UADB: Unsupervised Anomaly Detection Booster

Hangting Ye[1*], Zhining Liu[2*], Xinyi Shen[3], Wei Cao[4], Shun Zheng[4], Xiaofan Gui[4], Huishuai Zhang[4], Yi Chang[1†], Jiang Bian[4†]
[1]School of Artificial Intelligence, Jilin University, Changchun, China
[2]University of Illinois Urbana-Champaign, Champaign, United States
[3]Renmin University of China, Beijing, China
[4]Microsoft Research, Beijing, China
yeht2118@mails.jlu.edu.cn, liu326@illinois.edu, xinyi_shen@ruc.edu.cn,
{Wei.Cao, shun.zheng, xiaofangui, huishuai.zhang, jiang.bian}@microsoft.com, yichang@jlu.edu.cn

***Abstract*— Unsupervised Anomaly Detection (UAD) is a key data mining problem owing to its wide real-world applications. Due to the complete absence of supervision signals, UAD methods rely on implicit assumptions about anomalous patterns (e.g., scattered/sparsely/densely clustered) to detect anomalies. However, real-world data are complex and vary significantly across different domains. No single assumption can describe such complexity and be valid in all scenarios. This is also confirmed by recent research that shows no UAD method is omnipotent [1]. Based on above observations, instead of searching for a magic universal winner assumption, we seek to design a general UAD Booster (UADB) that empowers any UAD models with adaptability to different data. This is a challenging task given the heterogeneous model structures and assumptions adopted by existing UAD methods. To achieve this, we dive deep into the UAD problem and find that compared to normal data, anomalies (i) lack clear structure/pattern in feature space, thus (ii) harder to learn by model without a suitable assumption, and finally, leads to (iii) high variance between different learners. In light of these findings, we propose to (i) distill the knowledge of the source UAD model to an imitation learner (booster) that holds no data assumption, then (ii) exploit the variance between them to perform automatic correction, and thus (iii) improve the booster over the original UAD model. We use a neural network as the booster for its strong expressive power as a universal approximator and ability to perform flexible post-hoc tuning. Note that UADB is a model-agnostic framework that can enhance heterogeneous UAD models in a unified way. Extensive experiments on over 80 tabular datasets demonstrate the effectiveness of UADB. To facilitate further research, code, figures, and datasets are available at UADB's Github repository[1].**

***Index Terms*—unsupervised anomaly detection, unsupervised learning, outlier detection**

I. Introduction

Anomaly detection (AD), also known as Outlier Detection, aims to identify the data objects or behaviors that significantly deviate from the majority [2]. AD is considered as a crucial machine learning problem and has been researched in a variety of fields, including Web and Cyber Security (intrusion detection), Social Network Mining (malicious user/news discovery), and Healthcare (rare disease diagnosis) [3]. Anomalies in data can be translated into significant actionable information in a wide range of application domains. For example, in computer systems, unusual behaviors may show the presence of malicious activities; in clinical medicine, abnormal MRI images may indicate the presence of a malignant tumor [4].

For its wide applications, AD has been an active research area for several decades [5], [6], and numerous algorithms have been proposed for AD, including supervised, semi-supervised and unsupervised methods [1], [7]. However, ground-truth labels usually need to be manually annotated by domain experts, which is both expensive and time-consuming, and accurately marking all types of abnormal samples is usually unaffordable in practice [3]. Hence, the unsupervised anomaly detection (UAD) methods are the most widely applicable techniques as they do not require any label information [3]. This makes UAD a longtime research hotspot in the field, and new studies continue to appear in recent years [8]. In this paper, we focus on *unsupervised anomaly detection on tabular datasets*, which is a very challenging problem and has been the focus of most related works in the literature [1], [6], [9]–[13].

However, despite the extensive research efforts that have been made to address this problem, there still does not exist a single universal winner solution that consistently outperforms other counterparts due to the multifaceted complexity of the task [1]. Specifically, to achieve accurate UAD on tabular datasets, one faces following fundamental challenges:

- **Unsupervision**: In UAD's problem setting, the label information is completely absent in the training phase. Since there are no supervision signals that can provide the model with prior knowledge about the anomalous pattern, UAD models can only detect potential anomalies by making implicit assumptions about the anomaly data instances [14].
- **Assumption Misalignment**: Common assumptions adopted by UAD methods include: (i) anomalies occur far from their closest neighbors (neighbor-based, e.g., [15], [16]); (ii) anomalies does not belong to any cluster/far away from their closest cluster centroid/belong to small and/or sparse clusters (clustering-based, e.g., [17]–[19]); (iii) anomalies occur in the low probability regions of a stochastic model (statistical-based, e.g., [20]–[22]). UAD methods get satisfactory performance when their assumptions hold true, but unfortunately, this is usually not the case in practice. For instance, statistical-based UAD assume that the data is generated from a particular distribution, which often does not hold for high dimensional real datasets. The

*The two authors contributed equally to this work. †Corresponding authors.
[1]https://github.com/HangtingYe/UADB

nearest neighbor/clustering-based approach makes specific assumptions about the distribution of anomalies, but they are incompatible and even conflict with each other. For above reasons, UAD assumptions can be easily violated in real-world data and result in suboptimal performance [3].

- **Data Heterogeneity**: Furthermore, the heterogeneity, diversity, and complexity of tabular data also pose several great challenges to UAD. Unlike image/text/graph data that has natural contexts between pixels/channels/words/nodes, tabular data has no such shared contextual attributes that can help detecting anomalies. In tabular datasets, features are not explicitly linked to each other and often show heterogeneity: they may vary significantly in value distribution, range, and even space (e.g., continuous vs. categorical features) [23]. Therefore, even if a UAD method works well in one specific tabular dataset, its underlying assumptions are unlikely to hold in another tabular UAD task, since the two datasets can have very different characteristics. This is also confirmed by previous research efforts [1], [7] which show that *there is no universal winner for all UAD tasks*.

For the above reasons, we believe that the way to better UAD is not to look for a universal winner assumption which is unlikely to exist. To achieve generally better UAD on diverse and heterogeneous tabular data, the key is to go **beyond the static assumptions and empower the models with adaptability to different data**. In this direction, we propose our solution based on two key motivating observations: (i) *the power of proper assumptions* and (ii) *the high variance of anomalies*. Specifically, (i) [1] has shown that, with a proper data assumption, an unsupervised AD method can beat label-informed semi-supervised AD techniques. This indicates that the data assumption is a powerful tool for detecting specific types of anomalies and therefore should not be discarded outright, but it still needs to be adaptively enhanced so as to find anomalies that do not fit the assumption. (ii) Besides, unlike classification tasks where each class has a unique underlying distribution, in anomaly detection, anomalies are just irregular instances with no clear structure/pattern in the feature space. Therefore, compared to normal samples, anomalies are harder to learn by a model and are likely to induce a high variance, i.e., different models' predictions of anomalies will vary, see an example in Fig. 1. Such property can be used to support the adaptive augmentation of UAD models.

In light of these analysis, we propose UADB (Unsupervised Anomaly Detection Booster), a surprisingly simple yet effective framework that can *boost the prediction accuracy of any mainstream UAD methods on all types of tabular datasets*. Specifically, given any UAD model $f_T(\cdot)$, we introduce a booster model $f_B(\cdot)$, then iteratively (i) train f_B in a supervised manner using f_T's predictions as pseudo labels $\hat{\mathbf{y}}$; (ii) estimate sample variances $\mathbf{v}$ using the output of f_T and f_B; (iii) perform error correction by exploiting the sample variance. Note that having the observation in Fig. 1, the error correction can be simply done by adding sample variance to the pseudo labels. This will result in a large increase in the anomaly score

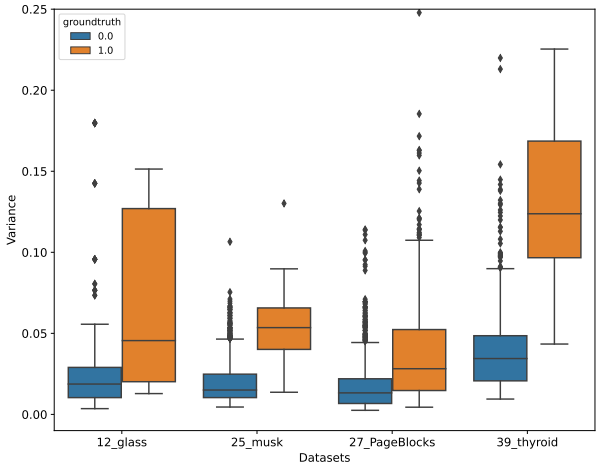


Fig. 1: The sample variance of normal (blue) and abnormal (orange) instances in real-world datasets, estimated by an UAD model IForest [24] and its imitation learner (a MLP trained with IForest's outputs as pseudo labels). The sample variance is the variance of MLP's prediction and IForest's output, and it is calculated per instance. Here, groundtruth=1.0 stands for anomalies, while groundtruth=0.0 stands for inliers. Anomalies consistently demonstrate higher variance across different datasets due to their weak structure/pattern in feature space.

(higher score indicates higher confidence that the sample is an anomaly) for false negatives (mispredicting anomalies) but a small increment for false positives (mispredicting normal samples), thus gradually narrowing the prediction gap between them, and finally altering their rankings in the pseudo labels to achieve error correction. We thus obtain a booster model f_B that is improved over its prototype f_T as it benefits from both knowledge distillation and adaptive error correction. We highlight that UADB does not make any assumptions about the input UAD prototype f_T, it is a model-agnostic framework that can generally enhance any f_T in a unified way. Extensive experiments and analyses conducted on over 80 tabular UAD datasets demonstrate the effectiveness of UADB.

To sum up, this paper makes following 3-fold contributions:

- We investigate the key challenges of Unsupervised Anomaly Detection (UAD) on tabular data, such as assumption misalignment and data heterogeneity. They prompt us to explore a new direction: empower static-assumption-based UAD models with adaptability to different data.
- We propose UADB, a model-agnostic framework that can effectively boost any UAD model's performance on tabular datasets via knowledge distillation and adaptive error correction. To our best knowledge, UADB is the first of its kind as a general augmentation framework for UAD models.
- We conducted extensive experiments on more than 80 tabular datasets along with comprehensive analysis and visualization. These results validate the effectiveness of the proposed UADB framework and may provide valuable insights for further research on versatile UAD boosters.

The rest of this paper is organized as follows: Section II re-

views closely related works in unsupervised anomaly detection and knowledge distillation. Section III introduces the notations and describes the proposed UADB framework. Section IV presents the experimental results as well as related discussions and analysis. And finally, section V concludes the paper.

