Reconstruction-based Anomaly Detection with Completely Random Forest

Yi-Xuan Xu* Ming Pang*† Ji Feng*‡§ Kai Ming Ting* Yuan Jiang* Zhi-Hua Zhou*

Abstract

Reconstruction-based anomaly detectors have drawn much attention recently. Existing methods rely almost universally on the neural network autoencoder and its variants. Their performance is limited by the facts that the neural network autoencoder requires a large training set in order to achieve high accuracy and has high computational cost. In addition, its performance depends heavily on tuning a large number of hyper-parameters. Our work is motivated by recent studies showing that a forest model is also capable of capturing information about the dataset, comparable to that of a neural network autoencoder. We propose a novel reconstruction-based anomaly detector solely based on a completely random forest. The proposed method, RecForest, has three advantages over existing methods. First, the forest model has much higher training efficiency and significantly fewer hyper-parameters, addressing the two above-mentioned issues of neural network autoencoders. Second, RecForest has two new capabilities compared with existing forest-based anomaly detectors, i.e., RecForest can mine outlying attributes and handle irrelevant attributes in high-dimensional datasets. Third, in terms of mining outlying attributes, RecForest runs orders of magnitude faster than state-of-the-art outlying aspect miners on large datasets. We verify the effectiveness and efficiency of the proposed method through extensive experiments.

1 Introduction

Anomaly detection is a learning problem that aims to identify anomalous samples in a dataset. It has been successfully applied in many domains such as financial fraud detection and network intrusion detection [10].

Reconstruction-based anomaly detectors have attracted much attention in the research community recently. In order to compute an anomaly score for each sample, a reconstructed sample will first be generated by reconstruction-based anomaly detectors. It is well established that anomalies are much harder to be accurately reconstructed than normal samples, leading to large reconstruction error [3]. Apart from its verified effectiveness on anomaly detection, reconstructed samples can also be used as the augmented data for performance enhancement on anomaly detection [4], which cannot be achieved by most anomaly detectors that only produce an anomaly score for each sample.

Existing reconstruction-based anomaly detectors almost universally adopt the neural network model of autoencoder. The neural network autoencoder excels at capturing complex structures of normal data; and its ability to learn representations for anomaly detection has increased its appeal. However, its practical use in anomaly detection is restricted by two deficiencies: (1) The neural network autoencoder is with high model complexity and demands a large training set in order to perform well [3]. Therefore, its effectiveness remains in doubt on a large number of datasets where the data sizes are insufficient for it to learn well from the normal data; (2) The performance of a neural network autoencoder depends heavily on tuning a large number of hyper-parameters. Since priori labels on anomalies are not available, it is practically hard to fine tune its hyper-parameters to achieve good performance. As a result of these deficiencies, the application of reconstruction-based anomaly detectors is also inhibited.

Recent studies show that a forest model is also capable of capturing information about the dataset, comparable to that of a neural network autoencoder [12, 19, 20]. Exploiting such information enables the forest model to be applied to various tasks, even those initially believed to be the specialty of neural networks, such as feature induction [16], learning distributed representations [7], and data encoding [6]. Motivated by these advancements, we believe that a forest model can also be used as a reconstruction-based anomaly detector. As a result, the key advantages of forest are retained, e.g., having small training cost and very few hyper-parameters. Another advantage is that the model complexity of forest is proportional to the data size, making it more robust against over-fitting [18].

Although forest-based anomaly detectors are widely used in the literature, there is a dearth of works that attempt to exploit information in the forest for anomaly detection. Existing forest-based methods usually rely on a single statistic derived from the forest for anomaly

*National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China. {xuyx, pangm, fengj, tingkm, jiangy, zhouzh}@lamda.nju.edu.cn

†JD.com, Beijing, China

‡Baiont Technology, Nanjing, China.

§Sinovation Ventures AI Institute, Beijing, China

detection, e.g., tree path length [11], collusive displacement score [9], and partial identification score [8]. The use of a single statistic is insufficient to distinguish anomalies from normal data in many cases [21], and it also does not allow us to identify outlying aspects of anomalies for better interpretability, called outlying aspects mining in the literature [15].

This paper proposes a novel reconstruction-based anomaly detector called RecForest (abbreviation for Reconstruction-based Forest). Similar to using the neural network autoencoder, anomaly detection is achieved by identifying a poorly reconstructed sample for each anomaly. But it has a totally different scheme of sample reconstruction, which fully exploits information in the forest model to produce a bounding box that contains the reconstructed result for each test sample.

Our contributions are:

- Proposing a novel reconstruction-based anomaly detector based on a completely random forest. It has much smaller training cost, and significantly fewer hyper-parameters than the neural network autoencoder, while achieving consistently better detection accuracy across benchmark datasets.
- Revealing that RecForest can also be used to mine outlying aspects for a given anomaly. Unlike existing outlying aspect miners, it does not require a systematic search. As a result, it runs orders of magnitude faster than existing miners with comparable mining outcomes.
- Providing an insight on how a forest model can be used for anomaly detection: the bounding boxes in trees for a test sample can be exploited to improve the accuracy of forest-based anomaly detectors; and the resulting model is also robust against irrelevant attributes in high-dimensional datasets.

The rest of the paper is organized as follows: The related work is first discussed in Section 2. Section 3 introduces the proposed method based on a completely random forest, and how it is used for anomaly detection and outlying aspects mining. The experimental settings and the empirical evaluations are presented in the next two sections. We conclude in the last section.

2 Related Work

In this section, we briefly introduce subareas related to the RecForest: reconstruction-based/forest-based anomaly detectors, and outlying aspects mining.

