AutoOD: Automatic Outlier Detection

Paper ID: 79

ABSTRACT

Outlier detection is critical in enterprises. Due to the existence of many outlier detection techniques which often return different results for the same data set, the users have to address the problem of determining which among these techniques is the best suited for their task and tune its parameters. This is particularly challenging in the unsupervised setting, where *no labels* are available for cross-validation needed for such method and parameter optimization. In this work, we propose AutoOD which uses the existing unsupervised detection techniques to automatically produce high quality outliers without any human tuning. AutoOD's fundamentally new strategy *unifies the merits* of unsupervised outlier detection and supervised classification within one integrated solution. It automatically tests a diverse set of unsupervised outlier techniques on a target data set, extracts useful signals from their combined detection results to reliably capture key differences between outliers and inliers. It then uses these signals to produce a "custom anomaly classifier" to classify anomalies, with its accuracy comparable to supervised outlier classification models trained with ground truth labels – without having access to the much needed labels. On a diverse set of benchmark outlier detection datasets, AutoOD consistently outperforms the best unsupervised outlier detector selected from hundreds of detectors. It also outperforms other tuning-free approaches we adapted to this unsupervised outlier setting from 12 to 97 points (out of 100) in the F-1 score.

1 INTRODUCTION

As data volumes continue to grow with the rise of social networks, digital currency, smart phones, connected vehicles, and other devices, there is an increasing need for techniques to support the discovery of outliers in data. Outliers correspond to rare items, events or observations which differ significantly from the majority of the data [3] and may indicate a problem, such as fraud, malfunctioning devices, or future catastrophic failures. With financial fraud causing a multi-billion dollar loss to global economy each year [7] and Internet of Things (IoT) applications from fleet management, security surveillance to inventory management predicted by Forrester to become a many trillion dollar market in the next 15 years [1], the need for effective outlier detection technology is abundant.

This has led to a significant surge in developing *outlier detection* techniques [3] over the past decade. They include fitting data to a statistical distribution and highlighting values far from the mean or median [10, 11], clustering data and identifying values outside common clusters [33], and discovering objects far from their neighbors [18, 67]. While previous research has resulted in a plethora of algorithms for detecting particular types of outliers, there are still challenging problems that hinder these algorithms from being useful in real applications by practitioners.

State-of-the-art and Its Limitations. One critical challenge is how to choose the most effective solution from this stew of available techniques and tune their parameters [3]. Users face several problems: first, no single algorithm adequately captures outliers across diverse data sets and problem domains. Thus customized algorithms have been proposed targeting different settings [3]. An outlier detection method that works well on one data set might yield poor results on another data set. Selecting a method appropriate to the given task is challenging for domain scientists, requiring not only thorough domain understanding, familiarity with the data at hand and knowledge about the most critical differences between outliers and inliers, but also a good understanding of the wealth of available outlier detection methods and their characteristics. This complexity is compounded by the fact that often the data characteristics for which certain algorithms work well are not known or properly documented.

Second, choosing the best outlier detection method is further complicated by the fact that many such methods are governed by a number of parameters. Without appropriately tuned parameter settings, detection algorithms tend to be ineffective at identifying outliers [21]. Although automatic parameter tuning methods have been proposed in Automated Machine Learning (AutoML) for supervised classification [25, 32, 61], these techniques are not adequate for solving the parameter tuning problem in the context of outlier detection. This is because outliers are rare events, making it hard to manually acquire a sufficient number of high quality outlier examples (labels) required for supervised learning. This is one reason why outlier detection techniques tend to be *unsupervised* [3]. Unfortunately, labels are required by the state-of-the-art AutoML methods [25, 32, 61] for automatic cross validation. This renders AutoML ineffective at tuning these unsupervised outlier detection methods.

Proposed Approach. To solve the above problems, we propose a automatic outlier detection approach (AutoOD). AutoOD is *not* a new outlier detection algorithm – instead it is a tuning-free approach that aims to best use existing outlier detection algorithms yet without requiring human labeling input. Our key intuition is that selecting one model from many alternate unsupervised anomaly detection models may not always work well. Instead, AutoOD targets combining the best of them.

The AutoOD Strategy. AutoOD uses a fundamentally new strategy that unifies the merits of unsupervised outlier detection techniques and supervised classification models. Unsupervised outlier detection does not require labeled data, but the accuracy of unsupervised techniques is often low due to the lack of supervision with domain knowledge [16, 19]. Compared to unsupervised techniques, supervised classification tends to achieve better accuracy, as long as a sufficient number of high quality labels are available [3].

Instead of first carefully selecting an appropriate outlier detection method for a given task and then tuning its parameters, AutoOD turns the unsupervised problem into a supervised problem. Specifically, it uses the results from many unsupervised outlier detectors in combination to *automatically* infer high quality labels by discovering objects from the input data that can reliably be detected as an outlier or an inlier. Using these automatically generated labels, AutoOD then trains a *supervised classification* model that makes inference on the remaining objects to produce the final outlier detection results.

In this way, AutoOD leverages supervised classification to achieve high accuracy in outlier detection, while no longer having to rely on domain experts to manually supply labels.

Methods to Automatically Produce Labels. The number and quality of the automatically produced labels are critical to AutoOD's effectiveness. To this end, we design two complementary methods, *AutoOD-Augment* and *AutoOD-Clean*.

AutoOD-Augment starts with discovering a small but reliable set of labels based on the strong consensus of *all* unsupervised outlier detectors. Next, leveraging the outlierness scores these detectors assign to each data object, AutoOD-Augment forms a feature space where each attribute represents one detector. It uses the set of reliable labels acquired so far and the outlierness score features to learn a machine learning model that predicts the performance of each detector for the data at hand. Leveraging this model, AutoOD-Augment iteratively prunes 'bad' detectors. AutoOD-Augment then continues to produce more labels based on the agreement among the predictions produced by the remaining 'good' outlier detectors. This way, the labels set is progressively augmented.

In contrast, AutoOD-Clean starts with a large but noisy set of labels and uses it to train a neural network. This set of labels could be formed by, for example, ensembling the results of all detectors. Leveraging the observation that the training loss on "correctly labeled" objects tends to be larger than that on any "mislabeled" objects in early epochs of training a deep neural network [62], AutoOD-Clean iteratively purifies the training data through its learning process, resulting in the removal of the objects with large early loss from the training data. As proven in Sec. 5.2, a deep neural network enhanced with this proposed self-cleaning strategy is guaranteed to converge. Further, our experimental study demonstrates that the resulting classification model learned by AutoOD-Clean shows high accuracy. AutoOD-Clean *complements* AutoOD-Augment, especially when it is hard to get initial reliable labels.

Experimental Results. We demonstrate the effectiveness of AutoOD using a variety of benchmark outlier detection data sets [16, 60]. In particular, as we show in Sec. 6.2, AutoOD consistently detects outliers with an accuracy higher than the *best outlier detector* among hundreds. Of note, AutoOD is able to do this without requiring any manual tuning nor human input in terms of pre-determined labels. Further, AutoOD significantly outperforms ensemble-based methods [55, 58] and other tuning-free approaches by 12 to 97 points (out of 100) in the F-1 score.

Contributions. In summary, key contributions of this work include:

- We propose AutoOD that uses a set of *unsupervised outlier detectors* to automatically produce high quality outliers, requiring zero human input nor ground-truth labels.
- AutoOD unifies the merits of unsupervised outlier detection and supervised classification, achieving a high accuracy in detecting outliers, while requiring zero human input nor ground-truth labels.
- We propose two complementary solutions, AutoOD-Augment and AutoOD-Clean, to realize the AutoOD framework, making AutoOD highly effective and robust in a variety of scenarios. Our theoretical analysis show that AutoOD-Augment and AutoOD-Clean are guaranteed to converge to a set of high quality labels.
- Our experiments on benchmark outlier detection datasets show that AutoOD consistently outperforms the best outlier detector among all candidate detectors. In fact, it achieves an accuracy comparable to supervised outlier classifiers trained with ground truth labels – without having the access and thus benefit of such ground truth knowledge!

2 PRELIMINARIES

Below, we briefly overview popular outlier detection techniques [3] including statistical-based outlier detection [3], distance-based outlier detection [35, 54], density-based outlier detection [15, 48], and Isolation Forest [44]. AutoOD supports these techniques as build-in libraries, although other techniques could also simply be plugged in.

Statistical-based Outlier Detection. Statistical-based methods detect outliers by discovering extreme values. In particular, the Mahalanobis method [3] models the entire dataset to be normally distributed around its mean in the form of a multivariate Gaussian distribution. Let $\overline{\mu}$ be the d-dimensional mean vector of a d-dimensional dataset, and Σ be its $d \times d$ covariance matrix. In this case, the (i, j)th entry of the covariance matrix is equal to the covariance between the dimensions i and j. Then the mahalanobis distance from a d-dimensional data object $\overline{X}$ to this distribution can be defined as:

$$Mahalanobis(\overline{X}, \overline{\mu}, \overline{\Sigma}) = \sqrt{(\overline{X} - \overline{\mu})\Sigma^{-1}(\overline{X} - \overline{\mu})^T} \quad (1)$$

The Mahalanobis distance is used as outlier score. Meaning, the larger the Mahalanobis distance is, the more the object deviates from the data set distribution and thus more likely is an outlier.

Distance-based Outlier Detection. Distance-based outlier detection computes outlier scores on the basis of nearest neighbor distances. Among many variations [9, 36, 54], the kNN outlier [54] is very popular. Let $D^k(p)$ denote the distance of object p from its k-th nearest neighbor. The kNN outlier ranks objects based on their $D^k(p)$ distance. The top n objects in this ranking are then considered to be outliers. These objects have fewer points close to them and are thus intuitively stronger outliers.

Density-based Outlier Detection. Density-based approaches consider ratios between the local density around an object and the local density around its neighboring objects. These approaches introduce the notion of *local outliers* as opposed to the *global outliers* discovered by distance-based outlier techniques. The concept of a local outlier is important since in many applications, different portions of a dataset can exhibit very different characteristics. It is thus meaningful to decide on the outlying possibility of an object based on other objects in its neighborhood. The most popular density-based outlier approach [15], called LOF, is to assign a local outlier factor (LOF) to each object of the dataset denoting its degree of outlierness. In general, LOF corresponds to the average of the ratio of the local density of an object p and those of p's k-nearest-neighbors. Intuitively, p's local outlier factor will be very high if its local density is much lower than those of its neighbors.

Isolation Forest. An isolation forest is an ensemble of a set of isolation trees [44]. In an isolation tree, the data is recursively partitioned with axis-parallel cuts at randomly chosen partition objects within randomly selected attributes. The aim is to isolate the instances into nodes with fewer and fewer instances until the objects are isolated into singleton nodes containing one instance only. In such cases, the tree branches containing outliers are noticeably less deep, because these objects are located in sparse regions. Here thus the distance of

the leaf to the root is used as the outlier score. The final outlier score then is computed by averaging the path lengths of the objects in the different trees of the isolation forest.

3 OVERVIEW OF THE AUTOOD FRAMEWORK

In this section, we introduce the overall design of AutoOD.

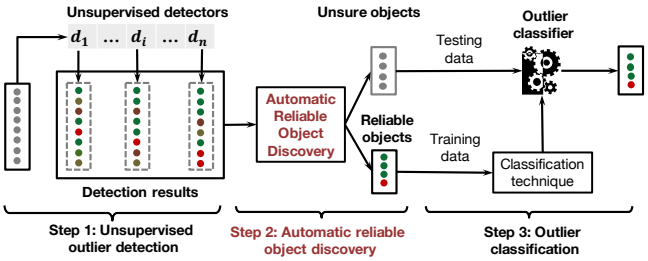

Figure 1: AutoOD Framework

3.1 Fundamentals Underlying AutoOD

AutoOD Strategy. AutoOD uses a fundamentally new strategy to solve the method selection and parameter tuning problem in unsupervised outlier detection. This strategy unifies the merits of unsupervised outlier detection and supervised classification. Namely, unsupervised methods do not rely on a human expert to supply labels about ground truth anomalies – which often are hard to come by. However, the accuracy of unsupervised outlier detection is often low [16] due to the lack of human supervision. On the other hand, supervised classification techniques are able to achieve higher accuracy by training a binary classifier that classifies the objects into outliers or inliers. However, they rely on the availability of a sufficient number of high quality labels for training [21].

