A Transductive Forest for Anomaly Detection with Few Labels

Jingrui Zhang[1], Ninh Pham[1] ✉[0000−0001−5768−9900], and Gillian Dobbie[1]

School of Computer Science, University of Auckland
{jzha968, ninh.pham, g.dobbie}@auckland.ac.nz

Abstract. Extensive labeled training data for anomaly detection is enormously expensive and often unavailable in data-sensitive applications due to privacy constraints. We propose TransForest, a transductive forest for anomaly detection, in the semi-supervised setting where few labels are available. Guided by little label information, TransForest pushes classification boundaries toward sensitive areas where abnormal and normal points are located, increasing learning capacity. Empirically, TransForest is competitive with other unsupervised and semi-supervised representative detectors given a small number of labeled points. TransForest also offers a feature importance ranking consistent with the rankings provided by popular supervised forests on low-dimensional data sets. Our code is available at `https://github.com/jzha968/transForest`.

1 Introduction

Anomaly detection is a fundamental data mining task with many applications in several domains, such as banking fraud detection, system health monitoring, medical diagnosis, and law enforcement [1]. In these applications, the data features are represented as a high-dimensional vector, and anomalous behaviors tend to be masked by the noise effects of irrelevant features. More importantly, the label information is often limited due to privacy constraints and significant human effort. For instance, labeling medical images is very expensive, and releasing their label information can be against the privacy rights [2,7]. Therefore, it is challenging to detect anomalies effectively and efficiently, given a limited amount of labeled points, and provide explanations regarding detected anomalous patterns to support end-users decisions.

Due to the difficulty of accessing the ground truth, unsupervised methods are popular techniques to detect anomalies in the last decades [1,25]. These approaches are based on a common assumption that anomalies are likely located in relatively sparse regions while normal points are often distributed in dense neighborhoods. Unsupervised methods proposed different notions to measure the sparsity of a point, for example, distance/density-based models [4,24,28], isolation-based models [10,11,17], histogram-based models [9,23,27], and distribution-based models [15,16]. Since anomaly detection is performed in high-dimensional space and anomalies are masked by multiple irrelevant dimensions, anomaly detectors in relevant subspaces show superiority over full space counterpart solutions [13,14,17].

However, the exponential complexity of the number of subspaces is the main bottleneck in detecting anomalous patterns accurately and efficiently.

Instead of finding relevant subspaces, recent semi-supervised models [21,26,30] use a little label information to extract rich feature representations for the data points. These features preserve discriminative information of abnormal and normal points and are helpful for anomaly detection. XGBOD [30], a boosting-based approach, combines unsupervised feature representations with the original features to enrich the augmented feature space. Deep learning models [20,26] couple rich feature representation learning with some specific anomaly score. These methods significantly improve unsupervised models by leveraging advances in boosting and deep neural networks. A recent benchmark [12] shows that the strongest baseline XGBOD [30] can improve up to 10% of the average detection accuracy over unsupervised competitors with just 5% labeled anomalies. However, we find that these semi-supervised methods require many labeled points for training to achieve reasonable accuracy.

This work studies tree-based semi-supervised ensembles for anomaly detection. We propose TransForest, a novel transductive semi-supervised forest, that requires few labeled points to select relevant subspaces for constructing the forest. Unlike the conventional transductive decision forest [6] that maximizes the *mixed* information gain derived from the unlabeled and labeled points, TransForest estimates the information gain by spreading the known label information to the other unlabeled points during the feature selection process. In particular, at each tree level, we construct a histogram on each of the randomly selected features. Then, we pseudo-label unlabeled points based on the data distribution of the histogram and labeled points. Both labeled and pseudo-labeled points are used to estimate the information gain for the selected feature and its corresponding splitting value.

Given few labeled abnormal and normal points, TransForest can not only discriminate the sparse subspace areas that contain normal points or anomalies, but also push the classification boundaries toward dense subspace areas where points from both classes constitute. Empirically, TransForest is fast and outperforms other unsupervised tree-based forests with just 10 labeled points. Given the same amount of label information, TransForest achieves competitive accuracy compared with current advanced semi-supervised methods. Especially, TransForest offers a feature importance ranking consistent with the rankings provided by well-known supervised methods, including Random Forest [3] and Extra Trees [8] trained on fully labeled data.

2 Related work

We briefly review popular unsupervised and semi-supervised anomaly detectors.

Unsupervised models. Due to the difficulty of accessing the ground truth, traditional unsupervised models compute an anomaly score for each data point such that anomalies are likely scored larger than normal points. The anomaly score of a point $\mathbf{q}$ is derived from the sparsity of the local area around $\mathbf{q}$. Different

approaches use different notions to measure the sparsity of $\mathbf{q}$ in high-dimensional space; for example, distance-based methods [24] using the k-nearest neighbor distance and density-based methods [4] using the difference ratio between the sparsity of $\mathbf{q}$ and the average sparsity of its local neighborhood.