Reconstruction-based anomaly detectors assume that anomalies are much harder to be accurately reconstructed than normal samples. Under the framework of deep learning, recent work mainly explores the neural network autoencoder for reconstruction-based anomaly detection. The basic idea is to find a reduced space that better clusters normal data. As a result, anomalies are much harder to be reconstructed when projecting them from this reduced space back to the input space [1]. The variational autoencoder extends this idea and further finds a distribution that better fits normal data [5]. To mitigate the problem that the performance of neural network autoencoder is unstable, Chen et al. [3] combine many neural network autoencoders via ensemble learning. However, the computational cost also increases drastically. To overcome the information loss caused by the projection, Zong et al. [21] propose to concatenate the neural network autoencoder with gaussian mixture model. These methods all strive to improve the neural network autoencoder, and their effectiveness is still restricted by the above-mentioned deficiencies. Despite the fact that our work also follows the one-class setting with a clean training set, it explores that whether reconstruction-based anomaly detectors can be built on other models instead of neural network autoencoders.

Another related line of work to ours is forest-based anomaly detectors. Most related to our work is isolation forest (iForest), which also uses a completely random forest and computes the average tree path length as the anomaly score [11]. PIDForest is another forest-based anomaly detector based on a novel anomaly score called partial identification score [8]. Instead of randomly choosing attributes for splitting as in iForest, PIDForest adopts a new splitting criterion to find splits for internal nodes heuristically. Our work is different in that we focus on how to exploit information in the forest instead of designing a better forest model. The underlying model of RecForest is arguably in the simplest form of forest models: completely random forest.

Interpretability on anomaly detection has received growing interests recently. Given a dataset of normal samples and a query anomaly, an outlying aspects miner aims to identify the attributes that make this anomaly outlying. Vinh et al. [17] combine Beam search with anomaly detectors to examine subspaces in the input space and find the most outlying one. Treating the separation of the anomaly from normal data as a binary classification problem, Micenková et al. [13] use feature selection algorithms to identify outlying attributes. In contrast, we aim to use the same anomaly detector to perform anomaly detection as well as outlying aspects mining. We show that the proposed method is more efficient than existing outlying aspects miners.

3 Reconstruction-based Forest: RecForest

The proposed method is described in the next three subsections. Section 3.1 introduces the idea of bounding box and the model used to implement the idea, i.e., completely random forest. Section 3.2 presents the proposed method based on bounding boxes derived from a forest. We show that the proposed method can be used to perform outlying aspects mining in Section 3.3.

3.1 Bounding Boxes and Completely Random Forest. The bounding box for a sample $\mathbf{x}$ is a rectangular box in the input space that bounds the neighboring region of the target $\mathbf{x}$. Given the bounding box, the reconstructed sample for $\mathbf{x}$ is defined as the center point of the bounding box. RecForest aims to produce a bounding box for each sample to generate the reconstructed sample, and the idea of RecForest works only if the bounding box is small enough to capture the local neighborhood relevant to the sample $\mathbf{x}$.

To efficiently produce such a bounding box, we propose to use the completely random tree, where the split at every internal node is generated randomly (for both the splitting attribute and the cut-off). The tree generation process is constrained by the training data that every split must produce two non-empty subsets.

Given a tree, the bounding box for $\mathbf{x}$ is defined as the region of the leaf node to which $\mathbf{x}$ traverses from the root node of the tree. Since the completely random tree adopts axis-aligned splitting at internal nodes, the region of the leaf node corresponds to a rectangular box in the input space, and it can be identified through retrieving the decision path to the leaf node. To further reduce the bounding box, multiple trees are employed (i.e., a completely random forest). The bounding box of a forest for $\mathbf{x}$ is defined as the intersection of bounding boxes from all completely random trees.

The tree generation process begins by computing a bounding box $\mathbf{B}_{\mathcal{D}}$ at the root node of the tree, which covers the input range of the training set $\mathcal{D}$. As the tree grows, each subsequent bounding box defining the region of an internal or leaf node is getting smaller, as the depth of the tree increases.

The key symbols used are given in Table 1.

Table 1: Key symbols used.

Symbol	Definition
$\mathcal{D}$	A dataset with n samples
$\mathbf{x}$	A sample $[x_1, \ldots, x_d]^T$ in the input space $\mathbb{R}^d$
$\mathbf{x}^{\text{rec}}$	A reconstructed sample of $\mathbf{x}$
$\mathbf{B}$	A bounding box in the input space $\mathbb{R}^d$
$\mathbf{L}^{\mathbf{B}}, \mathbf{U}^{\mathbf{B}}$	The lower and upper bound of a bounding box $\mathbf{B}$
$\mathbf{T}(\mathbf{x})$	The bounding box of sample $\mathbf{x}$ from a tree $\mathcal{T}$
$\mathbf{F}(\mathbf{x})$	The bounding box of sample $\mathbf{x}$ from a forest $\mathcal{F}$

We provide the formal definitions here and in the next two subsections.

Definition 1. A bounding box $\mathbf{B} \subset \mathbb{R}^d$ is defined to be a rectangular box in the input space, which has a lower bound $\mathbf{L}$ and an upper bound $\mathbf{U}$. Concretely, for each dimension i, we have:

$$\mathbf{B} = \{\mathbf{x} \in \mathbb{R}^d \mid L_i \leq x_i \leq U_i,\ \forall i = 1, \ldots, d\}. \tag{3.1}$$

Definition 2. $\mathbf{B}_{\mathcal{D}}$ is the smallest bounding box that covers the dataset $\mathcal{D}$, which has the following lower and upper bounds for each dimension $i = 1, \ldots, d$:

$$\begin{aligned} L_i^{\mathbf{B}_{\mathcal{D}}} &= \min\{x_i \mid \forall \mathbf{x} \in \mathcal{D}\}, \\ U_i^{\mathbf{B}_{\mathcal{D}}} &= \max\{x_i \mid \forall \mathbf{x} \in \mathcal{D}\}. \end{aligned} \tag{3.2}$$

Definition 3. A completely random tree $\mathcal{T}$ is generated from $\mathcal{D}$ by recursively using a randomly selected attribute i and cut-off c_i with split $x_i > c_i$ to subdivide the dataset into two non-empty subsets until the maximum height h is reached or the data run out.

In summary, in the training stage, a forest with m completely random trees that have maximum height h is produced from a training set $\mathcal{D}$ of normal samples.