II. Related Work

In this section, we provide a systematical review of the existing works related to unsupervised anomaly detection and knowledge distillation applications.

A. Unsupervised Anomaly Detection

Anomaly detection is a big topic in machine learning, including supervised, semi-supervised, and unsupervised methods. Due to the advantage of not requiring ground-truth labels, unsupervised anomaly detection methods are widely used. In the setting of unsupervised anomaly detection, we have no prior knowledge about which type of data is normal or which is abnormal, i.e. the training data is without true labels. Our task is to find instances that deviate the most from the other instances among all dataset [25]. Since unsupervised anomaly detection has drawn interest in the academic community [26], [27], numerous unsupervised anomaly detection methods have been proposed. These methods could be roughly grouped into shallow and deep methods, with the details as follows.

We list some representative shallow methods: (i) Isolation Forest (IForest) [24] builds an ensemble of trees for a given data set, then use the distance of instance to the root as anomaly score; (ii) Histogram-based Outlier Score (HBOS) [28]. The basic assumption is that the dataset's dimensions are independent. Each dimension would be divided into intervals. The higher density represents the lower anomaly score; (iii) Empirical-Cumulative-distribution-based Outlier Detection (ECOD) [29] first computes the empirical cumulative distribution for each dimension of the input data. Then for each dimension, ECOD aggregates the tail probabilities to compute the anomaly score.

Some representative deep methods are as follows: (i) Deep Support Vector Data Description (DeepSVDD) [30] trains a neural network while minimizing the volume of a hypersphere that encloses the network representations of the data, and the distance of the transformed embedding to the hypersphere's center is used to calculate the anomaly score; (ii) Deep Autoencoding Gaussian Mixture Model (DAGMM) [12] jointly optimizes the parameters of the deep autoencoder and the mixture model simultaneously in an end-to-end fashion, leveraging a separate estimation network to help with the parameter learning of the mixture model. The joint optimization eliminates the need for pre-training by assisting the autoencoder escape from less attractive local optima and further reducing reconstruction errors.

B. Knowledge Distillation

Knowledge Distillation (KD) is a family of techniques that aim to transfer knowledge from a trained source (teacher) model(s) to a target (student) model [31]. The student models are usually smaller but perform similarly or even better than the large teacher models. Such a training scheme is also known as the teacher-student architecture and has been proven to be effective in numerous applications [32], [33]. Starting from the success in image classification [34]–[36] and other visual recognition tasks [37], [38], more knowledge distillation systems are designed for broader applications such as neural machine translation [39], [40], feature selection [41], [42], text generation [43], speech recognition [44]–[46] and so on.

A number of research efforts have explored the application of knowledge distillation to anomaly detection tasks. *Salehi et al.* [47] tried to explore the multi-layer feature information in distillation, so as to better exploit teacher model's multi-resolution knowledge and get a better student network for image anomaly detection task. Also for image anomaly detection, *Bergmann et al.* [48] proposed to include multiple student networks and learn from the teacher model's knowledge in an ensemble mannner. *Wang et al.* [49] designed a multi-scale feature matching strategy to enable student learning with hierarchical supervision, thus improve the detection accuracy of anomalies of various sizes in images. Motivated by the fact that practitioners often build a large number of UAD models rather than a single model for reliable further combination and analysis, *Zhao et al.* [50] developed a system for accelerating UAD with large-scale heterogeneous models. They train a pseudo-supervised simple regressor student model to approximate the large ensemble of heterogeneous UAD models, thus accelerating the inference.

We note that there are several fundamental differences between our proposed UADB framework and the aforementioned KD + UAD techniques: Many methods are specifically designed for distilling knowledge from large multi-layere neural networks (e.g., [47], [49]), and for a specific AD task, such as detecting pixel-level anomaly objects in images. While in this work, we consider a more general case, to improve any teacher UAD model on heterogeneous tabular datasets. Furthermore, existing works directly transfer the knowledge from the teacher model without modification and let the student mimic the teacher's behaviors. In UADB, we estimate the variance using the discrepancy between teacher and student and exploit this information to adjust the pseudo labels, thus achieving adaptive error correction in knowledge distillation. The details of UADB will be covered in the next section.

III. Methodology

In this section, we first introduce the notations and formalize the unsupervised anomaly detection booster problem considered in this paper. We then demonstrate why anomalies are likely to have high variance and show empirical evidence collected from 80 tabular datasets. After that, we present how UADB exploits information from both the teacher model and the variance to achieve effective knowledge transfer and error correction. Finally, we formalize UADB in Algorithm 1, and Fig 3 gives an overview of the UADB framework.

TABLE I: Notation Definitions

Notation	Definition
d	Number of data features
$\mathbf{x} : [x_1, x_2, \cdots, x_d]$	A data sample
n	Number of data samples
$\mathbf{X} \in \mathbb{R}^{n \times d}$	A UAD dataset
$f_S(\cdot) : \mathbf{x} \to \mathbb{R}^{[0,1]}$	Source UAD (teacher) model
$f_B(\cdot;\theta) : \mathbf{x} \to \mathbb{R}^{[0,1]}$	Target Booster (student) model
θ	Parameters of the booster model
T	Number of student training steps
$\hat{y}_i$	Pseudo label of the i-th sample
$\hat{\mathbf{y}} : [\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_n] \in \mathbb{R}^n$	Pseudo label vector
$\hat{\mathbf{y}}^{(t)} : [\hat{y}_1^{(t)}, \hat{y}_2^{(t)}, \cdots, \hat{y}_n^{(t)}] \in \mathbb{R}^n$	Pseudo label vector at t-th iteration

A. Notations and problem definition

Notations. We first introduce the notations used in this paper. Let d be the number of input features, a data sample can be represented by its feature vector $\mathbf{x} : [x_1, x_2, \cdots, x_d]$, then we can denote a dataset for UAD as $\mathbf{X} \in \mathbb{R}^{n \times d}$, where n is the number of samples in the dataset. Note that there is no ground truth label y in the unsupervised setting. The goal of unsupervised anomaly detection is to learn a detection model $f(\cdot)$ without ground truth labels. A model $f(\cdot)$ takes a feature vector $\mathbf{x}$ as input and outputs $\hat{y} \in \mathbb{R}^{[0,1]}$, i.e., the predicted anomaly score of $\mathbf{x}$, higher score indicates higher confidence that $\mathbf{x}$ is an anomaly.

Problem Definition. In this paper, we consider the problem of finding a booster model $f_B(\cdot)$ for a given source UAD model $f_S(\cdot)$. As described before, this is achieved by iterative knowledge distillation with error correction by estimating and exploiting the variance between teacher and student models. Formally, given a source UAD model $f_S(\cdot) : \mathbf{x} \to \mathbb{R}^{[0,1]}$, we consider a parameterized booster model $f_B(\cdot;\theta) : \mathbf{x} \to \mathbb{R}^{[0,1]}$ with θ denoting its parameters. The goal of learning a UAD booster model is to find a parameter set θ^* that maximize $f_B(\cdot;\theta)$'s prediction accuracy on $\mathbf{X}$ given the source UAD model $f_S(\cdot)$ and dataset $\mathbf{X}$.

More specifically, we use the predictions of the source model $f_S(\mathbf{X})$ as the initial pseudo label vector $\hat{\mathbf{y}}$ for knowledge distillation. The target of distillation, booster model $f_B(\cdot;\theta)$ is a neural network. Note that unlike typical knowledge distillation settings, the source knowledge, i.e., $\hat{\mathbf{y}}$ will be adjusted for error correction during the booster model $f_B(\cdot;\theta)$'s pseudo-supervised training process. Suppose the total number of training steps is T, we denote the adjusted pseudo label vector at step/iteration t as $\hat{\mathbf{y}}^{(t)}$. Let $\mathcal{L}$ be the loss function, then in each training step, we need to (i) optimize θ to minimize $\mathcal{L}(f_B(\mathbf{X};\theta), \hat{\mathbf{y}}^{(t)})$ for knowledge transfer, and (ii) update $\hat{\mathbf{y}}^{(t)}$ for error correction. The former is a standard supervised learning objective that can be handled by many optimizers, but the latter objective has no straightforward solution. We will introduce our solution in the following sections.

B. Anomalies pattern on variance

Motivations. In order to perform error correction, we need information or statistics that can distinguish between abnormal and normal samples. Normally such information can be obtained from (partial) ground truth labels or prior knowledge about the anomalies' pattern provided by domain experts. But unfortunately, none of them is available in UAD. We have to find a new discriminative feature for error correction.

To achieve this, we look back to the fundamental difference between anomaly detection tasks and classification tasks. In classification, each class has a unique underlying distribution/structure in the feature space, and classifiers can distinguish them by learning the difference between class distributions. While in anomaly detection, only normal data has a meaningful underlying structure, and anomalies are just "abnormal" instances without a clear pattern, as they can be caused by multiple unknown, hidden, and even random factors (e.g., data corruption, sensor failure) [14]. This is also the reason why one-class learning (only learning from the pattern of normal data) obtains great success in anomaly detection.

Now we know that compared to normal samples, anomalies lack a clear structure/pattern in the feature space, which can be a sharp knife for distinguishing anomalies. Specifically, data samples that lack a clear pattern usually have high variance in predictions since they are hard to fit by a simple hypothesis (model). Therefore, anomalies should likely have high variances, i.e., different models' predictions for one anomaly may vary significantly, and this property can be used as a discriminative feature to support error correction.

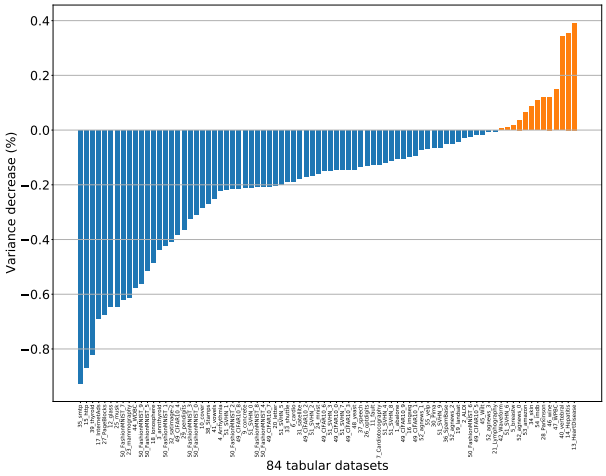


Fig. 2: Comparison of average variance of normal samples and anomalies on 84 different tabular datasets. Negative value indicates that anomalies have higher average variance compared to normal samples, which holds true on 71 out of 84 datasets.