Instead of selecting an appropriate outlier detection method and then tuning its parameters to get good results, AutoOD uses a large set of diverse unsupervised outlier detectors as labeling sources to automatically produce high quality labels, where an outlier detector corresponds to one specific unsupervised outlier detection method instantiated with a particular parameter setting. We then use these automatically produced labels to train a classification model to finally classify each object as being either an outlier or an inlier. As we will demonstrate in Sec. 6.2, this allows AutoOD to achieve high accuracy in detecting outliers, without having to rely on human experts to supply high-quality labels. AutoOD thus captures the benefits of both unsupervised and supervised outlier detection.

AutoOD Intuition. Without any manually supplied labels indicating ground truth, it appears extremely difficult to automatically discover the best unsupervised outlier detector among many alternatives. Even if it were possible to identify the best one, it would not be guaranteed to detect outliers with high accuracy. In fact, among all detectors there is at times no one *clear winner* that dominates all the other detectors. Given a diverse set of unsupervised detectors, each detector might discover some anomalies which other detectors would miss.

However, we observe that there typically tend to be some objects in the *data* that are *clear outliers and inliers*. Clear outliers, for example, often correspond to the objects that are well isolated from other objects, while clear inliers often correspond to objects residing deeply inside dense data clusters. Our intuition is that it is to be much easier to automatically identify such clear outliers or clear inliers compared to identifying the best overall detector method itself. Using these objects as reliable labels that characterize key differences between outliers and inliers, AutoOD thereby can fully explore the generalization ability of supervised machine learning [30] to learn a classification boundary that effectively infers the status of the remaining *unsure objects*.

3.2 Components of the AutoOD Framework

Fig. 1 depicts the overall AutoOD Framework. AutoOD is composed of three key components, including *unsupervised outlier detection*, *automatic reliable object discovery*, and *outlier classification*.

(1) Unsupervised Outlier Detection.

Given an input data set, AutoOD first uses a set of unsupervised outlier detectors to detect outliers. Each detector corresponds to one outlier detection method in the built-in AutoOD library with a particular configuration of parameter values instantiated. For simplicity and ease of use, for each detection method, AutoOD uniformly picks some parameter configurations from a reasonable parameter range recommended by [16].

(2) Automatic Reliable Object Discovery. Next, based on these unsupervised detection results, AutoOD divides the input data into two subsets, "reliable objects" and "unsure objects". The reliable objects are those which AutoOD is confident are clearly inliers or outliers based on the detection results produced by the detectors.

(3) Outlier Classification. Finally, the automatically discovered reliable objects are used as training data for a binary outlier classification model. This model then produces predictions for the "unsure" objects whose labels remain to be unknown. This way, eventually our AutoOD assigns labels to all objects.

AutoOD-Augment and AutoOD-Clean. Clearly, the effectiveness of AutoOD relies on the number and quality of the reliable objects discovered in the second step and used as labeled training data thereafter. In this work, we design two approaches to discover these reliable objects that **complement** each other. First, we propose an augmentation-based method, called AutoOD-Augment (Sec. 4). AutoOD-Augment starts by automatically discovering a small but reliable set of objects to label and keeps augmenting this set iteratively. Second, we propose a cleaning-based method, called AutoOD-Clean (Sec. 5). As opposed to AutoOD-Augment, AutoOD-Clean starts with a large set of noisy labels and keeps cleaning this set into an increasingly reliable set. In the next two sections, we introduce the two approaches in detail.

4 AUGMENTATION-BASED RELIABLE OBJECT DISCOVERY

AutoOD-Augment starts with discovering a small set of reliable objects and then *iteratively augments* this set until no new reliable objects can be found.

4.1 The Overview of AutoOD-Augment

Next, we overview the AutoOD-Augment approach that includes three key components, namely *initial reliable object discovery*, *learning-based poor detector pruning*, and *reliable object set update*.

(1) Initial Reliable Object Discovery. AutoOD-Augment identifies an initial label set of *stable* outliers/inliers using the strategy described below. In AutoOD, an object is considered to be reliably-decidable about its label status, if all unsupervised outlier detectors agree on its (outlier/inlier) label status. We call these the *stable objects*. The stable objects typically correspond to the clear inliers and outliers in the data. The intuition is that although different outlier detection methods use distinct techniques to detect outliers, they are based on the same principle. That is, an object is an outlier if it deviates significantly from the other observations [31]. Therefore, they all tend to be good at capturing the clear inliers that are deeply resided inside of some big data clusters and the clear outliers that are far way from any other objects.

(2) Learning-based Poor Detector Pruning. Second, treating the stable outliers/inliers as reliable objects and hence ground truth labels, AutoOD trains a machine learning model to estimate the performance of the unsupervised detectors. Progressively pruning the bad detectors enables AutoOD to collect more and more stable inliers and outliers, thus gradually discovering more reliable objects. Note although AutoOD estimates the performance of each detector, the ultimate goal is not to find the best detector, but instead it is to automatically discover more reliable objects.

(3) Reliable Object Set Update. AutoOD-Augment leverages the concept of multi-view analysis [26] to update the reliable objects. That is, AutoOD learns multiple distinct outlier classification models that are trained on the same set of labels but that use different sets of features. Features we can leverage here include not only the attributes of the data itself, but also the intermediate results produced by the outlier detectors. The intuition is that if the classification models learned from different views of the data agree with each other on the prediction of some objects, potentially these objects are reliable objects. On the other hand, the objects on which the classification models have conflicting predictions will be removed from the set of reliable objects to purify the automatically produced labels.

Algorithm 1 AutoOD-Augment

```
1:  function AutoODAugment(Detectors, Scores, X)
2:      prevIds = []
3:      Ids, D_r = initReliableObj(Scores)
4:      counter = 0
5:      while True do
6:          Scores = PruneDetector(Detectors, Scores, Ids, D_r)
7:          Ids, D_r = UpdateReliableObj(Scores, Ids, D_r)
8:          if prevIds == Ids then
9:              break
10:         prevIds = Ids
11:     return Ids, D_r
```

Alg. 1 illustrate the overall process of AutoOD-Augment. First, n different outlier detectors $d_1, d_2, \ldots, d_n$ generate a list of outlierness scores $s_1, s_2, \ldots, s_n$ for each object p_i, respectively. Based on these outlierness scores, AutoOD-Augment puts together its initial set of reliable objects $\mathbb{D}_r$ (Line 3, Alg. 1). Our learning-based detector pruning strategy then is applied to discover and prune the poor detectors (Line 6, Alg. 1). Based on the remaining outlier detectors, AutoOD-Augment leverages the multi-view outlier classification to update reliable objects $\mathbb{D}_r$ (Line 7, Alg. 1). The new $\mathbb{D}_r$ is used in the next iteration as training data to further prune the poor detectors. In this process, $\mathbb{D}_r$ gets larger and more accurate. AuotoOD-Augment terminates when $\mathbb{D}_r$ does not change any more (Lines 8–9, Alg. 1).

Next, we introduce the three key components of AutoOD-Augment in more detail, namely *initial reliable object discovery*, *learning-based poor detector pruning*, and *reliable object set update*.

4.2 Initial Reliable Object Discovery

AutoOD-Augment starts by identifying stable inliers and stable outliers as initial set of what we call "reliable objects". Because outliers typically correspond to only a very small fraction of the data set, in some cases AutoOD may not be able to identify stable outliers in this initial set – especially when handling small datasets. To solve this issue, we *relax* the strict requirement of stable outliers when necessary. The first relaxation is if all detectors that use the same detection method M_i but different parameter settings $pt_1 \ldots pt_m$ mark an object as outlier, then we will consider this object to be a stable outlier. For example, suppose AutoOD uses two outlier detection methods LOF [15] and Isolation Forest [44], and there is no stable outlier based on the strict stable outlier requirement. In this case, an object will be considered as a stable outlier if all detectors using LOF believe it is an outlier. While this or other relaxations we may explore in the future may introduce errors into the set of reliable objects, our AutoOD-Augment method is designed to fix these errors using an iterative learning process described in Sec. 4.4.

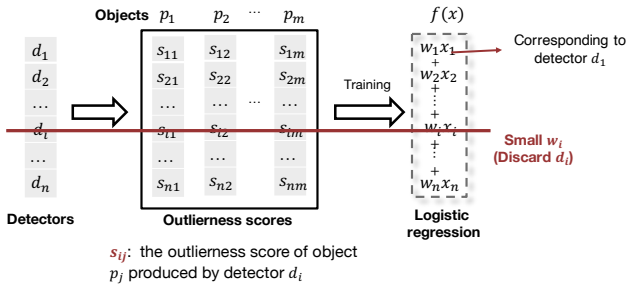

Figure 2: AutoOD-Augment: Prune Poor Detectors

4.3 Learning-based Poor Detector Pruning

One principle of AutoOD-Augment is to tackle the problem of pruning poor detectors using machine learning, more specifically logistic regression. In this work, leveraging the unique properties of the logistic regression model in conjunction of unsupervised outlier detection AutoOD-Augment effectively discover poor detectors with a *theoretical guarantee*.

Logistic regression (LR) [34] is a classical binary classification model that predicts the probabilities of possible outputs. Consider a single input observation x, represented by a vector of features $[x_1, x_2, \ldots, x_n]$. The classifier consuming x then outputs outcome y, with $y \in \{0, 1\}$ with 1 meaning the observation is a member of the class C and 0 the observation is not a member of the class C. We want to know the probability $P(y = 1|x)$ that the observation x is a member of the class C. In the outlier detection case, $P(y = 1|x)$ represents the probability that x is an outlier, while $P(y = 0|x)$ represents the probability that x is an inlier.

Given a training set of objects with class labels, logistic regression (LR) solves this classification problem by learning a vector of **weights** $w = [w_1, w_2, \ldots, w_n]$ and a bias term. Each weight w_i is

a real number, and is associated with one of the input features x_i. Formally, the LR model is defined as:

$$log\frac{P(y=1|x)}{1-P(y=1|x)} = w \cdot x + b \tag{2}$$

where w denotes the weight vector and b the bias term. Solving for $P(y=1|x)$, this gives:

$$P(y=1|x;b,w) = \frac{e^{w\cdot x+b}}{1+e^{w\cdot x+b}} = \frac{1}{1+e^{-(w\cdot x+b)}} \tag{3}$$

The weights and bias of the LR model are estimated from the training data. After we have learned the weights through the training process, the probabilities for each class can be computed by Eq. 3. If P(y=1|x; b,w) $\gg$ 0.5, the object is classified as an outlier. Alternatively, if P(y=1|x; b,w) is close to 0, the object is an inlier. On the other hand, if P(y=1|x; b,w) is around 0.5, the LR model is not certain enough about the status of the given object.

Outlier Detector Pruning With Logistic Regression. The outlier detector d_i produces an outlierness score $s_{i,j}$ for each object p_j in the dataset. As a real number, the larger the $s_{i,j}$, the more possible that d_i believes p_j is an outlier. This outlierness score can be treated as an additional derived feature x'_i of object p produced by detector d_i. Thus, in total, n outlier detectors will produce n such derived features $\{x'_1, x'_2, ..., x'_n\}$. As depicted in Fig. 2, the outlier scores $s_{i,j}$ produced by n detectors for m objects form a $n \times m$ matrix. Element $s_{i,j}$ represents the outlierness score that detector d_i assigns to object p_j. Correspondingly, the ith row corresponds to feature x'_i produced by detector d_i across all objects, while the jth column corresponds to features produced for object p_j by all the detectors.

As a preprocessing step, AutoOD normalizes the outlierness scores produced by different detectors into the same range to allow for ease of comparison. In the implementation, AutoOD uses RobustScaler in scikit-learn [50] that is known to be robust to outliers. AutoOD then uses these derived features $\{x'_1, x'_2, ..., x'_n\}$ produced by the n detectors and the set of stable labels produced in the first step to train an LR model.

