1	73.2 ± 0.8	**95.8 ± 0.3**
Pima	49.8 ± 0.8	51.8 ± 0.6	**53.1 ± 0.8**	49.6 ± 4.1	51.2 ± 1.5	51.7 ± 1.7	**52.6 ± 0.7**
Backdoor	4.5 ± 0.7	27.3 ± 2.6	**51.7 ± 8.7**	12.5 ± 1.7	3.5 ± 0.8	39.4 ± 3.3	**50.2 ± 5.8**
KDDCup99	**48.6 ± 7.0**	32.6 ± 1.1	43.0 ± 0.1	**44.1 ± 1.9**	15.4 ± 0.9	16.7 ± 0.5	40.1 ± 1.3
Campaign	28.5 ± 1.3	24.7 ± 1.0	**32.0 ± 0.5**	17.8 ± 3.9	**37.2 ± 0.9**	27.5 ± 1.3	29.8 ± 0.9
Creditcard	14.0 ± 3.4	13.3 ± 2.2	12.1 ± 3.9	**35.0 ± 1.2**	**36.5 ± 1.1**	14.7 ± 2.1	14.9 ± 3.2
Celeba	6.3 ± 0.9	6.8 ± 0.3	8.1 ± 1.0	**10.7 ± 1.8**	10.4 ± 0.6	5.5 ± 0.6	**10.6 ± 1.0**
Mnist	27.7 ± 3.2	38.3 ± 1.0	**40.7 ± 1.9**	20.4 ± 10.2	36.7 ± 2.4	33.0 ± 2.1	**39.2 ± 1.2**
Skin	4.3 ± 0.2	5.8 ± 0.3	**6.5 ± 0.8**	5.6 ± 1.0	5.1 ± 1.4	5.6 ± 1.0	**7.6 ± 1.0**
Census	7.1 ± 0.3	7.5 ± 0.1	**8.8 ± 0.2**	7.7 ± 0.2	8.6 ± 0.1	6.9 ± 0.2	**8.9 ± 0.2**
Donors	11.9 ± 0.8	10.5 ± 0.6	11.3 ± 1.6	15.5 ± 1.3	**43.2 ± 6.1**	8.0 ± 0.3	**16.5 ± 1.2**
Cover	6.4 ± 0.9	**9.0 ± 0.8**	6.5 ± 0.7	5.3 ± 2.0	4.5 ± 1.1	4.0 ± 0.8	**10.6 ± 0.5**
Http	**88.2 ± 8.5**	34.2 ± 0.6	35.4 ± 1.5	39.5 ± 2.4	36.2 ± 0.8	35.1 ± 0.4	**44.5 ± 1.1**
Smtp	0.4 ± 0	**56.9 ± 3.1**	36.9 ± 13.7	26.9 ± 10.8	21.7 ± 3.9	**55.2 ± 7.0**	48.2 ± 5.1
Satellite	65.0 ± 2.1	63.5 ± 0.6	**71.5 ± 0.9**	**70.3 ± 3.9**	61.3 ± 0.7	47.5 ± 1.3	63.5 ± 1.1
Vowel	12.6 ± 1.7	29.5 ± 2.1	32.4 ± 4.3	14.5 ± 6.5	8.9 ± 3.2	**41.1 ± 9.0**	**43.0 ± 3.1**
Waveform	9.7 ± 0.9	9.8 ± 0.4	**11.3 ± 1.0**	7.6 ± 2.4	8.8 ± 1.5	11.2 ± 3.3	**12.4 ± 0.6**
Wine	7.8 ± 0.4	**8.9 ± 0.2**	8.1 ± 0.2	8.1 ± 0.7	8.1 ± 0.2	8.5 ± 0.5	**9.5 ± 0.2**
Ave AUC	29.5 ± 1.9	30.5 ± 1.0	34.5 ± 2.4	28.9 ± 3.6	28.9 ± 1.5	29.1 ± 2.0	**37.1 ± 1.6**

are more robust on all datasets compared with other methods. The AUC-PR results demonstrate a similar pattern to the AUC-ROC results, reinforcing the notion that DOIForest consistently outperforms others in terms of the effectiveness and robustness.

Ablation Study Results and Discussion To understand how the number of layers in the deep model and the number of isolation tree influence the performance of DOIForest, we conduct detailed ablation studies on six datasets with a range of data types and sizes.

Number of Trees t. The number of isolation trees will influence the learning process

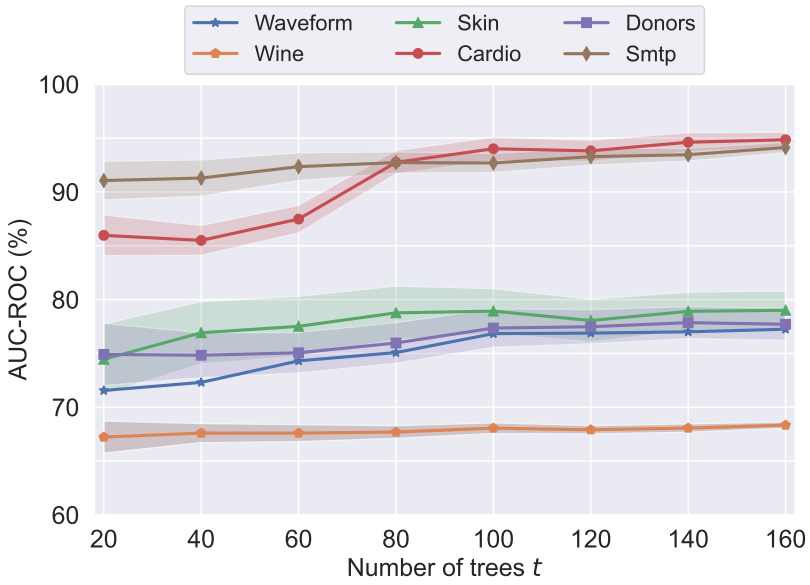


Figure 6.6: Detection performance changes w.r.t. number of trees t. In general, the performance is saturated when t reaches 100.

of our method since few trees can not provide enough variant structure and too many trees will affect time efficiency. We set the number of trees from 20 to 160 to observe the change of detection performance, shown in Fig. 6.6. It can be observed that the AUC-ROC scores improve with an increasing number of trees and eventually stabilise on most datasets. Only the "wine" dataset shows less sensitivity to changes in the number of trees, but there is still a slight improvement in the detection performance as the number of trees increases on this dataset. According to the above results in Fig. 6.6, the number of trees is set as 100 in our experiments, which can achieve a good detection performance with the balanced time cost.

Layers of Deep model l. To save the time on model training, we need to observe the influence of the layers in our model and find the optimal solution within limited layers. Herein, we set the layers from 10 to 120 to observe the change in AUC-ROC scores. It can be seen in Fig. 6.7 that AUC-ROC score gets higher with an increasing number of layers and keeps stable when the number of layers reaches to 80 or more. According to the above results, the number of layers in our method is set as 80. Compared with deep neural network based anomaly detection methods, our approach can reduce the

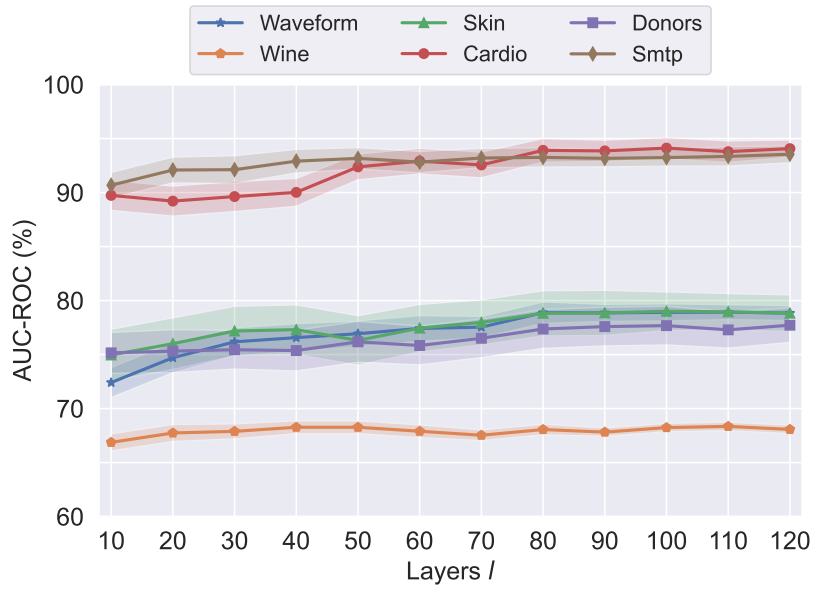


Figure 6.7: Detection performance changes w.r.t. layers l in deep model. In general, the performance is saturated when l reaches 80.

number of training layers since the initial model is built based on traditional isolation forest methods, which have a certain level of detection accuracy. This process can also reduce the search space to speed up the search for the optimal isolation forest, while we have incorporated random variable mutations into our method to escape from the local optimal solutions.