Empirical Evidence. To verify whether the variance can be used for error correction, we examine 84 real-world tabular datasets (described in Table III) to collect some empirical evidence. Specifically, for each dataset, we train a teacher UAD model f_S on it, then use its predictions as static labels to train a pseudo-supervised student model f_B. We estimate the sample variance by measuring the difference between the predictions of the teacher and the student model $\hat{v}_i = variance([f_S(\mathbf{x}_i), f_B(\mathbf{x}_i)])$ ($\mathbf{x}_i$ is the i-th

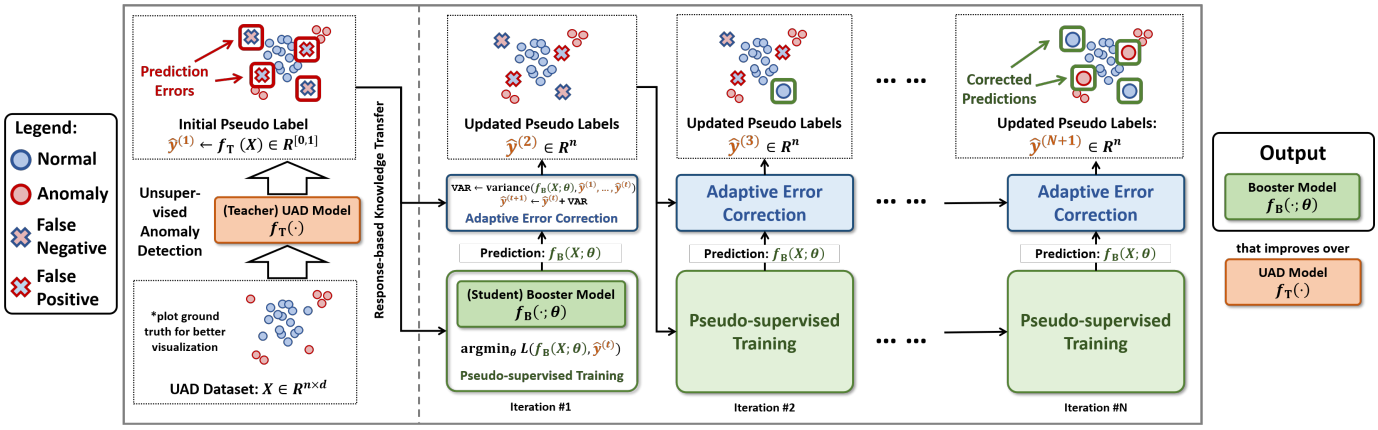


Fig. 3: Overview of the proposed UADB framework. Best viewed in color.

sample). We then compute the average variance of normal samples and anomalies, respectively, i.e., $\hat{v}_{\text{normal/abnormal}} = \sum_{i \in \text{normal/abnormal}}(\hat{v}_i)/S$, S is a normalization term that equals to the number of normal/abnormal instances. To improve the visual display of the results, we show the relative average variance difference between normal and anomalies, i.e., $(\hat{v}_{\text{normal}} - \hat{v}_{\text{abnormal}})/\hat{v}_{\text{abnormal}}$. Negative value indicates that anomalies have higher average variance than normal samples. Results are shown in Fig. 2.

As observed in Fig. 2, compared to normal samples, anomalies have higher average variance on 85% (71/84) tabular datasets, which directly validates our previous thoughts. The relative differences are considerably high ($> 5\%$) on 60/84 datasets. We note that the *variance can be naturally estimated in the teacher-student architecture*, between teacher and student model and/or student model checkpoints at different steps. This allows us to exploit the variance difference between normal samples and anomalies, achieving dynamic error correction during knowledge distillation.

C. Knowledge transfer and error correction

UADB Procedure. With the above analysis, we now elaborate on the technical details of UADB training procedure. As discussed before, appropriate data assumptions are powerful tools for detecting specific types of anomalies and therefore should not be discarded outright, but they need to be enhanced with adaptive error correction so as to handle anomalies that do not fit the assumption. Accordingly, UADB is designed to:

- keep the prior knowledge of the UAD model and its assumption by knowledge transfer;
- perform adaptive error correction during transfer by exploiting the sample variance at the same time.

Specifically, given a source UAD model $f_S(\cdot) : \mathbf{x} \rightarrow \mathbb{R}^{[0,1]}$ fitted on dataset $\mathbf{X}$, we consider a parameterized booster model $f_B(\cdot;\theta) : \mathbf{x} \rightarrow \mathbb{R}^{[0,1]}$ with θ denoting its parameters. In UADB, we use a neural network as the parameterized booster model for its strong expressive power as a universal approximator, this is important for handling diverse UAD models with different architectures. We use the source model's predictions as the initial pseudo label vector $\hat{\mathbf{y}}^{(1)}$, i.e., $\hat{\mathbf{y}}^{(1)} = f_S(\mathbf{X})$. Then in t-th iteration, repeat:

- Update θ w.r.t objective $\text{argmin}_\theta \mathcal{L}(f_B(\mathbf{X};\theta), \hat{\mathbf{y}}^{(t)})$;
- Estimate variance vector $\hat{\mathbf{v}}$ using $f_B(\mathbf{X};\theta)$ and all previous pseudo label vectors $\widehat{\mathbf{Y}} : \{\hat{\mathbf{y}}^{(1)}, \cdots, \hat{\mathbf{y}}^{(t)}\}$ (calculated per instance);
- Update the pseudo label simply by:
  - Adding variance $\hat{\mathbf{v}}$ to the pseudo label $\hat{\mathbf{y}}^{(t)}$
  - Renormalize to guarantee $\hat{y}_i \in \mathbb{R}^{[0,1]}, \forall i$
  - i.e., $\hat{\mathbf{y}}^{(t+1)} = \text{MinMaxScale}(\hat{\mathbf{y}}^{(t)} + \hat{\mathbf{v}})$

Note that the error correction mechanism in UADB is surprisingly simple. More complex error correction mechanisms can be designed in many different ways, but following Occam's razor principle of parsimony, we prefer a simpler solution for its elegance, interpretability, and wide applicability. Algorithm 1 formalizes the proposed UADB framework.

Algorithm 1 Unsupervised Anomaly Detection Booster

Input: UAD dataset $\mathbf{X}$, source UAD model $f_S(\cdot) : \mathbf{x} \rightarrow [0,1]$, number of booster training steps T

Initialize:
target booster model $f_B(\cdot;\theta)$
pseudo label vector $\hat{\mathbf{y}}^{(1)} \leftarrow f_S(\mathbf{X})$
pseudo label matrix $\widehat{\mathbf{Y}} \leftarrow [\hat{\mathbf{y}}^{(1)}] \in \mathbb{R}^{n \times 1}$
for $t \leftarrow 1$ to T **do**
  train $f_B(\cdot;\theta)$ by $\text{argmin}_\theta \mathcal{L}(f_B(\mathbf{X};\theta), \hat{\mathbf{y}}^{(t)})$
  compute variance $\hat{\mathbf{v}} \leftarrow variance([\widehat{\mathbf{Y}}, f_B(\mathbf{X};\theta)])$ [2]
  update $\hat{\mathbf{y}}^{(t+1)} \leftarrow \text{MinMaxScale}(\hat{\mathbf{y}}^{(t)} + \hat{\mathbf{v}})$
  update $\widehat{\mathbf{Y}} \leftarrow [\widehat{\mathbf{Y}}, \hat{\mathbf{y}}^{(t+1)}] \in \mathbb{R}^{n \times (t+1)}$
end for
return the booster UAD model $f_B(\cdot;\theta)$

Variance for Error Correction. To better illustrate why this simple pseudo-label updating rule can achieve error correction, we now present related discussions and case studies.

As demonstrated before, compared to normal data instances, anomalies are likely to have higher prediction variance. Since

[2] Calculated per instance.

UADB adopts a teacher-student architecture for booster model training, the variance can be naturally estimated using the source model f_S and the target model f_B. However, we do not simply compute the variance using only the 2 entries from f_S and f_B, since the student model is being updated during the training process, which makes the variance estimation vulnerable to unknown training dynamics. Inspired by the concept of self-teaching in knowledge distillation research [51], we record all the pseudo-label vectors ($\hat{\mathbf{y}}$) in previous steps and compute the variance on all $\hat{\mathbf{y}}$ and the current student prediction $f_B(\mathbf{X};\theta)$. This provides us with a more reliable variance estimation by including more predictions from model checkpoints, while mitigating the influence of random training dynamics by using previous $\hat{\mathbf{y}}$s instead of student model outputs. But even with reliable variance estimation $\hat{\mathbf{v}}$, why UADB can correct errors by simply updating $\hat{\mathbf{y}}^{(t+1)} \leftarrow \text{MinMaxScale}(\hat{\mathbf{y}}^{(t)} + \hat{\mathbf{v}})$?

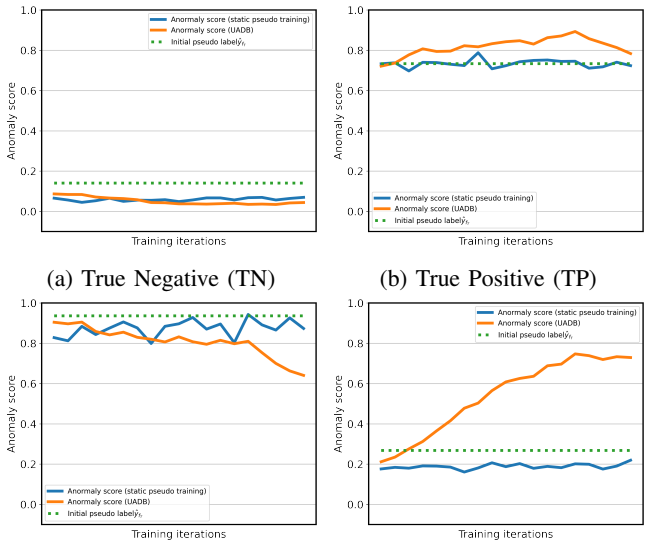
Case Study. Let us consider four cases for the source UAD model $f_T(\cdot)$ (without loss of generality, we assume anomalies are positive, i.e., $y = 1$ in the ground truth label and the detection threshold is 0.5): (i) True Positive (TP): $y = 1, \hat{y}_{f_T} = 1$; (ii) False Negative (FN): $y = 1, \hat{y}_{f_T} = 0$; (iii) False Positive (FP): $y = 0, \hat{y}_{f_T} = 1$; (iv) True Negative (TN): $y = 0, \hat{y}_{f_T} = 0$. Table II summarizes the four cases.

