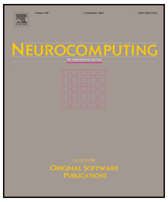
Neurocomputing 581 (2024) 127525

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Layered isolation forest: A multi-level subspace algorithm for improving isolation forest

Tao Liu, Zhen Zhou *, Lijun Yang

School of Computer Science and Engineering, Southwest Minzu University, Chengdu, 610225, China

ARTICLE INFO

Communicated by R. Zhu

Keywords:
Anomaly detection
Isolation forest
Subspace

ABSTRACT

Anomaly detection is an important field in data science that has been widely researched and applied, generating many methods. Among these methods, the isolation forest algorithm is outstanding because of its efficiency and effectiveness, especially in regard to large-scale data. Unfortunately, this algorithm has some drawbacks, such as being unable to effectively handle local outliers, possibly leading to normal data masking the detection of outliers due to space partitioning along the coordinate axes, and low utilization of training data. To address these issues, in this paper we propose an improved isolation forest algorithm based on multilayer subspace dividing, named layered isolation forest, which adapts to the different distributions of the dataset by dividing the sample space into subspaces and evaluating the anomaly degree of data in different spatial ranges. This algorithm obtains a more accurate and reasonable anomaly score, avoids the problems of the original algorithm, and improves the performance metrics. According to the experimental results, the proposed method maintains the efficiency of the original algorithm and exhibits the best comprehensive performance compared with similar algorithms on artificial synthetic datasets and real-world datasets from multiple domains.

1. Introduction

With the continuous development of an information society, vast amounts of data are generated in all aspects of our lives. These data have spawned a series of research directions in data mining and pattern recognition, among which anomaly detection has always been an important field of attention and research [1]. Anomaly detection, also known as outlier detection [2], involves discovering and identifying new or unusual patterns of behavior; it involves various fields, including monitoring credit card fraud [3], predicting and diagnosing diseases [4], detecting network intrusions [5,6], exploring astronomy [7], and even environmental change [8], so further research on anomaly detection methods is necessary.

The isolation forest (iForest or IF) [9,10] proposed in 2008 is an excellent method based on the concept that outliers can be easily isolated from a dataset. This method randomly partitions the sample space along the axis, repeats this operation and abstracts this process as a tree structure to effectively identify anomalies in the dataset. Due to its outstanding performance and linear time complexity [9], this algorithm attracts extensive research and application. However, as the saying goes in the field of software engineering, there is no "silver bullet" [11]. The algorithm still suffers from certain limitations. According to the survey [12], we summarized some problems of the iForest: the original algorithm uses a global measure and is insensitive to detect local anomaly data; because it uses axis-parallel partitioning in the entire sample space, the anomaly data points are masked by normal data points; it randomly selects features to partition, does not fully utilize the training data and may miss important information. These problems make it difficult to assess the anomaly degree of data points and have poor performance in some cases [13], especially in complex data distribution situations, such as multimodal datasets. Despite some researchers' efforts to address the issues of the original algorithm [13], they also introduced higher computational costs [12], weakening the advantage of the original algorithm. With the aim of addressing this issue, in this paper we make the following contributions:

(1) We analyze the process of isolating normal data points and local anomaly data points by isolation forest, and summarize the reason why the original isolation forest algorithm has difficulty detecting local anomalies.

(2) We propose an enhanced method based on the isolation forest, named the layered isolation forest (LIF), which divides the sample space into equally sized subspaces and constructs local isolation forests. Finally, the global and local anomaly scores are combined to make more accurate judgments on the degree of anomaly of the data points.

(3) We conduct experiments to guide the selection of algorithm parameters in different datasets and provide some recommendations.

* Corresponding author.
E-mail addresses: liutaop@stu.swun.edu.cn (T. Liu), zhouzhen1302@163.com (Z. Zhou), ylijun@swun.edu.cn (L. Yang).

https://doi.org/10.1016/j.neucom.2024.127525
Received 29 July 2023; Received in revised form 3 December 2023; Accepted 4 March 2024
Available online 7 March 2024

(4) A series of experiments demonstrate that this method effectively overcomes the limitations of the original algorithm, enhances the algorithm performance metrics, and maintains a low computational complexity.

This paper is organized into five sections. Section 2 provides a brief overview of the iForest, its limitations, and related work. Section 3 explains the principle and procedure of the LIF algorithm and discusses the parameters. Section 4 presents a comparison and evaluation of our algorithm with other competing algorithms using heatmaps of four synthetic datasets and area under the receiver operating characteristic curve ($ROC - AUC$) metrics and execution time of eight real-world benchmark data. Finally, Section 5 concludes the paper with a succinct summary and discussion.

2. Problem analysis and related work

$$s(x,n) = 2^{-E(h(x))/c(n)} \tag{1}$$

$$E(h(x)) = \sum_{i=0}^{t} h_i(x)/t \tag{2}$$

$$c(n) = 2H(n-1) - (2(n-1)/n) \tag{3}$$

$$H(n) = \ln(n) + \gamma \tag{4}$$

2.1. Original isolation forest

The isolation forest algorithm is an ensemble learning method [2] and is an unsupervised anomaly detection algorithm [10]. Similar to random forest, this algorithm uses binary search trees instead of decision trees. Based on the assumption that outliers are few and different from normal data in the sample space [14], they are easier to separate from the sample than normal data. Unlike most algorithms that attempt to build a distribution of normal data, this algorithm focuses more on outlier data [12], isolating data by randomly selecting features and splitting values to partition the sample space without calculating metrics such as distance or density between data points. Those data points that are easier to isolate are more likely to be outliers [9]. Furthermore, the algorithm achieves high efficiency and linear time complexity by using randomly selected subsamples that can be very small [9].

The basic algorithm flow is as follows [10]: In the training phase, some subsamples are extracted to build "isolation tree", which are partitioned by randomly selecting a dimension and a value on that dimension. The partition space is repeatedly processed until only one sample remains or a certain number is attained. The whole process is abstracted as a binary search tree, and the number of partitions needed to isolate a data point is equivalent to the depth of the leaf node where it is located. This depth is used to evaluate the anomaly score. Multiple trees are built to form a forest to ensure more accurate results. In the evaluation phase, the test samples are placed into all constructed trees, and their anomaly scores are evaluated according to the Eq. (1) based on the depth of their nodes. n denotes the number of samples, $h(x)$ represents the depth of data point x in an isolation tree, and $E(h(x))$ in Eq. (2) indicates the average depth of data point x in t isolation trees. As the isolation forest utilizes binary search trees, the average depth of n samples is equivalent to the average path length of unsuccessful search in binary search trees, denoted by Eq. (3). Here, $H(x)$ represents the harmonic number, which can be calculated using Eq. (4). Euler's constant γ is approximately equal to 0.577215665 [10]. To normalize $h(x)$, this algorithm employs the normalization factor $c(n)$. The score s is between (0, 1), and the closer it is to 1, the higher the degree of anomaly is considered.

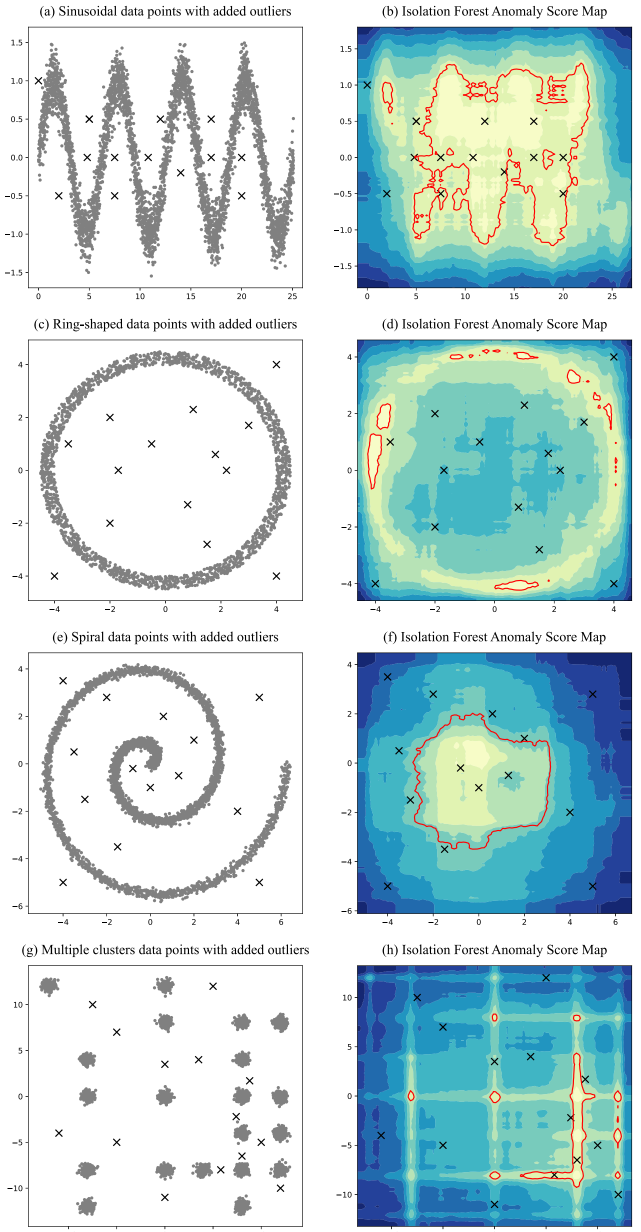


Fig. 1. Four artificial datasets were evaluated using the isolation forest algorithm, and the evaluation results were visualized as heatmaps. Where gray dots represent normal data points and black crosses represent artificially added anomalous data with different patterns from normal data. The heatmaps display different shades of color, with lighter areas indicating lower anomaly levels and darker areas indicating higher anomaly levels. In the heatmaps, the red line is drawn at an anomaly score threshold of 0.5. The enclosed region within the line is considered the region of normal data distribution, and the region outside represents the anomalous region.