6.6 Conclusion

Current deep anomaly detection methods, including the traditional neural network based and the isolation forest based methods, focus on learning good feature representations to minimise the loss of mapping data to a new space in order to obtain effective anomaly scores for anomaly detection. However, the backpropagation based scheme in neural networks is complex and time-consuming, and isolation forest based deep models fail to take the tree structure into consideration although the tree structure plays a decisive role in detection performance on isolation forest based methods. In this work, we have theoretically analysed the difficulty of finding the optimal isolation

tree structure in practical applications. Then, we propose the optimization solution through the deep evolutionary model to train the optimal forest and simultaneously optimise the feature representation for anomaly detection. Extensive experiments are conducted on both synthetic datasets and a variety of real-world datasets. The results have confirmed the effectiveness and robustness of DOIForest.

In the future, we intend to utilise parallel computing techniques to accelerate our method and employ industrial applications to enhance and implement the approach.

7

Conclusion and Future Work

7.1 Conclusion

In this dissertation, we mainly focus on the isolation forest-based anomaly detection with learning to hash scheme (i.e., data dependent techniques). In the first chapter, we introduce the background of our research and analyse the existing problems relevant to our work. Then, we summarise the contributions of this dissertation and organise the structure. In the second chapter, a large number of studies on anomaly detection, especially on unsupervised methods, are reviewed, and then isolation forest based anomaly detection methods are summarised and analysed. On the basis of the above, we propose a novel anomaly detection method using the order preserving hashing based isolation forest, which aims to fully learn the data information and produce higher effectiveness for anomaly detection in the big data era. Next, we explore the optimal

tree structure for an isolation forest with respect to the branching factor since there is no theoretical work answering this fundamentally and practically important question. Based on the theoretical underpinning, we design a practical optimal isolation forest OptIForest incorporating clustering-based learning to hash which enables more information to be learned from data for better isolation quality. We also explore the possibility of extending the shallow isolation forest scheme to the deep paradigm, i.e., we try to explore an unsupervised non-neural network deep model for anomaly detection based on the experience of the deep forest. Thus, a deep isolation tree ensemble approach, named DeepiForest, is proposed for robust anomaly detection by enhanced representation learning and multi-layer cascaded architecture. Finally, we analyse the search space of isolation trees under specific data instances and address the challenges in finding optimal isolation forest. Based on the theoretical underpinning and genetic algorithm, we design a deep model DOIForest with two mutation schemes and solution selection, which learns the optimal isolation forest and optimises the parameters in data partitioning.

We leverage the learning to hash scheme to isolation forest based anomaly detection in this dissertation, which solves the problems of data-independent feature learning and extends isolation forest to a better learning framework for anomaly detection. Besides, we are the first work that explores the deep structure of isolation forest, rather than the simple combination of deep neural networks and isolation forest. The performance of the deep isolation forest is influenced by the feature learning of data embedding and the isolation tree structure that is relevant to our proposed optimal isolation forest. Finally, extensive experiments on both real-world benchmarking datasets and ablation studies validate the effectiveness and robustness of our approaches.

7.2 Future Work

In this part, we will discuss a few open challenges and give a vision of future work. Firstly, the relationship between completed work and future work is shown in Figure 7.1. We have finished four core parts (shown in yellow boxes) of the isolation forest

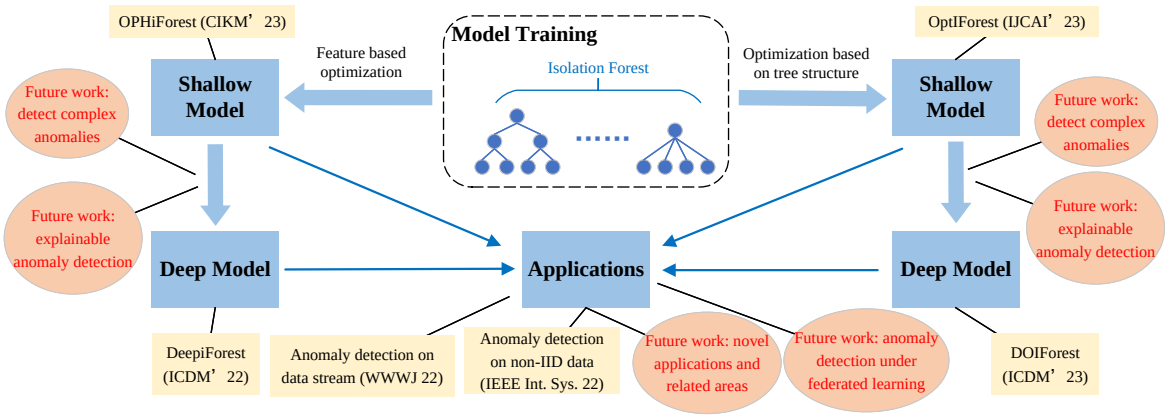


Figure 7.1: Relationship diagram between completed work and future work.

based model design for anomaly detection in this thesis, including two models in tree structure optimization and two models in feature optimisation. However, there are four challenges (shown in orange circles) that can be addressed in future work. Specifically, complex anomaly detection and explainable anomaly detection pose challenges for model design. Some novel scenarios, including federated learning and innovative applications, raise new problems for anomaly detection. The detailed explanation is shown as follows:

- **Detection of Complex Anomalies.** While most isolation forest based anomaly detection methods excel in detecting point anomalies and outperform other techniques, there has been relatively little exploration into using isolation forest for conditional or group anomalies. Although isolation forest based methods have been applied in intricate temporal and spatial datasets, and unordered data points, it is still a challenge whether they can achieve high detecting performance in identifying complex anomalies. Current isolation forest based anomaly detection primarily focuses on single data sources, while the realm of multimodal anomaly detection remains largely unexplored. Other traditional approaches struggle to bridge the gap presented by multimodal data. However, isolation forest with learning to hash has demonstrated tremendous success in learning feature representations from different types of raw data for anomaly detection.

By exploring deep layers for isolation forest, it can also concatenate representations from different data sources to learn unified representations, presenting significant opportunities in multimodal anomaly detection.

- **Explainable Anomaly Detection.** While current anomaly detection methods primarily emphasise detection accuracy, it is increasingly crucial to incorporate explainability into these models. Explainable anomaly detection is vital for comprehending model decisions, addressing potential biases or risks for human users, and facilitating decision-making actions. Some recent studies have begun to tackle anomaly explanation by identifying subsets of features that render reported anomalies as the most abnormal [9, 203, 10, 182]. These methods for abnormal feature selection can serve as tools for providing explanations for anomalies. However, this model-agnostic approach may limit the usefulness of explanations since it does not offer a genuine understanding of the inner workings of specific detection models, resulting in limited explainability, such as quantifying the impact of detected anomalies and potential mitigation measures [145]. It's essential to explore inherent explainability in anomaly detection models for anomaly explanation, accelerating anomaly detection models with sufficient explanation and detection accuracy.