Subspace methods. Since proximity-based methods suffer the "curse of dimensionality" and detected anomalies tend to be uninteresting noise instances, their subspace variants [13,14] compute anomaly scores on specific subsets of dimensions where interesting anomalies tend to appear. Histogram-based approaches construct a histogram on each dimension or random subsets of dimensions [9,27], and use the bin size of the histogram where $\mathbf{q}$ locates to estimate the sparsity of $\mathbf{q}$. iForest [17] and its variants [10,11] are tree-based solutions that measure the subspace sparsity of a point via the isolation concept. Since anomalies are few and different on a specific subset of dimensions, they will be likely isolated on the leaf node of a decision tree built on randomly selected features and random splitting values. Combined with subsampling, tree-based ensembles are the most efficient and accurate unsupervised anomaly detectors [12].

Semi-supervised models. Semi-supervised models combine a small amount of labeled data with unlabeled data to learn better classification boundaries or extract rich representations to distinguish abnormal and normal points.

Tree-based methods. Popular supervised tree-based ensembles such as Random Forest [3] and Extra Trees [8] build the tree via the information gain derived from labeled points. Given the limit of label information, their transductive semi-supervised variants [6] select the feature and its splitting value that maximizes the *mixed* information gain derived from both unlabeled and labeled points. The unlabeled information gain component is derived from the Gaussian density estimation learned by the maximum likelihood over many samples. This step suffers substantial computational overheads, significantly increasing the training time. A semi-supervised variant Hybrid iForest [19] incorporates a new distance-based score from the test point to the labeled points into the iForest's anomaly score to improve the accuracy. Given little label information, its performance is unstable due to the high dependence on the data distribution.

Boosting-based methods. XGBOD [30] is a recent semi-supervised approach that leverages XGBoost [5], a fast and accurate gradient boosting library, to improve detection performance. XGBOD first constructs a new set of features based on unsupervised anomaly scores. This new set of features augmented with the original ones forms a new rich feature space where XGBoost is used to learn a binary classification. Though XGBOD achieves good performance, it does not provide any explanation for detected anomalies.

Deep learning methods. Different from XGBOD, several deep learning-based approaches [20,22,26] couple the learning representations and anomaly scores to improve the detection accuracy. For example, REPEN [20] learns low-dimensional representations that separate normal and abnormal distance-based behaviors. DeepSAD [26] maps data points into the sphere of minimum volume and learns a hyperplane that separates the sphere center and the projected data points.

Algorithm 1 An ExtraTree Construction

1: **procedure** BUILDTREE(A subset of points S, h_{max}, n_{min}, the current depth $h = 0$)
2: If $h \geq h_{max}$ or $|S| \leq n_{min}$, create a leaf node and return
3: Generate k pairs (f_i, v_{f_i}) where f_i is the feature and v_{f_i} is a random splitting value uniformly within the empirical range of f_i
4: $(f_*, v_{f_*}) = \arg\max_i I(S, f_i, v_{f_i})$ using Eq. (1)
5: Split S into S^l and S^r using (f_*, v_{f_*})
6: BuildTree($S^l, h + 1$)
7: BuildTree($S^r, h + 1$)

Although semi-supervised deep learning approaches require a significant number of labeled points, their learned representations are often difficult to interpret.

3 Preliminary

We present the preliminary tree-based ensemble methods, including Extra Trees (ET) [8], Random Forest (RF) [3], and iForest [17]. Since these forest variants build an ensemble of t trees, each over a random sample of s points, for simplicity, we will discuss the generic tree construction and point out their key differences.

Tree construction. Both RF and ET grow a tree by computing the feature and its splitting values that maximize the information gain for a given node. The information gain I measures the impurity gain of a node S after splitting S into two nodes S^l and S^r using the feature f_i and splitting value v_{f_i}. Let p_0 and p_1 be the empirical probability of the normal and abnormal classes in any node A, the impurity of A is derived from its entropy value $H(A) = -p_0 \log_2(p_0) - p_1 \log_2(p_1)$. The information gain of S for the feature f_i and splitting value v_{f_i} is as follows.

$$I(S, f_i, v_{f_i}) = H(S) - \frac{|S^l|}{|S|} H(S^l) - \frac{|S^r|}{|S|} H(S^r)\,. \tag{1}$$

Denote by n_{min} and h_{max} the maximum number of points on leaf nodes and the maximum tree depth. These parameters are used to control the over-fitting in the forests. Algorithm 1 shows how to build an ET tree using information gain.

Difference between ET and RF. While both ET and RF build a tree on a *subset* of features, their primary difference is on generating the splitting values. ET picks the splitting value uniformly within the empirical range. RF computes the local optimal splitting value from randomly selected features. Since ET uses more randomness than RF, ET offers slightly lower classification accuracy but runs significantly faster than RF.

Difference between ET and iForest. In the unsupervised setting, building a tree of iForest is identical to the ET process without Step 4. Since there is no label information, Step 3 just needs $k = 1$ to reduce the training time. The key difference between iForest and ET is that iForest sets $n_{min} = 1$, $h_{max} = \log_2(s)$ and uses the path length from the leaf node to the root as an anomaly score. Since anomalies tend to differ from the other points on a specific subspace, they

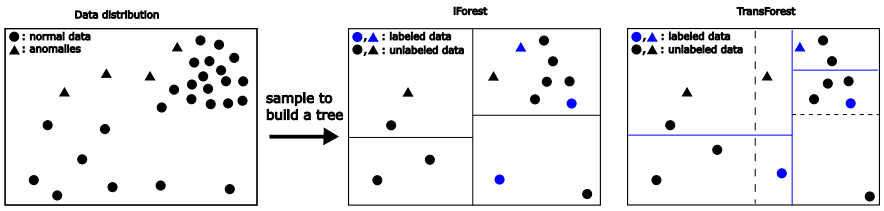


Fig. 1: An illustration of the tree constructions. Without label information, iForest's splits tend to separate dense/sparse regions. Guided by few labeled points and pseudo-labeled points via constructed histograms (dashed black lines), TransForest tends to split the node (blue lines) on sensitive regions where labeled points of two classes appear (Rule 3).

are likely isolated at shallow leaf nodes while normal points are likely located on deeper leaves. Building a forest will boost the performance since each tree deals with a random subspace derived from randomly selected features.