2.2. Exist problems

In this work we applied the isolation forest algorithm to four artificially generated datasets with complex distributions; they are sinusoidal datasets consisting of 4000 data points, ring-shaped datasets consisting of 4000 data points, spiral datasets consisting of 4000 data points and multicluster datasets with 20 clusters, each cluster having 200 data points, totaling 4000. These data are represented by gray circles, and we also manually added some anomalies, which are marked by black crosses, as shown in Fig. 1(a)(c)(e)(g). Then, we sampled uniformly in the plotting space and used the previously constructed

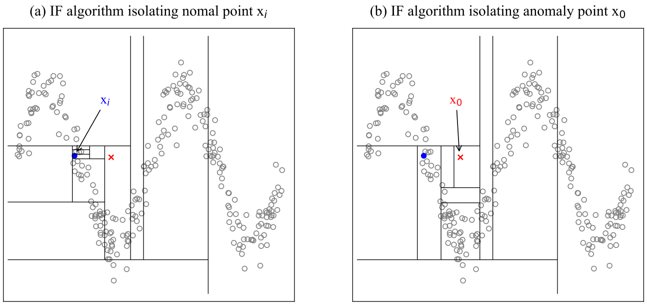


Fig. 2. We choose one random process of isolating the normal data points x_i (blue) and the abnormal data points x_0 (red) in a given sine dataset with 256 points as an example; other gray dots represent the sample data, and every black solid line is a partition process.

isolation forest to evaluate the anomaly scores of these sampled data, generating heatmaps of anomaly scores. Ideally, the anomaly scores would accurately reflect the distribution of the original data [13]. According to the definition of anomalies, data points in dense regions should have lower scores, while outliers far from dense regions should have higher scores [10]. However, as shown in Fig. 1(b)(d)(f)(h), because of the global axis partitioning of the sample space and the lack of attention to local details in datasets, the algorithm could not effectively detect all the outliers, and misclassification occurred.

In the sinusoidal dataset, the algorithm has difficulty identifying outliers at the bottom of the sine wave and fails to accurately identify the distribution of the data. It is clear that the "normal area" recognized by the algorithm has an irregular shape. In the ring-shaped dataset, the algorithm also reflects the distribution of the data inaccurately, and it can be seen that due to the axial splitting, the algorithm tends to identify the ring-shaped data as close to a rectangle, which is undoubtedly incorrect. The spiral-shaped dataset has a higher density of data points closer to the center, so the algorithm identifies The central area as "normal", ignoring the spiral-shaped data distribution. Finally, we observe the "ghost" [13] phenomenon mentioned earlier in the multicluster dataset, where the area between two clusters without normal data points is also included in the "normal area", but some data clusters are missed. It can be seen that the classic isolation forest algorithm in these cases has difficulty identifying the outliers we added and does not reflect the distribution of the original normal data well. We chose these datasets, which can more fully show the defects of the classic algorithm.

We further studied and analyzed the process of isolating data by the classical isolation forest algorithm. Taking Fig. 2 as an example, (a) and (b) are the random processes of isolating normal data x_i (blue) and abnormal data x_0 (red) in the artificial dataset using the original algorithm, respectively. The complexity of the dataset distribution, especially the nonconvexity of the dataset, can easily produce the situation in which normal data cover the abnormal data along the axis, resulting in "ghosts". At the same time, as shown in the figure, normal data such as x_i have local outliers x_0 with close distance, and the algorithm also has difficulty judging such local anomalies and cannot effectively and quickly isolate them. In the experiment, among the 100 randomly generated isolation trees, the average depth needed to isolate the local abnormal data point x_0 was approximately 10.4, close to the 10.5 needed for normal data such as x_i to be isolated, and both were higher than the maximum average depth $c(n) = 8$ set by the algorithm. This experiment illustrates that the isolation depth required by local anomaly data points and normal data points in the global scope is too close, and the original algorithm is difficult to distinguish between them. Therefore, in the global scope, they are all judged as normal data points, which is contradictory to our observation and artificial judgment.

Based on our previous experiments, analysis and literature review, we summarized several limitations of the original isolation forest algorithm:

(1) The iForest algorithm misjudges some data points due to the masking of anomalies, which is called the "ghost" phenomenon, by normal data along the axes as a result of axial partitioning of the sample space; this makes the anomaly scores of some data points unreasonable and unfair [13]. As a result, the distribution of anomaly scores does not match the data distributions.

(2) The algorithm shows low sensitivity to outliers that are close to normal data points, resulting in a failure to accurately identify local anomalies and normal data.

(3) The algorithm randomly selects features and values for partitioning, so after training, there is still much unused information in the data samples used for training. This may miss important information for detecting anomalies.

These drawbacks reveal the limitations of the original algorithm in dealing with different data distributions and identifying anomalies precisely.

2.3. Overview of previous research

Unfortunately, it is difficult to directly identify anomalous data points in a dataset. Therefore, the popular method is to build a model to describe the majority of normal data points and use it to identify the anomalous data points that differ from their pattern [15].

There are many methods for anomaly detection, such as distance-based methods, which determine the degree of anomaly by the difference in distance between data [16,17], such as the k-nearest neighbor algorithm [18]; density-based methods, which are similar to distance-based methods, must calculate the density between data points and identify the anomalous data points, such as Local Outlier Factor(LOF) [19], etc. These two types of methods generally require considerable computation. Probability-based methods assume the distribution of the data. Under this strong assumption [20,21], they consider the data in the low-probability distribution areas as anomalous data, but the distribution of the data is often complex and does not necessarily follow this type of assumption. Cluster-based methods, which first build clusters by calculating distance or density and then use clusters to identify anomalous patterns [22], such as Density-Based Spatial Clustering of Applications with Noise(DBSCAN) [23], also have computational complexity problems. In recent years, many methods based on neural networks have been developed, and these methods can achieve excellent results through a large amount of neural network inference [24,25]. For example, Autoencoder [26], which is similar to principal component analysis, overcomes the linear limitation of Principal Component Analysis(PCA) when using nonlinear activation functions. Because of the use of neural networks, the algorithm requires a large amount of high-quality training data; in addition, the training is difficult, the algorithm cost is enormous, and it is not applicable in some scenarios.

The isolation forest is a special and outstanding algorithm. To address the limitations of the iForest algorithm and apply it to more fields, many researchers have proposed improvement measures based on the original algorithm and made outstanding contributions: Similar to SCiForest [27] proposed by Liu FT et al. or EgiTree [28] proposed by Shen Y et al. these methods no longer use random partitioning but introduce a new evaluation mechanism to find the "optimal" partition hyperplane, which improves the efficiency of isolation and makes more full use of the sample information; MassAD [29,30] or Ordinal Isolation [31] that use different ways to partition the sample space, but the effect of these methods depends on the distribution of the datasets; Zhang et al. combined the isolation mechanism and hash function and proposed LSHIForest [32], which uses local sensitive hashing forest (LSH) as the framework. This method can measure the degree of data isolation from multiple distance metrics, including the Manhattan distance, Euclidean distance, angle, and even kernel function. However, the algorithm must use hash functions to calculate the anomalies of data points, which is very costly. There are also improvements that combine other algorithms, such as iNNE [33] and K-means isolation forest [34], which combine the idea of isolation with other anomaly detection methods and isolate abnormal data by calculating the distance between data and achieve good results. Hariri et al. [13] proposed the extended isolation forest (EIF) algorithm, which uses hyperplanes with random slopes to partition the sample space and solve the problems of iForest. However, the EIF algorithm generates "empty branches" when cutting the subsample space using hyperplanes with random slopes [35], conflicting with the goal of iForest to isolate anomalies as soon as possible, resulting in reduced accuracy and efficiency. In addition, the algorithm needs to calculate the position information between data points and the partition hyperplane in the evaluation stage, leading to greater computational cost. Lesouple et al. [35] improved the EIF algorithm by selecting a slope and projecting the data in the normal direction to partition the sample between the maximum and minimum values, thereby solving the empty branch problem and improving the efficiency and stability of the algorithm. However, since EIF and generalized iForest (GIF) use random hyperplanes to partition the sample space, they are very sensitive to the scale of different dimensions and need to obtain zero mean and unit variance for each dimension [35]; this adds a preprocessing step compared to the original algorithm and slightly reduces the efficiency of the two algorithms above. Mondrian Forest [36] was proposed to better handle batch data and online stream data; it uses Mondrian trees to partition the sample space finely and capture the detailed data distribution. However, this method has limited performance, and Mondrian trees are inherently complex. Recently, an excellent algorithm called Deep Isolation Forest (DIF) [37] was proposed, which uses only an initialized deep network, without the need for deep network inference, to randomly map the data space and better isolate the anomalous data points. This algorithm achieves good results on multiple datasets [38], but despite the authors using some methods to reduce the complexity and execution time of the algorithm, it still cannot escape the high execution time due to the construction of the deep network and the mapping process when evaluating the anomaly scores.