- **Anomaly Detection Under Federated Learning.** Most of the current anomaly detection models are trained on a single server or homogeneous data, so the server resources and data types will constrain the model training. In addition, the data gathered often encompasses sensitive information, thereby raising significant privacy concerns about data eavesdropping and unauthorised data leakage. To tackle the above problems, federated learning (FL) has become a promising supplement for anomaly detection, which enables multiple clients to train a collaborative model locally without sharing raw training data. However, applying anomaly detection models directly to FL can often lead to compatibility issues. There have been many works that try to develop a privacy-enhanced FL framework for anomaly detection [119, 101, 213, 107]. Still, these approaches always

have more challenges: 1) The communication overhead of FL models will seriously increase due to interactive decryption. 2) Some FL frameworks require data in different clients to be shared secretly among noncolluding servers, which has become vulnerable to inference attacks. 3) FL schemes may add too much noise to the training data or model parameters, which may result in poor detection performance for anomaly detection.

- **Novel Applications and Related Areas.** Many other applications and areas provide essential opportunities to extend the existing isolation forest based anomaly detection approaches. Firstly, as closely related, out-of-distribution (OOD) detection [69, 98] focuses on identifying data instances that deviate significantly from the training distribution. This plays a crucial role in allowing machine learning systems to handle instances from new classes in open-world scenarios. OOD detection shares similarities with anomaly detection but typically assumes access to detailed normal class labels during training, emphasizing the need to maintain classification accuracy for these normal classes while effectively detecting OOD instances. In addition, many shallow and deep models for anomaly detection assume that anomalies in data instances follow an independent and identically distributed (IID) pattern [142]. However, real-world anomalies often exhibit non-IID characteristics. For example, the anomalies of different instances or features may be interdependent and heterogeneous. Non-IID anomaly detection is crucial in complex scenarios, such as cases where anomalies exhibit subtle deviations and are concealed within the data space without considering these non-IID abnormal characteristics. Finally, there are diverse and intriguing applications for anomaly detection, including the detection of adversarial examples, intrusion detection of cybersecurity, and early detection of rare catastrophic events such as fraud detection and government hacking. These application domains offer unique challenges and opportunities for deploying deep anomaly detection techniques.

References

[1] D. Abati, A. Porrello, S. Calderara, and R. Cucchiara. Latent space autoregression for novelty detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 481–490, 2019.

[2] C. C. Aggarwal. Outlier analysis. In *Data mining*, pages 237–263. Springer, 2015.

[3] C. C. Aggarwal and C. C. Aggarwal. Proximity-based outlier detection. *Outlier analysis*, pages 111–147, 2017.

[4] C. C. Aggarwal and S. Sathe. Theoretical foundations and algorithms for outlier ensembles. In *ACM SIGKDD*, volume 17, pages 24–47. ACM, 2015.

[5] C. C. Aggarwal and S. Sathe. Theoretical foundations and algorithms for outlier ensembles. *Acm sigkdd explorations newsletter*, 17(1):24–47, 2015.

[6] M. Ahmed, A. N. Mahmood, and M. R. Islam. A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems*, 55:278–288, 2016.

[7] E. Alpaydin. *Introduction to machine learning*. MIT press, 2020.

[8] F. Angiulli and C. Pizzuti. Fast outlier detection in high dimensional spaces. In *European conference on principles of data mining and knowledge discovery (PKDD)*, pages 15–27. Springer, 2002.

[9] F. Angiulli, F. Fassetti, and L. Palopoli. Detecting outlying properties of exceptional objects. *Acm transactions on database systems (tods)*, 34(1):1–62, 2009.

[10] F. Angiulli, F. Fassetti, G. Manco, and L. Palopoli. Outlying property detection with numerical attributes. *Data mining and knowledge discovery*, 31:134–163, 2017.

[11] S. Aryal, K. M. Ting, J. R. Wells, and T. Washio. Improving iforest with relative mass. In *Advances in Knowledge Discovery and Data Mining: 18th Pacific-Asia Conference (PAKDD)*, pages 510–521. Springer, 2014.

[12] S. Athey. The impact of machine learning on economics. In *The economics of artificial intelligence: An agenda*, pages 507–547. University of Chicago Press, 2018.

[13] T. R. Bandaragoda, K. M. Ting, D. Albrecht, F. T. Liu, and J. R. Wells. Efficient anomaly detection by isolation using nearest neighbour ensemble. In *2014 IEEE International conference on data mining workshop (ICDM Workshop)*, pages 698–705. IEEE, 2014.

[14] T. R. Bandaragoda, K. M. Ting, D. Albrecht, F. T. Liu, Y. Zhu, and J. R. Wells. Isolation-based anomaly detection using nearest-neighbor ensembles. *Computational Intelligence*, 34(4):968–998, 2018.

[15] H. B. Barlow. Unsupervised learning. *Neural computation*, 1(3):295–311, 1989.

[16] M. Bawa, T. Condie, and P. Ganesan. Lsh forest: self-tuning indexes for similarity search. In *ACM WWW*, pages 651–660, 2005.

[17] S. D. Bay and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (SIGKDD)*, pages 29–38, 2003.

[18] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. Network anomaly detection: methods, systems and tools. *Ieee communications surveys & tutorials*, 16 (1):303–336, 2013.

[19] G. Bonaccorso. *Machine learning algorithms*. Packt Publishing Ltd, 2017.

[20] I. Bose and R. K. Mahapatra. Business data mining—a machine learning perspective. *Information & management*, 39(3):211–225, 2001.

[21] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data (SIGMOD)*, pages 93–104, 2000.

[22] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová. Machine learning and the physical sciences. *Reviews of Modern Physics*, 91(4):045002, 2019.

[23] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149, 2018.

[24] R. Chalapathy, A. K. Menon, and S. Chawla. Anomaly detection using one-class neural networks. *arXiv preprint arXiv:1802.06360*, 2018.

[25] R. Chalapathy, N. L. D. Khoa, and S. Chawla. Robust deep learning methods for anomaly detection. In *ACM SIGKDD*, pages 3507–3508, 2020.

[26] V. Chandola, A. Banerjee, and V. Kumar. Outlier detection: A survey. *ACM Computing Surveys (CSUR)*, 14:15, 2007.

[27] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[28] G. Chen, Y. L. Cai, and J. Shi. Ordinal isolation: An efficient and effective intelligent outlier detection algorithm. In *2011 IEEE International Conference*

on Cyber Technology in Automation, Control, and Intelligent Systems (ICCT), pages 21–26. IEEE, 2011.

[29] 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 (SDM)*, pages 90–98. SIAM, 2017.

[30] T. Chen and C. Tsourakakis. Antibenford subgraphs: Unsupervised anomaly detection in financial networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 2762–2770, 2022.

[31] T. Chen, L.-A. Tang, Y. Sun, Z. Chen, and K. Zhang. Entity embedding-based anomaly detection for heterogeneous categorical events. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.

[32] Y. Chen, X. S. Zhou, and T. S. Huang. One-class svm for learning in image retrieval. In *IEEE ICIP*, pages 34–37, 2001.

[33] X. Cheng, M. Zhang, S. Lin, K. Zhou, S. Zhao, and H. Wang. Two-stream isolation forest based on deep features for hyperspectral anomaly detection. *IEEE Geoscience and Remote Sensing Letters*, 2023.

[34] P. Cunningham, M. Cord, and S. J. Delany. Supervised learning. In *Machine learning techniques for multimedia*, pages 21–49. Springer, 2008.

[35] N. Dilokthanakul, P. A. Mediano, M. Garnelo, M. C. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *International Conference on Learning Representations (ICLR)*, 2017.

[36] K. Ding, J. Li, R. Bhanushali, and H. Liu. Deep anomaly detection on attributed networks. In *Proceedings of the 2019 SIAM International Conference on Data Mining (SDM)*, pages 594–602. SIAM, 2019.

[37] Y. Djenouri, A. Zimek, and M. Chiarandini. Outlier detection in urban traffic flow distributions. In *IEEE ICDM*, pages 935–940, 2018.