TABLE II: Four types of instances in UAD.

Case	Ground Truth	Label	Prediction	Variance
TP	abnormal	$y = 1$	$\hat{y}_{f_T} = 1$	high
FN	abnormal	$y = 1$	$\hat{y}_{f_T} = 0$	high
FP	normal	$y = 0$	$\hat{y}_{f_T} = 1$	low
TN	normal	$y = 0$	$\hat{y}_{f_T} = 0$	low

It is straightforward to observe from the Table II that the goal of error correction is to correct FP and FN in the pseudo labels. Specifically, in the booster model $f_B(\cdot)$, we want to increase its prediction of FN while decreasing its prediction of FP. Also, note that the variance of FN is more likely to be higher than FP, so intuitively, adding the respective variance to the pseudo-label will naturally reduce the error gap between FN and FP. This procedure can be repeated until the gap between FN and FP is eliminated and their relative relationship is inverted in the pseudo-label values, i.e., errors are corrected. We now discuss these cases in detail.

1) True Positive (TP) and True Negative (TN): For a TP instance $\mathbf{x}_{\text{TP}}$, the initial pseudo label $\hat{y}_{\text{TP}} = f_T(\mathbf{x}_{\text{TP}})$ is close to 1, and it has a high variance $\hat{v}_{\text{abnormal}}$. After adding the variance, $\hat{y}_{\text{TP}}(\to 1) + \hat{v}_{\text{abnormal}}(> \hat{v}_{\text{normal}})$ is likely to be a large value that greater than any other types of instances (FN, FP, TN), so after min-max scaling, its new pseudo label $\hat{y}^*_{\text{TP}}$ will still be close to 1. Oppositely, for a TN instance $\mathbf{x}_{\text{TN}}$, the initial pseudo label $\hat{y}_{\text{TN}} = f_T(\mathbf{x}_{\text{TN}})$ is close to 0, and it has a low variance $\hat{v}_{\text{normal}} < \hat{v}_{\text{abnormal}}$. Therefore, after adding its variance, $\hat{y}_{\text{TN}}(\to 0) + \hat{v}_{\text{normal}}(< \hat{v}_{\text{abnormal}})$ will still be the smallest values among all cases. After min-max scaling, its new pseudo label $\hat{y}^*_{\text{TN}}$ will still be close to 0. Thus the correct knowledge in $f_T(\cdot)$ will be maintained.



(a) True Negative (TN) (b) True Positive (TP)

(c) False Positive (FP) (d) False Negative (FN)

Fig. 4: Error correction during the UADB training process. We compare UADB's behavior (orange line) with a variant that learns a student model by static pseudo-supervised training without error correction (blue line). The dashed line indicates the initial pseudo label $\hat{y}_{f_T} \in [0, 1]$. We can observe that the student model without error correction simply mimics the teacher model's behavior (including errors), while UADB can gradually correct the booster's predictions on FP (Fig. 4c) and FN (Fig. 4d) by exploiting the variance difference between normal and abnormal instances.

2) False Positive (FP) and False Negative (FN): The key function of error correction is to correct the pseudo labels of FPs and FNs. Specifically, for a FP instance $\mathbf{x}_{\text{FP}}$, its ground truth label $y_{\text{FP}} = 0$ but the pseudo label $\hat{y}_{\text{FP}} = f_T(\mathbf{x}_{\text{FP}})$ is close to 1. Thus we want to gradually decrease its score for error correction and expect a smaller score after updating, i.e., $\hat{y}^*_{\text{FP}} < \hat{y}_{\text{FP}}$. To prove this, let's jointly consider FP and TP. After adding variance, we have the unnormalized scores, which is $s_{\text{FP}} : \hat{y}_{\text{FP}}(\to 1) + \hat{v}_{\text{normal}}(< \hat{v}_{\text{abnormal}})$ for FP and $s_{\text{TP}} : \hat{y}_{\text{TP}}(\to 1) + \hat{v}_{\text{abnormal}}(> \hat{v}_{\text{normal}})$ for TP. Note that although the pseudo labels of FP and TP are both close to 1, after adding the variance term, FP's unnormalized score will be smaller than TP due to its low variance, i.e., $s_{\text{FP}} < s_{\text{TP}}$. Then after the scaling, we have $\hat{y}^*_{\text{FP}} = \frac{s_{\text{FP}} - s_{\text{TN}}}{s_{\text{TP}} - s_{\text{TN}}} < \hat{y}_{\text{FP}}(\hat{y}_{\text{FP}} \to 1)$, which means $\hat{y}_{\text{FP}}$ will decrease in updates. Likewise, a FN instance's updated unnormalized score will be larger than that of TN due to its larger variance term, i.e., $s_{\text{FN}} > s_{\text{TN}}$, thus $\hat{y}^*_{\text{FN}} = \frac{s_{\text{FN}} - s_{\text{TN}}}{s_{\text{TP}} - s_{\text{TN}}} > \hat{y}_{\text{FN}}(\hat{y}_{\text{FN}} \to 0)$. Hence, the error gap between FN and FP can be narrowed by repeatedly applying this updating rule, thus finally achieving error correction.

To validate previous analysis, for each case, we show an example of how the booster's prediction changes during the UADB training process. Please see details in Fig. 4.

IV. Experiments & Analysis

In this section, we conduct comprehensive experiments to validate the effectiveness of the proposed UADB framework. We first introduce the experiment setup and the included 14 mainstream UAD models and 84 real-world datasets. To provide an intuitive understanding of UADB, we construct synthetic datasets with different types of anomalies, and visualize the behaviors of example UAD models and their boosters. After that, we test UADB on all 84 real-world tabular datasets and present the empirical results and corresponding analysis. Finally, we conduct ablation study and further compare UADB with several intuitive booster frameworks.

A. Experiment Setup Details.

Source UAD Models. As the UADB is a model-agnostic framework, we include 14 mainstream UAD techniques to test UADB's applicability and effectiveness to different UAD models with diverse architecture. These UAD algorithms include IForest [24], HBOS [28], LOF [52], KNN [53], PCA [54], OCSVM [55], CBLOF [56], COF [57], SOD [58], ECOD [29], GMM [59], LODA [60], COPOD [61] and DeepSVDD [30]. More details could be found in UADB's Github repository[3].

The aforementioned 14 UAD methods are widely used in practice and vary significantly in terms of both methodologies (e.g., neighbor-based [52], [53], [57], clustering-based [56], [59], density-based [28], [29]) and model architectures (e.g., tree [24], support vector machine [55], neural network [30]). Thus they can be used to perform a comprehensive test of the effectiveness and applicability of UADB.

Real-world Datasets. As described before, UAD on tabular datasets are challenging due to the heterogeneity, complexity, and diversity of tabular data. Table III shows the statistics of the 84 heterogeneous tabular datasets that are included for a comprehensive evaluation. Note that, in addition to the native tabular datasets (i.e., from abalone to yeast), datasets from larger and high-dimensional CV and NLP tasks are also included. However, as many UAD models such as IForest [24] and OCSVM [55] cannot directly handle CV or NLP task, we follow previous work [1] and use a CV/NLP feature extractor to generate tabular versions of these datasets. These datasets vary significantly in properties and application domains, including health care (e.g., disease diagnosis), finance (e.g. credit card fraud detection), image processing (e.g. object identification), language processing (e.g. speech recognition) and more.

UADB Setup. As mentioned before, UADB adopt a neural network as the booster model for its strong expressive power as a universal approximator and ability to perform flexible post-hoc tuning. We fix the parameter of source UAD model, and only optimize and keep the booster $f_B(\cdot;\theta)$ as the final UAD model. Specifically, the booster model is a simple 3-layer fully-connected MLP (Multi-layer Perceptron) with 128 neurons in each hidden layer. In t-th training step/iteration of UADB, the booster $f_B(\cdot;\theta)$ is updated w.r.t objective $\operatorname{argmin}_\theta \mathcal{L}(f_B(\mathbf{X};\theta), \hat{\mathbf{y}}^{(t)})$ for 10 epochs with batch size set to 256, optimized by Adam optimizer with a learning rate of 0.001, and the total number of UADB training steps $T = 10$. In addition, to prevent the booster model from overfitting the source model, we train 3 booster models in a 3-fold cross-validation manner (i.e., each model was trained on different 2 out of 3 splits of data and pseudo-labels). At inference time, we average the outputs of the 3 booster models as the final predictions. To reduce the effect of randomness, the reported performance is averaged over 10 independent runs.

Evaluation Metric & Implementation Details. Following the common practice of previous works, we use *Area Under the Curve of Receiver Characteristic Operator* (AUCROC) and *Average Precision* (AP) to evaluate UAD models' performance. Larger AUCROC/AP score indicates better detection performance. Following [1], we use the popular PyOD Python package [62] to implement all source UAD models, and apply their default parameter settings in PyOD[4]. The booster model is implemented using the PyTorch [63] framework.

B. Visualization on Synthetic Datasets

Before diving into the results on real-world datasets, we first show some visualizations on synthetic datasets to provide an intuitive understanding of how UADB works and improves over the original model. Previous research [1], [64] have demonstrated that anomalies in real-world applications could be roughly divided into four specific types, i.e. clustered, global, local, and dependency anomalies. Accordingly, we generate 4 synthetic datasets, each contains a specific type of anomaly, as shown in the rows of Fig. 5. For each type of anomaly, we select two UAD models that generally perform best on the given anomaly, then apply UADB to get two booster models, and show their predictions and errors in Fig. 5.

It can be observed that UADB generally maintains the predictions of the teacher model by knowledge transfer. But more importantly, the booster models benefit from the adaptive correction mechanism and thus are able to correct the errors when learning from the teacher model, e.g., the false positives in IForest-Clustered (1th-row left) and the false negatives in HBOS-Global (2nd-row right). On average, UADB achieves 38.94% error correction rate on all 8 Model-Anomaly pairs, with a maximum at 86.36% on IForest-Clustered (1st-row left), where 38 out of 44 errors of the source IForest model are corrected in its booster. These results show that UADB is able to handle different type of anomalies, and can make improvements even over the best-performing UAD models.