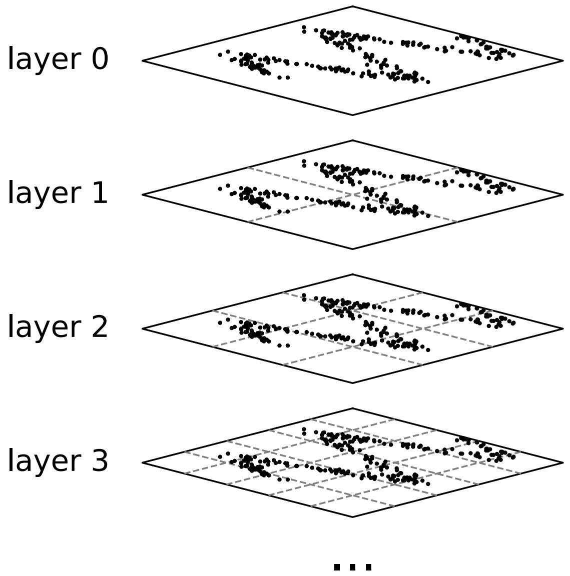


Fig. 3. An example of layerwise dividing of the sample space in a two-dimensional dataset, where the black dots represent the sample data and gray dashed lines indicate the dividing process in each layer.

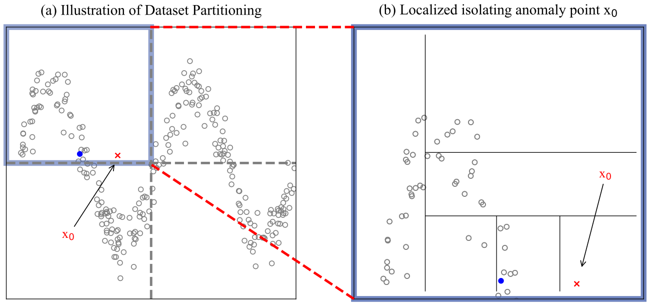


Fig. 4. A schematic of dataset splitting and the process of isolating the abnormal data point x_0 (red) in the local space. The blue rectangular area is the subspace where one of the local isolation forests was built. The average depth of 100 isolation trees isolating x_0 in the local space is 5.4, while the average depth of the data points in that space is $c(n) = 6$ (the normal data point x_0 is 8.1).

3. Layered isolation forest

3.1. Principle explanation

The above problems mainly stem from the fact that the iForest algorithm uses a global measure to construct isolation trees in the entire sample space and are also affected by the axial partitioning of the algorithm. To address these issues and maintain the simplicity of the algorithm as much as possible, we propose dividing the sample space into smaller subspaces of equal size and applying iForest both globally and locally. In detail, at the training stage, after constructing the global isolation forest, we take the sample data randomly selected when building the global iForest as the sample space that needs to be divided. Divide each dimension into equal parts in the sample space, resulting in several subspaces or hypercubes. Then, we build local isolation forests for each subspace, and in the evaluating stage, we evaluate the anomaly scores of data points in both global and local regions; this helps to address the problem of the original algorithm being unable to detect local anomalies and alleviates the axial masking of abnormal data by normal data points under the global method due to the introduction of local subspaces. By doing so, we can estimate a more precise and rational final anomaly score for data points and increase the usage of the data that is not completely utilized. Following this method, we can further divide each dimension of the sample space to generate more and smaller subspaces of equal size, as shown in Fig. 3, and then combine the anomaly scores of data at different layers using Eq. (6), where w_i represents the weight assigned to layer i. As shown in Fig. 4, it is evident that by increasing our attention to local data distributions, we increase the contrast of the sparsity degree between the abnormal data and the normal data, making the local anomaly easier to isolate. Compared with Fig. 2, we observe that in the original iForest, the outlier x_0, which is close to normal data points and difficult to isolate. However, within its corresponding subspace, it can be easily distinguished. The example in Fig. 4(b) represents a local isolation process, where the average depth $c(n)$ of data points in the subspace is 6, the depth of normal data points is 8.1, and the depth of the outlier x_0 is 5.4. Consequently, applying Eq. (1), we can calculate the local and global anomaly scores for data points. Here, we only select a simple and low-complexity weighting method[39], integrating local and global information of the sample space. Other methods can also be applied, such as methods based on kernel functions or methods based on multiscale analysis; however, these methods are not described here.

Based on the assumption that most normal data points are densely distributed while anomalies are rare and sparse, not all subspaces will contain data points when dividing the sample space. Therefore, it is only necessary to construct isolation forests for the subspaces that have data points; this significantly reduces the number of isolation forests needed, especially in higher-dimensional data where sparsity is more prevalent. Additionally, as the dividing goes deeper, the number of subspaces increases; the range of each subspace decreases, and the number of data points falling into them also decreases, resulting in lower depths of the isolation trees constructed.

$$W = \left\{ w_i \mid w_i \in (0,1), \sum_{i=0}^{Length(L)} w_i = 1 \right\} \tag{5}$$

$$S_{final} = \sum_{i=0}^{length(L)} w_i * s_i(x,n) \tag{6}$$

```
Algorithm 1 $iTree(X, e, l)$
Input: $X$ - input data, $e$ - current tree height, $l$ - height limit
Output: an iTree
if $e \geq l$ or $|X| \leq 1$ then
    return $exNode\{Size \leftarrow |X|\}$
else
    let $Q$ be a list of attributes in $X$
    randomly select $q \in Q$
    randomly select a split point $p$ between the min and max values of $q$ in $X$
    $X_l \leftarrow filter(X, q < p)$
    $X_r \leftarrow filter(X, q \geq p)$
    return $inNode\{Left \leftarrow iTree(X_l, e+1, l)$,
                   $Right \leftarrow iTree(X_r, e+1, l)$,
                   $SplitAtt \leftarrow q$,
                   $SplitValue \leftarrow p\}$
end if
```

3.2. Algorithm process and costs

```
Algorithm 2 $LiForest$
Input: $X$ - input data, $t$ - number of trees, $\psi$ - subsampling size,
       $L = [l_1, l_2 ... l_l]$ - list of split nums
Output: A LiForest composed by a set of iForest.
Initialize $Forest$
set height limit $d = ceiling(\log_2 \psi)$
for $i = 1$ to $t$ do
    $X' \leftarrow sample(X, \psi)$
    $Forest \leftarrow Forest \cup iTree(X', 0, d)$
    $X_u \leftarrow X_u \cup X'$
end for
$LiForest \leftarrow LiForest \cup Forest$
for $l_i (i = 2$ to $l)$ in $L$ do
    for $x$ in $X_u$ do
        use $l_i$ to divide $x$ in subspace hypercube $c$
                                   ▷ each $x$ corresponds to one $c$ in every $l_i$
        $c \leftarrow c \cup x$
        $C \leftarrow C \cup c$
    end for
    for $c$ in $C$ do
        Initialize $Forests$
        set height limit $d = ceiling(\log_2 \psi)$
        for $i = 1$ to $t$ do
            $X' \leftarrow sample(X, \psi)$
            $Forest \leftarrow Forest \cup iTree(X', 0, d)$
        end for
        $LiForest \leftarrow LiForest \cup Forest$
    end for
end for
```

The layered isolation forest (LIF) algorithm introduces a minor modification to the original algorithm flow. The tree construction process in Algorithm 1 remains unchanged. In Algorithm 2, after building an iForest, the training data of this iForest needs to be divided, and for each divided hypercube c, the isolation forest algorithm is applied while keeping other parameters constant. The resulting isolation forests are added to the LIForest, forming the LIF constructed from multiple isolation forests. Algorithm 3 evaluates the path in the same way as the original algorithm. In Algorithm 4, when evaluating data, the anomaly degree of data point x at different levels is computed. Therefore, x needs to be evaluated by the number of isolation forests corresponding to the number of times the sample space was divided, denoted as $len(L)$.

The original iForest algorithm with a complexity of $O(t\psi \log \psi)$ in the training stage and $O(nt \log \psi)$ in the evaluating stage, where ψ represents the number of subsamples, t represents the number of isolation trees constructed and n is the evaluating instances. Afterward, it was divided into hypercubes. The IF algorithm was applied again to the hypercubes that had sample points. The complexity of this step depended on the distribution of datasets, which was reduced by the fact that normal data were usually clustered, and thus, only a small proportion of hypercubes had sample points in high-dimensional spaces. For example, in the shuttle dataset with 9 dimensions, each dimension was split into 9 small intervals, leading to 9^9 possible hypercubes. However, only approximately 54 of them had sample points. The relationship between the time complexity of the algorithm and the dimensions and data size is observed to be negligible. Hence, the overall complexity was $(c \cdot \psi t \log \psi) + (l \cdot nt \log \psi)$, where c is the number of hypercubes to be processed and l is the length of the divide layers. Still maintains the linear time complexity, and the space complexity is $c\psi t$. The experimental results validated this analysis, and the running time has a very low connection with the dimension and scale of the dataset.

```
Algorithm 3 $PathLength(x, T, e)$
Input: $x$ - an instance, $T$ - an iTree, $e$ - current path length,
       to be initialized to zero when first called
Output: path length of $x$
if T is an external node then
    return $e + c(T.size)$ {$c(.)$ is defind in Eq. (1)}
end if
$a \leftarrow T.SplittAtt$
if $x_a < T.splitValue$ then
    return $PathLength(x, T.left, e + 1)$
else $x_a \geq T.splitValue$
    return $PathLength(x, T.right, e + 1)$
end if
```

```
Algorithm 4 $Evaluate\ Score$
Input: $x$ - an instance, $LiForest$ - a set of iForest
       $L$ - list of split nums
Output: anomaly scores of $x$ in different layer
for $l$ in $L$ do
    use $l_i$ divide $x$ in which $iForest$ in layer $i$
    calculate $s_i$ in Eq. (1)
    $Scores \leftarrow s_i$
end for
return $Scores$
```

By employing this modification, we improve the performance of the original algorithm and solve problems such as local anomaly insensitivity, the "ghost" phenomenon and insufficient utilization of training data while keeping the algorithm at low time complexity. The implementation code of the paper will be released later.