[38] B. Du, X. Sun, J. Ye, K. Cheng, J. Wang, and L. Sun. Gan-based anomaly detection for multivariate time series using polluted training set. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[39] M. Du, F. Li, G. Zheng, and V. Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *ACM SIGSAC*, pages 1285–1298, 2017.

[40] A. F. Emmott, S. Das, T. Dietterich, A. Fern, and W.-K. Wong. Systematic construction of anomaly detection benchmarks from real data. In *Proceedings of the ACM SIGKDD workshop on outlier detection and description (SIGKDD)*, pages 16–21, 2013.

[41] 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*, 58:121–134, 2016.

[42] S. Fan, C. Shi, and X. Wang. Abnormal event detection via heterogeneous information network embedding. In *Proceedings of the 27th ACM international conference on information and knowledge management (CIKM)*, pages 1483–1486, 2018.

[43] T. Fernando, H. Gammulle, S. Denman, S. Sridharan, and C. Fookes. Deep learning for medical anomaly detection–a survey. *ACM Computing Surveys (CSUR)*, 54(7):1–37, 2021.

[44] J. Gao, W. Hu, Z. Zhang, X. Zhang, and O. Wu. Rkof: robust kernel-based local outlier detection. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 270–283. Springer, 2011.

[45] R. Gao, T. Zhang, S. Sun, and Z. Liu. Research and improvement of isolation

forest in detection of local anomaly points. In *Journal of Physics: Conference Series*, volume 1237. IOP Publishing, 2019.

[46] M. Gebski and R. K. Wong. An efficient histogram method for outlier detection. In *Advances in Databases: Concepts, Systems and Applications: 12th International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 176–187. Springer, 2007.

[47] Z. Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, 2015.

[48] K. Ghasedi Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 5736–5745, 2017.

[49] W. Ghorbani et al. Theoretical foundation of detection network intrusion detection and prevention. In *Concepts and Techniques Advances in Information Security*, volume 47, pages 47–114. Springer Science, 2010.

[50] A. Ghoting, S. Parthasarathy, and M. E. Otey. Fast mining of distance-based outliers in high-dimensional datasets. *Data Mining and Knowledge Discovery*, 16:349–364, 2008.

[51] P. Gogoi, D. K. Bhattacharyya, B. Borah, and J. K. Kalita. A survey of outlier detection methods in network anomaly identification. *The Computer Journal*, 54 (4):570–588, 2011.

[52] I. Golan and R. El-Yaniv. Deep anomaly detection using geometric transformations. *Advances in neural information processing systems*, 31, 2018.

[53] M. Goldstein. Fastlof: an expectation-maximization based local outlier detection algorithm. In *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*, pages 2282–2285. IEEE, 2012.

[54] M. Goldstein and A. Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI-2012: poster and demo track*, 1:59–63, 2012.

[55] P. Gopalan, V. Sharan, and U. Wieder. Pidforest: anomaly detection via partial identification. *Advances in Neural Information Processing Systems*, 32, 2019.

[56] N. Görnitz, M. Kloft, K. Rieck, and U. Brefeld. Toward supervised anomaly detection. *Journal of Artificial Intelligence Research*, 46:235–262, 2013.

[57] J. G. Greener, S. M. Kandathil, L. Moffat, and D. T. Jones. A guide to machine learning for biologists. *Nature Reviews Molecular Cell Biology*, 23(1):40–55, 2022.

[58] D. Guthrie, L. Guthrie, B. Allison, and Y. Wilks. Unsupervised anomaly detection. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1624–1628, 2007.

[59] J. Haase and U. Brefeld. Finding similar movements in positional data streams. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 49–57, 2013.

[60] S. Han and S. S. Woo. Learning sparse latent graph representations for anomaly detection in multivariate time series. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 2977–2986, 2022.

[61] S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao. Adbench: Anomaly detection benchmark. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[62] X. Han, X. Chen, and L.-P. Liu. Gan ensemble for anomaly detection. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, pages 4090–4097, 2021.

[63] S. Hariri, M. C. Kind, and R. J. Brunner. Extended isolation forest. *IEEE transactions on knowledge and data engineering*, 33(4):1479–1489, 2019.

[64] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. Hashem. Attack and anomaly detection in iot sensors in iot sites using machine learning approaches. In *Internet of Things*, volume 7, pages 1–14. Elsevier, 2019.

[65] S. Hawkins, H. He, G. Williams, and R. Baxter. Outlier detection using replicator neural networks. In *International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*, pages 170–180. Springer, 2002.

[66] S. He, F. Chen, and B. Jiang. Physical intrusion monitoring via local-global network and deep isolation forest based on heterogeneous signals. *Neurocomputing*, 441:25–35, 2021.

[67] Z. He, X. Xu, and S. Deng. Discovering cluster-based local outliers. *Pattern recognition letters*, 24(9-10):1641–1650, 2003.

[68] M. Heigl, K. A. Anand, A. Urmann, D. Fiala, M. Schramm, and R. Hable. On the improvement of the isolation forest algorithm for outlier detection with streaming data. *Electronics*, 10(13):1534, 2021.

[69] D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations (ICLR)*, 2016.

[70] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[71] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[72] V. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2):85–126, 2004.

[73] C. Huang, Y. Wu, Y. Zuo, K. Pei, and G. Min. Towards experienced anomaly detector through reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 32, 2018.

[74] Y.-a. Huang, W. Fan, W. Lee, and P. S. Yu. Cross-feature analysis for detecting ad-hoc routing anomalies. In *23rd International Conference on Distributed Computing Systems (ICDCS)*, pages 478–487. IEEE, 2003.

[75] R. T. Ionescu, F. S. Khan, M.-I. Georgescu, and L. Shao. Object-centric auto-encoders and dummy anomalies for abnormal event detection in video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7842–7851, 2019.

[76] M.-F. Jiang, S.-S. Tseng, and C.-M. Su. Two-phase clustering process for outliers detection. *Pattern recognition letters*, 22(6-7):691–700, 2001.

[77] S.-y. Jiang and Q.-b. An. Clustering-based outlier detection method. In *2008 Fifth international conference on fuzzy systems and knowledge discovery (ICFSKD)*, volume 2, pages 429–433. IEEE, 2008.

[78] A. Joly and O. Buisson. Random maximum margin hashing. In *IEEE CVPR*, pages 873–880, 2011.

[79] M. I. Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[80] P. Karczmarek, A. Kiersztyn, W. Pedrycz, and E. Al. K-means-based isolation forest. *Knowledge-based systems*, 195:105659, 2020.

[81] P. Karczmarek, A. Kiersztyn, W. Pedrycz, and D. Czerwiński. Fuzzy c-means-based isolation forest. *Applied Soft Computing*, 106:107354, 2021.

[82] T. Kieu, B. Yang, C. Guo, and C. S. Jensen. Outlier detection for time series with recurrent autoencoder ensembles. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2725–2732, 2019.

[83] J. Kim and C. D. Scott. Robust kernel density estimation. *The Journal of Machine Learning Research*, 13(1):2529–2565, 2012.

[84] A. Kind, M. P. Stoecklin, and X. Dimitropoulos. Histogram-based traffic anomaly detection. *IEEE Transactions on Network and Service Management*, 6(2):110–121, 2009.

[85] B. R. Kiran, D. M. Thomas, and R. Parakkal. An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos. *Journal of Imaging*, 4(2):36, 2018.

[86] E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based outliers: algorithms and applications. *The VLDB Journal*, 8(3):237–253, 2000.

[87] E. M. Knox and R. T. Ng. Algorithms for mining distancebased outliers in large datasets. In *Proceedings of the international conference on very large data bases*, pages 392–403. Citeseer, 1998.

[88] H.-P. Kriegel, M. Schubert, and A. Zimek. Angle-based outlier detection in high-dimensional data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (SIGKDD)*, pages 444–452, 2008.