As discussed above, LR assigns a weight w_i to each feature x'_i corresponding to one detector d_i. Intuitively, the weight w_i represents *how important* that input feature x'_i is to the classification decision [30]. It can be positive or negative which would mean that the feature is (or is not) associated with the outlier class, respectively. Leveraging this insight, we define the pruning rule in Def. 4.1 that prunes detectors based on the importance of their evidence to the LR model's classification decision.

DEFINITION 4.1. ***Pruning Rule.*** *Given the weights $\mathbb{WT} = \{w_1, w_2, ..., w_n\}$ corresponding to features $\{x'_1, x'_2, ..., x'_n\}$ produced by detectors $\{d_1, d_2, ..., d_n\}$, AutoOD prunes detector d_i if:*

(1) weight $w_i < 0$; or

(2) $w_i < mean(\mathbb{WT}) - std(\mathbb{WT})$ when $\forall\ w_i \in \mathbb{WT}$, $w_i > 0$.

By the pruning rule, if there are negative weights, AutoOD prunes the corresponding detectors immediately. Otherwise, AutoOD prunes the detectors whose weights are at least one standard deviation smaller than the average weight over all detectors. Our experiments in Sec. 6.6 confirm that this is effective in pruning the poor detectors.

Theoretical Guarantee. Next, we formally show in Lemma 4.1 that the detectors discarded by the pruning rule are guaranteed to *not* perform better than the remaining detectors under the *comparison criteria* defined in Def. 4.2.

DEFINITION 4.2. ***Comparison Criteria.*** *Let $\mathbb{O}$ and $\mathbb{I}$ represent an outlier set and an inlier set, respectively. We say detector d_i is better than detector d_j, if and only if:*

$$\sum_{p_k\in\mathbb{O}} s_{i,k} > \sum_{p_k\in\mathbb{O}} s_{j,k} \text{ and } \sum_{p_k\in\mathbb{I}} s_{i,k} < \sum_{p_k\in\mathbb{I}} s_{j,k} \tag{4}$$

where $s_{i,k}$ corresponds to the outlierness score which detector d_i assigns to object p_k.

Intuitively, given a set of outliers and inliers, we say detector d_i is better than detector d_j, if compared to d_j, d_i in total assigns larger outlierness scores to outliers and smaller outlierness scores to inliers.

Next, we prove Lemma 4.1.

LEMMA 4.1. *If detector d_i is better than d_j by the comparison criteria in Def. 4.1, then $w_i > w_j$.*

PROOF. Focusing on w_i and w_j, we re-write the objective function of logistic regression as:

$$\max \sum_{p_k\in\mathbb{O}} (w_i \cdot s_{i,k} + w_j \cdot s_{j,k}) - \sum_{p_k\in\mathbb{I}} (w_i \cdot s_{i,k} + w_j \cdot s_{j,k}) \tag{5}$$

We use C_i to denote $\sum_{p_k\in\mathbb{O}} s_{i,k} - \sum_{p_k\in\mathbb{I}} s_{i,k}$ and C_j to denote $\sum_{p_k\in\mathbb{O}} s_{j,k} - \sum_{p_k\in\mathbb{I}} s_{j,k}$. By Equation 4, if d_i is better than d_j, we have $C_i > C_j$. Accordingly, we re-write the objective as:

$$\max C_i w_i + C_j w_j,\ s.t.\ C_i > C_j \tag{6}$$

If we set $||w||_2^2$ to be *Const*, then using the Lagrangian multiplier method, we have:

$$f(w) = C_i w_i + C_j w_j + \lambda(w_{u_1}^2 + w_{u_2}^2) \tag{7}$$

Then setting the derivative of $f(w)$ w.r.t. w_i and w_j to zero will yield:

$$w_i = -\frac{C_i}{2\lambda},\ w_j = -\frac{C_j}{2\lambda},\ \lambda < 0 \Longrightarrow w_i > w_j \tag{8}$$

This concludes the proof. □

By Lemma 4.1, weight w_i represents the relative performance of detector d_i. This in turn justifies AutoOD's pruning rule.

Algorithm 2 Update Reliable Object Set

```
highConfThr = t, lowConfThr = 1 - t
function UpdateReliableObject(Scores, trainIds, 𝔻_r, LRModel)
    DataFtModel = Train(X[trainIds], 𝔻_r)
    DataFtPredict = DataFtModel(X)
    OutlierScorePredict = OutlierScoreModel(Scores)
    highConfIn = Intersect(DataFtPredict < lowConfThres, OutlierScorePredict < lowConfThr)
    highConfOut = Intersect(DataFtPredict > highConfThres, OutlierScorePredict > highConfThr)
    disagrees = (DataFtPredict > 0.5) != (OutlierScorePredict > 0.5)
    stableIn, stableOut = GetStable(Scores)
    inliers = Union(stableIn, highConfIn)
    outliers = Union(stableOut, highConfOut)
    inliers = Setdiff(inliers, disagrees)
    outliers = Setdiff(outliers, disagrees)
    trainIds = [inliers, outliers]
    𝔻_r = [0 {len(inliers)}, 1 {len(outliers)}]
    return trainIds, 𝔻_r
```

4.4 Reliable Object Set Update

In Alg. 2, we leverage the idea of multi-view outlier classification to update the set of reliable objects. For this, beyond training the logistic regression (LR) model on the outlierness score features (the outlier score model), AutoOD-Augment trains an additional outlier classifier using only the raw features (attributes) of the stable data set (Line 3), called the data feature model.

At testing phase, this classifier takes an object as input, and based on the similarity of its attributes to objects in the stable data set, emits a confidence that the object is an outlier (or, not an outlier). In this work, while any binary classification model could be plugged in, in our implementation we utilize a Support Vector Machine (SVM) model as our classifier. This is because it does not require careful hyper-parameter tuning and can handle high dimensional data.

The outlier score model and data feature model are used to infer two different sets of probabilities of the outlier class for *all objects* in the entire input dataset (Lines 4,5). Thereafter, AutoOD-Augment iterates over the process below using the prediction results of the outlier score and data feature models as well as the remaining high-quality detectors to discover the reliable objects. The intuition here is that the two models and the outlier detectors produce predictions for each object from different views of the data. If they are consistent on the prediction of the outlier status for one object, the label of this object is considered to be reliable.

In the update process, AutoOD-Augment first discovers the new stable outliers/inliers (Line 9). After pruning some poor detectors, the numbers of stable inliers and outliers naturally increase. Then from the prediction results of the two classification models, AutoOD-Augment identifies the confident inliers (Line 6) and outliers (Line 7). An object is a confident inlier or outlier if both classification models are very confident about their prediction of its status.

DEFINITION 4.3. ***Confident Inliers/Outliers.*** *Given an object p and some threshold t ($0 < t < 1$), if the classification model trained on the outlierness scores and the classification model trained on the raw features of data both predict p to be an inlier or outlier with a probability higher than t, then p is a confident inlier or outlier.*

By default, AutoOD sets the value of threshold t in Def. 4.3 as 0.99. This corresponds to a very strict criteria for finding confident outliers and inliers to ensure their reliability. Our experiments show this value works well in all cases (suggesting this is not a parameter that needs to be tuned).

As discussed in Sec. 4.3, the logistic regression model trained on the outlierness score features uses Equation 3 to assign each object a probability. It corresponds to the confidence of the model in its prediction of the object's class label. Naturally, we can use this probability to determine if the object is likely an outlier (or inliers). That is, an object is potentially a confident inlier/outlier if and only if $P(y = 1|x; b, w) > t$ or $P(y = 1|x; b, w) < 1 - t$.

However, standard SVMs trained on the raw features of the data do not produce a probability for each object, representing how confident SVM is in its prediction of this object's class. In this work, we use the well-known Platt scaling [49, 53] method to calibrate the probabilities. Platt scaling trains the parameters of an additional sigmoid function to map the SVM outputs into probabilities.

AutoOD-Augment then unions the current stable and the confident outliers/inliers sets into two refreshed set of reliable outliers and inliers (Lines 10-11). Moreover, AutoOD-Augment removes the objects from the reliable object set if the two classification models have conflicting predictions on these objects (Lines 12-13). This further purifies the set of reliable objects.

Termination Condition. AutoOD-Augment proceeds until the set of reliable objects does not change (Line 9 in Alg. 1). Next, we intuitively show that this process is *guaranteed to converge*. First, the detector pruning would stop, in the worst case when only one detector were to remain. Then the stable outliers and inliers thus would not change anymore. Now the update of the reliable object set is only driven by the predictions of the two classification models. The diminishing update of the set of reliable objects in turn gradually stabilizes these two models trained on it. This eventually leads to the stabilization of the set of confident inliers/outliers and the termination of AutoOD-Augment. Now the two models and the remaining unsupervised outlier detectors highly agree with each other on the predictions of the current reliable objects. Therefore, they tend to be indeed reliable.

5 AUTOOD-CLEAN: CLEANING-BASED RELIABLE OBJECT DISCOVERY

AutoOD-Clean trains a neural network to combine the predictions of the unsupervised detectors to produce a set of reliable objects. We first describe an important observation about the training process of a deep neural network which we leverage, and then we introduce the details of the AutoOD-Clean approach.

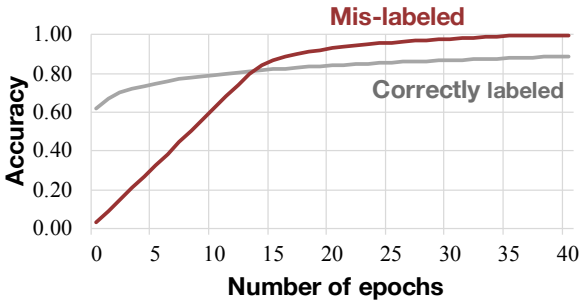

Figure 3: Observation: Evolution of Training Accuracy for Mislabeled and Correctly Labeled Samples.

Key Observation. Fig. 3 shows the evolution of model accuracy for "mislabeled" and "correctly labeled" objects as a function of training epochs when training a neural network model using a *noisy* training dataset [62]. We consider the common setup where training proceeds in epochs. We then inspect the evolution of the accuracy of the model on the training objects. That is, after each epoch, we take the model at that stage and see whether or not it makes an error for each of the training objects. Here we assume we have access to the ground truth of this noisy training dataset. As shown in Fig. 3, the accuracy on "correctly labeled" objects is higher than on the "mislabeled" objects, especially in the initial epochs of training.

Leveraging the above observation, we are now ready to propose an approach to discover reliable training data. This approach AutoOD-Clean starts with a large but noisy training dataset and iteratively cleans it. AutoOD-Clean is inspired by the theory of minimizing the **trimmed loss** [47]. Given a set of n samples, standard model-fitting involves choosing model parameters θ to minimize a loss

function over all n samples. In contrast, the trimmed loss estimator involves jointly choosing a subset of αn samples and parameters θ such that the loss on the subset is minimized over all choices of subsets and parameters. While this objective is intractable in general, AutoOD-Clean can be considered as an iterative methodology for minimizing this trimmed loss. It simultaneously generates a reliable training dataset and an effective neural network model, as iteratively minimizing the trimmed loss.

Trimmed Loss Let $\mathbb{D} = p_1, \ldots, p_n$ be the set of training objects, θ be the model parameters to be learned, and $f_\theta(.)$ be the loss function. With this setting, the standard approach is to minimize the total loss of all objects, i.e., $min_\theta \sum_i f_\theta(p_i)$. In contrast, the least trimmed loss (TL) estimator is given by:

$$\hat{\theta}^{(TL)} = arg \min_{\theta \in \mathcal{B}} \min_{\mathbb{D}:|S|=\lfloor \alpha n \rfloor} \sum_{i \in \mathbb{D}} f_\theta(p_i) \tag{9}$$

To find $\hat{\theta}^{(TL)}$, we need to minimize over both the set $\mathbb{D}$ of size $\lfloor \alpha n \rfloor$ – where $\alpha \in (0, 1)$ is the fraction of objects we want to fit – and the set of parameters θ. In general, solving for the least trimmed loss estimator is hard, even in the linear regression setting [47], i.e., even when $p = (x, y)$ and $f_\theta(x, y) = (y - \theta^T x)^2$.