4 TransForest: A Transductive Forest

We observe that the unsupervised isolation-based mechanism tends to detect anomalies in sparse subspace areas. When two anomalies in a dense anomalous cluster are sampled to build trees, this cluster of anomalies receives similar anomaly scores as normal points. Since high-dimensional data sets contain multiple irrelevant features caused by measurement noise, sparse regions in randomly selected subspaces consist of both abnormal and normal points. In this case, normal points will be flagged as anomalies by the isolation-based mechanism.

We present TransForest to overcome these drawbacks by leveraging few labeled abnormal and normal points. TransForest is a novel transductive tree-based ensemble that spreads the little known label information to the other unlabeled points. Guided by pseudo-labeled and labeled points, each tree of TransForest selects more relevant subspaces and splitting values to investigate, providing higher performance with negligible computational overheads.

We denote the unlabeled and labeled sets by X_u and X_l, respectively. These two sets of sizes $|X_u| = n_u, |X_l| = n_l$ form the data set X of $n = n_u + n_l$ points in d dimensions. Since the number of labeled points is tiny, i.e. $n_l \ll n_u$, we *always* use X_l to build every tree of TransForest to learn more discriminate subspaces via the feature selection and its corresponding splitting value.

Overview. Providing the class probability is essential for enabling information gain-based splitting criterion. Given few labeled points, the information gain guides the feature selection on just a few shallow nodes in supervised tree-based methods. After a certain level, supervised models switch to unsupervised learning which unfortunately limits the use of label information. To maximize the use of labeled points, we propose a label propagation that spreads label information to their local regions to improve the tree construction. We implement

Algorithm 2 Compute Information Gain $I(S, f_i, v_{f_i})$

1: **function** ComputeGain(The node S, X_l, f_i, v_{f_i})
2: Create a fixed bin width histogram for S on f_i using $\log_2(|S|)+1$ bins
3: Split the bin containing v_{f_i} into two new bins
4: Compute the density threshold $\Delta = 0.1|S|$
5: **for each** bin B in the histogram **do**
6: Count pseudo-labeled abnormal and normal points in B using Pseudo-label Counting(B, Δ, X_l) in Algorithm 3
7: Given the number of abnormal and normal points in each bin, compute the entropy $H(S)$, $H(S^l), H(S^r)$ for the pair (f_i, v_{f_i})
8: **return** $I(S, f_i, v_{f_i})$ using Eq.(1)

Algorithm 3 Counting # pseudo-labels of each class

1: **function** Pseudo-label Counting(Bin B, Δ, X_l)
Ensure: # normal points, # anomalies in bin B
2: **if** B does not contain labeled points **then**
3: **if** $|B| \geq \Delta$ **then return** $(|B|, 0)$ ▷ Dense area: label all points as normal
4: **else return** $(0, |B|)$ ▷ Sparse area: label all points as anomalies
5: **if** $|B| \geq \Delta$ and B contains only anomalies **then return** $(0.9|B|, 0.1|B|)$
6: **if** B has $m_0 \geq 0$ normal points and $m_1 \geq 0$ anomalies **then return** $\left(\frac{m_0}{m_0+m_1}|B|, \frac{m_1}{m_0+m_1}|B|\right)$

the spreading mechanism via a histogram built from the empirical range of each feature. The constructed histograms will be used to *pseudo-label* unlabeled points based on their distribution and labeled points. Given labeled and pseudo-labeled points, we can compute the information gain and resort to standard supervised learning Extra Trees to build learning trees. We also output approximate feature importance ranking as an interpretation effort for TransForest. Figure 1 shows how TransForest leverages few labels to learn better splitting values than iForest.

Training phase. TransForest uses Algorithm 1 to build trees. Each tree uses *all* labeled points, and the information gain in Step 4 is derived from a histogram-based label propagation. On the tree node S, for a randomly selected feature f_i and random splitting value v_{f_i}, TransForest constructs a histogram and pseudo-labels *all* unlabeled points by propagating labeled points to the other points in every histogram bin, as shown in Algorithm 2. In particular, we count the number of pseudo-labeled normal and abnormal points in each bin and use them to compute the information gain for the pair (f_i, v_{f_i}).

Counting pseudo-labels of each class. We construct a fixed bin width histogram on the empirical range of random feature f_i to pseudo-label unlabeled points due to the fact that many histogram-based unsupervised methods [9,23,27] are efficient and effective on detecting anomalies in high-dimensional space. For a node S, we construct a histogram using $\log_2(|S|)+1$ bins. If there are no labeled points in the bin, we resort to the commonly assumed prior of unsupervised

learning, which is anomalies tend to be located in sparse local regions whereas dense local regions tend to contain normal points. We use the threshold $\Delta = 0.1|S|$ to determine a dense/sparse bin. For a bin B of size $|B|$, Algorithm 3 shows how to count the number of pseudo-labels for each class using the following rules.