[89] H.-P. Kriegel, P. Kroger, E. Schubert, and A. Zimek. Interpreting and unifying outlier scores. In *Proceedings of the 2011 SIAM International Conference on Data Mining (SDM)*, pages 13–24. SIAM, 2011.

[90] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[91] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(6):1092–1104, 2011.

[92] M. N. Kurt, Y. Yilmaz, and X. Wang. Real-time nonparametric anomaly detection in high-dimensional settings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[93] Y. Lai, Y. Han, and Y. Wang. Anomaly detection with prototype-guided discriminative latent embeddings. In *IEEE international conference on data mining (ICDM)*, pages 300–309. IEEE, 2021.

[94] M. T. R. Laskar, J. X. Huang, V. Smetana, C. Stewart, K. Pouw, A. An, S. Chan, and L. Liu. Extending isolation forest for anomaly detection in big data via k-means. *ACM Transactions on Cyber-Physical Systems*, 5(4):1–26, 2021.

[95] L. J. Latecki, A. Lazarevic, and D. Pokrajac. Outlier detection with kernel density functions. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 61–75. Springer, 2007.

[96] A. Lazarevic and V. Kumar. Feature bagging for outlier detection. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining (SIGKDD)*, pages 157–166, 2005.

[97] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[98] K. LEE, K. Lee, H. Lee, and J. Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *International Conference on Learning Representations (ICLR)*. ICLR 2018, 2018.

[99] J. Lesouple, C. Baudoin, M. Spigai, and J.-Y. Tourneret. Generalized isolation forest for anomaly detection. *Pattern Recognition Letters*, 149:109–119, 2021.

[100] K. Leung and C. Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science (ACCS)*, pages 333–342, 2005.

[101] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao. Deepfed: Federated deep learning for intrusion detection in industrial cyber–physical systems. *IEEE Transactions on Industrial Informatics*, 17(8):5615–5624, 2020.

[102] S. Li, K. Zhang, P. Duan, and X. Kang. Hyperspectral anomaly detection with kernel isolation forest. *IEEE Transactions on Geoscience and Remote Sensing*, 58(1):319–329, 2019.

[103] Z. Li, Y. Zhao, X. Hu, N. Botta, C. Ionescu, and G. Chen. Ecod: Unsupervised outlier detection using empirical cumulative distribution functions. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[104] J. Liang, Q. Liang, Z. Wu, H. Chen, S. Zhang, and F. Jiang. A novel unsupervised deep transfer learning method with isolation forest for machine fault diagnosis. *IEEE Transactions on Industrial Informatics*, 2023.

[105] L. Liao and B. Luo. Entropy isolation forest based on dimension entropy for anomaly detection. In *Computational Intelligence and Intelligent Systems: 10th International Symposium (ISICA)*, pages 365–376. Springer, 2019.

[106] W. Liao, Y. Guo, X. Chen, and P. Li. A unified unsupervised gaussian mixture variational autoencoder for high dimensional outlier detection. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1208–1217. IEEE, 2018.

[107] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22(3):2031–2063, 2020.

[108] Z. Lin, X. Liu, and M. Collu. Wind power prediction based on high-frequency scada data along with isolation forest and deep learning neural networks. *International Journal of Electrical Power & Energy Systems*, 118:105835, 2020.

[109] F. Liu, Y. Yu, P. Song, Y. Fan, and X. Tong. Scalable kde-based top-n local outlier detection over large-scale data streams. *Knowledge-Based Systems*, 204: 106186, 2020.

[110] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *IEEE International Conference on Data Mining (ICDM)*, pages 413–422, 2008.

[111] F. T. Liu, K. M. Ting, and Z.-H. Zhou. On detecting clustered anomalies using sciforest. In *Machine Learning and Knowledge Discovery in Databases: European Conference (ECML PKDD)*, pages 274–290. Springer, 2010.

[112] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation-based anomaly detection. In *ACM Transactions on Knowledge Discovery from Data*, pages 1–39. ACM, 2012.

[113] H. Liu, R. Wang, S. Shan, and X. Chen. Deep supervised hashing for fast image retrieval. In *IEEE CVPR*, pages 2064–2072, 2016.

[114] S.-C. Liu, Z.-G. Chen, Z.-H. Zhan, S.-W. Jeon, S. Kwong, and J. Zhang. Many-objective job-shop scheduling: a multiple populations for multiple objectives-based genetic algorithm approach. *IEEE Transactions on Cybernetics*, 2021.

[115] W. Liu, W. Luo, D. Lian, and S. Gao. Future frame prediction for anomaly detection–a new baseline. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 6536–6545, 2018.

[116] X. Liu, X. Nie, Q. Zhou, L. Nie, and Y. Yin. Model optimization boosting framework for linear model hash learning. In *IEEE Transactions on Image Processing*, volume 29, pages 4254–4268. IEEE, 2020.

[117] Y. Liu, C.-L. Li, and B. Póczos. Classifier two sample test for video anomaly detections. In *BMVC*, page 71, 2018.

[118] Y. Liu, Z. Li, C. Zhou, Y. Jiang, J. Sun, M. Wang, and X. He. Generative adversarial active learning for unsupervised outlier detection. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1517–1528, 2019.

[119] Y. Liu, S. Garg, J. Nie, Y. Zhang, Z. Xiong, J. Kang, and M. S. Hossain. Deep anomaly detection for time-series data in industrial iot: A communication-efficient on-device federated learning approach. *IEEE Internet of Things Journal*, 8(8):6348–6358, 2020.

[120] W. Lu, Y. Cheng, C. Xiao, S. Chang, S. Huang, B. Liang, and T. Huang. Unsupervised sequential outlier detection with deep architectures. *IEEE transactions on image processing*, 26(9):4321–4330, 2017.

[121] Y. Luo, Y. Xiao, L. Cheng, G. Peng, and D. Yao. Deep learning-based anomaly detection in cyber-physical systems: Progress and opportunities. *ACM Computing Surveys*, 54(5):1–36, 2021.

[122] L. Lyu, J. Jin, S. Rajasegarar, X. He, and M. Palaniswami. Fog-empowered anomaly detection in iot using hyperellipsoidal clustering. In *IEEE Internet of Things Journal*, volume 4, pages 1174–1184. IEEE, 2017.

[123] J. Ma, L. Sun, H. Wang, Y. Zhang, and U. Aickelin. Supervised anomaly detection in uncertain pseudoperiodic data streams. *ACM Transactions on Internet Technology (TOIT)*, 16(1):1–20, 2016.

[124] R. Ma, G. Pang, L. Chen, and A. van den Hengel. Deep graph-level anomaly detection by glocal knowledge distillation. In *The Fifteenth ACM International Conference on Web Search and Data Mining (WSDM)*, 2022.

[125] B. Mahesh. Machine learning algorithms-a review. *International Journal of Science and Research*, 9(1):381–386, 2020.

[126] A. Makhzani and B. Frey. K-sparse autoencoders. In *International conference on learning representations (ICLR)*, 2014.

[127] C. Manikopoulos and S. Papavassiliou. Network intrusion and fault detection: a statistical anomaly approach. *IEEE Communications Magazine*, 40(10):76–82, 2002.

[128] R. Mao, H. Xu, W. Wu, J. Li, Y. Li, and M. Lu. Overcoming the challenge of variety: big data abstraction, the next evolution of data management for aal communication systems. *IEEE Communications Magazine*, 53(1):42–47, 2015.

[129] E. Marchi, F. Vesperini, F. Weninger, F. Eyben, S. Squartini, and B. Schuller. Non-linear prediction with lstm recurrent neural networks for acoustic novelty detection. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2015.

[130] A. Mensi and M. Bicego. A novel anomaly score for isolation forests. In *Image Analysis and Processing–ICIAP 2019: 20th International Conference (ICIAP)*, pages 152–163. Springer, 2019.

[131] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[132] G. Montavon, W. Samek, and K.-R. Müller. Methods for interpreting and understanding deep neural networks. *Digital signal processing*, 73:1–15, 2018.

[133] G. Münz, S. Li, and G. Carle. Traffic anomaly detection using k-means clustering. In *Gi/itg workshop mmbnet*, volume 7, 2007.