3.2 Anomaly Detection with RecForest. This section describes how we use the reconstructed sample defined in RecForest for anomaly detection. To identify anomalies, the key ideas used in RecForest are: (1) Each bounding box in the forest contains at least a normal sample from the training set. (2) The reconstructed sample is always within the bounding boxes from the forest and $\mathbf{B}_{\mathcal{D}}$. (3) An anomaly is likely to be outside the bound. Therefore, the center point of the bounding box will be a poorly reconstructed sample for each anomaly, leading to a large reconstruction error.

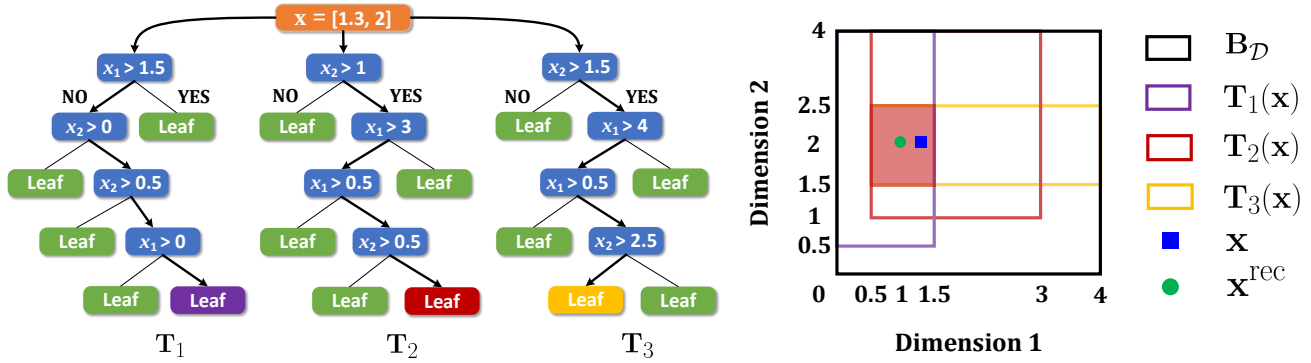
Here we provide formal definitions on the bounding box for a given sample from a tree and from a forest; the reconstructed sample; and the reconstruction error. Notice that this process is the evaluating stage for a test sample using a forest produced from the training stage.

Given a sample $\mathbf{x}$, let $A(\mathbf{x})$ be the set of all attributes and their cut-offs used in the internal nodes of a tree $\mathcal{T}$ along $\mathbf{x}$'s traversal to a leaf node; and $\overline{A}(\mathbf{x})$ be the set of attributes excluding those in $A(\mathbf{x})$.

Definition 4. Given a completely random tree $\mathcal{T}$ produced from $\mathcal{D}$, the bounding box $\mathbf{T}(\mathbf{x})$ is determined in combination with the bounding box $\mathbf{B}_{\mathcal{D}}$ as follows:

- For each attribute i in $A(\mathbf{x})$, let the cut-offs c_i of p internal nodes that use attribute i for splitting be ordered with respect to x_i as follows:

$$L_i^{\mathbf{B}_{\mathcal{D}}} < c_i^1 < \cdots, c_i^j \leq x_i < c_i^{j+1}, \cdots < c_i^p < U_i^{\mathbf{B}_{\mathcal{D}}}.$$



(a) Examples on completely random trees **(b) Demonstration on sample reconstruction**

Figure 1: Demonstration of sample reconstruction using a completely random forest. (a) The structure of completely random trees in the forest. Decision paths of a given test sample $\mathbf{x}$ are bolded. (b) Bounding boxes from trees $\{\mathbf{T}_1(\mathbf{x}), \mathbf{T}_2(\mathbf{x}), \mathbf{T}_3(\mathbf{x})\}$, and from the training data $\mathbf{B}_\mathcal{D}$. The intersection area is colored red.

Then, the lower and upper bound of $\mathbf{T}(\mathbf{x})$ on attribute i are defined as:

$$L_i^{\mathbf{T}(\mathbf{x})} = c_i^j \leq x_i < c_i^{j+1} = U_i^{\mathbf{T}(\mathbf{x})}. \tag{3.3}$$

- For each attribute i in $\overline{A}(\mathbf{x})$, the i-th lower and upper bound of $\mathbf{T}(\mathbf{x})$ are defined using $\mathbf{B}_\mathcal{D}$:

$$L_i^{\mathbf{T}(\mathbf{x})} = L_i^{\mathbf{B}_\mathcal{D}}; \quad U_i^{\mathbf{T}(\mathbf{x})} = U_i^{\mathbf{B}_\mathcal{D}}. \tag{3.4}$$

Since attributes in $A(\mathbf{x})$ and $\overline{A}(\mathbf{x})$ cover all dimensions in $\mathbb{R}^d$, given a test sample $\mathbf{x}$, its bounding box $\mathbf{T}(\mathbf{x})$ then can be determined using the tree $\mathcal{T}$.

Definition 5. Given a completely random forest $\mathcal{F}$ with m trees. The bounding box $\mathbf{F}(\mathbf{x})$ is defined as the intersection of bounding boxes $\mathbf{T}_i(\mathbf{x})$ from all trees:

$$\mathbf{F}(\mathbf{x}) = \cap_{i=1}^m \mathbf{T}_i(\mathbf{x}). \tag{3.5}$$

Finding the intersection of bounding boxes from all trees to produce $\mathbf{F}(\mathbf{x})$ is computationally efficient because each bounding box is a rectangular box. For example, the maximal-compatible rule can be used [6], which scales linearly with the number of trees m and the number of input dimensions d.