5.1 AutoOD-Clean: Iterative Trimmed Loss Minimization

As described in Alg. 3, AutoOD-Clean starts with a large but noisy training dataset and alternates between training a neural network based on trimmed loss and updating the training data using the early training loss as well as the prediction results of the neural network.

The AutoOD-Clean approach is composed of three steps: *initial training data generation*, *modeling*, and *training data update*.

Algorithm 3 AutoOD-Clean

```
1:  function AutoODClean(Scores, X)
2:      labels = Ensemble(Scores)
3:      cur = [0, ..., len(labels)]                ▷ Keeps track of remaining data indexes
4:      while true do
5:          model, lossList = runNN(X[cur], labels[cur], numEpochs = 3)
6:          model = runNN(X[cur], labels[cur], numEpochs=10, model = model)
7:          predictions = Infer(model, X)
8:          cur = PruneLoss(lossList, cur)
9:          cur, labels = AddConfObjs(predictions, cur, labels)
10:         if numRmv > numAdd then
11:             break
12: return cur, labels
```

(1) Initial Training Data Generation. As shown in Alg. 3 (Line 2), AutoOD-Clean starts by using all data objects as training data. The labels of these objects can be produced by ensembling the results of multiple detectors or using the results of any detector.

(2) Modeling. During the modeling process, we train a neural network using the current training data (Lines 5–7, Alg. 3). Specifically, we first train a neural network just for a few epochs and keep track of the loss for each training object (Line 5, Alg. 3). Given an training object, its loss is measured as the difference between the prediction and its label using a typical loss function such as cross entropy [29].

AutoOD-Clean uses the loss for each object to prune mislabeled objects. That is, it leverages the observation that mislabeled objects tend to have higher loss than correctly labeled objects, especially during the early epochs [62]. This stage of the process does not require for the model to have converged; thus a handful of initial epochs is sufficient. In our experiments, we use a small network with 3 hidden layers for all datasets; and thus set the number of epochs to be 3. If using a larger network, we would set the number of epochs to be a bit larger, say 5 or 10.

After recording the *early loss* for each training data object – the *average* training loss it incurred in the early epochs, AutoOD-Clean continues to train the network until it converges (Line 6 in Alg. 3). Then AutoOD-Clean uses the converged model to make inference on the entire dataset. In the outlier detection setting, the output is a probability that can be interpreted as the confidences of the current model on the object being an outlier or a inlier. Both the early losses and the confidence scores will be used in the next step to update the training data (Lines 8-9, Alg. 3).

(3) Training Data Update. First, AutoOD-Clean removes the training points with large early losses from the training data using the *cleaning rule* defined below.

DEFINITION 5.1. ***Cleaning Rule.*** *Let $\mathbb{L}_I = \{l_1^I, l_2^I, ..., l_n^I\}$ denote the early losses of the training objects $\mathbb{I}$ that are classified as inliers and $\mathbb{L}_O = \{l_1^O, l_2^O, ..., l_n^O\}$ denote the early losses of the training objects $\mathbb{O}$ that are classified as outliers. A training object p_i with early loss l_i will be removed if:*

(1) $l_i > mean(\mathbb{L}_I) + std(\mathbb{L}_I)$ when $p_i \in \mathbb{I}$; or (2) $l_i > mean(\mathbb{L}_O) + std(\mathbb{L}_O)$ when $p_i \in \mathbb{O}$

Algorithm 4 Pruning Objects with Large Early Losses

```
1: function PruneLargeLoss(lossList, cur, labels)
2:     outliers = labels[cur] == 1
3:     lossThrOut = mean(lossList[outliers]) + std(lossList[outliers])
4:     rmvIdx = cur[outliers][lossList[outliers] > lossThrOut]
5:     inliers = labels[cur] == 0
6:     lossThrIn = mean(lossList[inliers]) + std(lossList[inliers])
7:     rmvIdx += cur[inliers][lossList[inliers] > lossThrIn)
8:     cur = cur - rmvIdx
9:     return cur
```

Alg. 4 shows the process of applying the cleaning rule to prune objects. Suppose an object is temporarily considered as an outlier (Line 2, Alg. 4). AutoOD-Clean will remove it if its loss is one standard deviation larger than the mean loss of the *outier labels*. On the other hand, AutoOD-Clean will remove an inlier if its loss is one standard deviation larger than the mean loss of the *inlier labels*. Note we use different criteria to prune mis-labeled inliers and outliers, because outliers and inliers tend to show different patterns in their early losses due to their distinct data characteristics. Our empirical study shows that the early loss of outlier labels in average is larger than inlier labels.

Fixing Erroneous Pruned Objects. AutoOD-Clean might erroneously remove some training objects due to the prediction errors of the network. This in particular may arise during early training stages, when the neural network model is trained on noisy training data and thus may possibly have low accuracy. Because the training data continuously gets improved in this iterative process, the accuracy of the neural network model is also expected to increase over time. This provides us with opportunities to fix the earlier mistakes made by adding the high confidence objects back into the training data.

Next, we describe the process of adding objects with high prediction confidence back into the training data set. We train the neural

network until it fully converges. Then we make prediction on all input data. That is, we apply the latest fully converged neural network to classify the data into outliers or inliers. If an object is not in the current training data, but it is being classified as an outlier or inlier with a confidence higher than a threshold t, we will add it to the training data. On the other hand, if the label of a current training object is different from the inference output, it will be removed from the training data even if its loss is not large. The threshold t by default is set as 0.99, corresponding to a very strict requirement on confidence level. Our experiments show it works well in all cases.

The Termination Condition. After updating the training data, AutoOD-Clean will iteratively repeat this process, that is, it will re-train the neural network in the next iteration using the new training data. This process continues until the number of removed objects is *smaller* than the number of added objects (Lines 10–11, Alg. 3). Next, we show that with this termination condition, AutoOD-Clean is guaranteed to converge to a reliable training data.

5.2 Convergence Analysis

We show the convergence of AutoOD-Clean by proving that AutoOD-Clean is able to recover the ground truth labels with a linear convergence rate. Here we use the generalized linear model to represent the neural network model which AutoOD-Clean uses. This is a common practice [62], because the problem of analyzing a general least trimmed loss estimator is intractable.

We analyze AutoOD-Clean with errors in the labels. We represent the training objects in the form of (x, y) such that:

$$\begin{aligned} y &= w(\phi(x)^T \cdot \theta^*) + e, \text{(correctly labeled objects)} \\ y &= r + e \qquad\qquad\quad \text{(mislabeled objects)} \end{aligned} \tag{10}$$

Here x represents the features of a training objects and y denotes the output, embedding function ϕ and link function w are known, e is random subgaussian noise with parameter σ^2 [65], and θ^* is the ground truth. Let α^* be the fraction of correctly labeled objects in the training data.

For AutoOD-Clean, we use squared loss, i.e., $f_\theta(x, y) = (y - w(\phi(x)^T \cdot \theta))^2$. We assume the feature matrices are regular, as defined below.

Definition 5.2. *Let $\Phi(X) \in \mathbb{R}^{n\times d}$ be the features matrix for all training objects, where the ith row is $\phi(x_i)^T$. Let $\mathcal{W}_k = \{W \in \mathbb{R}^{n\times n} | W_{i,j} = 0, W_{i,i} \in \{0, 1\}, \mathrm{Tr}(W) = k\}$. Define*

$$\begin{aligned} \psi^-(k) &= \min_{W:W\in} \sigma_{min}(\Phi(X)^T W \Phi(X)), \\ \psi^+(k) &= \max_{W:W\in} \sigma_{max}(\Phi(X)^T W \Phi(X)) \end{aligned} \tag{11}$$

$\Phi(X)$ is a regular feature matrix if for $k = \alpha n$, $\alpha \in [c, 1]$, $\psi^-(k) = \psi^+(k) = \Theta(n)$ for $n = \Omega(dlogd)$. Regularity states that every large enough subset of training objects results in a well conditioned $\Phi(X)$.

We prove the convergence claim by using an one-iteration update lemma for the linear case [62].

Lemma 5.1. *Assume $w(x) = x$ and we are using AutoOD-Clean. The following holds per iteration:*

$$||\theta_{t+1} - \theta^*||_2 \le \frac{\sqrt{2}\gamma_t}{\psi^-(\alpha n)}||\theta_t - \theta^*||_2 + \frac{\sqrt{2}\varphi_t + c\xi_t\sigma}{\psi^-(\alpha n)} \tag{12}$$

where α denotes the fraction of objects that are being removed in each iteration, $\varphi_t = \| \sum_{i\in S_t\setminus S^*} (\phi(x_i)^T\theta_t - r_i - e_i)\phi(x_i)\|_2$, $\gamma_t = \psi^+(|S_t\setminus S^*|)$, and $\xi_t = \sqrt{\sum_{i=1}^n \|\phi(x_i)_2^2 logn\|}$.

Note that in Lemma 5.1, α is always larger than 1, because AutoOD-Clean makes sure that the number of removed objects is always larger than the number of added objects. By Lemma 5.1 a α larger than 1 ensures that AutoOD-Clean bounds the error in the next step based on the error in the current step. AutoOD-Clean thus is guaranteed to converge.

Proof. Let θ_t be the learned parameter at round t, and θ_{t+1} be the learned parameter in the next round, following Alg. 3. More specifically, a subset S_t with the smallest losses $(y_i - \theta^T \cdot \phi(x_i))^2$ is selected. θ_{t+1} is the minimizer on the selected set of sample losses. Denote W_t as the diagonal matrix whose diagonal entry $W_{t,ii}$ equals 1 when the i-th sample is in set S_t, otherwise 0. Then, assume that we take infinite steps and reach the optimal solution, we have:

$$\theta_{t+1} = (\Phi(X)^T W_t \Phi(X))^{-1}\Phi(X)^T W_t y$$

where $\Phi(X)$ is an $n \times d$ matrix, whose i-th row is $\phi(x_i)^T$, and we have used the fact that $W_t^2 = W_t$. Remind that the feature matrix $\Phi(X)$ is defined in Equation 11. For $\Phi(X)$ whose every row follows i.i.d. sub-Gaussian random vector, by using concentration of the spectral norm of Gaussian matrices, and uniform bound, $\Phi(X)$ is a regular feature matrix.

On the other hand, denote W^* as the ground truth diagonal matrix for the samples, i.e., $W^*_{ii} = 1$ if the i-th sample is a clean sample, otherwise $W^*_{ii} = 0$. Accordingly, define S^* as the ground truth set of clean samples. For clearness of the presentation, we may drop the subscript t when there is no ambiguation. For bad samples, the output is written in the form of $y_i = r_i + e_i$, where e_i represents the observation noise, and r_i depends on the specific setting we consider. Under this general representation, we can re-write the term θ_{t+1} as

$$\begin{aligned} \theta_{t+1} &= (\Phi(X)^T W \Phi(X))^{-1}\Phi(X)^T W \\ &\quad (W^*\Phi(X)\theta^* + (I - W^*)r + e) \\ &= \theta^* + (\Phi(X)^T W \Phi(X))^{-1}(\Phi(X)^T W W^* \Phi(X)\theta^* \\ &\quad + \Phi(X)^T W r - \Phi(X)^T W \Phi(X)^T W W^* r \\ &\quad - \Phi(X)^T W \Phi(X)\theta^* + \Phi(X)^T W e) \\ &= \theta^* + (\Phi(X)^T W \Phi(X))^{-1}\Phi(X)^T (W W^* - W) \\ &\quad (\Phi(X)\theta^* - r - e) + (\Phi(X)^T W \Phi(X))^{-1}\Phi(X)^T W W^* e \end{aligned}$$

Therefore, the l_2 distance between the learned parameter and ground truth parameter can be bounded by:

$$\begin{aligned}\|\theta_{t+1}-\theta^*\| &= \|(\Phi(X)^T W\Phi(X))^{-1}\Phi(X)^T(WW^*-W)\\ &\quad (\Phi(X)\theta^*-r-e)+(\Phi(X)^T W\Phi(X))^{-1}\\ &\quad \Phi(X)^T WW^* e\|_2\\ &\le \underbrace{\|(\Phi(X)^T W\Phi(X))^{-1}\|_2}_{\tau_1}\cdot\\ &\quad (\underbrace{\|\Phi(X)^T(WW^*-W)(\Phi(X)\theta^*-r-e)\|_2}_{\tau_2}\\ &\quad +\underbrace{\|\Phi(X)^T WW^* e\|_2}_{\tau_3})\end{aligned}$$

For the term τ_1, W selects αn rows of $\Phi(X)$, i.e., $\mathbf{Tr}(W) = \alpha n$. Therefore, $\tau_1 \le \frac{1}{\psi^-(\alpha n)}$.