[134] F. K. Nakano, K. Pliakos, and C. Vens. Deep tree-ensembles for multi-output prediction. *Pattern Recognition*, 121:108211, 2022.

[135] P. C. Ngo, A. A. Winarto, C. K. L. Kou, S. Park, F. Akram, and H. K. Lee. Fence gan: Towards better anomaly detection. In *2019 IEEE 31St International Conference on tools with artificial intelligence (ICTAI)*, pages 141–148. IEEE, 2019.

[136] M.-N. Nguyen and N. A. Vien. Scalable and interpretable one-class svms with deep learning and random fourier features. In *Machine Learning and Knowledge Discovery in Databases: European Conference (ECML PKDD)*, pages 157–172. Springer, 2019.

[137] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[138] K. Noto, C. Brodley, and D. Slonim. Frac: a feature-modeling approach for semi-supervised and unsupervised anomaly detection. *Data mining and knowledge discovery*, 25:109–133, 2012.

[139] J. H. Oh and J. Gao. A kernel-based approach for detecting outliers of high-dimensional biological data. In *BMC bioinformatics*, volume 10, pages 1–9. Springer, 2009.

[140] M.-h. Oh and G. Iyengar. Sequential anomaly detection using inverse reinforcement learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & data mining (SIGKDD)*, pages 1480–1490, 2019.

[141] S. Omar, A. Ngadi, and H. H. Jebur. Machine learning techniques for anomaly detection: an overview. *International Journal of Computer Applications*, 79(2), 2013.

[142] G. Pang. *Non-IID outlier detection with coupled outlier factors*. PhD thesis, 2019.

[143] G. Pang, L. Cao, L. Chen, and H. Liu. Learning representations of ultrahigh-dimensional data for random distance-based outlier detection. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining (SIGKDD)*, pages 2041–2050, 2018.

[144] G. Pang, C. Shen, and A. van den Hengel. Deep anomaly detection with deviation networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining (SIGKDD)*, pages 353–362, 2019.

[145] G. Pang, C. Yan, C. Shen, A. v. d. Hengel, and X. Bai. Self-trained deep ordinal regression for end-to-end video anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 12173–12182, 2020.

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

[147] G. Pang, A. van den Hengel, C. Shen, and L. Cao. Toward deep supervised anomaly detection: Reinforcement learning from partially labeled anomaly data. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining (SIGKDD)*, pages 1298–1308, 2021.

[148] G. Pang, J. Li, A. van den Hengel, L. Cao, and T. G. Dietterich. Andea: anomaly and novelty detection, explanation, and accommodation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 4892–4893, 2022.

[149] M. Pang, K.-M. Ting, P. Zhao, and Z.-H. Zhou. Improving deep forest by confidence screening. In *ICDM*, pages 1194–1199. IEEE, 2018.

[150] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *Proceedings 19th international conference on data engineering (ICDE)*, pages 315–326. IEEE, 2003.

[151] N. Paulauskas and A. Baskys. Application of histogram-based outlier scores to detect computer network anomalies. *Electronics*, 8(11):1251, 2019.

[152] M. Pavlidou and G. Zioutas. Kernel density outlier detector. In *Topics in Nonparametric Statistics: Proceedings of the First Conference of the International Society for Nonparametric Statistics*, pages 241–250. Springer, 2014.

[153] P. Perera, R. Nallapati, and B. Xiang. Ocgan: One-class novelty detection using gans with constrained latent representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pages 2898–2906, 2019.

[154] K. Pliakos and C. Vens. Mining features for biomedical data using clustering tree ensembles. *Journal of biomedical informatics*, 85:40–48, 2018.

[155] X. Qin, K. M. Ting, Y. Zhu, and V. C. Lee. Nearest-neighbour-induced isolation similarity and its impact on density-based clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 33, pages 4755–4762, 2019.

[156] H. Qu, Z. Li, and J. Wu. Integrated learning method for anomaly detection combining klsh and isolation principles. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–6. IEEE, 2020.

[157] M. Radovanović, A. Nanopoulos, and M. Ivanović. Reverse nearest neighbors in unsupervised distance-based outlier detection. *IEEE transactions on knowledge and data engineering*, 27(5):1369–1382, 2014.

[158] A. Rajaraman and J. D. Ullman. *Mining of Massive Datasets*. Cambridge University Press, New York, NY, USA, 2011. ISBN 1107015359, 9781107015357.

[159] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data (SIGMOD)*, pages 427–438, 2000.

[160] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré. Snorkel: Rapid training data creation with weak supervision. *Proceedings VLDB Endowment. International Conference on Very Large Data Bases*, 11(3):269–282, 2017.

[161] D. Ribeiro, L. M. Matos, G. Moreira, A. Pilastri, and P. Cortez. Isolation forests and deep autoencoders for industrial screw tightening anomaly detection. *Computers*, 11(4):54, 2022.

[162] P. J. Rousseeuw and A. M. Leroy. *Robust regression and outlier detection*. John wiley & sons, 2005.

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

[164] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795, 2021.

[165] K. Russell. Estimating the value of e by simulation. *The American Statistician*, 45(1):66–68, 1991.

[166] S. J. Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

[167] M. Sabokrou, M. Khalooei, M. Fathy, and E. Adeli. Adversarially learned one-class classifier for novelty detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3379–3388, 2018.

[168] D. Samariya and A. Thakkar. A comprehensive survey of anomaly detection algorithms. *Annals of Data Science*, 10(3):829–850, 2023.

[169] K. K. Santhosh, D. P. Dogra, and P. P. Roy. Anomaly detection in road traffic using visual surveillance: A survey. *ACM Computing Surveys (CSUR)*, 53(6): 1–26, 2020.

[170] M. H. Satman. A new algorithm for detecting outliers in linear regression. *International Journal of statistics and Probability*, 2(3):101, 2013.

[171] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International conference on information processing in medical imaging (IPML)*, pages 146–157. Springer, 2017.

[172] T. Schlegl, P. Seeböck, S. M. Waldstein, G. Langs, and U. Schmidt-Erfurth. f-anogan: Fast unsupervised anomaly detection with generative adversarial networks. *Medical image analysis*, 54:30–44, 2019.

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

[174] E. Schubert, A. Zimek, and H.-P. Kriegel. Generalized outlier detection with flexible kernel density estimates. In *Proceedings of the 2014 SIAM International Conference on Data Mining (SDM)*, pages 542–550. SIAM, 2014.

[175] E. Schubert, A. Zimek, and H.-P. Kriegel. Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection. *Data mining and knowledge discovery*, 28:190–237, 2014.

[176] A. Shabtai, Y. Elovici, and L. Rokach. *A survey of data leakage detection and prevention solutions*. Springer Science & Business Media, 2012.

[177] C. Shao, X. Du, J. Yu, and J. Chen. Cluster-based improved isolation forest. *Entropy*, 24(5):611, 2022.

[178] S. Shekhar, C.-T. Lu, and P. Zhang. Detecting graph-based spatial outliers: algorithms and applications (a summary of results). In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 371–376. ACM, 2001.

[179] Y. Shen, H. Liu, Y. Wang, Z. Chen, and G. Sun. A novel isolation-based outlier detection method. In *Trends in Artificial Intelligence: 14th Pacific Rim International Conference on Artificial Intelligence (PRICAI)*, pages 446–456. Springer, 2016.

[180] T. Shon and J. Moon. A hybrid machine learning approach to network anomaly detection. *Information Sciences*, 177(18):3799–3821, 2007.

[181] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang. A novel anomaly detection scheme based on principal component classifier. In *Proceedings of the IEEE foundations and new directions of data mining workshop*, pages 172–179. IEEE Press, 2003.

[182] M. A. Siddiqui, A. Fern, T. G. Dietterich, and W.-K. Wong. Sequential feature explanations for anomaly detection. In *ACM Transactions on Knowledge Discovery from Data*, volume 13, pages 1–22. ACM New York, NY, USA, 2019.