Definition 6. The reconstructed sample $\mathbf{x}^{\text{rec}}$ of $\mathbf{x}$ from the forest $\mathcal{F}$ is defined as the center point of the bounding box $\mathbf{F}(\mathbf{x})$, i.e., $\forall i = 1, \ldots, d$:

$$x_i^{\text{rec}} = \frac{1}{2}\left(L_i^{\mathbf{F}(\mathbf{x})} + U_i^{\mathbf{F}(\mathbf{x})}\right). \tag{3.6}$$

Definition 7. The reconstruction error for a test sample $\mathbf{x} \in \mathbb{R}^d$ and its reconstructed sample $\mathbf{x}^{\text{rec}}$ is:

$$s = \left(\mathbf{x} - \mathbf{x}^{\text{rec}}\right)^\top \left(\mathbf{x} - \mathbf{x}^{\text{rec}}\right). \tag{3.7}$$

Algorithm 1: Evaluating Stage of RecForest

Input: Test sample $\mathbf{x}$, Forest $\mathcal{F}$
Output: Anomaly score s

$\mathbf{F}(\mathbf{x}) = \cap_{i=1}^m \mathbf{T}_i(\mathbf{x})$;
$\mathbf{x}^{\text{rec}} = \frac{1}{2}\left(\mathbf{L}^{\mathbf{F}(\mathbf{x})} + \mathbf{U}^{\mathbf{F}(\mathbf{x})}\right)$;
$s = \left(\mathbf{x} - \mathbf{x}^{\text{rec}}\right)^\top \left(\mathbf{x} - \mathbf{x}^{\text{rec}}\right)$;
Return s

The sample reconstruction error is then used as the anomaly score in RecForest for each test sample. Fig. 1 presents a detailed example on sample reconstruction with three completely random trees. We assume that the bounding box $\mathbf{B}_\mathcal{D}$ as given is $\mathbf{L}^{\mathbf{B}_\mathcal{D}} = [0, 0]$ and $\mathbf{U}^{\mathbf{B}_\mathcal{D}} = [4, 4]$, respectively. After passing the test sample $\mathbf{x} = [1.3, 2]$ through three trees shown in Fig. 1 (a), a bounding box can be obtained per tree according to equations (3.3) and (3.4). Concretely, we have:

$$\mathbf{L}^{\mathbf{T}_1(\mathbf{x})} = [0.0, 0.5], \ \mathbf{U}^{\mathbf{T}_1(\mathbf{x})} = [1.5, 4.0],$$
$$\mathbf{L}^{\mathbf{T}_2(\mathbf{x})} = [0.5, 1.0], \ \mathbf{U}^{\mathbf{T}_2(\mathbf{x})} = [3.0, 4.0],$$
$$\mathbf{L}^{\mathbf{T}_3(\mathbf{x})} = [0.5, 1.5], \ \mathbf{U}^{\mathbf{T}_3(\mathbf{x})} = [4.0, 2.5].$$

The intersection of three bounding boxes, colored red in Fig. 1 (b), corresponds to the bounding box of the forest $\mathcal{F}$. The reconstructed sample is $\mathbf{x}^{\text{rec}} = [1, 2]$, and the corresponding reconstruction error is 0.09.

The procedure of scoring a test sample for anomaly detection is given in Algorithm 1, and the entire process of the training and evaluating stage is called RecForest. Given a testing set consisting of normal samples and anomalies with unknown labels, RecForest can be used to provide a score for every test sample, and then they

can be ranked in the descending order. Samples ranked at the top are regarded as anomalies.

Consider that the size of training/testing set is n and t, and the maximum tree height h and the number of trees m are both constants, the time complexity of RecForest during the training and evaluating stage is $O(nmh)$ and $O(tmh)$, respectively. Therefore, the runtime of RecForest scales linearly with the data size. In addition, RecForest can be easily parallelized through exploiting the independency between trees in the forest. In the supplementary materials, we provide further discussions on the relationships between RecForest and two related methods: the neural network autoencoder and isolation forest.

3.3 RecForest for Outlying Aspects Mining.

Given the reconstructed sample in RecForest, here we show a new capability of RecForest which is not usually available in existing forest-based anomaly detectors, i.e., identifying outlying attributes of an anomaly with respect to the given dataset. The outlying attributes significantly enhance the interpretability of the anomaly.

Definition 8. The outlying score of each attribute $i = 1, \ldots, d$ for a test sample $\mathbf{x}$ and its reconstructed sample $\mathbf{x}^{\text{rec}}$ is defined as follows:

$$O_i = \frac{\exp\left(x_i - x_i^{\text{rec}}\right)^2}{\sum_{k=1}^{d} \exp\left(x_k - x_k^{\text{rec}}\right)^2}. \tag{3.8}$$

Equation (3.8) normalizes the discrepancies between $\mathbf{x}$ and $\mathbf{x}^{\text{rec}}$ on all attributes into a distribution, and attributes with large O_i indicate that they are more outlying ones. These attributes can be used to explain what makes an anomaly stand out from the training set. Identifying outlying attributes in this way is efficient in that they are directly inferred based on the reconstruction error on each attribute.

4 Experimental Settings

Dataset: We use 11 benchmark datasets for performance comparison. They are selected for two reasons: (1) All of them are commonly used in the literature of anomaly detection. (2) They are publicly available at Outlier Detection DataSets (ODDS)[1].

Baseline: We use reconstruction-based, forest-based, and traditional anomaly detectors as baselines: (1) **Deep Autoencoder**: Neural network autoencoder with nonlinear dimensionality reduction. The reconstruction error is used as the anomaly score [1]; (2) **VAE**: Variational autoencoder is a neural network based generative model that maximizes the likelihood of training data on its latent distribution [5]; (3) **DAGMM**: An anomaly detector based on the variant of neural network autoencoder, where a deep autoencoder and a gaussian mixture model are jointly trained [21]; (4) **iForest**: Isolation forest is a popular forest-based anomaly detector, which adopts average tree path length in the forest as the anomaly score [11]; (5) **PIDForest**: Another forest-based anomaly detector based on a novel mechanism called partial identification [8]; (6) **OC-SVM**: A kernel-based method on estimating the support of the distribution of normal data [14].

Training methodology: Since RecForest is a reconstruction-based anomaly detector, we follow the experiment setting with clean training data [3, 21]. Concretely, 60% of normal samples are selected by random sampling as the training set, and the rest 40% are mixed up with all anomalies as the testing set.