1. **Rule 1:** If B does not have labeled points, we resort to the common unsupervised prior. That is, if B is dense, i.e. $|B| \geq \Delta$, all points in B are labeled as normal points; otherwise, they are all anomalies (Lines 2 – 4).
2. **Rule 2:** If B is dense but contains only anomalies, B will have $0.9|B|$ normal points and $0.1|B|$ anomalies (Line 5) (e.g. normal points dominate anomalies).
3. **Rule 3:** Otherwise, we use the ratio between labeled abnormal and normal points to compute the number of pseudo-labels for each class (Line 6).

Rule 1 reflects the unsupervised prior; hence, TransForest can run in the unsupervised setting. Rules 2 and 3 push the classification boundaries toward sensitive areas where abnormal and normal points appear. Since normal points dominate anomalies, pushing the learning boundary toward sensitive areas increases the chance of finding discriminate local sparse areas to isolate anomalies in shallow nodes, as shown in Figure 1.

Testing phase. Since few labeled points of both classes are available in every tree, we will adjust the anomaly scores on isolated leaves where labeled points locate. In particular, for the leaves that contain only labeled anomalies, we set $anomalyScore = 1$. We set $anomalyScore = h_{max}$ for the leaves containing only labeled normal points. In other words, the local areas with only one-side labeled points are classified as this one-sided class. For most leaves without labeled points, we set $anomalyScore$ as the path length from the leaf to the root, similar to iForest. This adjustment leverages the labeled points to significantly improve the accuracy of TransForest on real-world data sets where local areas tend to contain the same class information.

Hyperparameter setting. Since TransForest is a semi-supervised variant of iForest, we use $t = 100$ trees, each built on a subset of random points of size s. For each tree node, we select the best (f_i, v_{f_i}) among $k = 10$ random choices.

Since each tree of TransForest uses all labeled points X_l, and we need a sufficient unlabeled point to execute the spreading mechanism, we set $s = \texttt{max}(256, 2n_l)$ to use additional $s - n_l$ unlabeled points. Since X_l is often tiny, this setting will not affect the training time complexity of TransForest.

We heuristically use $\Delta = 0.1|S|$ as a density threshold for the node S in the training phase to determine dense/sparse regions. We observe that the fixed bin width histogram constructed on the empirical range of a feature has a very skew distribution. Most of the sampled points are distributed in a few bins, while the rest of the bins are almost empty. Hence, the performance is similar for any density threshold $\Delta \in \{0.05, 0.1, 0.2, 0.3\}|S|$, as shown in the experiment.

The last hyperparameter is the heuristic setting used for the dense area with only anomalies (Rule 2). Without the class imbalance ratio, we set the ratio of 90% normal points and 10% anomalies to illustrate the domination of the normal class. Otherwise, this ratio can be set as the known class imbalance ratio. We observe that the case of dense regions with only anomalies is very rare due to

the imbalanced property, changing the anomaly ratio from 10% to 30% would not affect the performance, as shown in the experiment.

Computational complexity. Given the same settings of t trees, each tree uses s samples, $n_{min} = 1$, and $h_{max} = \log_2(s)$, TransForest and iForest share the same asymptotically linear running time. Though both have $O(ts \log_2(s))$ training time, TransForest runs slower than iForest due to the additional $O(ks)$ cost of constructing the histogram and pseudo-labeling for feature selection. In the testing phase, their empirical running time is similar to $O(nt \log_2(s))$ time.

Feature importance ranking of TransForest. Given pseudo-labeled and labeled points and the information gain computed in each node while constructing each tree, TransForest computes an important feature ranking similar to supervised tree-based RF and ET. Note that TransForest can compute the feature importance ranking without any labeled points by applying the common unsupervised prior. The feature important ranking will facilitate decision-marking on detected anomalies and identify irrelevant features from the data set. Empirically, TransForest with few labeled points can offer the feature important ranking consistent with supervised RF and ET on low-dimensional data sets.

5 Experiments

We implement our TransForest in Python and compare its performance with other unsupervised models, including iForest [17], OCSVM [18] and HBOS [9], and with recent semi-supervised models, including XGBOD [30], DevNet [22], DeepSAD [26] and Hybrid iForest [19]. Regarding the recent benchmark ADBench [12], these approaches are the strongest competitors in unsupervised and semi-supervised anomaly detection. We conduct experiments on a 2.90 GHz core i7-10700 16GB of RAM with a single CPU.

We use the standard AUC score (i.e. area under the ROC curve) to evaluate the accuracy of unsupervised and semi-supervised ensemble detectors since they output the outlier rankings. All results of each algorithm are the average over 5 runs. Semi-supervised models take labeled points randomly for each run. We present empirical evaluations on real-world data sets, including tabular and continuous data sets from computer vision and natural language processing domains, with a variety of differences in sizes and dimensions from ADBench [1] (see Table 1) to verify our claims, including:

1. Given limited label information, TransForest outperforms XGBOD, Hybrid iForest, and recent deep learning approaches, including DevNet and DeepSAD, regarding detection accuracy.
2. Given just 20 labeled points, TransForest improves up to 10% AUC score compared to strong unsupervised baselines, including iForest and HBOS.
3. TransForest offers a feature importance ranking consistent with supervised RF and ET. TransForest is also robust to noisy and corrupted data sets while other deep learning-based approaches are not.