3.3. Parameters

One of the drawbacks of the original iForest is that it does not fully exploit the information in the data used for tree construction, as it randomly selects features and values for splitting. Some key information that isolates anomalies may be lost. Our method further splits the data used for tree construction and builds local isolation forests based on the

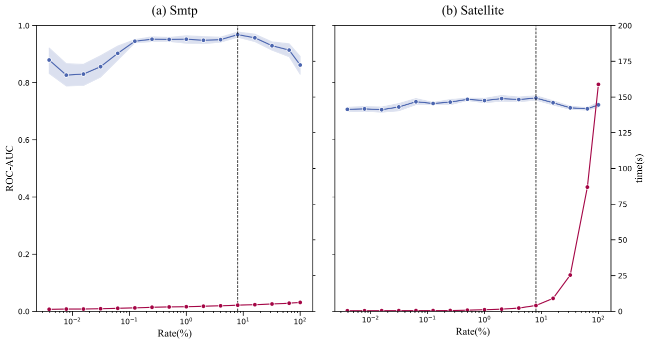


Fig. 5. Relationship between the rate of data usage and ROC-AUC, execution time. (a) is the experimental result in the Smtp dataset, and (b) is the Satellite. x-axis is the rate of usage in the dataset, the left y-axis and blue line are the values of the ROC-AUC, the right y-axis and red line are the execution time, and the colored area corresponding to the line indicates the 95% confidence interval.

original iForest, enhancing the utilization efficiency of the data. In the original algorithm, the number of isolation trees is usually set to 100, and each tree randomly draws some data points from the sample space to construct the isolation tree. This parameter is generally set to 256. We keep these parameters unchanged, so the data needs to be divided is 25 600. Considering the size of the original dataset, there may be duplicate data points among them. We also randomly extract the data points that must be divided from the datasets according to a certain proportion. To balance the performance and efficiency of the algorithm, we conducted experiments on the relationship between the proportion of datasets used for dividing, algorithm performance, and algorithm execution time on multiple datasets. As shown in the experimental results of the two datasets in Fig. 5, the figures selected low-dimensional and high-dimensional datasets, which are more representative. $ROC-AUC$ converged when using approximately 8% of the data points in the dataset Smtp; in the dataset Satellite, it also converged when using approximately 4% of the data points. Continuing to increase the proportion of datasets used will not achieve better algorithm performance but will further increase the algorithm execution time. In a nutshell, using a lower proportion of data (we use 5% in this paper uniformly) can achieve a balance between performance and efficiency.

$$W^{*} = \underset{W}{argmax}\, f(W) \tag{7}$$

In this work, we must determine the level of division $L = \{L_1, L_2 ..., L_i\}$ and the contribution of each level to the final anomaly score, i.e., the weight $W = \{W_1, W_2 ..., W_i\}$ in the training stage. It is obvious that, due to the different distribution situations of different datasets, the parameters L and W are different on different datasets. The number of data points that must be divided in the above text has been determined, and it can be easily seen that as the division of each dimension goes deeper, more hypercubes will be generated, and the average number of data points in each hypercube will also decrease. When the number of data points is too low, it is difficult for the isolation forest to construct an effective isolation forest; meanwhile, more hypercubes mean more local isolation forests, which also bring further algorithmic overhead. To balance the efficiency and performance of the algorithm, we calculate the average number of data points in the hypercube in different depths of divisions *average points* and use it to restrict the division depth of the algorithm. When the change of the average data points remains at a low level, we can observe that it is convergence. Fig. 6 shows the change in the average number of data points in hypercubes with the division depth of the 11 datasets we used.

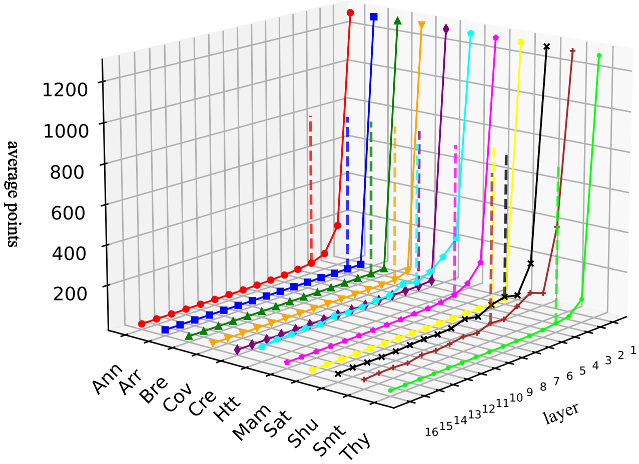


Fig. 6. This figure depicts the variation in average data points within hypercubes across different partition depths for the 11 datasets used in the experiment. The z-axis represents the quantity of $averagepoints$, and the y-axis corresponds to the algorithm's partition depth, denoted as L. Higher-dimensional datasets exhibit a rapid decrease in $averagepoints$, approaching convergence at lower partition depths, while lower-dimensional datasets with larger numbers tend to stabilize in average points at deeper partition depths. Therefore, the determination of L should be contingent upon the specific characteristics of the dataset. Once the $averagepoints$ stabilize without significant fluctuations, further partitioning beyond that point is unnecessary.

This figure shows the characteristics of high-dimensional datasets, such as *Arrhythmia*. The average number of points stabilizes at a relatively shallow level L_i, and it hardly changes after further division, which means that the number of hypercubes also hardly increases. Further division cannot provide a more accurate description of the data distribution but increases the algorithmic overhead. For the situation where the data size is large but the dimension is low, such as $Smtp$, it can be seen that as the division goes deeper, the average number of data points in the hypercubes continues to decline, and the average number of data points stabilizes when L_i is 8, so the parameter L of such datasets is 8. Therefore, we think that the depth of the division should be limited to 8. To reduce the impact of division on the original data distribution, we use $L = \{x \in \mathbb{N} \mid \gcd(x, y) = 1, \forall y \in L, y \neq x\}$ to restrict the choice of L because this prevents the algorithm from repeatedly dividing the same position in the global sample space, which may introduce new bias in the sample space. After that, we use $ROC-AUC$ to determine the parameter W and need to satisfy Eq. (7), f represents the calculation process of ROC-AUC. Considering the efficiency, this paper uses the simplest grid search method, with a step size of 10%, which balances the efficiency and performance of the algorithm and only needs to choose an approximate optimal value. Many convex optimization algorithms can be applied to solve this problem, and this paper refrains from delving into excessive details. In particular, for the weights of local and global anomalies, it should be noted that increasing the proportion of local anomalies does not always lead to better results. Taking Fig. 7 as an example, in the $Satellite$ dataset, the $ROC-AUC$ reaches the highest value when the weight of global anomalies is approximately 90%, while for datasets such as $Annthyroid$, the weight of global anomalies is optimal at approximately 30%. The difference in data distribution causes the importance of different levels of anomaly values to be different. The expression and research of data distribution can further explore this issue, representing future research directions.

4. Experiment

4.1. Anomaly score heatmaps

We employ heatmaps on the four datasets shown in Fig. 8 to illustrate the differences between the original Isolation Forest (IF) algorithm, one of the improved algorithms Extended Isolation Forest

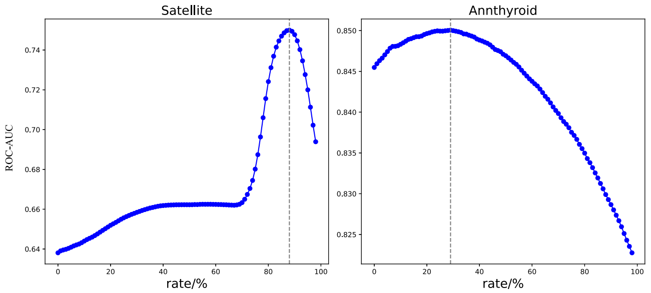


Fig. 7. This graph depicts the relationship between the proportion of global isolation forest anomaly scores on the x-axis, ranging from 0 to 100%, and the ROC-AUC on the y-axis. As the proportion of global anomaly scores increases gradually, the ROC-AUC achieves its maximum value at a certain proportion of global anomaly scores. Unrestricted increments or decrements in the global proportion make it challenging to attain the algorithm's optimal performance.

Table 1
Parameters of algorithms in the four artificial datasets.

Parameters		
iForest	Extended iForest	Layered iForest
sample_size *trees_number*	*sample_size* *trees_number* *extend_levels*	*split_nums* *layers_weight*
256 100	256 100 dimension-1	[1,3,5,7] [10%,10%,40%,40%]

(EIF) that we mentioned before, and we proposed the layered isolation forest (LIF) algorithm. The parameters of the three algorithms used in the four datasets are shown in Table 1. In the sinusoidal dataset, the original iForest fails to accurately identify all artificially added outlier data points, especially some outliers close to normal data located at the bottom of the sine wave. The representation of the normal data distribution is also not perfect. While EIF has greatly improved, because there are more data points in the center part, it is insufficient for characterizing the data close to both sides. Our proposed method, however, can still accurately identify anomalies and depict normal data. It can be observed that IF is able to identify artificially added anomalous data in the circular dataset; however, due to the axial partitioning in the global sample space, many normal data points fall within the anomaly range, and it fails to identify the distribution of the original data well. EIF, which eliminates the axial bias, performs better by accurately identifying the anomalies and most normal data points. LIF, although the bias caused by axial splitting still exists, combines the global and local anomaly scores to reduce the masking phenomenon of normal data on anomalous data. Provides a more accurate depiction of the circular dataset. The spiral dataset presents a greater challenge. Both IF and EIF algorithms classify only the densely populated central region as normal data, thereby missing the spiral's trailing end and exhibiting poor performance in data identification. Our method demonstrates significant improvement by correctly identifying the artificially added anomalous data points and capturing the basic shape of the spiral. Finally, in the multicluster dataset, the IF successfully identifies a certain number of data clusters and distinguishes most of the anomalous data points. However, due to axial partitioning, "ghost" clusters appear between the two actual clusters, incorrectly classifying some anomaly points as normal data. EIF, utilizing random hyperplane partitioning, also does not perform well in recognizing the data distribution in this dataset. Our LIF still shows excellent and clear identification results, not only identifying all data clusters but also judging outliers almost without errors.