C. Results & Analysis on Real-world Datasets

In this section, we carry out extensive experiments on 84 heterogeneous real-world datasets with 14 different mainstream UAD models. The results can provide a comprehensive evaluation of the effectiveness of UADB in real-world UAD tasks. The statistics of the 84 heterogeneous tabular datasets are shown in Table III. Specifically, we aim to answer the following research questions (**RQ**s) in this section:

[3]https://github.com/HangtingYe/UADB

[4]Please refer to https://pyod.readthedocs.io/en/latest/pyod.models.html.

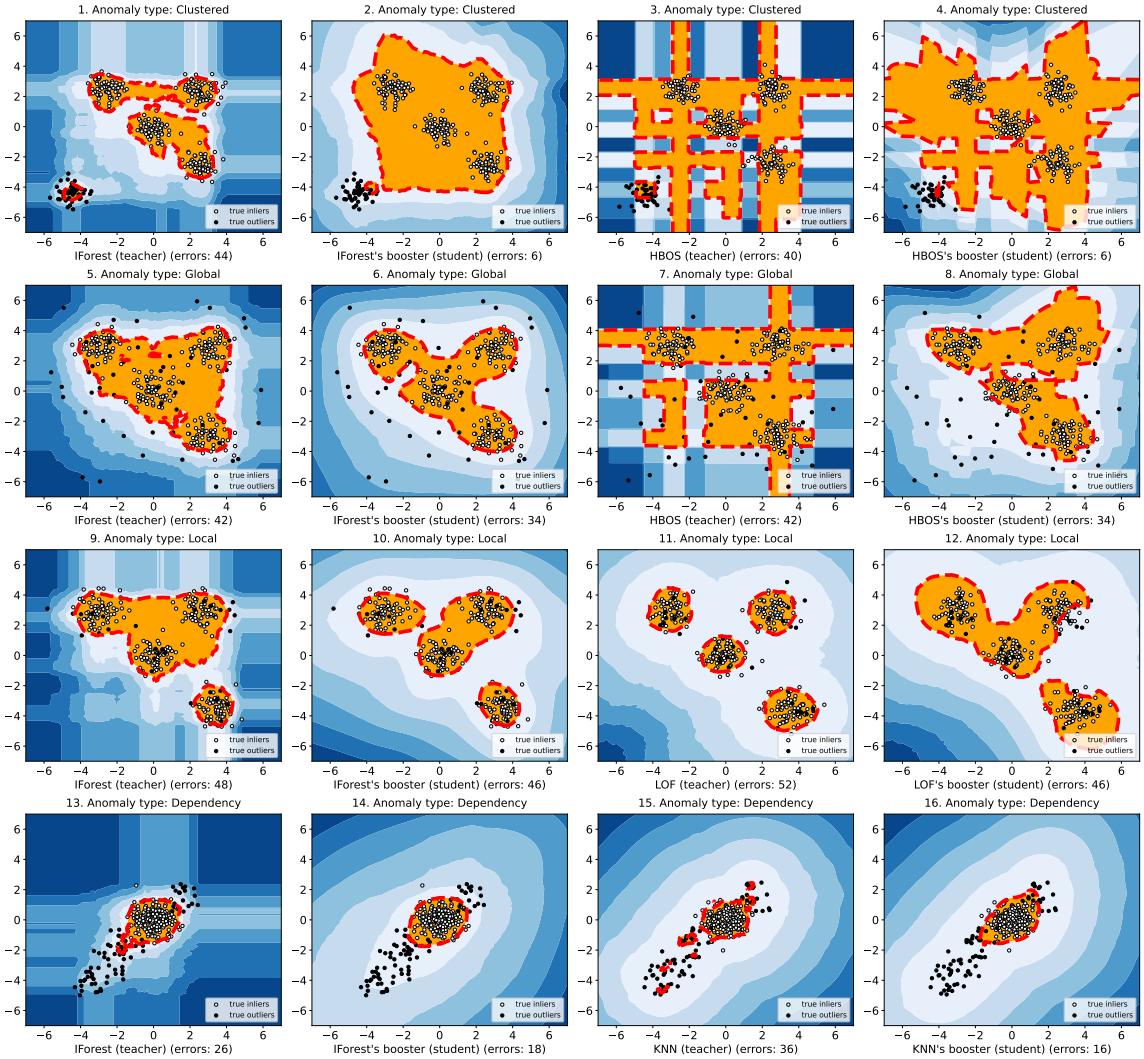


Fig. 5: Visualization of the detection results of UAD models and their UADB boosters when facing different types of anomalies. Specifically, we generate 4 synthetic datasets (in 4 rows) with different types of anomalies, i.e. Clustered, Global, Local and Dependency. For each dataset, we test two UAD models (1st & 3rd columns) that are best in handling the corresponding anomaly [1] and their boosters (2nd & 4th columns), we then plot their decision boundaries. In each figure, the orange/blue interface represents the normal/abnormal space predicted by the model. We can observe that UADB boosters can maintain the right decisions of the source model by knowledge transfer, while correcting the wrong predictions by adaptive error correction. UADB improves the UAD model's detection results on all 8 model-anomaly-type pairs, achieving up to 86% correction rate.

1) **RQ1**: Can UADB generally boost the performance of different UAD models in real-world applications?
2) **RQ2**: How does the error correction mechanism work? And what is the role of iterative training?
3) **RQ3**: How much did the iterative error correction mechanism contribute to UADB training?
4) **RQ4**: Can other intuitive mechanisms that exploit the variance also improve over UAD models?

Main Results (RQ1). First, we train the 14 different mainstream UAD models as well as their corresponding UADB boosters on all 84 tabular datasets. Note that this is a large-scale experiment that result in 14(models)×84(datasets)×2(metrics)×2(source & booster) = 4704 numerical results. We cannot fully display these results here due to space limitations. Therefore, we provide a summarization of the experimental results in Table IV, and part of the more detailed results are available in Table V.

From Table IV we can observe that:

- UADB achieves consistent improvements over the 14 source UAD models for 84 tabular datasets. It achieved a more than 1% average performance improvement for each model, whether using AUCROC or AP as the evaluation metric. In addition, we conduct Wilcoxon signed-rank test (with $\alpha = 0.05$) for each source UAD model and its booster over 84 datasets. In all settings, the improvement is statistically significant at the 95% confidence

TABLE III: Data description of the 84 real-world datasets.

Datasets	% Anomaly	Category	Datasets	% Anomaly	Category	Datasets	% Anomaly	Category	Datasets	% Anomaly	Category
abalone	49.82	Biology	**mammography**	2.32	Healthcare	**WDBC**	2.72	Healthcare	**FashionMNIST_6**	5.00	Image
ALOI	3.04	Image	**mnist**	9.21	Image	**Wilt**	5.33	Botany	**FashionMNIST_7**	5.00	Image
annthyroid	7.42	Healthcare	**musk**	3.17	Chemistry	**wine**	7.75	Chemistry	**FashionMNIST_8**	5.00	Image
Arrhythmia	45.78	Healthcare	**optdigits**	2.88	Image	**WPBC**	23.74	Healthcare	**FashionMNIST_9**	5.00	Image
breastw	34.99	Healthcare	**PageBlocks**	9.46	Document	**yeast**	34.16	Biology	**SVHN_0**	5.00	Image
cardio	9.61	Healthcare	**Parkinson**	75.38	Healthcare	**CIFAR10_0**	5.00	Image	**SVHN_1**	5.00	Image
Cardiotocography	22.04	Healthcare	**pendigits**	2.27	Image	**CIFAR10_1**	5.00	Image	**SVHN_2**	5.00	Image
concrete	50.00	Physical	**Pima**	34.90	Healthcare	**CIFAR10_2**	5.00	Image	**SVHN_3**	5.00	Image
cover	0.96	Botany	**satellite**	31.64	Astronautics	**CIFAR10_3**	5.00	Image	**SVHN_4**	5.00	Image
fault	34.67	Physical	**satimage-2**	1.22	Astronautics	**CIFAR10_4**	5.00	Image	**SVHN_5**	5.00	Image
glass	4.21	Forensic	**shuttle**	7.15	Astronautics	**CIFAR10_5**	5.00	Image	**SVHN_6**	5.00	Image
HeartDisease	44.44	Healthcare	**skin**	20.75	Image	**CIFAR10_6**	5.00	Image	**SVHN_7**	5.00	Image
Hepatitis	16.25	Healthcare	**smtp**	0.03	Web	**CIFAR10_7**	5.00	Image	**SVHN_8**	5.00	Image
http	0.39	Web	**SpamBase**	39.91	Document	**CIFAR10_8**	5.00	Image	**SVHN_9**	5.00	Image
imgseg	42.86	image	**speech**	1.65	Linguistics	**CIFAR10_9**	5.00	Image	**agnews_0**	5.00	NLP
InternetAds	18.72	Image	**Stamps**	9.12	Document	**FashionMNIST_0**	5.00	Image	**agnews_1**	5.00	NLP
Ionosphere	35.90	Oryctognosy	**thyroid**	2.47	Healthcare	**FashionMNIST_1**	5.00	Image	**agnews_2**	5.00	NLP
landsat	20.71	Astronautics	**vertebral**	12.50	Biology	**FashionMNIST_2**	5.00	Image	**agnews_3**	5.00	NLP
letter	6.25	Image	**vowels**	3.43	Linguistics	**FashionMNIST_3**	5.00	Image	**amazon**	5.00	NLP
Lymphography	4.05	Healthcare	**Waveform**	2.90	Physics	**FashionMNIST_4**	5.00	Image	**imdb**	5.00	NLP
magic.gamma	35.16	Physical	**WBC**	4.48	Healthcare	**FashionMNIST_5**	5.00	Image	**yelp**	5.00	NLP

TABLE IV: The detection performance improvement achieved by UADB over 14 source UAD models on 84 datasets. "**Original**" indicates the average score achieved by the corresponding source UAD model on all datasets. "**Improvement**" indicates the average score improvement achieved by the UADB booster over the UAD model, "**Improvement (%)**" indicates the average score improvement in percentage. "**Effects**" represents the number of datasets that UADB booster made improvements over the source model. "**P-value**" represents the results of the Wilcoxon signed-rank test (with $\alpha = 0.05$), "**P-value**" less than 0.05 indicates the improvement is statistically significant.