[1] https://github.com/Minqi824/ADBench/tree/main/datasets

5.1 Semi-supervised comparisons

We compare the AUC scores provided by TransForest and other semi-supervised approaches, including XGBOD, DeepSAD, DevNet, and Hybrid iForest.

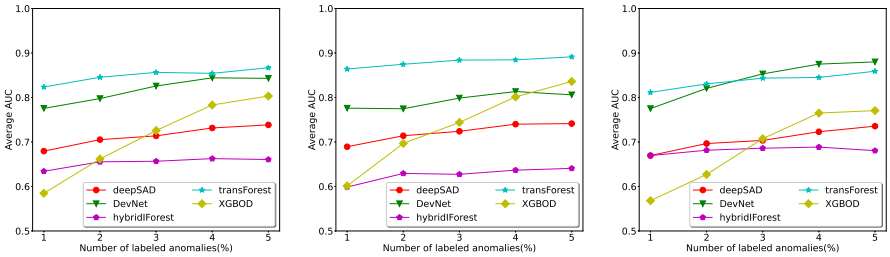
Parameter settings. TransForest uses $t = 100$ trees, $s = \texttt{max}(256, 2n_l)$ samples, $n_{min} = 1$, $h_{max} = \log_2(s)$, $k = 10$. We train Hybrid iForest with $t = 128, s = 128$. The hyperparameters in score aggregation, the coefficient of unsupervised and supervised scores $\alpha_1 = 0.2, \alpha_2 = 0.7$ are set as suggested in [19]. We use the DevNet implementation [22] with the setting $n_{epochs} = 50$, batch size $b = 512$, $\alpha = 5$, and 20 steps. Keras's RMSprop optimization has a learning rate of 0.001 and $\rho = 0.95$. The pre-trained AutoEncoder uses $n_{epochs} = 100$, $b = 128$. Adam optimization with a learning rate of 0.001 and weight decay $L^2 = 10^{-6}$ is applied in both training and pre-training stages. For XGBOD and DeepSAD, we follow the parameter settings from ADBench [12]. XGBOD uses default parameters with unsupervised estimators, including KNN, LOF, HBOS, OCSVM and iForest, and XGBoost, with a learning rate of 0.1. DeepSAD uses $n_{epochs} = 50$ with batch size $b = 128$.

Limited label information. Due to the difficulty of labeling training data, we limit the availability of labeled anomalies to $\{1\%, 2\%, \ldots, 10\%\}$ and use the same number of labeled normal points. In other words, we have $n_l/2$ labeled points from each class. We select labeled points randomly, train semi-supervised detectors on labeled and unlabeled points, and test on *all* unlabeled points.

Very limited labeled points on both classes. Figure 2 shows the average AUC scores provided by DevNet, DeepSAD, XGBOD, Hybrid iForest, and TransForest over a wide range of numbers of known anomalies. Overall, TransForest provides

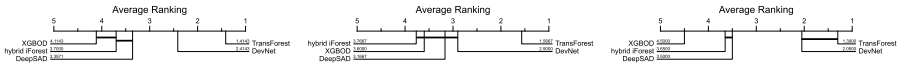
Table 1: Dataset descriptions: 15 tabular and 20 continuous data sets.

Dataset	Size	Dimension	Anomalies
ALOI	50000	27	1508 (3.02%)
Annthyroid	7200	6	534 (7.42%)
Breastw	683	9	239 (35%)
Cardio	1831	21	176 (9.6%)
Letter	1600	32	100 (6.25%)
Mammography	11183	6	260 (2.32%)
Mnist	7603	100	700 (9.2%)
Optdigits	5216	64	150 (3%)
Pendigits	6870	16	156 (2.27%)
Pima	768	8	268 (35%)
Satellite	6435	36	2036 (32%)
Satimage-2	5803	36	71 (1.2%)
Shuttle	49097	9	3511 (7%)
Speech	3686	400	61 (1.65%)
Thyroid	3772	6	93 (2.5%)
20news (5 versions)	3090	768	154 (5%)
agnews (5 versions)	10000	768	500 (5%)
FashionMNIST (10 versions)	6315	512	315 (5%)



(a) All datasets. (b) Tabular datasets. (c) CV & NLP datasets.

Fig. 2: Average AUC over different types of data with various percentages of labeled anomalies. We use the same amount of labeled points for both classes.



(a) All data sets. (b) Tabular data sets. (c) CV & NLP data sets.

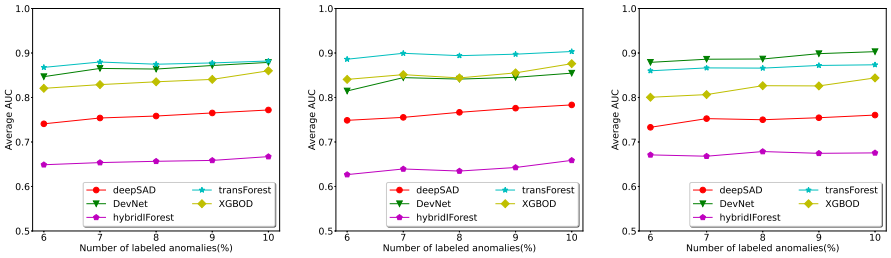
Fig. 3: Critical difference in average ranking of AUC scores over different types of data using 1% labeled anomalies. We use the same amount of labeled points for both classes.

higher accuracy than other competitors, especially on tabular data sets. The advantage of TransForest is significant when using less label information. While DevNet performs best in continuous data sets, TransForest offers a competitive performance by achieving a better accuracy with up to 3% labeled anomalies.