In summary, in scenarios with complex data distributions, IF fails to accurately detect anomalies due to its inherent limitations. EIF shows some improvement compared to IF, but it is not applicable in some distributions, as observed in the latter two datasets, where misclassification of "ghost" clusters still occurs and local anomaly detection is inaccurate. Our LIF characterizes the distribution of data more perfectly by splitting the sample space and accurately identifies local outliers.

Table 2
Real world benchmark datasets properties.

Dataset	Size	Dimension	Anomaly %
Http(KDD99)	567 479	3	0.4
Shuttle	49 094	9	7.15
Satellite	6435	36	31.6
Smtp	95 156	3	0.03
Forestcover	286 048	10	0.96
Mammography	11 183	6	2.32
Thyroid	3772	6	21
Arrhythmia	452	274	14.6
Annthyroid	7200	6	7.42
Breastw	683	9	35
Creditcard	284 807	30	0.17

$$TruePositiveRate = TruePositive/(TruePositive + FalseNegative) \tag{8}$$

$$FalsePositiveRate = FalsePositive/(FalsePositive + TrueNegative) \tag{9}$$

$$AUC = \int_0^1 TPR(FPR)\,\mathrm{d}(FPR) \tag{10}$$

4.2. Performance

To objectively validate the practical performance of the algorithms, we conducted experiments using eleven commonly encountered real-world datasets in the anomaly detection field. We compared seven different algorithms, including our proposed method. These algorithms are Local Outlier Factor (LOF) [19,21], the original Isolation Forest (IF), Extended Isolation Forest (EIF), an improved version of EIF called Generalized Isolation Forest (GIF), Local Sensitive Hashing Isolation Forest (LSHIF), which combines hashing methods, Deep Isolation Forest (DIF), which uses a neural network to randomly project the data space, and Layered Isolation Forest (LIF), which we proposed in this paper.

The comparison methods we chose in this paper are all based on random partitioning of the sample space, which solves or alleviates the problems existing in the original algorithm and improves the performance compared with the original algorithm. The experiment does not include methods that introduce complex mechanisms or solve other direction problems in the comparison scope. The LOF algorithm is chosen as the baseline comparison because it is one of the most common and widely used anomaly detection algorithms. The reason for comparison with the IF algorithm is obvious. We also added EIF and GIF as comparisons, which introduce the hyperplane partitioning mechanism. Moreover, we used LSHIF, which uses a local sensitive hashing forest. This method has improved the indicators on multiple datasets and introduced a novel mechanism of hashing. Finally, the deep isolation forest that uses an initialized deep network is selected. This algorithm is a relatively recent isolation-based method and has achieved the best results on multiple datasets. We believe that choosing these algorithms has a good comparative value and can better measure the effect of our algorithm.

For the parameter selection, set the parameter *n_neighbors* of LOF to 200, because when the parameter increases by an order of magnitude, the algorithm performance is better, but for large-scale datasets such as Http(KDD99) and Forestcover, the algorithm exceeds the memory space; EIF and GIF use the default parameters used in their authors' papers; LSHIF uses L2SH, which is the Euclidean distance as the measurement standard, and has the best comprehensive effect in its paper [32], while keeping other parameters unchanged; it should be pointed out that DIF is different from these methods. DIF uses 50

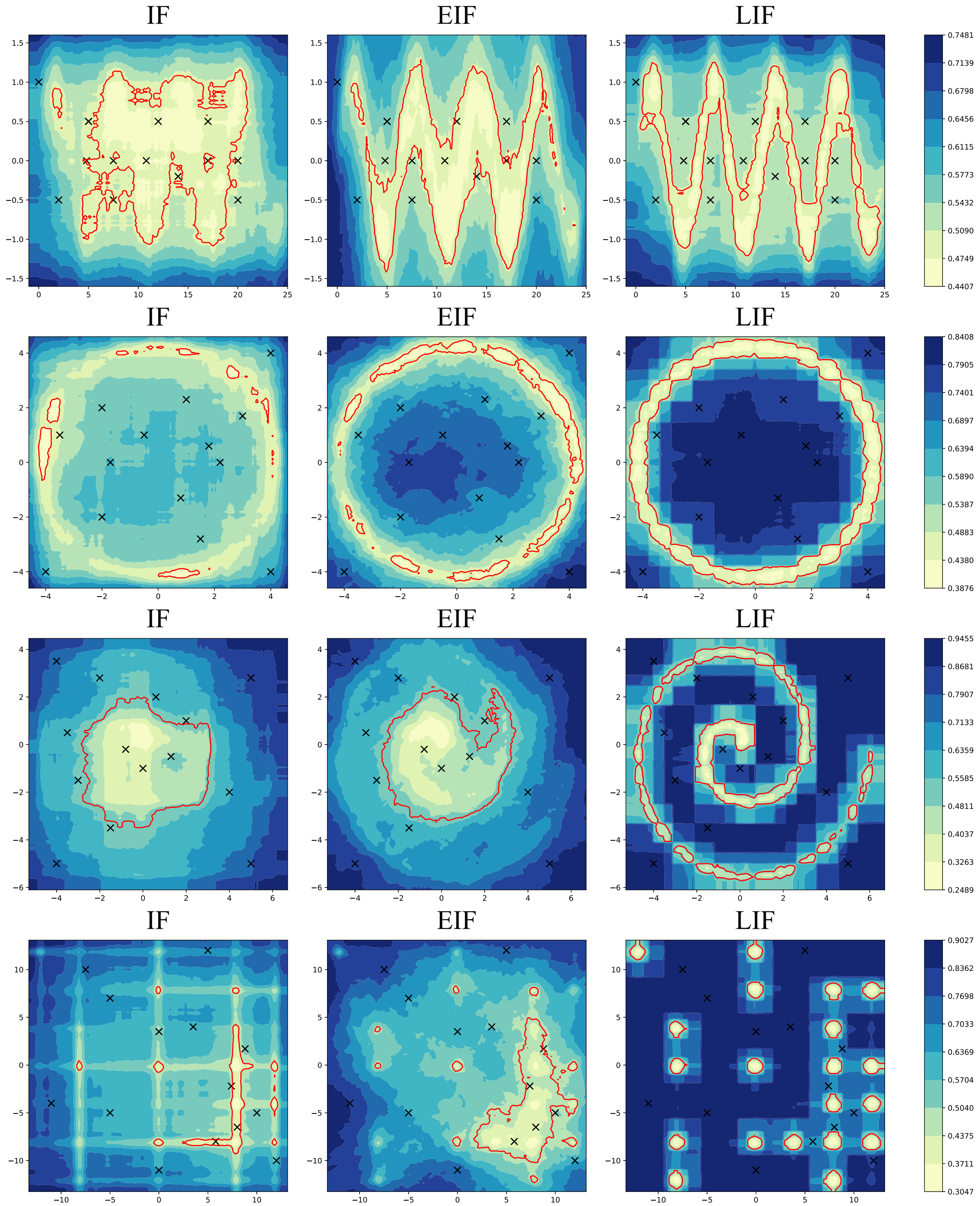


Fig. 8. This figure shows the anomaly score heatmaps generated by the evaluation of the four synthetic datasets in Fig. 1 using the isolation forest (IF), extended isolation forest (EIF), and layered isolation forest (LIF) algorithms. From left to right, they correspond to the IF, EIF, and LIF algorithms, respectively. The black crosses represent the artificial anomalies added previously to show the detection performance of the algorithms. In the heatmaps, lighter areas indicate a lower degree of anomaly, while darker areas indicate a higher degree of anomaly. A red line is drawn at an anomaly score threshold of 0.5. The enclosed region within the line, where the anomaly scores are below 0.5, is considered the normal region. Conversely, the region outside the line, where the anomaly scores exceed 0.5, is regarded as the anomalous region.

representations (r = 50) and 6 isolation trees per representation (t = 6) to process data by using a fully connected multilayer perceptron networks [37], so it actually builds more isolation trees (300). To reproduce the performance of the algorithm, we used these parameters. The parameters used for the seven algorithms in the experiment are shown in Table 5. The experiments were implemented using Python 3.10, and the datasets were obtained from publicly available datasets on ODDS or UCI Machine Learning Repository, similar to the above papers, which are described in Table 2. The LOF and IF algorithms were sourced from sklearn, while the EIF, GIF, LSHIF and DIF algorithms were obtained from the author's provided GitHub links or personal webpages. The experiments were conducted on a 2.3 GHz CPU. Based on these settings, we calculated the $ROC-AUC$ for each algorithm on each dataset. We also recorded the actual running time of the