Next, the term τ_2 can be bounded as:

$$\begin{aligned}\tau_2^2 &= \|\Phi(X)^T(WW^*-W)(\Phi(X)\theta^*-r-e)\|_2^2\\ &= (\Phi(X)\theta^*-r-e)^T[(W-WW^*)\Phi(X)\Phi(X)^T\\ &\quad (W-WW^*)](\Phi(X)\theta^*-r-e)\\ &\le 2(\Phi(X)\theta^*-\Phi(X)\theta_t)^T[(W-WW^*)\Phi(X)\Phi(X)^T\\ &\quad (W-WW^*)](\Phi(X)\theta^*-\Phi(X)\theta_t)\\ &\quad +2(\Phi(X)\theta_t-r-e)^T[(W-WW^*)\Phi(X)\Phi(X)^T\\ &\quad (W-WW^*)](\Phi(X)\theta_t-r-e)\\ &\le 2\sigma_{max}(\Phi(X)^T(W-WW^*)\Phi(X))^2\|\theta^*-\theta_t\|_2^2\\ &\quad +2(\Phi(X)\theta_t-r-e)^T[(W-WW^*)\Phi(X)\Phi(X)^T\\ &\quad (W-WW^*)](\Phi(X)\theta_t-r-e)\end{aligned} \tag{13}$$

For the first term in Equation 13, let $|S_t\backslash S^*|$ be the number of bad samples in S_t. Then the eigenvalue is bounded by $\psi^+(|S_t\backslash S^*|)$. The last term in Equation 13 is defined as $\varphi_t := \varphi(S_t, S^*, \|\theta^*-\theta_t\|_2) = \|\sum_{i\in S\backslash S^*}(\phi(x_i)^T\theta_t - r_i - e_i)\phi(x_i)\|_2$. The term τ_3 can be bounded as:

$$\begin{aligned}\tau_3^2 &= \|\Phi(X)^T WW^* e\|_2\\ &\le e^T\Phi(X)\Phi(X)^T e = \sum_{i=1}^{d}(\sum_{j=1}^{n} e_j\phi(x_j)_i)^2\\ &\le c\sum_{i=1}^{n}\|\phi(x_i)\|_2^2 log n\sigma^2\end{aligned}$$

where the last inequality holds with high probability by the sub-exponential concentration property, and all randomness comes from the measurement noise e.

Then, as a summary, combining the results for all three terms, we have:

$$\begin{aligned}||\theta_{t+1}-\theta^*||_2 \le{}& \frac{\sqrt{2}\psi^+(|S_t\backslash S^*|)}{\psi^-(\alpha n)}||\theta_t-\theta^*||_2\\ &+\frac{\sqrt{2}\varphi(S_t,S^*,\|\theta^*-\theta_t\|_2)}{\psi^-(\alpha n)}\\ &+\frac{c\sqrt{\sum_{i=1}^{n}\|\phi(x_i)\|_2^2 log n}}{\psi^-(\alpha n)}\sigma\end{aligned} \tag{14}$$

□

5.3 The Selection Between AutoOD-Augment and AutoOD-Clean

AutoOD-Augment and AutoOD-Clean work in two different ways: AutoOD-Augment starts with a small set of reliable labels and keeps augmenting it, while AutoOD-Clean starts with a large but noisy set of labels, and keeps cleaning it. Although in general both methods work well, as shown in Fig. 4 (Sec. 6.2), they complement each other in terms of applicability concerning the availability of hardware and data set types. In particular, we prefer AutoOD-Clean when it is hard to acquire reliable labels initially. This, for example, is the case on the Pendigits dataset where AutoOD-clean significantly outperforms AutoOD-Augment. If the user does not have any GPU resources available, then we recommend the users AutoOD-Augment, because AutoOD-Clean needs GPUs to train a neural net. In the current implementation, our AutoOD system starts with AutoOD-Augment. If AutoOD-Augment only gets very few reliable labels at beginning, it switches to AutoOD-Clean.

6 EXPERIMENTAL EVALUATION

In the experiments, we evaluate the effectiveness and efficiency of AutoOD. We also analyze the quality of the automatically produced reliable labels which are critical to the performance of AutoOD.

6.1 Experimental Methodology and Setup

Datasets. We evaluate the effectiveness of AutoOD using 11 outlier detection benchmark datasets [16, 60] with varying cardinality, numbers of dimensions, and proportions of outliers. The main characteristics of these datasets are summarized in Table 1.

Dataset	# Instances	Outlier Fract.	# of Dims
PageBlock	5,473	10%	10
SpamBase	4,601	40%	57
Pima	768	35%	8
Shuttle	49,097	7%	9
Http	567,498	0.4%	3
Mulcross	262,141	10%	4
Annthyroid	7,129	7%	21
Musk	3,062	3%	166
Satimage-2	5,803	1.2%	36
Pendigits	6,870	2.3%	16
SMTP	95,156	0.03%	3

Table 1: Ten Benchmark Data Sets for Outlier Detection

Experimental Setup. The experiments were run on a single machine with 32 Intel Xeon 2.30GHz cores, 120GB RAM and 500GB disk on Google Cloud. We use one P100 GPU to train the neural networks.

Outlier Detectors. We use four popular unsupervised outlier detection methods: Local Outlier Factor (LOF) [15], K-Nearest Neighbors (KNN), Isolation Forest [44], and the Mahalanobis method that cover diverse categories of outlier types including local outliers, global outliers, tree-based outliers, and statistical-based outliers.

For all four methods, we configure them with a variety of different parameter settings. First, all methods have a parameter N that controls the number of outliers it returns. In our experiments, we pick 6 N values, so that the fraction of the outliers each method returns falls into the range from 0.5% to 10%.

Next, we vary the parameters specific to each method. Both LOF and KNN have the number of neighbors k as parameter. We work with 10 different k values, with the actual values of k randomly picked in a range from 1 to 100. For the Isolation Forest, we vary the number of features $num\_features$ to train the base estimator of the forest. For all datasets, we vary $num\_features$ from 20% to 100% of the available features with a 20% interval. The Mahalanobis method does not have any specific parameter.

In summary, we use 156 instantiated outlier detectors (60 (6×10) LOF detectors, 60 (6×10) KNN detectors, 30 (6×5) Isolation Forest detectors, and 6 Mahalanobis method detectors.

Algorithms. We compare the following algorithms:

- **AutoOD-Augment** (Sec. 4) and **AutoOD-Clean** (Sec. 5): our two AutoOD methods.
- **LODA** [52] and **LSCP** [71]: state-of-the-art outlier ensemble methods. Like AutoOD, outlier ensemble methods use many detectors to detect outliers. So they are a natural point of comparison. However, these works only use the detectors from the same detection method, e.g., LOF [15].
- **Snorkel** [55–57]: given a set of noisy labeling sources, aims to assign label to each object with high accuracy. We use its most recent variant [57] to integrate results from different outlier detectors, treating these detectors as noisy labeling sources. This can be seen as an advanced ensemble method.
- **PyODDs** [43]: leverages AutoML to select outlier detector, with strong assumption that ground truth labels are available.
- **MetaOD** [72]: uses a pre-trained meta learning model to select a good detector from many detectors, in which these detectors are pre-encoded during the training phase. We use the pre-trained model that the authors have published as core ingredient of their approach (https://github.com/yzhao062/MetaOD).
- **Isolation Forest** [44]: we compare against Isolation Forest with a random configuration. Same as [46], we report its accuracy as the average accuracy of 30 Isolation Forest models we use in our experiment, which can be considered to be equivalent to an Isolation Forest with randomly chosen hyper-parameters.
- **Best_Unsupervised**: the best unsupervised outlier detector among all detectors discovered by an oracle, where we use ground truth labels to compute F-1 score of each detector.
- **Ground-Truth(GT)**: supervised classifier that is trained using the ground truth as labels. This corresponds to a best case scenario, as it knows upfront what are the outliers and inliers in the data set. As in AutoOD-Augment, we use SVM as supervised classification technique. We train SVM model by randomly sampling 50% of dataset as training data.

Metrics. We first evaluate effectiveness at detecting outliers by measuring the F-1 score for the outlier class, given F-1 is known to be robust to class imbalance. F-1 score considers both precision and recall, where precision is the number of correctly detected outliers divided by the number of all outliers returned by the detector, and recall is the number of correctly detected outliers divided by the number of all ground truth outliers. F-1 is computed as F-1 = $2 \times \frac{precision \times recall}{precision + recall}$. We also report the precision and recall separately. Next, we evaluate the efficiency of AutoOD including the running/-training time, its scalability to the number of detectors, and the memory consumption. In addition, we report the statistics of the reliable labels that AutoOD produces and the number and the types of detectors that AutoOD-Augment preserves. Finally, we evaluate if AutoOD-Augment is effective at identifying the bad detectors.

Parameters. For the methods that fall into the category of automatic detector selection including MetaOD and PyODDs we use the configuration suggested by their original authors in the literature. For example, in the AutoML-based PyODDs, there is no parameter to tune other than ensuring it uses the same set of detectors that AutoOD uses. Per the suggestion of the MetaOD [72] authors, we use the model they published, which in the training phase uses more types of outlier detectors than AutoOD. For the Isolation Forest, per the suggestion of [46], We report its accuracy as the average accuracy of 30 Isolation Forest detectors we use. For the ensemble-based methods including Snorkel, LODA, and LSCP, we tune the parameters per the instruction of the authors and report the best results on each dataset. Snorkel uses the same set of detectors that AutoOD uses. We set its learning rate as 0.001 and the number of epochs as 1000. LODA requires the users to determine the outlier rate which is set as the *true* outlier rate of each dataset in our experiments. Per the authors' recommendation, LSCP uses LOF as the detection method. For each LOF detector, we set its outlier rate in the the same way as our AutoOD does.

6.2 Effectiveness Evaluation

6.2.1 Summary of Effectiveness Results. In this experiment we measure the effectiveness of AutoOD using a variety of benchmark outlier detection data sets. We find that AutoOD is consistently able to detect outliers with an accuracy higher than the *best outlier detector* among hundreds of configured detectors, while all other approaches we evaluated perform significantly worse. In fact, AutoOD succeeds to achieve an accuracy comparable to supervised outlier classifiers trained with ground truth labels – yet without having any access to such ground truth knowledge. AutoOD significantly outperforms the two SOTA outlier ensemble methods (LODA and LSCP), Snorkel, PyODDs, MetaOD, and Isolation Forest by 12 to 97 points (out of 100) in the F-1 score.

6.2.2 Detailed Analysis of Effectiveness. Fig. 4 shows the results on ten benchmark datasets. At the x axis, the datasets are ordered by their sizes. Http is the largest dataset. AutoOD applies the same configuration to all data sets. On almost all datasets, AutoOD outperforms all other methods except GT – with the later the unfair advantage of full access to the ground truth to train its outlier classifier and hence it is expected to perform very well.