For the baseline methods, their hyper-parameter settings either use the default values or are determined via a grid search for a range of values. The details are provided in the supplementary materials. For RecForest, the number of trees is set as 100, and each tree grows until each leaf node contains precisely one training sample. Such a setting is purposely made to validate that RecForest does not require extensive hyper-parameter tuning like the neural network autoencoder.

In addition, we found that PIDForest failed to run if all training samples take the same value on a dimension. Therefore, such dimensions were removed in a dataset when evaluating PIDForest.

Evaluation metric: After the evaluating stage, labels on the testing set are made available to compute two performance measures: (1) AUC is the area under the Receiver Operating Characteristic curve. (2) Given K the number of anomalies in the testing set, Precision@K is the percentage of authentic anomalies among K test samples with the largest anomaly scores produced by an anomaly detector. Each experiment is evaluated over ten independent trials to report the average value and standard deviation.

5 Empirical Evaluation

The empirical evaluation aims to assess the relative performance of RecForest and different methods on anomaly detection, time efficiency, outlying aspects mining, and abilities on handling irrelevant attributes.

The empirical evaluation is divided into five subsections. First, RecForest is compared with the contenders on benchmark datasets. Second, the runtime of RecForest is evaluated. In the third and fourth experiments, we demonstrate the advantages of RecForest on mining outlying attributes and handling irrelevant attributes, respectively. The last experiment studies the sensitiv-

[1] http://odds.cs.stonybrook.edu

Table 2: Average AUC (%) with standard deviation per method. Brackets in the first column denote the data size N. Deep AE is short for the Deep AutoEncoder. The best results on datasets are indicated by (•).

Dataset	(N)	Deep AE [1]	DAGMM [21]	VAE [5]	OC-SVM [14]	PIDForest [8]	iForest [11]	RecForest
glass	(214)	60.08±4.22	69.54±11.42	57.32±2.95	45.66±1.59	68.80±3.90	74.57±3.20	• 77.70±3.56
ionosphere	(351)	90.76±1.70	79.21±8.72	83.74±1.68	92.75±1.22	82.44±3.64	91.98±2.24	• 96.57±0.63
wbc	(378)	94.10±0.75	78.07±12.50	93.43±0.90	• 95.53±0.86	95.14±0.75	95.28±1.26	95.31±1.42
vowels	(1456)	62.43±1.08	58.22±10.95	62.61±0.86	82.33±0.75	73.04±1.34	76.85±2.83	• 83.53±2.94
letter	(1600)	52.47±1.09	53.62±4.16	52.36±0.86	52.37±0.75	63.41±1.74	63.17±2.14	• 65.60±1.58
musk	(3062)	• 100.0±0.00	92.15±8.27	• 100.0±0.00	• 100.0±0.00	• 100.0±0.00	94.63±5.58	• 100.0±0.00
optdigits	(5216)	56.60±0.82	38.26±17.11	52.37±0.62	68.79±0.91	• 87.43±3.24	81.03±3.13	86.69±1.63
satimage-2	(5803)	97.88±0.04	78.38±16.70	97.75±0.04	99.66±0.01	82.22±7.30	99.43±0.10	• 99.71±0.12
satellite	(6435)	66.65±0.23	65.56±10.71	64.95±0.26	72.60±0.25	53.08±3.07	80.34±1.42	• 81.54±0.83
pendigits	(6870)	94.25±0.26	74.94±12.41	93.76±0.25	• 97.85±0.17	95.92±0.05	96.87±0.95	95.38±0.61
shuttle	(49097)	99.32±0.04	83.70±17.58	99.04±0.02	99.28±0.03	93.07±10.73	99.31±0.09	• 99.38±0.15

Table 3: Average Precision@K (%) with standard deviation per method. Brackets in the first column denote the data size N. Deep AE is short for the Deep AutoEncoder. The best results on datasets are indicated by (•).

Dataset	(N)	Deep AE [1]	DAGMM [21]	VAE [5]	OC-SVM [14]	PIDForest [8]	iForest [11]	RecForest
glass	(214)	15.55±9.37	15.55±13.91	13.33±7.03	15.55±5.74	11.11±7.41	15.55±7.77	• 18.89±7.50
ionosphere	(351)	83.10±2.77	71.11±25.38	77.86±2.41	86.19±1.60	62.43±6.57	85.56±3.26	• 92.06±1.35
wbc	(378)	69.05±4.63	44.29±18.92	66.67±3.18	68.57±4.02	69.05±3.37	• 71.91±3.51	70.00±2.30
vowels	(1456)	21.20±2.53	17.40±11.51	21.40±2.32	40.60±1.26	28.66±4.84	28.40±5.15	• 40.68±3.77
letter	(1600)	16.70±1.77	20.40±5.06	16.10±1.29	21.70±2.11	17.50±2.27	20.70±3.37	• 37.76±5.00
musk	(3062)	• 100.0±0.00	55.57±34.03	• 100.0±0.00	• 100.0±0.00	• 100.0±0.00	57.44±21.37	• 100.0±0.00
optdigits	(5216)	0.67±0.00	1.40±1.76	0.67±0.00	8.47±0.32	30.60±9.18	16.13±6.22	• 33.89±5.26
satimage-2	(5803)	85.92±1.15	22.68±27.31	84.23±1.11	• 93.81±0.98	52.69±12.04	88.03±1.37	91.41±1.40
satellite	(6435)	63.14±0.13	63.71±8.33	62.01±0.20	68.16±0.06	51.21±1.64	71.37±1.27	• 76.32±0.71
pendigits	(6870)	47.44±1.51	19.30±15.60	45.13±1.18	• 68.33±2.50	50.29±3.03	58.50±6.59	58.53±4.21
shuttle	(49097)	96.13±0.11	66.19±30.74	95.64±0.08	95.67±0.27	82.56±11.31	96.73±0.73	• 96.82±0.48

ity of hyper-parameters in RecForest.

An efficient implementation of RecForest along with the supplementary material are publicly available[2].