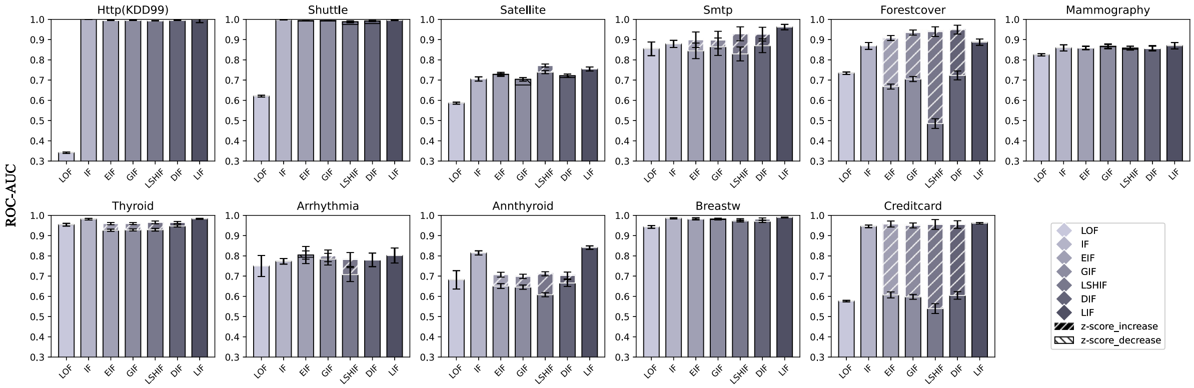


Fig. 9. The following are the ROC-AUC results of seven comparison algorithms on eleven datasets. The y-axis is the value of ROC-AUC, each color corresponds to one algorithm, the white slash area indicates the improvement after z score standardization; the black backslash area indicates the decrease after standardization, and the black error bars represent the 95% confidence intervals.

Table 3
The values of the area under the ROC curve for seven algorithms on eleven real-world datasets.

Datasets	ROC-AUC						
	LOF-200	iForest	Extended iForest	Generalized iForest	LSHIForest(L2SH)	Deep iForest	Layered iForest
Http(KDD99)	0.3412[10]	**0.9998 [1]**	0.9944[7](0.9967[5])	0.9949[6](0.9966[4])	0.9932[9](0.9934[8])	0.9949[6](0.9993[3])	0.9994[2]
Shuttle	0.6206[11]	**0.9973 [1]**	0.9934[4](0.9916[7])	0.9939[3](0.9919[6])	0.9880[8](0.9750[10])	0.9923[5](0.9793[9])	0.9952[2]
Satellite	0.5859[11]	0.7055[8]	0.7283[4](0.7277[5])	0.7037[9](0.6758[10])	0.7391[3](**0.7716 [1]**)	0.7214[6](0.7125[7])	0.7548[2]
Smtp	0.8544[9]	0.8787[6]	0.8444[10](0.8994[4])	0.8650[8](0.8980[5])	0.8294[11](0.9287[2])	0.8698[7](0.9268[3])	**0.9620 [1]**
Forestcover	0.7340[7]	0.8797[6]	0.6678[10](0.9077[4])	0.7052[9](0.9348[3])	0.4852[11](0.9394[2])	0.7231[8](**0.9492 [1]**)	0.8874[5]
Mammography	0.8248[11]	0.8592[5]	0.8578[8](0.8601[4])	0.8668[2](0.8652[3])	0.8579[7](0.8525[10])	0.8554[9](0.8591[6])	**0.8697 [1]**
Thyroid	0.9538[7]	0.9812[2]	0.9266[11](0.9593[6])	0.9281[10](0.9594[5])	0.9290[9](0.9653[3])	0.9479[8](0.9648[4])	**0.9831 [1]**
Arrhythmia	0.7499[10]	0.7728[9]	**0.8041 [1]** (0.7931[4])	0.7835[5](0.8002[3])	0.7073[11](0.7819[6])	0.7797[8](0.7805[7])	0.8016[2]
Annthyroid	0.6813[7]	0.8145[2]	0.6507[9](0.7069[4])	0.6454[10](0.6985[6])	0.6079[11](0.7115[3])	0.6659[8](0.7029[5])	**0.8408 [1]**
Breastw	0.9429[11]	0.9854[2]	0.9818[5](0.9841[3])	0.9819[4](0.9797[7])	0.9725[9](0.9785[8])	0.9707[10](0.9814[6])	**0.9901 [1]**
Creditcard	0.5766[10]	0.9457[6]	0.6071[7](0.9574[2])	0.5974[9](0.9504[5])	0.5394[11](0.9551[3])	0.6044[8](0.9547[4])	**0.9611 [1]**
Average	0.7021[11]	0.8927[5]	0.8233[8](0.8895[4])	0.8242[9](0.8864[6])	0.7863[10](0.8957[2])	0.8296[7](0.8919[3])	**0.9132 [1]**

algorithms. Each experiment, 60% of the data was randomly drawn as training data, 10% as validation set, and 30% as test data. That was repeated 12 times, excluding the highest and lowest results, and the remaining 10 results were averaged, taking the 95% confidence interval to obtain as fair and objective outcomes as possible.

4.2.1. ROC-AUC

The area under the receiver operating characteristic curve is a common metric for anomaly detection. Similar to many studies, we use this metric to measure the performance of the algorithm. The calculation method is shown in Eqs. (8)(9)(10).

Since the EIF, GIF, LSHIF, and DIF algorithms explicitly mentioned in the paper need to preprocess the datasets, including zero mean and unit variance, and LOF, IF and our method do not require this kind of processing, considering the fairness of the experiment, we conducted experiments without preprocessing for the four algorithms that need preprocessing, outside the parentheses; experiments after preprocessing, inside the parentheses; the numbers in [] after the performance indicators of the algorithm are their ranking. The experimental results shown in Table 3 show that overall, without preprocessing, LIF has a significant performance advantage over the other four IF-based improved algorithms, followed by IF.

This improved performance is due to the fact that our method also randomly selects and partitions features of different scales, inheriting the advantage of the original algorithm that it is not sensitive to feature scale differences. Fig. 9 shows the experimental results depicted in the graph. Fig. 9 represents the results without z score normalization of the data and comparison with the results after preprocessing the data with four algorithms that require preprocessing; other algorithms are left unprocessed. Among the four improved algorithms, EIF and GIF use hyperplanes and have higher requirements for feature standardization; LSHIF uses Euclidean distance as the evaluation indicator of the hash function and needs to preprocess the data; otherwise, the difference of different features will make it difficult for the algorithm to build local hash sensitive trees; DIF uses neural network random mapping of the data space, and data standardization is necessary. After preprocessing the data, we can see that the performance of the four improved methods of IF has improved significantly. LSHIF and DIF surpassed the original IF on multiple datasets. This feature is especially reflected in datasets $Smtp$, $Forestcover$, and $Creditcard$. This is because these three datasets have large scale differences in different dimensions.

Therefore, after standardization processing, scale-sensitive algorithms can achieve better results, such as the DIF algorithm after preprocessing; it achieved the best result on $Forestcover$. Nevertheless, our method can also adapt well to the complex distribution of different datasets by dividing the entire sample space; it achieved the best performance on multiple datasets, ranking first on 6 datasets, ranking second on 4 datasets, and ranking first on average performance.

4.2.2. Execution time

Table 4 shows the running time of these algorithms. Comparing the average running time of the algorithms, we observed that the isolation forest (IF) algorithm had the shortest execution time, followed by our proposed LIF, which had slightly increased computation time compared to IF. This is because we built more iForests than the original isolation forest algorithm, including local iForests. However, by utilizing the characteristics of the datasets in which there are few and different anomalous data points and combining the restriction on the degree of partitioning of the sample space, the actual number of iForests built was kept at a small number, which made the execution time of our algorithm still within an acceptable range. Consistent with our analysis results, the running time of the algorithm was weakly correlated with the dimension and scale of the data. Next was LOF, where the $n_neighbors$ parameter was set to 200, as our baseline reference method

Table 4
The values of the times for seven algorithms on eleven real-world datasets.

Datasets	Time (s)						
	LOF-200	iForest	Extended iForest	Generalized iForest	LSHIForest(L2SH)	Deep iForest	Layered iForest
Http(KDD99)	32.75 ± 1.24	8.52 ± 0.09	198.19 ± 4.12	288.09 ± 7.27	437.56 ± 21.96	1371.78 ± 31.61	10.65 ± 0.25
Shuttle	9.45 ± 0.14	0.73 ± 0.02	20.94 ± 0.15	30.42 ± 0.36	33.3 ± 2.28	82.78 ± 6.26	2.62 ± 0.14
Satellite	0.67 ± 0.02	0.21 ± 0.01	3.22 ± 0.05	4.31 ± 0.05	5.3 ± 0.79	12.66 ± 1.17	3.9 ± 0.04
Smtp	4.62 ± 0.19	1.48 ± 0.04	32.4 ± 0.14	49.47 ± 0.21	107.03 ± 7.83	212.9 ± 3.23	4.35 ± 0.06
Forestcover	31.79 ± 0.71	5.39 ± 0.07	126.49 ± 0.75	173.93 ± 1.93	127.26 ± 4.87	689.55 ± 26.55	9.21 ± 0.17
Mammography	0.62 ± 0.02	0.23 ± 0.01	4.07 ± 0.05	5.51 ± 0.04	10.7 ± 0.53	18.93 ± 2.63	4.34 ± 0.09
Thyroid	0.17 ± 0.01	0.15 ± 0.01	1.55 ± 0.04	2.14 ± 0.04	6.45 ± 0.9	7.89 ± 0.72	0.72 ± 0.04
Arrhythmia	0.03 ± 0.01	0.2 ± 0.01	1.34 ± 0.11	3.88 ± 0.35	1.16 ± 0.23	1.27 ± 0.11	0.9 ± 0.06
Annthyroid	0.31 ± 0	0.25 ± 0.09	2.56 ± 0.11	3.67 ± 0.1	6.14 ± 0.55	12.84 ± 0.63	1.67 ± 0.05
Breastw	0.02 ± 0	0.19 ± 0.01	0.61 ± 0.06	0.56 ± 0.06	2.32 ± 0.28	1.55 ± 0.13	0.81 ± 0.12
Creditcard	51.33 ± 1.32	10.21 ± 0.21	96.53 ± 1.33	138.23 ± 0.34	104.81 ± 6.56	762.13 ± 16.57	19.66 ± 2.21
Average	11.98 ± 0.33	2.51 ± 0.05	44.26 ± 0.63	63.66 ± 0.98	76.55 ± 4.25	288.57 ± 8.15	5.35 ± 0.29

Table 5
Parameters of algorithms in experiments.