Compared to XGBOD, the advantage of TransForest is significant when using less label information and on continuous data sets. Since XGBOD uses unsupervised outlier scores to enrich feature learning during training, it requires more labeled points to achieve reasonable accuracy. Similarly, deep learning approaches require significant labeled data on tabular data sets. Hybrid iForest shows inferior performance. Although it incorporates labeled and unlabeled points in training, its detection ability is emphasized for clustered anomalies, and therefore benefits less from little label information.

Figure 3 shows the critical difference diagram in average rank of AUC scores of the 5 semi-supervised approaches. While XGBOD, DeepSAD, and Hybrid iForest perform similarly, TransForest and DevNet are statistically better over all data sets when the label information is limited. Compared to DevNet, TransForest's ranking is significant on tabular data sets but is marginal on continuous ones.

More available labeled points. Figure 4 shows the increase in accuracy of all semi-supervised methods when using more labeled points. Though TransForest still offers superior performance over all (15 tabular + 20 continuous) data sets, its gap with DevNet is marginal since DevNet is well-designed for continuous data sets. Nevertheless, TransForest is still competitive with DeepSAD, XGBOD, and Hybric iForest on both tabular and continuous data sets.



(a) All data sets. (b) Tabular data sets. (c) CV & NLP data sets.

Fig. 4: Average AUC over different types of data with various numbers of labeled anomalies. We use the same amount of labeled points for both classes.

5.2 Effects of pseudo-labeling of TransForest

This subsection evaluates the effects of pseudo-labeling by comparing TransForest to several representative unsupervised methods, including iForest, OCSVM, and HBOS. To demonstrate the effectiveness of TransForest on utilizing label information, we vary both numbers of abnormal and normal points from 1 to 10. For other parameter settings of TransForest, we use the same as described above. For unsupervised methods, we use the default parameter setting provided in PyOD [31], and the whole data set as the training and testing sets.

Figure 5 presents the average AUC provided by iForest, OCSVM, HBOS and TransForest on tabular data sets. It is clear that TransForest significantly improves the unsupervised approach while leveraging limited label information. Given only 3 abnormal and 3 normal points, TransForest improves nearly 7% detection accuracy over iForest and HBOS, and 20% over OCSVM.

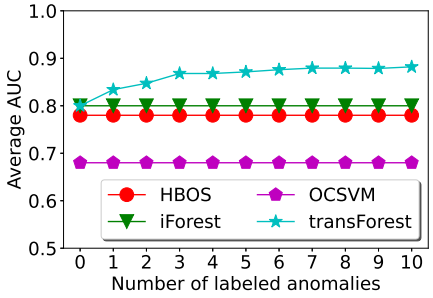
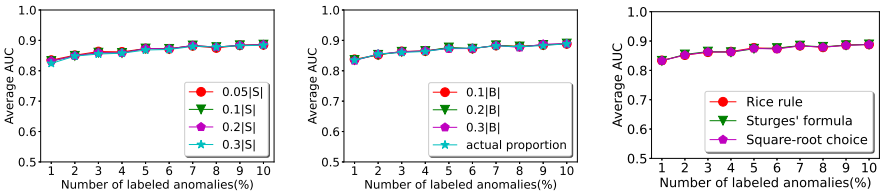


Fig. 5: Average AUC compared with unsupervised methods. We use the same amount of labeled points for both classes.



(a) Density threshold. (b) Anomaly ratio. (c) Number of bins.

Fig. 6: Average AUC over all data sets with different parameter configurations.

5.3 Parameter sensitivity of TransForest

We examine the sensitivity of core hyperparameters of TransForest, including the density threshold Δ, anomaly ratio in Rule 2, and the number of bins for constructing the histogram. We examine $\Delta = \{0.05, 0.1, 0.2, 0.3\}|S|$, the ratio of anomalies in dense areas with only labeled anomalies $\{0.1, 0.2, 0.3\}|B|$, and the actual class imbalance ratio of the data sets. For the number of bins, we use 3 popular recommendations, including square-root choice $\sqrt{|S|}$, Sturges' formula $\log_2(|S|) + 1$, and Rice rule $2\sqrt[3]{|S|}$ [29].

Fig 6 shows the stable performance by TransForest with different settings of hyperparameters over all data sets for a wide range of labeled points. In other words, TransForest does not need heavy parameter tuning.