5.1 Anomaly Detection Accuracy. The experimental results are presented in Tables 2 and 3 in terms of AUC and Precision@K, respectively. Both tables show that on most benchmark datasets, RecForest achieves better performance than other contenders.

Although RecForest and neural network autoencoder are both reconstruction-based anomaly detectors, RecForest is almost always better than the autoencoder. RecForest is also a better reconstruction-based anomaly detector especially on small datasets, winning by a large margin over other reconstruction-based methods. In addition, the variant of neural network autoencoder, i.e., DAGMM, suffers from large variance on several datasets. The reason for the better performance of RecForest lies in its different mechanism of anomaly detection. Experiments in the supplementary material show that RecForest is able to provide a much larger reconstruction gap between normal data and anomalies.

RecForest also achieves better performance than other forest-based anomaly detectors on most datasets with small variance. Considering that RecForest and iForest are both built on a completely random forest, such results validate that exploiting information in the forest is helpful on improving the detection accuracy.

5.2 Runtime Comparison. Here we compare the runtime of RecForest with iForest, PIDForest, and the neural network autoencoder. The goal is to verify that RecForest has small computational cost.

For autoencoder, we report results when it is trained on either CPU or GPU. For iForest, PIDForest, and RecForest, the number of trees used is 100; and their training and evaluating stages are parallelized on CPU. Concretely, all processes in CPU are used for the construction and inference on trees, and each process focuses on the routine of one tree at a time. Table 4 presents the training and total runtime of different methods on a randomly selected dataset: pendigits. The results on other datasets have similar trends.

The training cost of iForest and RecForest are the smallest, as each internal node in the completely random tree randomly selects an attribute and cut-off

[2] https://github.com/xuyxu/RecForest

Table 4: Runtime in seconds on pendigits dataset.

Model (Device)	Training (s)	Total (s)
Autoencoder (CPU)	62.2	62.5
Autoencoder (GPU)	32.5	32.6
PIDForest (CPU)	7.5	9.0
iForest (CPU)	0.4	1.8
RecForest (CPU)	0.3	2.1

for splitting. Training PIDForest is more expensive as a splitting criterion is used to find splits for internal nodes. Training the neural network autoencoder also has large training cost because of the iterative optimization on network parameters. The evaluating time of RecForest is 1.8 seconds on the pendigits dataset, which is close to that of iForest. Although the evaluating time of forest-based methods is larger, the overall runtime of RecForest and iForest is still the smallest, more than one order of magnitude faster than the autoencoder.

5.3 Mining Outlying Attributes. This section aims to evaluate the capability of RecForest on finding meaningful outlying attributes given an anomaly.

First experiment: We use the KDD Cup'99 dataset that contains 97,278 samples with normal network connections and 396,743 samples with different kinds of network attack. We produce RecForest using all normal samples, and the aim is to identify outlying attributes for samples belonging to each kind of network attack. The results are presented in Fig. 2.

Using the `guess_passwd` attack as an example, the key outlying attribute identified by RecForest is `num_failed_logins`. This result conforms to the domain knowledge that a `guess_passwd` attack is conducted by guessing the login password of user accounts.

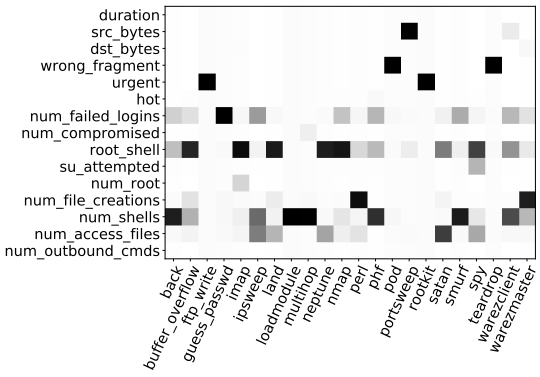


Figure 2: Verification on the KDD Cup'99 dataset, each row corresponds to an attribute, and each column corresponds to one kind of network attack. Darker grids indicate larger anomalousness identified by RecForest.

Table 5: Comparisons between RecForest, Autoencoder, and Beam Search with isolation forest on a benchmark dataset for outlying aspects mining.

ID	Ground Truth	RecForest	AE	Beam
1	$\{8,9\}$	$\{8,9\}$	$\{8,9\}$	$\{8,9\}$
2	$\{0,1\}$	$\{0,1\}$	$\{0,1\}$	$\{0,1\}$
3	$\{6,7\}$	$\{6,7\}$	$\{6,7\}$	$\{6,7\}$
4	$\{0,1\}$	$\{1,2\}$	$\{1,7\}$	$\{0,1\}$
5	$\{2,3,4,5\}$	$\{1,2,3,5\}$	$\{1,3,5,8\}$	$\{2,3,4,5,8\}$
6	$\{2,3,4,5\}$	$\{2,3,4,5\}$	$\{0,2,3,4\}$	$\{2,5\}$
7	$\{8,9\}$	$\{8,9\}$	$\{8,9\}$	$\{8,9\}$
8	$\{0,1\}$	$\{0,1\}$	$\{0,1\}$	$\{0,1\}$
9	$\{0,1\}$	$\{0,1\}$	$\{0,1\}$	$\{0,1\}$
10	$\{2,3,4,5\}$	$\{0,2,4,8\}$	$\{0,2,4,8\}$	$\{0,7\}$
11	$\{2,3,4,5\}$	$\{2,3,4,7\}$	$\{2,3,4,7\}$	$\{2,3,4,5\}$
12	$\{8,9\}$	$\{8,9\}$	$\{6,9\}$	$\{8,9\}$
13	$\{2,3,4,5\}$	$\{0,3,4,7\}$	$\{0,3,4,7\}$	$\{0,2,3,5,9\}$
14	$\{6,7\}$	$\{6,7\}$	$\{6,7\}$	$\{6,7\}$
15	$\{6,7\}$	$\{6,8\}$	$\{6,7\}$	$\{6,7\}$
16	$\{6,7\}$	$\{6,7\}$	$\{6,7\}$	$\{6,7\}$
17	$\{8,9\}$	$\{5,9\}$	$\{5,9\}$	$\{8,9\}$
18	$\{8,9\}$	$\{8,9\}$	$\{8,9\}$	$\{8,9\}$
	Count on Hitting	11/18	10/18	14/18
	Overall Runtime (s)	0.3	8.7	55.7