Datasets	Parameters						
	LOF-200	iForest	Extended iForest	Generalized iForest	LSHIForest	Deep iForest	Layered iForest

	n_neighbors	*itrees_nums*
		sample_nums
			itrees_nums
			sample_nums
			extend_level
				itrees_nums
				sample_nums
				extend_level
					subsampling_range
					LSH_family
						representations itrees_nums/rep network	*split_nums(L)*
							layers_weight(W)

Http(KDD99)	200	100 256	100 256 dimensions-1	100 256 dimensions-1	[64,1024] L2SH	50 6 Fully connected multi-layer-perceptron networks	[1,2] [0.8,0.2]
Shuttle	200	100 256	100 256 dimensions-1	100 256 dimensions-1	[64,1024] L2SH	50 6 Fully connected multi-layer-perceptron networks	[1,3] [0.9,0.1]
Satellite	200	100 256	100 256 dimensions-1	100 256 dimensions-1	[64,1024] L2SH	50 6 Fully connected multi-layer-perceptron networks	[1,3] [0.9,0.1]
Smtp	200	100 256	100 256 dimensions-1	100 256 dimensions-1	[64,1024] L2SH	50 6 Fully connected multi-layer-perceptron networks	[1,3,7] [0.5,0.3,0.2]
Forestcover	200	100 256	100 256 dimensions-1	100 256 dimensions-1	[64,1024] L2SH	50 6 Fully connected multi-layer-perceptron networks	[1,2] [0.8,0.2]
Mammography	200	100 256	100 256 dimensions-1	100 256 dimensions-1	[64,1024] L2SH	50 6 Fully connected multi-layer-perceptron networks	[1,3] [0.9,0.1]
Thyroid	200	100 256	100 256 dimensions-1	100 256 dimensions-1	[64,1024] L2SH	50 6 Fully connected multi-layer-perceptron networks	[1,2] [0.4,0.6]
Arrhythmia	200	100 256	100 256 dimensions-1	100 256 dimensions-1	[64,1024] L2SH	50 6 Fully connected multi-layer-perceptron networks	[1,3] [0.8,0.2]
Annthyroid	200	100 256	100 256 dimensions-1	100 256 dimensions-1	[64,1024] L2SH	50 6 Fully connected multi-layer-perceptron networks	[1,3] [0.3,0.7]
Breastw	200	100 256	100 256 dimensions-1	100 256 dimensions-1	[64,1024] L2SH	50 6 Fully connected multi-layer-perceptron networks	[1,2] [0.9,0.1]
Creditcard	200	100 256	100 256 dimensions-1	100 256 dimensions-1	[64,1024] L2SH	50 6 Fully connected multi-layer-perceptron networks	[1,2] [0.9,0.1]

because it is the most common anomaly detection algorithm. Next were other improved algorithms based on IF. Although EIF and GIF showed good time efficiency on smaller datasets, they spent a considerable amount of computation time in the evaluation stage due to the need to calculate the position of data points relative to the splitting hyperplane. Especially when dealing with larger datasets, such as $Http(KDD99)$ and $Forestcover$, they spent a long time evaluating the degree of anomaly. LSHIF took more time than the previous two methods because it needed

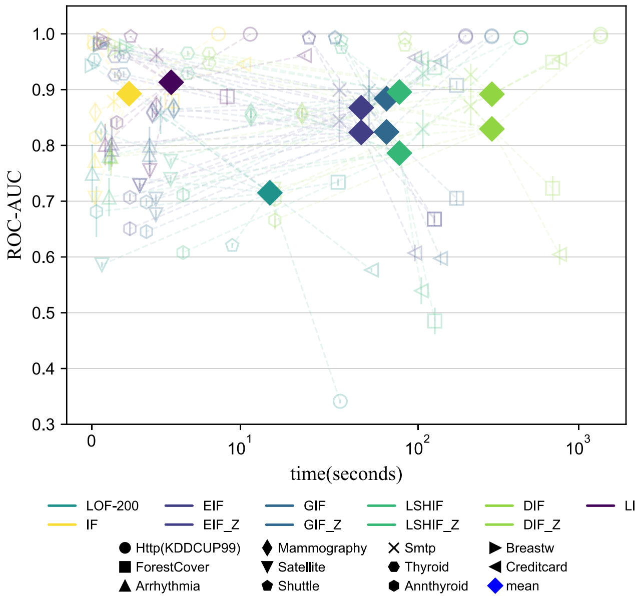


Fig. 10. The x-axis of the figure represents the algorithm's running time, while the y-axis represents the ROC-AUC values of the algorithms. Each shape in the figure corresponds to a specific experimental dataset, and each color represents a different algorithm used in the experiment. The algorithms before and after z score standardization are represented by the same color. The symbol "◆" denotes the average performance of an algorithm across all datasets, and a dashed line connects the average value to the corresponding datasets.

to calculate data points based on the Euclidean distance hash function and used a local sensitive hash tree to evaluate the anomaly degree. DIF spent a lot of time on both the mapping operation itself and evaluating the data points through the randomly mapped space afterward. It can be seen that it had the longest execution time on 9 datasets, except for *Arrhythmia* and *Breastw*, which are relatively small in scale.

We use Fig. 10 to visually show the relationship between algorithm performance and running time. Taking LOF as the baseline, we can see from the figure that the IF algorithm has a short running time and shows good performance on many datasets. EIF and GIF have better detection effects than the original IF on some datasets but fail to show good results on some other datasets and have long running times on large-scale datasets. LSHIF and DIF have better effects than the original IF after data preprocessing and have the best performance on several datasets but have long running times. Although the paper claims that they have linear complexity, they spend much time in the data evaluation stage due to the sophisticated complex mechanism, which is also the reason why they have long running times on large-scale datasets. Our method not only has the best comprehensive performance but also maintains a low running time in the data evaluation stage because it does not introduce complex mechanisms, and the running time is slightly higher than the original algorithm.

5. Conclusion

In this paper, we propose an improved method for anomaly detection based on the isolation forest algorithm. The proposed method divides the sample space into multiple layers and builds local isolation forests to adapt to the different distributions of the dataset. By integrating the anomaly scores of the data in different ranges of the sample space with different weights, our algorithm evaluates the anomaly degree of the data. Our experimental results show that the method improves the performance of the isolation forest algorithm and outperforms other algorithms in comparative experiments while effectively alleviating some of the drawbacks of the original isolation forest algorithm. In terms of algorithm consumption, the method increases the consumption of building local isolation forests and introduces some new parameters, but still maintains a linear time complexity. The size and dimension of the data have little impact on it, which is also verified by the experimental results. The idea of combining global and local space is more widely used in the tasks of deep learning, such as computer vision, and we can also achieve good results by applying these ideas to traditional machine learning algorithms. There are still some shortcomings in this method. In future works, we will focus on issues such as the hyperparameters of the algorithm and the distribution characteristics of the dataset, and some recent studies [40,41] have advanced this development. Meanwhile, in this work we only use the simplest weights to combine the anomaly degrees of data in different ranges. Introducing additional methods such as kernel functions may lead to better experimental results. Considering the mechanism of the isolation algorithm, research on the interpretability of the algorithm [42] can also help improve the performance of the algorithm and even discover new reliable methods.

CRediT authorship contribution statement

Tao Liu: Conceptualization, Data curation, Formal analysis, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Zhen Zhou:** Funding acquisition, Investigation, Project administration, Resources, Supervision, Writing – review & editing. **Lijun Yang:** Funding acquisition, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

Funding

This work is supported by Sichuan Science and Technology Program, China (Grant No. 2020JDRC0040); the Major Research and Development Projects of Sichuan Province (No. 2023YFSY0049)

References

[1] A. Emmott, S. Das, T. Dietterich, A. Fern, W.-K. Wong, A meta-analysis of the anomaly detection problem, 2015, arXiv preprint arXiv:1503.01158.

[2] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Comput. Surv. (CSUR) 41 (3) (2009) 1–58.

[3] H. John, S. Naaz, Credit card fraud detection using local outlier factor and isolation forest, Int. J. Comput. Sci. Eng. 7 (4) (2019) 1060–1064.

[4] L. Meneghetti, M. Terzi, S. Del Favero, G.A. Susto, C. Cobelli, Data-driven anomaly recognition for unsupervised model-free fault detection in artificial pancreas, IEEE Trans. Control Syst. Technol. 28 (1) (2018) 33–47.

[5] E. De la Hoz, E. De La Hoz, A. Ortiz, J. Ortega, A. Martínez-Álvarez, Feature selection by multi-objective optimisation: Application to network anomaly detection by hierarchical self-organising maps, Knowl.-Based Syst. 71 (2014) 322–338.