Comparison to Best Unsupervised. Compared to the best unsupervised detector (discovered by an oracle using the ground truth),

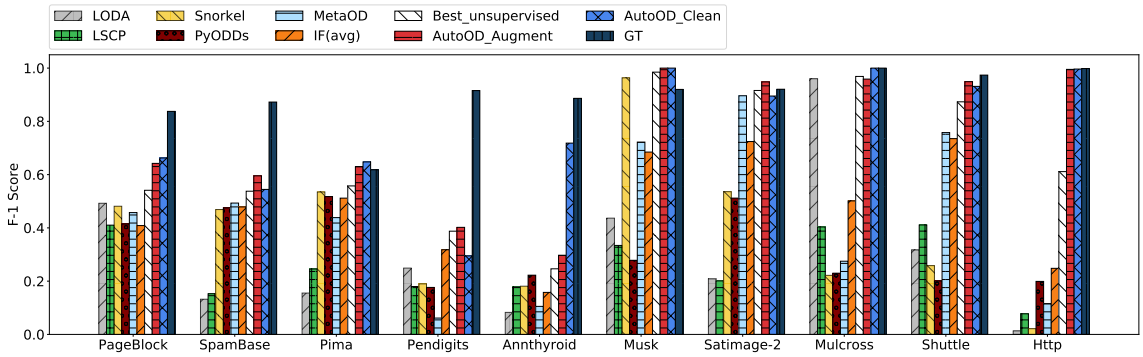

Figure 4: Effectiveness Evaluation of All Methods on Ten Benchmark Data Sets using F-1 Metric

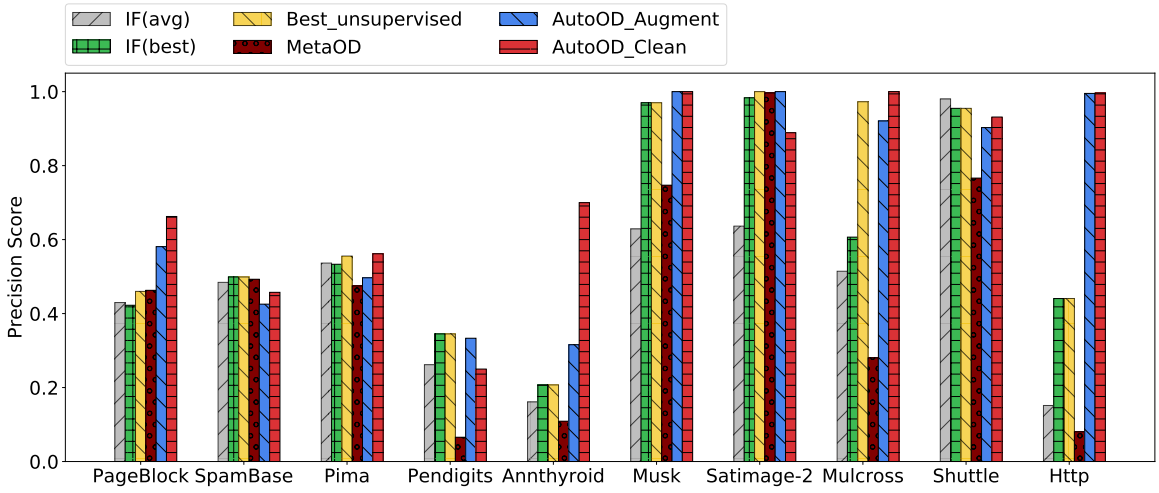

Figure 5: Effectiveness Evaluation of All Methods on Ten Benchmark Data Sets using Precision Metric

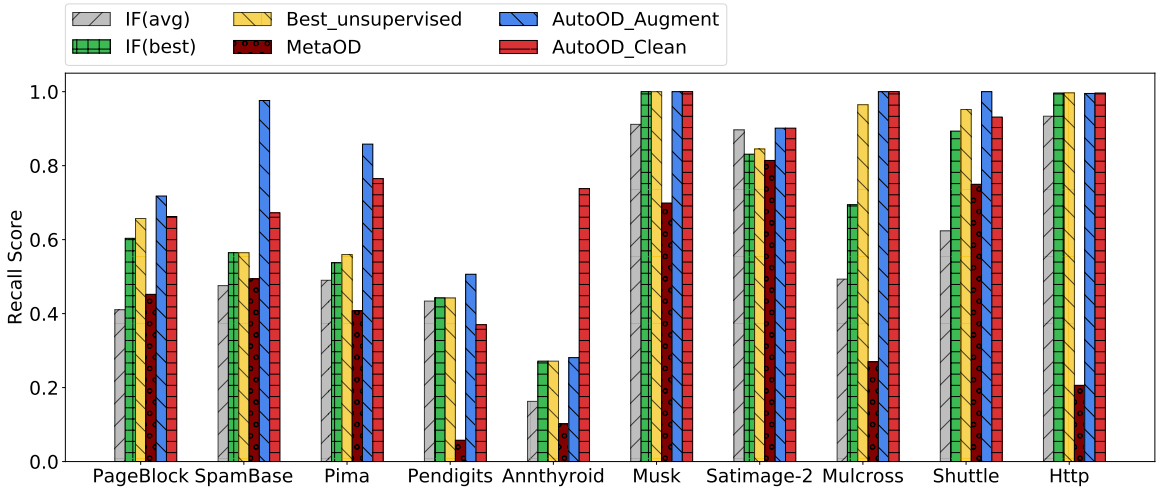

Figure 6: Effectiveness Evaluation of All Methods on Ten Benchmark Data Sets using Recall Metric

AutoOD-Augment achieves higher F-1 scores on 9 out of 10 datasets by up to 38 points and AutoOD-Clean gets better F-1 scores on 8 out of 10 datasets by up to 39 points, without relying on human tuning. This is because AutoOD intelligently combines the contributions of all unsupervised detectors in the process of automatically discovering reliable labels, instead of relying on one single detector to produce the final results.

Comparison to Ensemble. The two outlier ensemble methods (LODA, LSCP) and Snorkel perform consistently worse than the best outlier detector, often with a large margin. The reason may be

that the detectors often produce diverse results on many objects. It is thus challenging to make consensus-based inference on these objects – yet this is indeed the nature of these outlier ensemble methods.

Comparison to PyODDs. Although PyODDs uses domain knowledge in the form of *ground truth* labels (to which our methods do not have access), it cannot find the best detector in all cases and often ends up with a poor detector. Therefore, in many cases it performs even worse than LODA, LSCP, and Snorkel, especially for large datasets. Because our AutoOD consistently outperforms the best detector, it thus significantly outperforms all these methods from 12 to 97 points in F-1 score.

Comparison to MetaOD. We use the model published by the authors. During training MetaOD has already seen 8 out of the 10 testing datasets used in our experiments, which should bias performance in its favor. Overall, MetaOD slightly outperforms Snorkel and PyODD, showing that this interesting meta-learning based method works to some extent.

However, even on the 8 datasets seen before, MetaOD cannot find the best detector from the candidates, while it completely fails on the two previously unseen datasets (Mulcross and HTTP), ending up with choosing a detector that performs poorly. This indicates MetaOD has poor generalization ability, probably because the characteristics of outliers in different datasets can be rather distinct.

Comparison to Isolation Forest. For the Isolation Forest with a random configuration, we get a similar conclusion to the empirical study paper [46], namely, that it is comparable to the methods that select one method from a number of outlier detectors and to ensemble-based methods. However, this method still performs significantly worse than the best unsupervised detector and than our two AutoOD-based approaches, because Isolation Forests do not always perform the best among all detectors, as shown in Table 4.

Comparison to GT. Compared to GT, our two AutoOD methods achieve comparable or even slightly higher F-1 scores on 6 out of the 10 datasets, even though AutoOD does not have access to any ground truth training data. However, on the Pendigits and Annthyroid datasets, AutoOD does not perform as well as GT. This is because for these datasets, all unsupervised outlier detectors have very low F-1 scores. Therefore, AutoOD cannot identify many high quality labels from the output of these unsupervised detectors.

Scalability to Large Data & Robustness to Dimensionality. As shown in Fig. 4, the larger the datasets are, the better AutoOD performs, indicating its *scalability* to large data. This is because it is easier to produce a sufficient number of quality labels from large datasets. The *dimensionality* of the datasets used in our experiments falls in a large range from 10 to 166. AutoOD works well on the high dimensional Musk data. This shows AutoOD is robust to dimensionality. Because any outlier detection method can be plugged into AutoOD, base detectors less sensitive to data dimensionality thus can be used. For example, in our experiments we use Isolation Forest known to work well on high dimensional data.

The Convergence. In Sec. 4.4 (Termination Condition) and Sec. 5.2 we have shown that AutoOD-Augment and AutoOD-Clean are guaranteed to converge. In our experiments, we observe that on average AutoOD-Augment converges in 21 iterations, while AutoOD-Clean converges in 12 iterations.

The Performance Variance of AutoOD on Different Datasets. For some datasets, AutoOD is close to GT, while for others the difference is big. This is because of the quality of the labels that AutoOD automatically produces. When AutoOD is able to produce high quality labels, AutoOD tends to work as well as GT (Ground Truth) which uses ground truth labels to train an outlier classification model. As shown in Fig. 4, the performance of AutoOD is comparable to GT on the large datasets, because AutoOD has a larger chance to produce labels that are *indeed* reliable. Our analysis on the automatically produced labels (Table 2) confirms this. On the Musk, Stimage-2, Mulcross, Shuttle, and Http datasets, the labels produced by AutoOD have an accuracy close to 1. Interestingly, we observe that typically AutoOD does not need a large number of outlier labels to achieve a high accuracy in outlier detection. As long as it gets a sufficient number of accurate inlier labels, AutoOD performs well. Because outliers are typically rare, this makes the strategy used by AutoOD well-suited to outlier detection.

AutoOD-Augment VS AutoOD-Clean. As we have discussed in Sec. 5.3, AutoOD-Augment and AutoOD-Clean work in two different ways and complement each other. However, we observe that AutoOD-Augment and AutoOD-Clean achieve similar accuracy results on many datasets. Our analysis on the reliable labels confirms that this is because the reliable labels that the two methods produce tend to have a similar quality (Table 2) and in most cases, heavily overlap with each other. We observe that these common reliable labels typically correspond to the same set of *"clear"* outliers and inliers which are relatively easy for the unsupervised outlier detection detectors and hence AutoOD-Augment/AutoOD-Clean to capture.

6.2.3 Precision and Recall. We study the precision and recall of the methods that perform best on F-1 including our two AutoOD methods, Isolation Forests, MetaOD, and the best unsupervised detector. We omit other methods as including them makes the graph hard to read. As shown in Fig. 6 and 6, AutoOD significantly outperforms all other methods. Moreover, AutoOD performs even better on recall than on F-1. This is important to outlier detection, as missing outliers (false negatives) often causes bigger problems than false alarms (false positives).

6.3 Reliable Label Analysis

In Table 2, we show the statistics (the quality and the number) of reliable labels that AutoOD produces separately for inlier and outlier labels. We make the following observations:

- Both AutoOD-Augment and AutoOD-Clean do not need a large number of reliable outliers to achieve good performance; as long as a sufficient number of reliable inliers can be identified that the system can rely on, i.e., that are high quality, AutoOD-Augment and AutoOD-Clean work well.
- On large datasets, the reliable labels that AutoOD produces are near perfect. Therefore, AutoOD achieves an accuracy close to that of the supervised method GT. This is because the larger the dataset is, the better the chance that AutoOD is able to produce labels that are *indeed* reliable.
- The accuracy of reliable inlier labels that AutoOD finds is consistently high, indicating it is easier to find reliable inliers from the datasets than outliers.

Datasets	Accuracy						Number of Points			
	AutoOD-Augment			AutoOD-Clean			AutoOD-Augment		AutoOD-Clean	
	F-1	Outlier Acc	Inlier Acc	F-1	Outlier Acc	Inliner Acc	Outlier	Inliner	Outlier	Inlier
PageBlock	0.657	0.670	0.976	0.743	0.721	0.973	282	4413	258	2076
SpamBase	0.597	0.431	0.840	0.547	0.425	0.760	3761	275	2001	1063
Pima	0.691	0.531	0.980	0.756	0.791	0.874	196	50	43	103
Pendigits	0.421	0.400	0.989	0.362	0.350	0.972	140	6528	75	3105
Annthyroid	0.046	0.063	0.942	0.890	0.802	1.000	206	6101	247	1068
Musk	1.000	1.000	1.000	1.000	1.000	1.000	34	2658	22	1514
Satimage-2	0.984	1.000	1.000	0.943	0.926	1.000	60	5642	27	2488
mulcross	0.981	0.963	1.000	1.000	1.000	1.000	27229	231736	26213	34621
Shuttle	0.998	0.996	1.000	0.968	0.976	1.000	2058	44740	482	6791
Http	0.946	0.898	1.000	0.949	0.924	1.000	118	535406	189	43251
SMTP	0.690	0.714	1.000	0.644	0.633	1.000	28	94979	30	91856

Table 2: Reliable Label Analysis

Method	Precision	Recall	F-1
IForest	0	0	0
Best Unsupervised	0.667	0.667	0.667
MetaOD	0.536	0.44	0.481
AutoOD_Augment	0.714	0.667	0.690
AutoOD_Clean	0.76	0.633	0.691

Table 3: Evaluation on SMTP (Low Outlier Rate)

6.4 Effectiveness of AutoOD on Dataset with Low Outlier Rate

We experiment on the SMTP dataset [16, 60] with an extremely low outlier rate at 0.03%. Note this is the only outlier detection benchmark dataset we can find that is publicly available and has a very low outlier rate. As shown in Table 3, both AutoOD-Augment and AutoOD-Clean achieve an accuracy higher than the best unsupervised detector. They also significantly outperform other methods. This is similar to the results on other datasets with higher outlier rates. Note in the effective experiments we discussed in Sec. 6.2, we evaluated AutoOD on datasets with outlier rates varying from 0.4% to 40%, confirming that our AutoOD works well across a rich variety of outlier rates.