[183] G. Staerman, P. Mozharovskyi, S. Clémençon, and F. d'Alché Buc. Functional isolation forest. In *Asian Conference on Machine Learning (ACML)*, pages 332–347. PMLR, 2019.

[184] J. Sternby, E. Thormarker, and M. Liljenstam. Anomaly detection forest. In *European Conference on Artificial Intelligence (ECAI)*, pages 1507–1514. IOS Press, 2020.

[185] M. Sugiyama and K. Borgwardt. Rapid distance-based outlier detection via sampling. *Advances in neural information processing systems*, 26, 2013.

[186] W. Sultani, C. Chen, and M. Shah. Real-world anomaly detection in surveillance videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 6479–6488, 2018.

[187] H. Sun, Q. He, K. Liao, T. Sellis, L. Guo, X. Zhang, J. Shen, and F. Chen. Fast anomaly detection in multiple multi-dimensional data streams. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1218–1223. IEEE, 2019.

[188] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[189] W. Szpankowski. On the analysis of the average height of a digital trie: Another approach. pages 86–646, Department of Computer Science, Purdue University, Tech. Rep, 1986.

[190] 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)*, pages 535–548. Springer, 2002.

[191] T. Tayeh, S. Aburakhia, R. Myers, and A. Shami. Distance-based anomaly detection for industrial surfaces using triplet networks. In *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0372–0377. IEEE, 2020.

[192] L. Tenenboim-Chekina, L. Rokach, and B. Shapira. Ensemble of feature chains for anomaly detection. In *Multiple Classifier Systems: 11th International Workshop (MCS)*, pages 295–306. Springer, 2013.

[193] S. Thiprungsri and M. A. Vasarhelyi. Cluster analysis for anomaly detection in accounting data: An audit approach. *International Journal of Digital Accounting Research*, 11, 2011.

[194] K. M. Ting, Y. Zhu, and Z.-H. Zhou. Isolation kernel and its effect on svm. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (SIGKDD)*, pages 2329–2337, 2018.

[195] K. M. Ting, B.-C. Xu, T. Washio, and Z.-H. Zhou. Isolation distributional kernel: A new tool for kernel based anomaly detection. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (SIGKDD)*, pages 198–206, 2020.

[196] K. M. Ting, B.-C. Xu, T. Washio, and Z.-H. Zhou. Isolation distributional kernel a new tool for point & group anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[197] K. M. Ting, Z. Liu, H. Zhang, and Y. Zhu. A new distributional treatment for time series and an anomaly detection investigation. *Proceedings of the VLDB Endowment*, 15(11):2321–2333, 2022.

[198] K. M. Ting, T. Washio, J. Wells, H. Zhang, and Y. Zhu. Isolation kernel estimators. *Knowledge and Information Systems*, 65(2):759–787, 2023.

[199] R. Tudor Ionescu, S. Smeureanu, B. Alexe, and M. Popescu. Unmasking the abnormal events in video. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 2895–2903, 2017.

[200] L. Utkin, A. Ageev, A. Konstantinov, and V. Muliukha. Improved anomaly detection by using the attention-based isolation forest. *Algorithms*, 16(1):19, 2022.

[201] M. Van Otterlo and M. Wiering. Reinforcement learning and markov decision processes. In *Reinforcement learning: State-of-the-art*, pages 3–42. Springer, 2012.

[202] M. E. Villa-Pérez, M. A. Alvarez-Carmona, O. Loyola-Gonzalez, M. A. Medina-Pérez, J. C. Velazco-Rossell, and K.-K. R. Choo. Semi-supervised anomaly detection algorithms: A comparative summary and future research directions. *Knowledge-Based Systems*, 218:106878, 2021.

[203] N. X. Vinh, J. Chan, S. Romano, J. Bailey, C. Leckie, K. Ramamohanarao, and J. Pei. Discovering outlying aspects in large datasets. *Data mining and knowledge discovery*, 30:1520–1555, 2016.

[204] B. Wang, G. Xiao, H. Yu, and X. Yang. Distance-based outlier detection on uncertain data. In *2009 Ninth IEEE international conference on computer and information technology (ICCIT)*, volume 1, pages 293–298. IEEE, 2009.

[205] H. Wang, Z. Lei, X. Zhang, B. Zhou, and J. Peng. Machine learning basics. *Deep learning*, pages 98–164, 2016.

[206] H. Wang, G. Pang, C. Shen, and C. Ma. Unsupervised representation learning by predicting random distances. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2020.

[207] H. Wang, G. Pang, C. Shen, and C. Ma. Unsupervised representation learning by predicting random distances. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2950–2956, 2021.

[208] J. Wang, J. Wang, N. Yu, and S. Li. Order preserving hashing for approximate nearest neighbor search. In *ACM International Conference on Multimedia (ACM MM)*, pages 133–142, 2013.

[209] J. Wang, T. Zhang, N. Sebe, H. T. Shen, et al. A survey on learning to hash. In *IEEE transactions on pattern analysis and machine intelligence*, volume 40, pages 769–790. IEEE, 2017.

[210] R. Wang, K. Nie, T. Wang, Y. Yang, and B. Long. Deep learning for anomaly detection. In *WSDM*, pages 894–896, 2020.

[211] S. Wang, Y. Zeng, X. Liu, E. Zhu, J. Yin, C. Xu, and M. Kloft. Effective end-to-end unsupervised outlier detection via inlier priority of discriminative network. *Advances in neural information processing systems*, 32, 2019.

[212] X. Wang, T. Zhang, G.-J. Qi, J. Tang, and J. Wang. Supervised quantization for similarity search. In *IEEE CVPR*, pages 2018–2026, 2016.

[213] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

[214] A. S. Weigend, M. Mangeas, and A. N. Srivastava. Nonlinear gated experts for time series: Discovering regimes and avoiding overfitting. *International Journal of Neural Systems*, 6(04):373–399, 1995.

[215] K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480, 2006.

[216] P. H. Winston. *Artificial intelligence*. Addison-Wesley Longman Publishing Co., Inc., 1984.

[217] P. Wu, J. Liu, and F. Shen. A deep one-class neural network for anomalous event detection in complex scenes. *IEEE transactions on neural networks and learning systems*, 31(7):2609–2622, 2019.

[218] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding. Data mining with big data. In *IEEE transactions on knowledge and data engineering*, volume 26, pages 97–107. IEEE, 2013.

[219] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding. Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering*, 1(26):97–107, 2014.

[220] X. Wu, Y. Yang, M. Bilal, L. Qi, and X. Xu. 6g-enabled anomaly detection for metaverse healthcare analytics in internet of things. *IEEE Journal of Biomedical and Health Informatics*, 2023.

[221] H. Xiang and X. Zhang. Edge computing empowered anomaly detection framework with dynamic insertion and deletion schemes on data streams. *World Wide Web*, 25(5):2163–2183, 2022.

[222] H. Xiang, Z. Salcic, W. Dou, X. Xu, L. Qi, and X. Zhang. Ophiforest: order preserving hashing based isolation forest for robust and scalable anomaly detection. In *Proceedings of the 29th ACM international conference on information & knowledge management (CIKM)*, pages 1655–1664, 2020.

[223] H. Xiang, J. Wang, K. Ramamohanarao, Z. Salcic, W. Dou, and X. Zhang. Isolation forest based anomaly detection framework on non-iid data. *IEEE Intelligent Systems*, 36(3):31–40, 2021.

[224] H. Xiang, H. Hu, and X. Zhang. Deepiforest: A deep anomaly detection framework with hashing based isolation forest. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 1251–1256. IEEE, 2022.

[225] H. Xiang, X. Zhang, H. Hu, L. Qi, W. Dou, M. Dras, A. Beheshti, and X. Xu. Optiforest: Optimal isolation forest for anomaly detection. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2023.

[226] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning (ICML)*, pages 478–487. PMLR, 2016.