Second experiment: We follow [17] to use the same synthetic dataset to evaluate the ability of RecForest on mining outlying attributes. The synthetic dataset contains 981 normal samples and 18 anomalies in $\mathbb{R}^{10}$. In addition, ground truths on outlying attributes of each anomaly are available. Two contenders are included: (1) AutoEncoder (AE), which uses a neural network autoencoder to score the anomalousness of attributes; (2) Beam, a state-of-the-art outlying aspect miner that combines a beam search with the anomaly score of iForest to find outlying attributes [17]. The maximum number of attributes searched and the beam width in Beam are set as 5 and 200, respectively.

The result is presented in Table 5. For RecForest and Autoencoder, since they score each attribute, we report the top-K outlying attributes for each anomaly, where K is the actual number of outlying attributes.

The results show that Beam successfully identified outlying attributes for 14 anomalies; RecForest and Autoencoder identified correctly for 11 and 10 anomalies, respectively. This result is not surprising considering that Beam conducts a systematic search on subspaces.

Note that the overall time cost of RecForest is one to two orders of magnitude faster than the two contenders. The runtime complexity of Beam search on mining outlying aspects for an anomaly in a d-dimensional dataset is $O(d^2 + wdd_{max})$, where w and d_{max} are the beam width and the maximum number of subspace dimensions, respectively. In contrast, the overall time com-

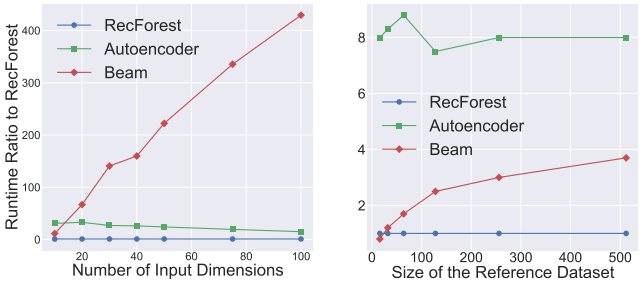


Figure 3: Runtime ratio wrt RecForest when increasing the number of dimensions and reference dataset size.

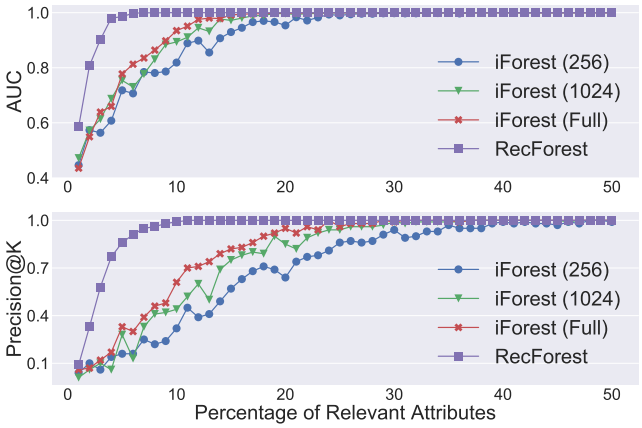


Figure 4: AUC and Precision@K of RecForest, iForest when increasing the percentage of relevant attributes.

plexity of finding outlying attributes with RecForest is equivalent to fitting a RecForest model and obtaining the reconstructed sample for the given anomaly. Therefore, the number of input dimensions d has a small impact on the time cost of RecForest.

Scale-up tests: To further examine the time efficiency of RecForest, we conduct two scale-up tests on increasing the number of input dimensions and the size of the reference dataset. The runtime ratios wrt RecForest presented in Fig. 3 show that RecForest is orders of magnitude faster than Beam as the number of dimensions increasing. RecForest has the highest runtime efficiency as the size of reference dataset increases.

5.4 Robustness against Irrelevant Attributes. This section aims to validate that despite RecForest and iForest are both based on a completely random forest, fully exploiting information in the forest enables RecForest to have stronger robustness against irrelevant attributes when handling high-dimensional datasets.

First, a synthetic dataset with 1000 dimensions is generated following the setting in [2]. For each sample, r percentage of attributes are relevant attributes while remaining attributes are noise. The details of the synthetic dataset are available in the supplementary material. Second, we increase the percentage of relevant attributes r from 1% to 50%.

Fig. 4 presents the results of RecForest and iForest with different sub-sampling sizes ("Full" indicates no sub-sampling) in terms of AUC and Precision@K. It shows that RecForest is much more robust against irrelevant attributes than iForest. Concretely, RecForest achieves AUC=1.0 when the percentage of relevant attributes r increases to 10%. The same thresholds are 41% and 28% when the sub-sampling size of iForest is set to 256 and "Full", respectively.

A large sub-sampling size effectively increases the average depth of decision tree in iForest, resulting in more attributes being used to identify anomalies (especially the relevant attributes). This explains why iForest without sub-sampling has slightly better performance than that with sub-sampling sizes 256 and 1024 when handling irrelevant attributes. Nevertheless, the overall performance of RecForest is still much better. Despite using the same completely random forest, RecForest is more robust against irrelevant attributes in that information on splitting attributes of internal nodes is incorporated into the anomaly detection process. Concretely, the error defined in Equation (3.7) can be decomposed into relevant and irrelevant components as follows:

$$l(\mathbf{x}, \mathbf{x}^{\text{rec}}) = \sum_{j \in A_1} (x_j - x_j^{\text{rec}})^2 + \sum_{k \in A_2} (x_k - x_k^{\text{rec}})^2,$$

where A_1 and A_2 denote the sets of relevant and irrelevant attributes of the dataset, respectively.