5.4 Robustness against irrelevant features

This subsection evaluates the robustness of semi-supervised approaches with noisy features. Following ADBench [12], we add irrelevant features up to 50% of the original features. We select a few representative data sets from a wide

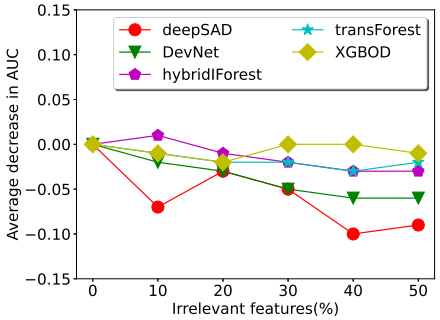
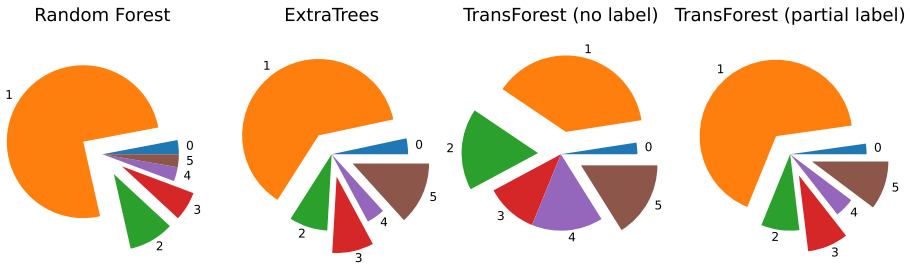


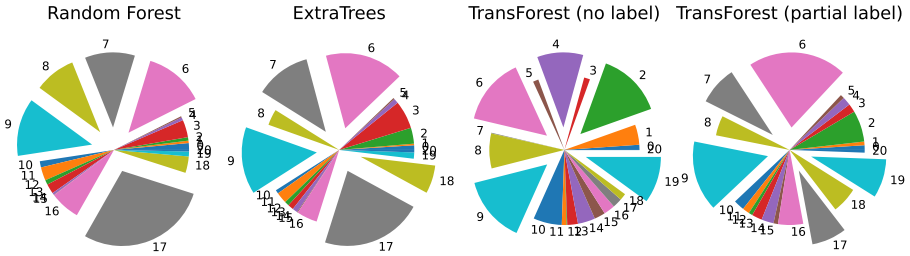
Fig. 7: Average decrease in AUC score with additional irrelevant features. We use 10% labeled anomalies and the same amount of labeled points for both classes.

range of dimensions, including Satellite, Mnist and 20news_0 to test the impact of irrelevant features. Similar to the previous setting, we use 10% labeled anomalies and use the same labeled points for each class. We vary the percentage of additional irrelevant features from 10% up to 50%. To generate an additional noisy feature, we randomly select a feature f_i, get the minimum and maximum values of f_i to generate the uniform noise from this range, and augment this noise feature into the original data.

Figure 7 shows the decrease in AUC scores of 5 detectors, including DevNet, TransForest, XGBOD, Hybrid iForest and DeepSAD when increasing the number of irrelevant features. Deep learning approaches, DevNet and DeepSAD, suffer a substantial downgrade in accuracy when the number of irrelevant features rises. With augmented 50% irrelevant features, their AUC scores are reduced by 5% and 8%, respectively. XGBOD performs superior due to its built-in feature selection to select the best splitting value, while semi-supervised forest variants show less but reasonable robustness against irrelevant features. TransForest selects the local-optimal feature and its splitting value among $k = 10$ random features to compute the information gain. We note that increasing k will improve the robustness of TransForest. In contrast, Hybrid iForest randomly selects a feature and a splitting value. Nonetheless, Hybrid iForest combines the ratio between a testing point to the centroid of labeled anomaly and normal points with iForest's



(a) Annthyroid.



(b) Cardio.

Fig. 8: Feature importance ranking on Annthyroid and Cardio. The number is the dimension index, and the wedge size reflects the feature's importance.

anomaly score. By setting $\alpha_2 = 0.7$, the supervised score dominates, and the use of ratio reduces the impact of irrelevant features.

5.5 Feature importance ranking

This subsection evaluates the feature importance ranking provided by TransForest and popular supervised RF and ET on the 2 medical data sets, Annthyroid and Cardio. Note that this utility is not available in XGBOD and deep learning approaches. For RF and ET, we use the whole data sets to train.

Figure 8 shows the feature importance ranking provided by RF, ET, unsupervised TransForest, TransForest with 10 anomalies and 100 normal points on Annthyroid and Cardio. On these 2 data sets, TransForest shows consistent feature importance rankings to that of RF and ET. On Annthyroid, the feature rankings provided by ET and TransForest are almost identical though TransForest uses just 2% labeled anomalies. On the higher dimensional Cardio, TransForest with 5% labeled anomalies shares similar top-4 important features with ET and RF (features 6, 7, 9, 17).

5.6 Running time

Table2 shows the total running time of DevNet, DeepSAD, TransForest, Hybrid iForest and XGBOD on the large 3 data sets using the same settings described above. Though our Python implementation of TransForest is not well-optimized, it still runs faster than Hybrid iForest and deep learning, and slightly slower than XGBOD.

6 Conclusion

We study the semi-supervised anomaly detection with the limit of label information. This setting suits many practical applications due to the high cost of accessing the ground truth. We propose TransForest, a transductive forest, that can learn feature selection from little label information. Empirically, TransForest with 1% labeled anomalies provides 5% improvement in AUC score compared with DevNet, and up to 20% compared to other semi-supervised learning approaches. Given 10 labeled anomalies and 100 normal points, the semi-supervised TransForest offers a feature importance ranking consistent with popular supervised models on several low-dimensional data sets.