[6] M. Zhang, B. Xu, J. Gong, An anomaly detection model based on one-class svm to detect network intrusions, in: 2015 11th International Conference on Mobile Ad-Hoc and Sensor Networks, MSN, IEEE, 2015, pp. 102–107.

[7] K.L. Malanchev, A.A. Volnova, M.V. Kornilov, M.V. Pruzhinskaya, E.E. Ishida, F. Mondon, V.S. Korolev, Use of machine learning for anomaly detection problem in large astronomical databases, in: DAMDID/RCDL, 2019, pp. 205–216.

[8] M. Das, S. Parthasarathy, Anomaly detection and spatio-temporal analysis of global climate system, in: Proceedings of the Third International Workshop on Knowledge Discovery from Sensor Data, 2009, pp. 142–150.

[9] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation forest, in: 2008 Eighth Ieee International Conference on Data Mining, IEEE, 2008, pp. 413–422.

[10] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation-based anomaly detection, ACM Trans. Knowl. Discov. Data (TKDD) 6 (1) (2012) 1–39.

[11] F.P. Brooks, N.S. Bullet, Essence and accidents of software engineering, IEEE Comput. 20 (4) (1987) 10–19.

[12] T. Barbariol, F.D. Chiara, D. Marcato, G.A. Susto, A review of tree-based approaches for anomaly detection, in: Control Charts and Machine Learning for Anomaly Detection in Manufacturing, Springer, 2022, pp. 149–185.

[13] S. Hariri, M.C. Kind, R.J. Brunner, Extended isolation forest, IEEE Trans. Knowl. Data Eng. 33 (4) (2019) 1479–1489.

[14] D.M. Hawkins, Identification of Outliers, vol. 11, Springer, 1980.

[15] A. Fernandez, J. Bella, J.R. Dorronsoro, Supervised outlier detection for classification and regression, Neurocomputing 486 (2022) 77–92, http://dx.doi.org/10.1016/j.neucom.2022.02.047.

[16] F. Angiulli, C. Pizzuti, Fast outlier detection in high dimensional spaces, in: European Conference on Principles of Data Mining and Knowledge Discovery, Springer, 2002, pp. 15–27.

[17] F. Angiulli, F. Fassetti, Dolphin: An efficient algorithm for mining distance-based outliers in very large datasets, ACM Trans. Knowl. Discov. Data (TKDD) 3 (1) (2009) 1–57.

[18] L.E. Peterson, K-nearest neighbor, Scholarpedia 4 (2) (2009) 1883.

[19] M.M. Breunig, H.-P. Kriegel, R.T. Ng, J. Sander, LOF: identifying density-based local outliers, in: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, 2000, pp. 93–104.

[20] E.W. Dereszynski, T.G. Dietterich, Spatiotemporal models for data-anomaly detection in dynamic environmental monitoring campaigns, ACM Trans. Sensor Netw. 8 (1) (2011) 1–36.

[21] O. Alghushairy, R. Alsini, T. Soule, X. Ma, A review of local outlier factor algorithms for outlier detection in big data streams, Big Data Cogn. Comput. 5 (1) (2020) 1.

[22] R.J. Campello, D. Moulavi, A. Zimek, J. Sander, Hierarchical density estimates for data clustering, visualization, and outlier detection, ACM Trans. Knowl. Discov. Data (TKDD) 10 (1) (2015) 1–51.

[23] M. Celik, F. Dadaser-Celik, A.S. Dokuz, Anomaly detection in temperature data using DBSCAN algorithm, in: 2011 International Symposium on Innovations in Intelligent Systems and Applications, IEEE, 2011, pp. 91–95.

[24] A. Aldweesh, A. Derhab, A.Z. Emam, Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues, Knowl.-Based Syst. 189 (2020) 105124.

[25] G. Pang, C. Shen, L. Cao, A.V.D. Hengel, Deep learning for anomaly detection: A review, ACM Comput. Surv. (CSUR) 54 (2) (2021) 1–38.

[26] C. Zhou, R.C. Paffenroth, Anomaly detection with robust deep autoencoders, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, Halifax NS Canada, 2017, pp. 665–674, http://dx.doi.org/10.1145/3097983.3098052.

[27] F.T. Liu, K.M. Ting, Z.-H. Zhou, On detecting clustered anomalies using SCiForest, in: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part II 21, Springer, 2010, pp. 274–290.

[28] Y. Shen, H. Liu, Y. Wang, Z. Chen, G. Sun, A novel isolation-based outlier detection method, in: PRICAI 2016: Trends in Artificial Intelligence: 14th Pacific Rim International Conference on Artificial Intelligence, Phuket, Thailand, August 22-26, 2016, Proceedings 14, Springer, 2016, pp. 446–456.

[29] K.M. Ting, G.-T. Zhou, F.T. Liu, J.S.C. Tan, Mass estimation and its applications, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010, pp. 989–998.

[30] K.M. Ting, G.-T. Zhou, F.T. Liu, S.C. Tan, Mass estimation, Mach. Learn. 90 (2013) 127–160.

[31] G. Chen, Y.L. Cai, 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, 2011, pp. 21–26, http://dx.doi.org/10.1109/CYBER.2011.6011757.

[32] X. Zhang, W. Dou, Q. He, R. Zhou, C. Leckie, R. Kotagiri, Z. Salcic, LSHiForest: A generic framework for fast tree isolation based ensemble anomaly analysis, in: 2017 IEEE 33rd International Conference on Data Engineering, ICDE, 2017, pp. 983–994, http://dx.doi.org/10.1109/ICDE.2017.145, ISSN: 2375-026X.

[33] T.R. Bandaragoda, K.M. Ting, D. Albrecht, F.T. Liu, Y. Zhu, J.R. Wells, Isolation-based anomaly detection using nearest-neighbor ensembles, Comput. Intell. 34 (4) (2018) 968–998.

[34] P. Karczmarek, A. Kiersztyn, W. Pedrycz, N-ary isolation forest: An experimental comparative analysis, in: Artificial Intelligence and Soft Computing: 19th International Conference, ICAISC 2020, Zakopane, Poland, October 12-14, 2020, Proceedings, Part II 19, Springer, 2020, pp. 188–198.

[35] J. Lesouple, C. Baudoin, M. Spigai, J.-Y. Tourneret, Generalized isolation forest for anomaly detection, Pattern Recognit. Lett. 149 (2021) 109–119.

[36] H. Ma, B. Ghojogh, M.N. Samad, D. Zheng, M. Crowley, Isolation mondrian forest for batch and online anomaly detection, in: 2020 IEEE International Conference on Systems, Man, and Cybernetics, SMC, 2020, pp. 3051–3058, http://dx.doi.org/10.1109/SMC42975.2020.9283073.

[37] H. Xu, G. Pang, Y. Wang, Y. Wang, Deep isolation forest for anomaly detection, IEEE Trans. Knowl. Data Eng. (2023) Publisher: IEEE.

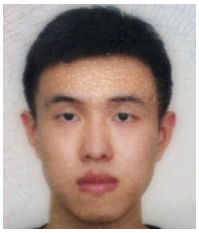
[38] S. He, F. Chen, B. Jiang, Physical intrusion monitoring via local-global network and deep isolation forest based on heterogeneous signals, Neurocomputing 441 (2021) 25–35, http://dx.doi.org/10.1016/j.neucom.2021.01.104.

[39] Y.-L. Zhang, L. Li, J. Zhou, X. Li, Z.-H. Zhou, Anomaly detection with partially observed anomalies, in: Companion Proceedings of the The Web Conference 2018, 2018, pp. 639–646.

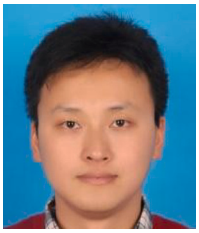
[40] W. Hu, J. Gao, B. Li, O. Wu, J. Du, S. Maybank, Anomaly detection using local kernel density estimation and context-based regression, IEEE Trans. Knowl. Data Eng. 32 (2) (2020) 218–233, http://dx.doi.org/10.1109/TKDE.2018.2882404, Conference Name: IEEE Transactions on Knowledge and Data Engineering.

[41] K.M. Ting, B.-C. Xu, T. Washio, Z.-H. Zhou, Isolation distributional kernel: A new tool for point and group anomaly detections, IEEE Trans. Knowl. Data Eng. 35 (3) (2023) 2697–2710, http://dx.doi.org/10.1109/TKDE.2021.3120277, Conference Name: IEEE Transactions on Knowledge and Data Engineering.

[42] M. Carletti, M. Terzi, G.A. Susto, Interpretable anomaly detection with DIFFI: Depth-based feature importance of isolation forest, Eng. Appl. Artif. Intell. 119 (2023) 105730, http://dx.doi.org/10.1016/j.engappai.2022.105730.



Tao Liu received the B.E. degree from Changzhou University, Changzhou, China. He is currently pursuing the master's degree in the School of Computer Science and Engineering at Southwest Minzu University. His research interests include machine learning, data mining and anomaly detection.



Zhen Zhou received the M.S. and Ph.D. degrees in Computer Science and Technology from Chongqing University, China. He is currently a lecturer at the Department of Computer Science and Engineering, Southwest Minzu University. His research interests include cloud computing, anomaly detection and machine learning.

Lijun Yang received the Ph.D. degree from Chongqing University, Chongqing, China, in 2017. He is currently a Lecturer with the School of Computer Science and Technology, Southwest Minzu University, Chengdu, China. His current research interests include classification, prototype reduction, and clustering analysis.