Source UAD Model		IForest	HBOS	LOF	KNN	PCA	OCSVM	CBLOF	COF	SOD	ECOD	GMM	LODA	COPOD	DeepSVDD
AUCROC	**Original**	0.7028	0.6848	0.6311	0.6794	0.6930	0.6750	0.7110	0.6105	0.6638	0.6866	0.7274	0.6571	0.6882	0.5346
	Improvement	0.0117	0.0153	0.0694	0.0452	0.0130	0.0241	0.0215	0.0580	0.0562	0.0144	0.0133	0.0293	0.0116	0.0979
	Improvement (%)	1.66	2.23	11.00	6.65	1.88	3.57	3.02	9.50	8.46	2.10	1.83	4.46	1.69	18.31
	Effects	49	59	51	46	53	58	51	57	57	57	47	62	52	68
	P-value	1.89e-2	1.99e-4	7.18e-4	8.51e-4	1.48e-3	2.61e-6	3.06e-3	5.71e-5	3.19e-6	1.31e-4	1.80e-2	4.62e-6	1.20e-2	3.53e-11
AP	**Original**	0.3012	0.2918	0.1903	0.2550	0.3051	0.2738	0.3057	0.1989	0.2322	0.2908	0.2805	0.2636	0.2832	0.1727
	Improvement	0.0134	0.0137	0.1146	0.0627	0.0010	0.0229	0.0184	0.0670	0.0742	0.0101	0.0283	0.0390	0.0146	0.0741
	Improvement (%)	4.45	4.69	60.22	24.59	0.32	8.36	6.02	33.69	31.96	3.47	10.09	14.80	5.16	42.91
	Effects	57	62	56	51	62	59	57	60	60	66	51	64	64	70
	P-value	1.77e-4	2.00e-6	1.21e-6	3.50e-6	3.64e-6	1.20e-8	1.96e-4	5.22e-7	7.23e-9	2.29e-8	1.26e-3	4.56e-7	2.38e-7	1.74e-10

level according to the Wilcoxon signed-rank test. This demonstrates the superior adaptability and generality of UADB to different models and tasks.

- There is no UAD model that consistently outperforms others on all tasks, i.e., the universal winner UAD solution does not exist. This is aligned with the findings in previous works [1].
- Using a different evaluation metric can lead to a different "best model". For example, GMM [59] obtains the best average performance in terms of AUCROC, but CBLOF [56] becomes the best performer if we change the evaluation metric to AP.
- UADB can make improvements even over the best performer UAD model (i.e., GMM [59] in terms of AUCROC and CBLOF [56] in terms of AP). Despite their relatively good performance, UADB still managed to achieve an average 1.83%/10.09% relative performance gain of AUCROC/AP over GMM, and average 3.02%/6.02% relative performance gain of AUCROC/AP over CBLOF.

As observed in Fig. 2, for a small portion of datasets, anomalies do not have higher average variance than normal samples. We further explore UADB's performance on these datasets, which are shown in Fig. 6. Though the empirical evidence does not hold, UADB could still achieve improvements over 12 out of 14 UAD models on more than half of these datasets.

Sensitivity Analysis. Fig. 7 demonstrated that for most UAD models, the performance of the UADB gradually improves with an increase in training iterations, and the UADB's performance reaches a stable level after 10 training iterations. Thus, it is reasonable to set the total training steps T to 10

TABLE V: UADB's performance on representative UAD models (teacher) in terms of AUCROC and AP (higher is better). Due to the space limitations, we select 4 widely used UAD techniques (i.e., IForest [24], HBOS [28], LOF [52], and KNN [53]) as representatives and show the performance of them and their UADB boosters on 5 example datasets. In each sub-table, we show the teacher model's performance as well as the booster's performance (during and after UADB training) on the 5 datasets. The "improvement" indicates the performance improvement that the UADB booster achieved over its source UAD model. In each sub-table row, we show the performance sub-table in terms of AUCROC and AP for a specific UAD model.

(a) Performance of (**IForest**) and its UADB booster in terms of **AUCROC**

Datasets	Teacher	iter 2	iter 4	iter 6	iter 8	iter 10	Improvement
speech	0.5057	0.579	0.6002	0.613	0.62	0.6233	0.1176
Wilt	0.4276	0.4407	0.4989	0.5305	0.5309	0.5364	0.1088
satellite	0.6668	0.6988	0.7117	0.725	0.7413	0.7625	0.0957
vowels	0.8118	0.8318	0.8488	0.8707	0.8918	0.9066	0.0949
abalone	0.4989	0.5435	0.5524	0.5589	0.5603	0.5663	0.0674

(b) Performance of (**IForest**) and its UADB booster in terms of **AP**

Datasets	Teacher	iter 2	iter 4	iter 6	iter 8	iter 10	Improvement
pendigits	0.3392	0.3995	0.4154	0.4505	0.4973	0.5524	0.2132
vowels	0.1825	0.1743	0.1835	0.2463	0.3143	0.3408	0.1582
satellite	0.6248	0.7074	0.7163	0.7252	0.7317	0.7399	0.1151
InternetAds	0.5078	0.5166	0.5221	0.5313	0.5468	0.5588	0.051
abalone	0.5111	0.531	0.5379	0.5443	0.5468	0.5529	0.0418

(c) Performance of (**HBOS**) and its UADB booster in terms of **AUCROC**

Datasets	Teacher	iter 2	iter 4	iter 6	iter 8	iter 10	Improvement
speech	0.4673	0.5931	0.612	0.626	0.6299	0.633	0.1657
Wilt	0.3849	0.5111	0.5311	0.5492	0.5464	0.5426	0.1577
vowels	0.6726	0.7341	0.7732	0.7878	0.7987	0.8051	0.1325
mnist	0.6141	0.6662	0.6623	0.6672	0.675	0.6779	0.0638
imgseg	0.656	0.6945	0.7012	0.7032	0.7058	0.7096	0.0536

(d) Performance of (**HBOS**) and its UADB booster in terms of **AP**

Datasets	Teacher	iter 2	iter 4	iter 6	iter 8	iter 10	Improvement
pendigits	0.2805	0.3397	0.359	0.3671	0.3779	0.389	0.1085
vowels	0.0883	0.0865	0.1052	0.1254	0.1524	0.182	0.0936
Ionosphere	0.4133	0.4516	0.5228	0.5313	0.5095	0.486	0.0727
imgseg	0.6105	0.644	0.6478	0.6476	0.6509	0.6538	0.0433
cardio	0.4679	0.5132	0.5182	0.5134	0.5099	0.508	0.0401

(e) Performance of (**LOF**) and its UADB booster in terms of **AUCROC**

Datasets	Teacher	iter 2	iter 4	iter 6	iter 8	iter 10	Improvement
http	0.3685	1	1	1	1	1	0.6315
shuttle	0.4886	0.9199	0.9244	0.9691	0.9537	0.9525	0.4638
satimage-2	0.4702	0.5677	0.6909	0.7618	0.8402	0.9146	0.4444
optdigits	0.5819	0.8488	0.8884	0.9128	0.9318	0.9438	0.3619
musk	0.4586	0.5008	0.5584	0.6137	0.6848	0.7528	0.2942

(f) Performance of (**LOF**) and its UADB booster in terms of **AP**

Datasets	Teacher	iter 2	iter 4	iter 6	iter 8	iter 10	Improvement
http	0.0603	1	1	1	1	1	0.9397
shuttle	0.0958	0.6814	0.7598	0.873	0.7902	0.7882	0.6924
optdigits	0.0732	0.1114	0.1825	0.2627	0.3593	0.4551	0.3819
satellite	0.3746	0.6268	0.675	0.6887	0.7047	0.7117	0.3371
WDBC	0.1026	0.1757	0.2627	0.3202	0.3604	0.3754	0.2727

(g) Performance of (**KNN**) and its UADB booster in terms of **AUCROC**

Datasets	Teacher	iter 2	iter 4	iter 6	iter 8	iter 10	Improvement
Stamps	0.6587	0.773	0.8458	0.8863	0.8876	0.8851	0.2264
SpamBase	0.5086	0.6249	0.6675	0.692	0.7087	0.7119	0.2033
pendigits	0.7133	0.8409	0.8484	0.8691	0.8861	0.9047	0.1914
musk	0.6845	0.775	0.7961	0.8153	0.8377	0.8551	0.1706
shuttle	0.6618	0.9908	0.9903	0.9365	0.8425	0.8134	0.1516

(h) Performance of (**KNN**) and its UADB booster in terms of **AP**

Datasets	Teacher	iter 2	iter 4	iter 6	iter 8	iter 10	Improvement
shuttle	0.1558	0.9055	0.9203	0.8229	0.7425	0.7379	0.5822
satimage-2	0.3294	0.4703	0.5803	0.6749	0.8082	0.8873	0.5579
satellite	0.5041	0.7098	0.7306	0.7448	0.7609	0.7683	0.2642
SpamBase	0.3967	0.4643	0.5138	0.5479	0.5746	0.5815	0.1848
WPBC	0.232	0.2343	0.2788	0.3481	0.3952	0.4123	0.1803

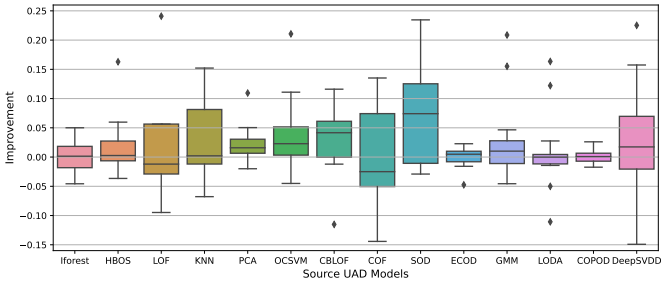


Fig. 6: UADB's performance on datasets that anomalies do not have higher average variance than normal samples. "Improvement" indicates the improvements achieved by the UADB booster over the UAD model on these datasets.

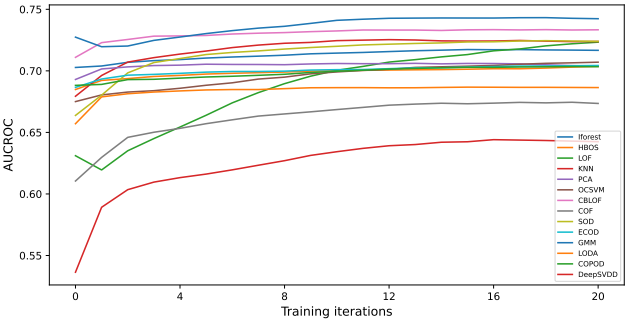


Fig. 7: UADB's performance (AUCROC) with different number of training iterations/steps. The results are averaged over 84 tabular datasets.

in Section IV-A. In addition, Fig. 8 shows the AUCROC performance of UADB w.r.t. different MLP layers. The results indicate that UADB performs very stably w.r.t. the number of MLP layers on all datasets.