Table 2: Average running time (s).

Dataset	Mammography	Mnist	agnews_0
DevNet	16.93	17.33	27.04
DeepSAD	35.86	28.88	52.18
TransForest	11.59	16.25	28.78
XGBOD	10.22	8.82	29.21
Hybrid iForest	52.1	55.28	81.6

Ethical Statement

Since we propose a new learning model for anomaly detection, there are no ethical issues.

References

1. Aggarwal, C.C.: Outlier Analysis. Springer (2013)
2. Bercea, C.I., Wiestler, B., Rueckert, D., Albarqouni, S.: Federated disentangled representation learning for unsupervised brain anomaly detection. Nat. Mach. Intell. **4**(8), 685–695 (2022)
3. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
4. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LOF: identifying density-based local outliers. In: SIGMOD. pp. 93–104 (2000)
5. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: KDD. pp. 785–794 (2016)
6. Criminisi, A., Shotton, J., Konukoglu, E.: Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. Found. Trends Comput. Graph. Vis. **7**(2-3), 81–227 (2012)
7. Dou, Q., et al.: Federated deep learning for detecting COVID-19 lung abnormalities in CT: a privacy-preserving multinational validation study. npj Digit. Med. **4**(60) (2021)
8. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Machine learning **63**(1), 3–42 (2006)
9. Goldstein, M., Dengel, A.: Histogram-based outlier score (HBOS): A fast unsupervised anomaly detection algorithm. KI-2012: poster and demo track **9** (2012)
10. Gopalan, P., Sharan, V., Wieder, U.: PIDForest: anomaly detection via partial identification. NeurIPS pp. 15783–15793 (2019)
11. Guha, S., Mishra, N., Roy, G., Schrijvers, O.: Robust random cut forest based anomaly detection on streams. In: ICML. pp. 2712–2721 (2016)
12. Han, S., Hu, X., Huang, H., Jiang, M., Zhao, Y.: ADBench: Anomaly detection benchmark. In: NeurIPS (2022)
13. Keller, F., Müller, E., Böhm, K.: HiCS: High contrast subspaces for density-based outlier ranking. In: ICDE. pp. 1037–1048 (2012)
14. Kriegel, H.P., Kröger, P., Schubert, E., Zimek, A.: Outlier detection in axis-parallel subspaces of high dimensional data. In: PAKDD. pp. 831–838 (2009)
15. Li, Z., Zhao, Y., Botta, N., Ionescu, C., Hu, X.: COPOD: copula-based outlier detection. In: ICDM. pp. 1118–1123 (2020)
16. Li, Z., Zhao, Y., Hu, X., Botta, N., Ionescu, C., Chen, G.: ECOD: unsupervised outlier detection using empirical cumulative distribution functions. TKDE pp. 1–1 (2022)
17. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: ICDM. pp. 413–422 (2008)
18. Manevitz, L.M., Yousef, M.: One-class svms for document classification. Journal of machine Learning research **2**, 139–154 (2001)
19. Marteau, P.F., Soheily-Khah, S., Béchet, N.: Hybrid isolation forest-application to intrusion detection. arXiv preprint arXiv:1705.03800 (2017)
20. Pang, G., Cao, L., Chen, L., Liu, H.: Learning representations of ultrahigh-dimensional data for random distance-based outlier detection. In: KDD. pp. 2041–2050 (2018)

21. Pang, G., Shen, C., Cao, L., Hengel, A.V.D.: Deep learning for anomaly detection: A review. ACM Computing Surveys (CSUR) **54**(2), 1–38 (2021)
22. Pang, G., Shen, C., van den Hengel, A.: Deep anomaly detection with deviation networks. In: KDD. pp. 353–362 (2019)
23. Pevnỳ, T.: LODA: lightweight on-line detector of anomalies. Machine Learning **102**(2), 275–304 (2016)
24. Ramaswamy, S., Rastogi, R., Shim, K.: Efficient algorithms for mining outliers from large data sets. In: SIGMOD. pp. 427–438 (2000)
25. Ruff, L., Kauffmann, J.R., Vandermeulen, R.A., Montavon, G., Samek, W., Kloft, M., Dietterich, T.G., Müller, K.R.: A unifying review of deep and shallow anomaly detection. Proceedings of the IEEE (2021)
26. Ruff, L., Vandermeulen, R.A., Görnitz, N., Binder, A., Müller, E., Müller, K., Kloft, M.: Deep semi-supervised anomaly detection. In: ICLR (2020)
27. Sathe, S., Aggarwal, C.C.: Subspace histograms for outlier detection in linear time. Knowl. Inf. Syst. **56**(3), 691–715 (2018)
28. Schubert, E., Zimek, A., Kriegel, H.: Generalized outlier detection with flexible kernel density estimates. In: SDM. pp. 542–550 (2014)
29. Scott, D.W.: Multivariate Density Estimation: Theory, Practice, and Visualization. Wiley Series in Probability and Statistics, Wiley (1992)
30. Zhao, Y., Hryniewicki, M.K.: XGBOD: improving supervised outlier detection with unsupervised representation learning. In: IJCNN. pp. 1–8 (2018)
31. Zhao, Y., Nasrullah, Z., Li, Z.: PyOD: a python toolbox for scalable outlier detection. JMLR **20**, 1–7 (2019)