As we have analyzed in Sec. 6.3, this is because AutoOD does not need a large number of outlier labels to achieve a good performance. As long as AutoOD obtains a sufficient number of accurate inlier labels, it works well. In outlier detection, getting reliable inlier labels tends to be much easier than getting reliable outlier labels. Therefore, AutoOD does not suffer as much from the rarity of outliers.

6.5 Efficiency Evaluation

We evaluate the running/training time of AutoOD, its scalability in the number of outlier detectors, and the memory requirement.

6.5.1 Comparison of Running time to Other Methods. We compared our AutoOD-Augment and AutoOD-Clean against MetaOD [72], PyODDs [43] and Snorkel [57], all of which, like AutoOD, use many different types of outlier detectors to produce the final detection results. As shown in Fig. 7, except on the Pendigit dataset, PyODDs is always the slowest because of the complex AutoML technique they use.

The total running time of AutoOD is composed of the time of first running the unsupervised outlier detectors and then second learning the reliable labels from the detection results, with the former typically dominating the latter, as shown in Fig. 7. Because both our methods AutoOD-Augment and AutoOD-Clean run the same set of unsupervised detectors, they typically have a similar total running time. This is why, compared to manual tuning, which first runs all detectors and then picks a good one, AutoOD does not have much overhead. This also explains why AutoOD tends to be better than or comparable to Snorkel in running time which runs the same set of outlier detectors as AutoOD.

For our comparison to MetaOD, we use the pre-trained model that the authors have published. Hence, we cannot measure the MetaOD training time. Thus, we only report the running time that it takes to process the targeted dataset in which outliers are to be identified. Clearly the complexity of MetaOD depends on the inference time of the meta learning model plus the final detector it selects. We note that MetaOD tends to be slow at inference phase and often ends up with an expensive outlier detector, in particular, ABOD [38]. Therefore, in many cases, it is slower than AutoOD. However, among all methods, this second phase of MetaOD (given we skip phase 1 of training) tends to be the fastest on large datasets.

Note all above methods we compare against have *much lower accuracy* than AutoOD as shown in Fig. 4, while our AutoOD achieves this gain without paying any significant cost in running time. We thus consider AutoOD*superior overall*.

6.5.2 Scalability in the Number of Outlier Detectors. Because AutoOD-Augment and AutoOD-Clean perform similarly in the running time (for the reasons we discussed above), here we only report on the results of AutoOD-Augment. We run this experiment on the MulCross dataset and vary the number of detectors from 50 to 250 by gradually increasing the number of detectors per detection method.

As we can see from Fig. 8, the running time of AutoOD-Augment increases, as expected. However, the slope becomes flatter as the number of detectors gets larger. This is because AutoOD improves on the running time of the unsupervised outlier detectors by sharing the common computation across different outlier detectors. In particular, for the slower ones among the detectors, namely, the kNN-based [54] and LOF [15] detectors that utilize expensive kNN search, our detectors share kNN search as much as possible. Therefore, the running time increases sub-linearly with the number of detectors.

6.5.3 Memory. In AutoOD, the memory consumption comes primarily from the data structure that keeps the anomaly scores that the outlier detectors produce per data object. This space complexity is $n \times d$, where n represents the number of data objects and d is the number of the unsupervised detectors. On the HTTP dataset, which is the largest dataset we tested, the peak memory consumption is 708M. Therefore, memory is not a performance bottle in AutoOD.

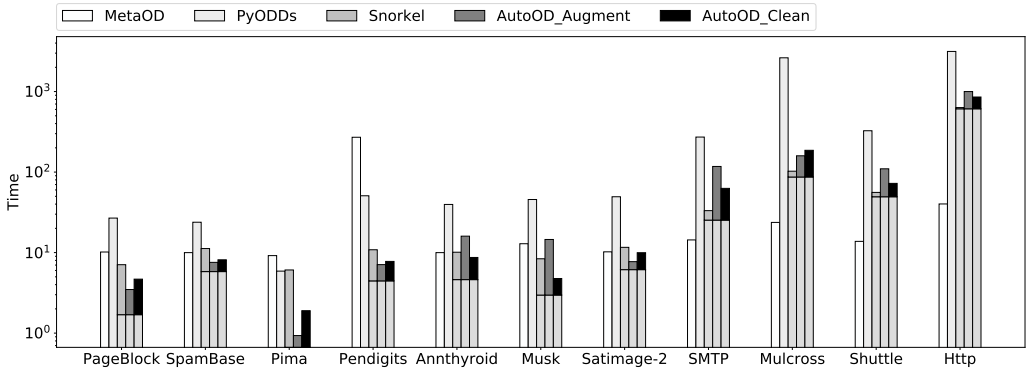

Figure 7: Runtime Evaluation

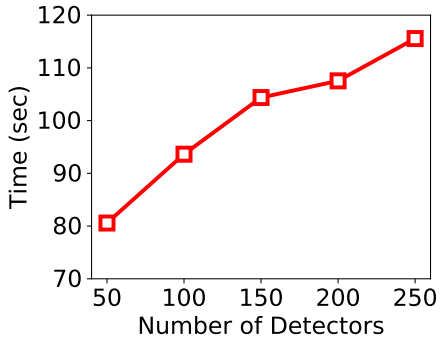

Figure 8: Scalability on the Number of Detectors

Dataset	Rank of Detector	Final Detector
PageBlock	2nd	KNN
SpamBase	1st	LOF
Pima	1st	IForest
Shuttle	1st	IForest
Http	3rd	KNN
Mulcross	1st	Mahalanobis
Annthyroid	2nd	LOF
Musk	1st	IForest
Satimage-2	1st	LOF
Pendigits	1st	IForest
SMTP	1st	LOF

Table 4: AutoOD-Augment: the Remaining Detector

6.6 Evaluation of Detector Pruning of AutoOD-Augment

As shown in Sec. 4, AutoOD-Augment iteratively prunes outlier detectors. In this experiment we evaluate the number and the types of detectors preserved by AutoOD-Augment. On average AutoOD-Augment preserves 1.4 detectors and in most cases preserves 1 detector, while their types vary across different datasets, as shown in Table 4. The detection methods that AutoOD most frequently preserves are the Isolation Forest and LOF. This is aligned with empirical studies that evaluate different unsupervised outlier detection methods [16], which find: (1) no method routinely outperforms others; (2) Isolation Forest and LOF tend to work better in general.

We further evaluate the quality of the outlier detector that AutoOD-Augment preserves. For this, we add an additional constraint into its termination condition to ensure that only one detector will survive. Tab. 4 shows the rank of this remaining outlier detector among all outlier detectors based on their F-1 scores. AutoOD-Augment selects the best outlier detector on 7 out of 10 datasets. On the remaining 3 datasets, AutoOD-Augment picks the second or third best detector.

7 RELATED WORK

Automated Machine Learning (AutoML). AutoOD targets a similar problem as AutoML in that it aims to learn which of many models is the best. However, in contrast to our work, existing AutoML systems [13, 14, 25, 28, 37, 40, 42, 45, 63, 64, 74] all focus on (1) *supervised* machine learning and (2) require labeled training data for automatic cross validation. PyODDS [43] leverages AutoML techniques to select one outlier detection method and their corresponding input parameter setting, *but assuming the existence of labeled data*. Further, our experiments show that PyODDS is not very effective in finding good outlier detectors among candidate detectors even using ground truth labels. AutoOD significantly outperforms PyODDS even in this setting.

Outlier Ensemble. Ensemble has been studied in the context of outlier detection [4, 5, 39, 52, 71]. In [4, 5], the authors investigated the theoretical underpinnings of outlier ensemble. In [39], the authors use feature bagging for outlier ensemble. However, it relies on the strong assumption that they "have information about normal behavior (class) in the data set", which often does not hold.

LSCP [71] and LODA [52] are the recent outlier ensemble works. LSCP defines a local region around an instance using the consensus of its nearest neighbors in randomly selected feature sub-spaces. It then combines the detection results produced by multiple base detectors in this local region as the model's final output using traditional ensemble tricks [5]. In its evaluation LSCP outperforms feature bagging [39].

LODA [52] first uses many one-dimensional histograms to produce outlier candidates. An object is considered as an outlie candidate if its value on one dimension is out of distribution. LODA then aggregates the sets of outlier candidates to produce the final results, again based on consensus. LSCP and LODA perform much worse than our AutoOD, because based on our empirical study the base detectors often produce very diverse results on many of the objects and thus make consensus-based inference challenging. AutoOD does not rely on the ensemble to discover all outliers. Instead it focuses

on discovering the reliable outliers/inliers and then uses those as pseudo-labels to train a classifier, hence much more effective.

Weak Supervision. Given a set of noisy labeling functions that cover different overlapping subsets of a given data set, Snorkel [55, 57] assigns a label to each object in the dataset by weighting and then ensembling the results produced by these labeling functions. More specifically, using a matrix completion technique, Snorkel learns a weight for each function based on their agreement rates. If a function agrees with a lot of other functions, Snorkel tends to assign a large weight to it. So it can be considered as an advanced *ensemble method*. As confirmed in our experiments (Sec. 6), Snorkel typically performs much worse than the best individual detector for the reason we discussed above on outlier ensembles.

Parameter Tuning in Outlier Detection. ONION [17] is an online system that, given a data set, allows the users to interactively tune the parameters of distance-based outlier detection. The key idea is to pre-compute and index results of parameterized distance-based outlier detectors into a compact index. Using index-lookup, ONION is able to answer outlier detection requests with different parameter settings in near real-time. Unlike our AutoOD, it still relies on the humans to manually decide which parameter setting to request and thus to tune the parameters. Further, it only supports one specific outlier technique, that is, distance-based outliers [35].

MetaOD [72] is a meta-learning based method. Its key idea is to run a large number of outlier detectors on historical outlier detection benchmark datasets with ground truth labels and use this prior experience to automatically select an effective model to be employed on a new dataset. To capture task similarity, it introduces specialized meta-features to model the characteristics of the outliers in a dataset. However, as confirmed in our experiments (Sec. 6.2.2), even on the 8 testing datasets seen in the training process, MetaOD cannot find the best detector from the candidates, while on the 2 unseen datasets, it ends up with choosing a detector that performs poorly, indicating its poor generalization ability.

Human-in-the-Loop Outlier Detection. HOD [19], a crowd-based method, relies on humans to find outliers. HOD first uses many unsupervised outlier detectors to produce a large outlier candidate set, taking the union of the detected outliers. To reduce human labor costs, it designs some questions that once answered by humans help verify the status of multiple outlier candidates. In contrast, AutoOD does not make use of any human supervision, yet succeeds to achieve accuracy comparable to supervised outlier classification.

Deep Anomaly Detection. Deep learning has been used to detect outliers from data in complex format such as image or timeseries, typically by learning a representation that better distinguishes outliers from inliers. Some of these techniques [8, 22, 59, 69] use the reconstruction errors of Auto-Encoder as the anomalous score to detect outliers, assuming that Auto-Encoders incur larger reconstruction errors on outliers than inliers. Some other techniques use the same principle, but apply different deep learning techniques to learn the data representation, such as Generative Adversarial Networks [6, 51, 68], self-learning models [27] and Auto-regressive models [2]. AutoOD is compatible with these methods.