Case Study on Real-world Datasets (RQ2). To further validate the role of iterative training, we show some real-world dataset cases for better understanding. In Fig. 9, we show the ranking development of 4 types of instances (i.e. True Positive (TP), True Negative (TN), False Positive (FP),

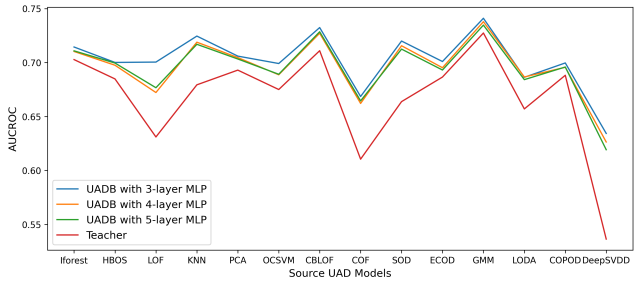
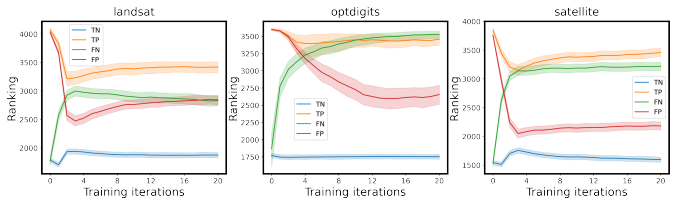


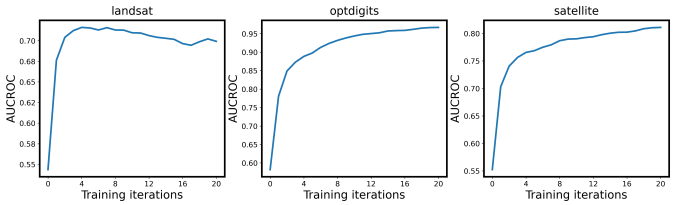
Fig. 8: UADB's performance (AUCROC) with different MLP layers. The results are averaged over 84 tabular datasets.

and False Negative (FN)) on several real-world datasets, and also show the respective performance of UAD booster during multiple iterations. Higher ranking represents higher anomaly score. Let's first consider TP and FP, the initial pseudo labels (teacher's prediction) are closer to 1 (i.e. the ranking is high), however, the variance of TP is higher than FP (anomalies has higher average variance compared to normal samples), thus after adding the variance to pseudo labels, the ranking of FP would decrease compared to TP. After multiple iterations, the difference in ranking between FP and TP will increase (TP maintains high ranking while the ranking of FP decreases). Likewise, the difference in ranking between FN and TN will also increase. Therefore, after multiple training iterations, UADB could maintain the right decisions of the source model, while correcting the wrong predictions by error correction mechanism.

Ablation Study (RQ3). To further validate the effectiveness of the error correction mechanism in UADB, we carry out ablation study on real-world datasets. Note that in our case, removing error correction in UADB results in no knowledge correction, which forces the booster models to have identical



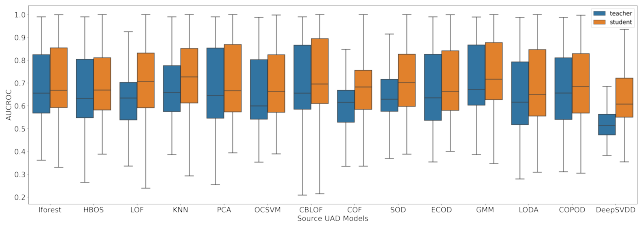
(a) The development of ranking.



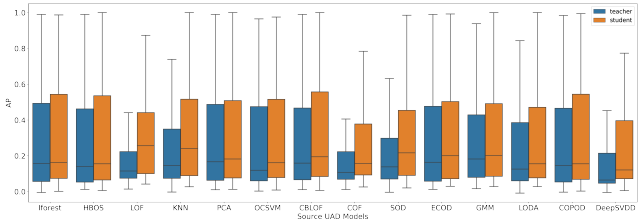
(b) The development of AUCROC performance.

Fig. 9: The development of instances' ranking and UADB's performance on *landsat, optdigits and satellite*. Here, we adopt LOF as UAD model and the number of training steps is set to 20. We reported the average ranking of 4 types of instances (TP, TN, FP, FN) respectively.

output with the teacher model. So the ablation study can be done by simply comparing the performance of teacher UAD models and their UADB boosters. Same as above, to obtain reliable conclusions, we conduct the ablation study on all 84 real-world tabular datasets. We summarize the ablation study results in boxplots of performance, as shown in Fig. 10.



(a) Evaluation metric: Area Under the Curve of Receiver Characteristic Operator (AUCROC)



(b) Evaluation metric: Average Precision (AP)

Fig. 10: Boxplots of performance of 14 teacher UAD models and their UADB boosters on 84 tabular datasets.

We can observe that:

- In terms of both evaluation metrics, removing the error correction mechanism results in obvious average performance degradation for all tested UAD techniques.
- The degradation is more significant in terms of AP, where not only the average, but also the best and worst performance among the 84 datasets get degraded.
- The performance gain from error correction is especially important for those UAD models that do not perform well by themselves, such as LOF [52], COF [57], KNN [53], SOD [58], and DeepSVDD [30] (in terms of AP).

Comparison with Other Intuitive Mechanisms (RQ4). Finally, to further validate the effectiveness of UADB's design, we consider several variants of UADB and compare their performance. Motivated by previous works that directly use the discrepancy between multiple models' output as the predicted anomaly score, we apply different training and inference schema to generate multiple variant booster frameworks. Specifically, we consider 4 different alternative booster frameworks:

1) Naïve Booster: it only uses the source model's output as static pseudo labels, then use it to train the booster model without any in-iteration adjustments on pseudo labels. The booster's output are used directly as the predicted anomaly score at inference.
2) Discrepancy Booster: similar to the Naïve Booster, it also adopt the source model's output as static labels for pseudo-supervised training of the booster model. However, when inference, it uses the discrepancy (Standard

TABLE VI: Ablation study results. The average AUCROC and AP over 84 datasets are reported. For each UAD model, we show the results of 6 variants (UAD itself and 5 different types of booster). The best results are highlighted in bold.

Comparison between different booster training strategies in terms of AUCROC

Source UAD Model	Iforest	HBOS	LOF	KNN	PCA	OCSVM	CBLOF	COF	SOD	ECOD	GMM	LODA	COPOD	DeepSVDD	Average
Origin	0.7028	0.6848	0.6311	0.6794	0.6930	0.6750	0.7110	0.6105	0.6638	0.6866	0.7274	0.6571	0.6882	0.5346	0.6777
Naïve Booster	0.6896	0.6728	0.6365	0.6778	0.6943	0.6733	0.7093	0.6108	0.6813	0.6863	0.7173	0.6751	0.6713	0.5641	0.6766
Discrepancy Booster	0.5589	0.5526	0.5807	0.5850	0.6065	0.6019	0.5356	0.5439	0.6025	0.5497	0.6362	0.5677	0.5607	0.5851	0.5755
Self Booster	0.6838	0.6688	0.6334	0.6686	0.6809	0.6618	0.7008	0.6073	0.6701	0.6722	0.7031	0.6736	0.6658	0.5667	0.6685
Discrepancy Booster*	0.6235	0.5904	0.5731	0.6144	0.6195	0.5970	0.5679	0.5670	0.6358	0.6122	0.6597	0.5761	0.6039	0.5580	0.6031
UADB	**0.7144**	**0.7001**	**0.7004**	**0.7245**	**0.7060**	**0.6991**	**0.7324**	**0.6686**	**0.7199**	**0.7010**	**0.7407**	**0.6864**	**0.6998**	**0.6343**	**0.7072**

Comparison between different booster training strategies in terms of AP

Source UAD Model	Iforest	HBOS	LOF	KNN	PCA	OCSVM	CBLOF	COF	SOD	ECOD	GMM	LODA	COPOD	DeepSVDD	Average
Origin	0.3012	0.2918	0.1903	0.2550	0.3051	0.2738	0.3057	0.1989	0.2322	0.2908	0.2805	0.2623	0.2832	0.1727	0.2670
Naïve Booster	0.2966	0.2926	0.2257	0.2572	0.2923	0.2735	0.3086	0.2154	0.2608	0.2838	0.2901	0.2865	0.2841	0.2187	0.2744
Discrepancy Booster	0.1588	0.1629	0.1739	0.1837	0.1946	0.1854	0.1580	0.1647	0.1881	0.1537	0.2080	0.1813	0.1627	0.1856	0.1751
Self Booster	0.2946	0.2911	0.2127	0.2594	0.2938	0.2760	0.3095	0.2132	0.2647	0.2837	0.3033	0.2887	0.2817	0.2132	0.2748
Discrepancy Booster*	0.2078	0.2057	0.1854	0.2076	0.2255	0.1999	0.1825	0.1802	0.2260	0.1940	0.2564	0.1897	0.2015	0.1988	0.2048
UADB	**0.3146**	**0.3055**	**0.3049**	**0.3178**	**0.3061**	**0.2967**	**0.3241**	**0.2659**	**0.3064**	**0.3009**	**0.3092**	**0.3013**	**0.2978**	**0.2468**	**0.3039**

deviation) between booster's output and source model's output as the predicted anomaly score.

3) Self Booster: it involves multiple booster training iterations like UADB, but in each iteration, it does not perform error correction on the pseudo labels. Instead, it uses booster's output after MinMax normalization as the pseudo label in the next iterations. At inference, booster's output is used as the predicted anomaly score.
4) Discrepancy Booster*: it also involves multiple booster training iterations like UADB. The booster training strategy is identical to that of Self Booster, but at inference, the discrepancy between booster's output and source model's output is used as the predicted anomaly score.

Following the previous setting, we apply all booster frameworks to 14 UAD models and 84 tabular datasets to get reliable results. The comparison results between UADB and these alternative booster frameworks are shown in Table VI. We can obeserve that:

- UADB is the best performer among all booster frameworks, it generally outperforms other counterparts by a large margin.
- Directly use the discrepancy between a teacher and a student model's output as the anomaly score cannot get good anomaly detection performance. This may due to the fact that only two models are included causing inaccurate discrepancy estimation.
- Self Booster obtains better results than other counter parts, indicating the importance of multi-stage training and pseudo labels adjustments.

To this point, we answered all the research questions that were proposed at the start of this section. Through comprehensive experiments and analysis on multiple real-world datasets and UAD models, we show that UADB can generally boost the performance of different UAD models on heterogeneous tabular datasets, and the error correction mechanism can effectively prevent the transfer of the wrong knowledge in the teacher model. UADB generally shows wide applicability and great performance in various real-world applications.