[227] M. Xie, J. Hu, and B. Tian. Histogram-based online anomaly detection in hierarchical wireless sensor networks. In *2012 IEEE 11th international conference*

on trust, security and privacy in computing and communications (ICTSPCC), pages 751–759. IEEE, 2012.

[228] D. Xu, E. Ricci, Y. Yan, J. Song, and N. Sebe. Learning deep representations of appearance and motion for anomalous event detection. *arXiv preprint arXiv:1510.01553*, 2015.

[229] H. Xu, Y. Wang, L. Cheng, Y. Wang, and X. Ma. Exploring a high-quality outlying feature value set for noise-resilient outlier detection in categorical data. In *ACM CIKM*, pages 17–26, 2018.

[230] H. Xu, G. Pang, Y. Wang, and Y. Wang. Deep isolation forest for anomaly detection. *arXiv preprint arXiv:2206.06602*, 2022.

[231] H. Xu, G. Pang, Y. Wang, and Y. Wang. Deep isolation forest for anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–14, 2023.

[232] H. Xu, G. Pang, Y. Wang, and Y. Wang. Deep isolation forest for anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–14, 2023.

[233] X. Yang, L. J. Latecki, and D. Pokrajac. Outlier detection with globally optimal exemplar-based gmm. In *Proceedings of the 2009 SIAM international conference on data mining (SDM)*, pages 145–154. SIAM, 2009.

[234] Y. Yang, X. Xu, L. Wang, W. Zhong, C. Yan, and L. Qi. Fast anomaly detection based on data stream in network intrusion detection system. In *Proceedings of the ACM Turing Award Celebration Conference-China (TACC)*, pages 87–91, 2021.

[235] Y. Yang, X. Yang, M. Heidari, M. A. Khan, G. Srivastava, M. Khosravi, and L. Qi. Astream: Data-stream-driven scalable anomaly detection with accuracy guarantee in iiot environment. *IEEE Transactions on Network Science and Engineering*, 2022.

[236] C. Yao, X. Ma, B. Chen, X. Zhao, and G. Bai. Distribution forest: An anomaly detection method based on isolation forest. In *Advanced Parallel Processing Technologies: 13th International Symposium (APPT)*, pages 135–147. Springer, 2019.

[237] M. Ye, X. Peng, W. Gan, W. Wu, and Y. Qiao. Anopcn: Video anomaly detection via deep predictive coding network. In *Proceedings of the 27th ACM International Conference on Multimedia (ACM MM)*, pages 1805–1813, 2019.

[238] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27, 2014.

[239] D. Yu, G. Sheikholeslami, and A. Zhang. Findout: Finding outliers in very large datasets. *Knowledge and information Systems*, 4:387–412, 2002.

[240] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, and W. Wang. Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (ACM SIGKDD)*, pages 2672–2681, 2018.

[241] M. Z. Zaheer, J.-h. Lee, M. Astrid, and S.-I. Lee. Old is gold: Redefining the adversarially learned one-class classifier training paradigm. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14183–14193, 2020.

[242] V. Zavrtanik, M. Kristan, and D. Skočaj. Reconstruction by inpainting for visual anomaly detection. *Pattern Recognition*, 112:107706, 2021.

[243] H. Zenati, C. S. Foo, B. Lecouat, G. Manek, and V. R. Chandrasekhar. Efficient gan-based anomaly detection. *arXiv preprint arXiv:1802.06222*, 2018.

[244] H. Zenati, M. Romain, C.-S. Foo, B. Lecouat, and V. Chandrasekhar. Adversarially learned anomaly detection. In *2018 IEEE International conference on data mining (ICDM)*, pages 727–736. IEEE, 2018.

[245] D. Zha, K.-H. Lai, M. Wan, and X. Hu. Meta-aad: Active anomaly detection with deep reinforcement learning. In *IEEE International conference on data mining (ICDM)*, pages 771–780. IEEE, 2020.

[246] B. Zhang, H. Huang, L. E. Tibbs-Cortes, A. Vanous, Z. Zhang, K. Sanguinet, K. A. Garland-Campbell, J. Yu, and X. Li. Streamline unsupervised machine learning to survey and graph indel-based haplotypes from pan-genomes. *Molecular Plant*, 16(6):975–978, 2023.

[247] H. Zhang, L. Cao, P. VanNostrand, S. Madden, and E. A. Rundensteiner. Elite: Robust deep anomaly detection with meta gradient. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (SIGKDD)*, pages 2174–2182, 2021.

[248] K. Zhang, M. Hutter, and H. Jin. A new local distance-based outlier detection approach for scattered real-world data. In *Advances in Knowledge Discovery and Data Mining: 13th Pacific-Asia Conference (PAKDD)*, pages 813–822. Springer, 2009.

[249] X. Zhang, W. Dou, Q. He, R. Zhou, C. Leckie, R. Kotagiri, and Z. Salcic. Lshiforest: a generic framework for fast tree isolation based ensemble anomaly analysis. In *IEEE International Conference on Data Engineering (ICDE)*, pages 983–994. IEEE, 2017.

[250] Y. Zhang, N. A. Hamm, N. Meratnia, A. Stein, M. Van de Voort, and P. J. Havinga. Statistics-based outlier detection for wireless sensor networks. *International Journal of Geographical Information Science*, 26(8):1373–1392, 2012.

[251] Y. Zhang, Y. Chen, J. Wang, and Z. Pan. Unsupervised deep anomaly detection for multi-sensor time-series signals. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[252] Y. Zhang, J. Wang, Y. Chen, H. Yu, and T. Qin. Adaptive memory networks with

self-supervised learning for unsupervised anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[253] Y.-L. Zhang, J. Zhou, W. Zheng, J. Feng, L. Li, Z. Liu, M. Li, Z. Zhang, C. Chen, X. Li, et al. Distributed deep forest and its application to automatic detection of cash-out fraud. *ACM Transactions on Intelligent Systems and Technology*, 10 (5):1–19, 2019.

[254] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *IEEE CVPR*, pages 1556–1564, 2015.

[255] Y. Zhao, Z. Nasrullah, M. K. Hryniewicki, and Z. Li. Lscp: Locally selective combination in parallel outlier ensembles. In *Proceedings of the 2019 SIAM International Conference on Data Mining (SDM)*, pages 585–593. SIAM, 2019.

[256] Y. Zhao, R. Rossi, and L. Akoglu. Automatic unsupervised outlier model selection. *Advances in Neural Information Processing Systems*, 34:4489–4502, 2021.

[257] P. Zheng, S. Yuan, X. Wu, J. Li, and A. Lu. One-class adversarial nets for fraud detection. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 33, pages 1286–1293, 2019.

[258] 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 (SIGKDD)*, pages 665–674, 2017.

[259] J. T. Zhou, J. Du, H. Zhu, X. Peng, Y. Liu, and R. S. M. Goh. Anomalynet: An anomaly detection network for video surveillance. *IEEE Transactions on Information Forensics and Security*, 14(10):2537–2550, 2019.

[260] Z.-H. Zhou. *Machine learning*. Springer Nature, 2021.

[261] Z.-H. Zhou and J. Feng. Deep forest. *arXiv preprint arXiv:1702.08835*, 2017.

[262] Z.-H. Zhou and J. Feng. Deep forest: towards an alternative to deep neural networks. In *IJCAI*, pages 3553–3559, 2017.

[263] Z.-H. Zhou and J. Feng. Deep forest. *National science review*, 6(1):74–86, 2019.

[264] H. Zhu and Y. Jin. Multi-objective evolutionary federated learning. *IEEE transactions on neural networks and learning systems*, 31(4):1310–1322, 2019.

[265] A. Zimek and P. Filzmoser. There and back again: Outlier detection between statistical reasoning and data mining algorithms. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(6):e1280, 2018.

[266] A. Zimek, M. Gaudet, R. J. Campello, and J. Sander. Subsampling for efficient and effective unsupervised outlier detection ensembles. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining (SIGKDD)*, pages 428–436, 2013.

[267] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International conference on learning representations (ICLR)*, 2018.