It turns out that using RecForest, the reconstruction error of anomalies on relevant attributes A_1 remains much larger than normal data, whereas their differences on irrelevant attributes A_2 are small. Therefore, irrelevant attributes have little impacts on RecForest as long as they corrupt "equally" to both anomalies and normal data. However, this cannot be identified using iForest because it does not incorporate information on the used splitting attributes, all attributes therefore contribute equally to its anomaly score: tree path length.

5.5 Parameter Sensitivity. In this section, we study the sensitivity of two hyper-parameters in RecForest: tree depth and the number of trees. Three datasets are used: wbc, letter, and pendigits. For the tree depth, we increase its value from 3 to 30 with a step size 3. For the number of trees, we increase its value from 20 to 200 with a step size 20.

Fig. 5 presents the trends on average AUC and

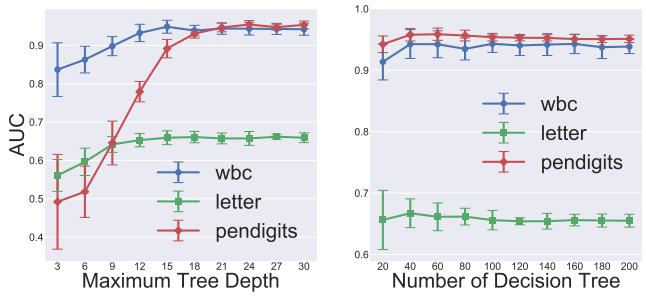


Figure 5: Parameter sensitivity of RecForest.

standard deviation over 10 independent trails. First, the result shows that a better performance can be achieved by increasing the tree depth. When the data size is large, a higher tree depth is recommended to better capture information about the dataset. Second, the performance of RecForest converges quickly with a small number of trees. On all datasets, the performance does not improve or deteriorate when the number of trees is higher than 100. It can also be observed that increasing the values of two hyper-parameters both leads to a smaller standard deviation.

6 Conclusion

This paper proposes a novel reconstruction-based anomaly detector called RecForest. Unlike existing works that universally adopt the neural network autoencoder, RecForest is built on a completely random forest; and the sample reconstruction is established by exploiting the intersection of the bounding boxes in the forest for each sample.

It has three advantages over existing anomaly detectors. First, RecForest has significantly fewer hyper-parameters and smaller training cost compared with the neural network autoencoder. As a result, RecForest can be applied to large scale datasets that would otherwise be impossible for neural network autoencoder. Second, RecForest fully exploits the information in the forest and achieves two new capabilities not available in existing forest-based methods, i.e., RecForest can effectively mine outlying attributes and handle irrelevant attributes in high-dimensional datasets. Third, in terms of mining outlying attributes, RecForest runs orders of magnitude faster than the existing outlying aspects miner using Beam search.

Acknowledgement. This research was supported by the National Science Foundation of China (61921006).

The authors would like to thank Peng Zhao and Yu-Cheng He for helpful discussions, and thank the anonymous reviewers for their valuable comments.

References

[1] C. C. Aggarwal, "Outlier analysis," Springer, 2017.
[2] T. R. Bandaragoda, K.-M. Ting, D. Albrecht, F.-T. Liu, Y. Zhu et al., "Isolation-based anomaly detection using nearest-neighbor ensembles," *Computational Intelligence*, vol. 34, no. 4, pp. 968-998, 2018.
[3] J.-H. Chen, S. Sathe, C. C. Aggarwal and D. Turaga, "Outlier detection with autoencoder ensembles," in *SDM*, 2017, pp. 90-98.
[4] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," arXiv:1901.03407, 2019.
[5] D. P. Kingma, M. Welling, "Auto-encoding variational bayes," arXiv:1312.6114, 2013.
[6] J. Feng and Z.-H. Zhou, "Autoencoder by forest," in *AAAI*, 2018, pp. 2967-2973.
[7] J. Feng, Y. Yu and Z.-H. Zhou, "Multi-layered gradient boosting decision trees," in *NeurIPS*, 2018, pp. 3551-3561.
[8] P. Gopalan, V. Sharan and U. Wieder, "PIDForest: Anomaly detection via partial identification," in *NeurIPS*, 2019, pp. 15783-15793.
[9] S. Guha, N. Mishra, G. Roy, O. Schrijvers, "Robust random cut forest based anomaly detection on streams," in *ICML*, 2016, pp. 2712-2721.
[10] G.-T. Pedro, D.-V. Jesus, M.-F. Gabriel and V. Enrique, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers and Security*, vol. 28, no. 1-2, pp. 18-28, 2009.
[11] F.-T. Liu, K.-M. Ting and Z.-H. Zhou, "Isolation forest," in *ICDM*, 2008, pp. 413-422.
[12] S.-H. Lyu, L. Yang and Z.-H. Zhou, "A refined margin distribution analysis for forest representation learning," in *NeurIPS*, 2019, pp. 5531-5541.
[13] B. Micenková, R. T. Ng, X.-H. Dang, and I. Assent, "Explaining outliers by subspace separability," in *ICDM*, 2013, pp. 518–527.
[14] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor and J. C. Platt, "Support vector method for novelty detection," in *NIPS*, 2000, pp. 582-588.
[15] D. Samariya, J. Ma, S. Aryal and K.-M. Ting, "A comprehensive survey on outlying aspect mining methods," arXiv: 2005.02637, 2020.
[16] C. Vens, F. Costa, "Random forest based feature induction," in *ICDM*, 2011, pp. 744-753.
[17] N. X. Vinh, J. Chan, S. Romano, J. Bailey, C. Leckie, et al., "Discovering outlying aspects in large datasets," *Data Mining and Knowledge Discovery*, vol. 30, no. 6, pp. 1520-1555, 2016.
[18] Z.-H. Zhou, "Ensemble methods: Foundations and algorithms," CRC press, 2012.
[19] Z.-H. Zhou and J. Feng, "Deep Forest: Towards an alternative to deep neural networks," in *IJCAI*, 2017.
[20] Z.-H. Zhou and J. Feng, "Deep Forest," *National Science Review*, vol. 6, no. 1, pp. 74-86, 2018.
[21] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu et al., "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *ICLR*, 2018.