To learn a representation effective in separating outliers, most of these methods require a clean training data set – a data set not containing any outliers. However, such clean training data rarely exist in real applications. Robust deep anomaly detection [12, 20, 24, 66, 73] targets this problem by finding data with potentially *corrupted features* in the training process, while AutoOD-Clean finds *mislabeled objects* from automatically produced outlier/inlier labels. Elite [70] instead uses a small number of ground truth labels to mitigate the impact of outliers on representation learning. However, ground truth labels are *not available* in the unsupervised setting which AutoOD targets.

Data Fusion. Data fusion integrates data from different sources which potentially conflict with each other to obtain better description about the same objects. To exclude the effect of low quality data sources, in [23] the authors select a subset of data sources that balance the cost and the gain, where the gain is approximated on a set of samples with their truths known beforehand. However, AutoOD does not assume the availability of the ground truth. CRH [41] estimates the source reliability by iterating truth estimation and source weighting. However, it relies on the users to define a cost function and a regularization funcition based on the characteristics of the data sources and the source reliability distributions. However, such knowledge does not exist in AutoOD setting.

8 CONCLUSION

In this work, we propose AutoOD that elegantly unifies the merits of both unsupervised outlier detection and supervised classification into one solution for tackling outlier detection. AutoOD leverages a diverse set of unsupervised outlier detectors to iteratively generates high quality labels for the data. Using these automatically generated labels, it then trains a classification model to reliably discover outliers. We design two strategies for realizing the AutoOD methodology, namely AutoOD-Augment and AutoOD-Clean, which differ in the way they generate the training set for the classifier. Our experiments show the effectiveness of the two AutoOD-based methods – achieving consistently 12 to 97 points gain in the F-1 score compared to a wide range of alternate solutions on 11 benchmark outlier data sets.

REFERENCES

[1] Forrester prediction. https://go.forrester.com, 2017.
[2] D. Abati, A. Porrello, S. Calderara, and R. Cucchiara. Latent space autoregression for novelty detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 481–490, 2019.
[3] C. C. Aggarwal. *Outlier Analysis: Second Edition*. Springer, 2017.
[4] C. C. Aggarwal and S. Sathe. Theoretical foundations and algorithms for outlier ensembles. *ACM SIGKDD Explorations Newsletter*, 17(1):24–47, 2015.
[5] C. C. Aggarwal and S. Sathe. *Outlier ensembles: An introduction*. Springer, 2017.
[6] S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon. Ganomaly: Semi-supervised anomaly detection via adversarial training. In *Asian conference on computer vision*, pages 622–637. Springer, 2018.
[7] S. M. Al Pascual, Kyle Marchini. Identity fraud: Securing the connected life. 2017.
[8] J. T. Andrews, E. J. Morton, and L. D. Griffin. Detecting anomalous data using auto-encoders. *International Journal of Machine Learning and Computing*, 6(1):21, 2016.
[9] F. Angiulli and C. Pizzuti. Fast outlier detection in high dimensional spaces. In *PKDD*, pages 15–26, 2002.
[10] P. Bailis, E. Gan, S. Madden, D. Narayanan, K. Rong, and S. Suri. Macrobase: Prioritizing attention in fast data. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 541–556. ACM, 2017.
[11] V. Barnett, T. Lewis, et al. *Outliers in statistical data*, volume 3. Wiley New York, 1994.
[12] L. Beggel, M. Pfeiffer, and B. Bischl. Robust anomaly detection in images using adversarial autoencoders. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 206–222. Springer, 2019.
[13] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb):281–305, 2012.

[14] C. Binnig, B. Buratti, Y. Chung, C. Cousins, T. Kraska, Z. Shang, E. Upfal, R. Zeleznik, and E. Zgraggen. Towards interactive curation & automatic tuning of ml pipelines. In *Proceedings of the Second Workshop on Data Management for End-To-End Machine Learning*, pages 1–4, 2018.

[15] M. M. Breunig, H. Kriegel, R. T. Ng, and J. Sander. LOF: identifying density-based local outliers. In *SIGMOD*, pages 93–104, 2000.

[16] G. O. Campos, A. Zimek, J. Sander, R. J. G. B. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Min. Knowl. Discov.*, 30(4):891–927, 2016.

[17] L. Cao, M. Wei, D. Yang, and E. A. Rundensteiner. Online outlier exploration over large datasets. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 89–98. ACM, 2015.

[18] L. Cao, Y. Yan, C. Kuhlman, Q. Wang, E. A. Rundensteiner, and M. Eltabakh. Multi-tactic distance-based outlier detection. In *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*, pages 959–970. IEEE, 2017.

[19] C. Chai, L. Cao, G. Li, J. Li, Y. Luo, and S. Madden. Human-in-the-loop outlier detection. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, page 19–33, New York, NY, USA, 2020. Association for Computing Machinery.

[20] R. Chalapathy, A. K. Menon, and S. Chawla. Robust, deep and inductive anomaly detection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 36–51. Springer, 2017.

[21] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection. *ACM Computing Surveys*, 41(3):1–58, 2009.

[22] J. Chen, S. Sathe, C. Aggarwal, and D. Turaga. Outlier detection with autoencoder ensembles. In *Proceedings of the 2017 SIAM international conference on data mining*, pages 90–98. SIAM, 2017.

[23] X. L. Dong, B. Saha, and D. Srivastava. Less is more: Selecting sources wisely for integration. *Proc. VLDB Endow.*, 6(2):37–48, 2012.

[24] S. Eduardo, A. Nazábal, C. K. I. Williams, and C. Sutton. Robust variational autoencoders for outlier detection and repair of mixed-type data. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, pages 4056–4066, 2020.

[25] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In *Advances in neural information processing systems*, pages 2962–2970, 2015.

[26] J. Gao, J. Han, J. Liu, and C. Wang. Multi-view clustering via joint nonnegative matrix factorization. In *SDM, May 2-4, 2013. Austin, Texas, USA*, pages 252–260, 2013.

[27] I. Golan and R. El-Yaniv. Deep anomaly detection using geometric transformations. In *NeurIPS*, pages 9758–9769, 2018.

[28] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1487–1495, 2017.

[29] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[30] T. Hastie, R. Tibshirani, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction, 2nd Edition*. Springer series in statistics. Springer, 2009.

[31] D. M. Hawkins. *Identification of outliers*, volume 11. Springer, 1980.

[32] X. He, K. Zhao, and X. Chu. Automl: A survey of the state-of-the-art. *arXiv preprint arXiv:1908.00709*, 2019.

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

[34] D. G. Kleinbaum, K. Dietz, M. Gail, M. Klein, and M. Klein. *Logistic regression*. Springer, 2002.

[35] E. M. Knorr and R. T. Ng. Algorithms for mining distance-based outliers in large datasets. In *VLDB*, pages 392–403, 1998.

[36] E. M. Knorr and R. T. Ng. Finding intensional knowledge of distance-based outliers. In *VLDB*, volume 99, pages 211–222, 1999.

[37] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown. Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. *The Journal of Machine Learning Research*, 18(1):826–830, 2017.

[38] H. Kriegel, M. Schubert, and A. Zimek. Angle-based outlier detection in high-dimensional data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, pages 444–452, 2008.

[39] A. Lazarevic and V. Kumar. Feature bagging for outlier detection. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*, pages 157–166, 2005.

[40] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

[41] Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J. Han. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 1187–1198, 2014.

[42] T. Li, J. Zhong, J. Liu, W. Wu, and C. Zhang. Ease. ml: Towards multi-tenant resource sharing for machine learning workloads. *Proceedings of the VLDB Endowment*, 11(5):607–620, 2018.

[43] Y. Li, D. Zha, P. Venugopal, N. Zou, and X. Hu. Pyodds: An end-to-end outlier detection system with automated machine learning. In *Companion Proceedings of the Web Conference 2020*, pages 153–157, 2020.

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

[45] G. Luo. A review of automatic selection methods for machine learning algorithms and hyper-parameter values. *Network Modeling Analysis in Health Informatics and Bioinformatics*, 5(1):18, 2016.

[46] M. Q. Ma, Y. Zhao, X. Zhang, and L. Akoglu. A large-scale study on unsupervised outlier model selection: Do internal strategies suffice? *ArXiv*, abs/2104.01422, 2021.

[47] D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. On the least trimmed squares estimator. *Algorithmica*, 69(1):148–183, 2014.

[48] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos. LOCI: fast outlier detection using the local correlation integral. In *Proceedings of the 19th International Conference on Data Engineering, March 5-8, 2003, Bangalore, India*, pages 315–326, 2003.

[49] F. Pedregosa, G. Varoquaux, and etc. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[50] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[51] P. Perera, R. Nallapati, and B. Xiang. Ocgan: One-class novelty detection using gans with constrained latent representations. In *CVPR*, pages 2898–2906, 2019.

[52] T. Pevnỳ. Loda: Lightweight on-line detector of anomalies. *Machine Learning*, 102(2):275–304, 2016.

[53] J. Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[54] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *SIGMOD*, pages 427–438, 2000.

[55] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré. Snorkel: Rapid training data creation with weak supervision. *The VLDB Journal*, pages 1–22, 2019.

[56] A. Ratner, S. H. Bach, H. R. Ehrenberg, J. A. Fries, S. Wu, and C. Ré. Snorkel: Rapid training data creation with weak super. *PVLDB*, 11(3):269–282, 2017.

[57] A. Ratner, B. Hancock, J. Dunnmon, F. Sala, S. Pandey, and C. Ré. Training complex models with multi-task weak supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4763–4771, 2019.

[58] L. Rokach. Ensemble-based classifiers. *Artif. Intell. Rev.*, 33(1-2):1–39, 2010.

[59] M. Sakurada and T. Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, pages 4–11, 2014.

[60] E. Schubert, R. Wojdanowski, A. Zimek, and H. Kriegel. On evaluation of outlier rankings and outlier scores. In *SDM*, pages 1047–1058, 2012.

[61] Z. Shang, E. Zgraggen, B. Buratti, F. Kossmann, P. Eichmann, Y. Chung, C. Binnig, E. Upfal, and T. Kraska. Democratizing data science through interactive curation of ML pipelines. In *SIGMOD*, pages 1171–1188, 2019.

[62] Y. Shen and S. Sanghavi. Learning with bad training data via iterative trimmed loss minimization. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5739–5748, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[63] E. R. Sparks, A. Talwalkar, D. Haas, M. J. Franklin, M. I. Jordan, and T. Kraska. Automating model search for large scale machine learning. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*, pages 368–380. ACM, 2015.

[64] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855, 2013.

[65] R. Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

[66] Y. Xia, X. Cao, F. Wen, G. Hua, and J. Sun. Learning discriminative reconstructions for unsupervised outlier removal. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1511–1519, 2015.

[67] Y. Yan, L. Cao, and E. A. Rundensteiner. Scalable top-n local outlier detection. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1235–1244. ACM, 2017.

[68] H. Zenati, C. S. Foo, B. Lecouat, G. Manek, and V. R. Chandrasekhar. Efficient gan-based anomaly detection. *arXiv preprint arXiv:1802.06222*, 2018.

[69] S. Zhai, Y. Cheng, W. Lu, and Z. Zhang. Deep structured energy based models for anomaly detection. *arXiv preprint arXiv:1605.07717*, 2016.

[70] H. Zhang, L. Cao, P. VanNostrand, S. Madden, and E. A. Rundensteiner. ELITE: robust deep anomaly detection with meta gradient. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pages 2174–2182, 2021.

[71] Y. Zhao, Z. Nasrullah, M. K. Hryniewicki, and Z. Li. Lscp: Locally selective combination in parallel outlier ensembles. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 585–593. SIAM, 2019.

[72] Y. Zhao, R. Rossi, and L. Akoglu. Automatic unsupervised outlier model selection. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 4489–4502. Curran Associates, Inc., 2021.

[73] C. Zhou and R. C. Paffenroth. Anomaly detection with robust deep autoencoders. In *SIGKDD*, pages 665–674, 2017.

[74] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.