V. Conclusion

In this paper, we aim to design better unsupervised anomaly detection (UAD) techniques for tabular datasets. This task faces several fundamental challenges including (i) unsupervision: lack of prior knowledge about the anomalous pattern; (ii) assumption misalignment: the assumptions of UAD methods can be easily violated in real-world data and result in suboptimal performance; (iii) data heterogeneity: tabular data's properties vary greatly across different domains. For above reasons, a universal winner that consistently outperforms other solutions does not exist due to the multifaceted complexity of the task. We argue that the key to generally better UAD on diverse and heterogeneous tabular data is to go beyond the static assumptions and empower UAD models with adaptability to different data.

In light of this, we propose UADB (Unsupervised Anomaly Detection Booster), a versatile framework for improving any UAD model's general performance on tabular datasets. Specifically, UADB is designed to (i) keep the prior knowledge of the UAD model and its assumption by knowledge transfer, and (ii) perform adaptive error correction during transfer by exploiting the sample variance at the same time. The structure of UADB is conceptually simple, but has proven to be very effective in real-world UAD tasks. Extensive experiments show that UADB can generally achieve significant performance improvement over the 14 different source UAD models on 84 heterogeneous tabular datasets. To our best knowledge, UADB is the first of its kind as a general framework for enhancing UAD models. To sum up, we carry out a preliminary exploration on designing a model-agnostic booster framework to enhance UAD on tabular datasets. We hope our findings can shed some light on developing better versatile UAD solutions.

REFERENCES

[1] S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao, "Adbench: Anomaly detection benchmark," *arXiv preprint arXiv:2206.09426*, 2022.

[2] D. Zha, K.-H. Lai, M. Wan, and X. Hu, "Meta-aad: Active anomaly detection with deep reinforcement learning," in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 771–780.

[3] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[4] C. Spence, L. Parra, and P. Sajda, "Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model," in *Proceedings IEEE workshop on mathematical methods in biomedical image analysis (MMBIA 2001)*. IEEE, 2001, pp. 3–10.

[5] F. E. Grubbs, "Procedures for detecting outlying observations in samples," *Technometrics*, vol. 11, no. 1, pp. 1–21, 1969.

[6] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[7] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

[8] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller, "A unifying review of deep and shallow anomaly detection," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 756–795, 2021.

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

[10] H. Wang, G. Pang, C. Shen, and C. Ma, "Unsupervised representation learning by predicting random distances," *arXiv preprint arXiv:1912.12186*, 2019.

[11] M.-N. Nguyen and N. A. Vien, "Scalable and interpretable one-class svms with deep learning and random fourier features," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2018, pp. 157–172.

[12] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *International conference on learning representations*, 2018.

[13] T. Chen, L.-A. Tang, Y. Sun, Z. Chen, and K. Zhang, "Entity embedding-based anomaly detection for heterogeneous categorical events," *arXiv preprint arXiv:1608.07502*, 2016.

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

[15] F. Angiulli and C. Pizzuti, "Fast outlier detection in high dimensional spaces," in *European conference on principles of data mining and knowledge discovery*. Springer, 2002, pp. 15–27.

[16] J. Zhang and H. Wang, "Detecting outlying subspaces for high-dimensional data: the new task, algorithms, and performance," *Knowledge and information systems*, vol. 10, no. 3, pp. 333–355, 2006.

[17] D. Yu, G. Sheikholeslami, and A. Zhang, "Findout: Finding outliers in very large datasets," *Knowledge and information Systems*, vol. 4, no. 4, pp. 387–412, 2002.

[18] R. Smith, A. Bivens, M. Embrechts, C. Palagiri, and B. Szymanski, "Clustering approaches for anomaly based intrusion detection," *Proceedings of intelligent engineering systems through artificial neural networks*, vol. 9, 2002.

[19] A. Pires and C. Santos-Pereira, "Using clustering and robust estimators to detect outliers in multivariate data," 2005.

[20] N. Ye and Q. Chen, "An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems," *Quality and reliability engineering international*, vol. 17, no. 2, pp. 105–112, 2001.

[21] C. C. Aggarwal, "On abnormality detection in spuriously populated data streams," in *Proceedings of the 2005 siam international conference on data mining*. SIAM, 2005, pp. 80–91.

[22] D. Agarwal, "Detecting anomalies in cross-classified streams: a bayesian approach," *Knowledge and information systems*, vol. 11, no. 1, pp. 29–44, 2007.

[23] T. Shenkar and L. Wolf, "Anomaly detection for tabular data with internal contrastive learning," in *International Conference on Learning Representations*, 2021.

[24] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*. IEEE, 2008, pp. 413–422.

[25] D. Guthrie, L. Guthrie, B. Allison, and Y. Wilks, "Unsupervised anomaly detection." in *IJCAI*, 2007, pp. 1624–1628.

[26] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A geometric framework for unsupervised anomaly detection," in *Applications of data mining in computer security*. Springer, 2002, pp. 77–101.

[27] K. Leung and C. Leckie, "Unsupervised anomaly detection in network intrusion detection using clusters," in *Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38*, 2005, pp. 333–342.

[28] M. Goldstein and A. Dengel, "Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm," *KI-2012: poster and demo track*, vol. 9, 2012.

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

[30] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *International conference on machine learning*. PMLR, 2018, pp. 4393–4402.

[31] G. Hinton, O. Vinyals, J. Dean *et al.*, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

[32] M. Phuong and C. Lampert, "Towards understanding knowledge distillation," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5142–5151.

[33] J. H. Cho and B. Hariharan, "On the efficacy of knowledge distillation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 4794–4802.

[34] Y. Li, J. Yang, Y. Song, L. Cao, J. Luo, and L.-J. Li, "Learning from noisy labels with distillation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1910–1918.

[35] J. Wang, L. Gou, W. Zhang, H. Yang, and H.-W. Shen, "Deepvid: Deep visual interpretation and diagnosis for image classifiers via knowledge distillation," *IEEE transactions on visualization and computer graphics*, vol. 25, no. 6, pp. 2168–2180, 2019.

[36] M. Zhu, K. Han, C. Zhang, J. Lin, and Y. Wang, "Low-resolution visual recognition via deep feature distillation," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3762–3766.

[37] H. Kong, J. Zhao, X. Tu, J. Xing, S. Shen, and J. Feng, "Cross-resolution face recognition via prior-aided face hallucination and residual knowledge distillation," *arXiv preprint arXiv:1905.10777*, 2019.

[38] M. Yan, M. Zhao, Z. Xu, Q. Zhang, G. Wang, and Z. Su, "Vargfacenet: An efficient variable group convolutional neural network for lightweight face recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[39] S. Hahn and H. Choi, "Self-knowledge distillation in natural language processing," *arXiv preprint arXiv:1908.01851*, 2019.

[40] C. Zhou, G. Neubig, and J. Gu, "Understanding knowledge distillation in non-autoregressive machine translation," *arXiv preprint arXiv:1911.02727*, 2019.

[41] W. Fan, K. Liu, H. Liu, P. Wang, Y. Ge, and Y. Fu, "Autofs: Automated feature selection via diversity-aware interactive reinforcement learning," in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 1008–1013.

[42] W. Fan, K. Liu, H. Liu, Y. Ge, H. Xiong, and Y. Fu, "Interactive reinforcement learning for feature selection with decision tree in the loop," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[43] Y.-C. Chen, Z. Gan, Y. Cheng, J. Liu, and J. Liu, "Distilling knowledge learned in bert for text generation," *arXiv preprint arXiv:1911.03829*, 2019.

[44] A. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. Driessche, E. Lockhart, L. Cobo, F. Stimberg *et al.*, "Parallel wavenet: Fast high-fidelity speech synthesis," in *International conference on machine learning*. PMLR, 2018, pp. 3918–3926.

[45] K. Kwon, H. Na, H. Lee, and N. S. Kim, "Adaptive knowledge distillation based on entropy," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7409–7413.

[46] P. Shen, X. Lu, S. Li, and H. Kawai, "Knowledge distillation-based representation learning for short-utterance spoken language identification,"

IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 28, pp. 2674–2683, 2020.

[47] M. Salehi, N. Sadjadi, S. Baselizadeh, M. H. Rohban, and H. R. Rabiee, "Multiresolution knowledge distillation for anomaly detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 14 902–14 912.

[48] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4183–4192.

[49] G. Wang, S. Han, E. Ding, and D. Huang, "Student-teacher feature pyramid matching for unsupervised anomaly detection," *arXiv preprint arXiv:2103.04257*, 2021.

[50] Y. Zhao, X. Hu, C. Cheng, C. Wang, C. Wan, W. Wang, J. Yang, H. Bai, Z. Li, C. Xiao *et al.*, "Suod: Accelerating large-scale unsupervised heterogeneous outlier detection," *Proceedings of Machine Learning and Systems*, vol. 3, pp. 463–478, 2021.

[51] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021.

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

[53] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 427–438.

[54] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, "A novel anomaly detection scheme based on principal component classifier," Miami Univ Coral Gables Fl Dept of Electrical and Computer Engineering, Tech. Rep., 2003.

[55] B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, "Support vector method for novelty detection," *Advances in neural information processing systems*, vol. 12, 1999.

[56] Z. He, X. Xu, and S. Deng, "Discovering cluster-based local outliers," *Pattern recognition letters*, vol. 24, no. 9-10, pp. 1641–1650, 2003.

[57] J. Tang, Z. Chen, A. W.-C. Fu, and D. W. Cheung, "Enhancing effectiveness of outlier detections for low density patterns," in *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 2002, pp. 535–548.

[58] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "Outlier detection in axis-parallel subspaces of high dimensional data," in *Pacific-asia conference on knowledge discovery and data mining*. Springer, 2009, pp. 831–838.

[59] D. A. Reynolds, "Gaussian mixture models." *Encyclopedia of biometrics*, vol. 741, no. 659-663, 2009.

[60] T. Pevnỳ, "Loda: Lightweight on-line detector of anomalies," *Machine Learning*, vol. 102, no. 2, pp. 275–304, 2016.

[61] Z. Li, Y. Zhao, N. Botta, C. Ionescu, and X. Hu, "Copod: copula-based outlier detection," in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 1118–1123.

[62] Y. Zhao, Z. Nasrullah, and Z. Li, "Pyod: A python toolbox for scalable outlier detection," *arXiv preprint arXiv:1901.01588*, 2019.

[63] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

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