Fascinating Supervisory Signals and Where to Find Them: Deep Anomaly Detection with Scale Learning

Hongzuo Xu [1 2] **Yijie Wang** [1 2] **Juhui Wei** [3] **Songlei Jian** [2] **Yizhou Li** [1 2] **Ning Liu** [1 2]

Abstract

Due to the unsupervised nature of anomaly detection, the key to fueling deep models is finding supervisory signals. Different from current reconstruction-guided generative models and transformation-based contrastive models, we devise novel data-driven supervision for tabular data by introducing a characteristic – *scale* – as data labels. By representing varied sub-vectors of data instances, we define *scale* as the relationship between the dimensionality of original sub-vectors and that of representations. Scales serve as labels attached to transformed representations, thus offering ample labeled data for neural network training. This paper further proposes a scale learning-based anomaly detection method. Supervised by the learning objective of scale distribution alignment, our approach learns the ranking of representations converted from varied subspaces of each data instance. Through this proxy task, our approach models inherent regularities and patterns within data, which well describes data "normality". Abnormal degrees of testing instances are obtained by measuring whether they fit these learned patterns. Extensive experiments show that our approach leads to significant improvement over state-of-the-art generative/contrastive anomaly detection methods.

1. Introduction

Anomaly detection, the task of discovering exceptional data that deviate significantly from the majority (Aggarwal, 2017), has been successfully applied in many real-world domains when there is a need to identify both negative and positive rare events (e.g., diseases, cyberspace intrusions, financial frauds, industrial faults, and marketing opportunities). Deep learning has shown strong modeling capability, which enables deep anomaly detection methods to yield drastic performance improvement over traditional methods (Pang et al., 2021). Due to the unsupervised nature of anomaly detection, *designing deep anomaly detection models is a journey of finding reasonable supervisory signals* .

Many deep anomaly detection methods (Xia et al., 2015; Chen et al., 2017; Zhou & Paffenroth, 2017; Liu et al., 2019; 2021) construct various kinds of autoencoders, generative adversarial networks, or prediction models. Their learning objectives are adapted to anomaly detection by treating reconstruction errors of incoming data as abnormal degrees. These generative methods are intuitive and show favorable performance on several popular benchmarks. However, as has been discussed in (Larsen et al., 2016; Ruff et al., 2018; Wang et al., 2022), one imperfection is that their learning target is primarily designed to reconstruct/generate the whole data. That is, they are forced to focus on reducing errors in each fine-grained point, but overemphasizing low-level details may make the model hard to converge when the normal class is complicated. Besides, some underlying hard anomalies can be only identified by investigating high-level pattern information in inlier data.

To address this limitation, recent efforts (Golan & El-Yaniv, 2018; Li et al., 2021; Wang et al., 2022; Ristea et al., 2022) have been made to liberate deep anomaly detection from the above reconstruction-based pipeline. They define various transformations and design pretext tasks, learning to classify, compare, or map these transformations. They either use the learned representations for independent abnormality estimators or utilize loss values as anomaly scores. These models can embed high-level semantic information into the learned representations, thus leading to stronger anomaly detectors with promising detection accuracy. However, most popular transformation operations (e.g., rotations, cropping, and flip) can be only applied to image data. As for non-perceptual tabular data, it is still a non-trivial task to define suitable supervisory signals to actuate deep learning models.

To this end, we introduce a new data characteristic – *scale* – to devise a novel kind of data-driven supervision. Generally,
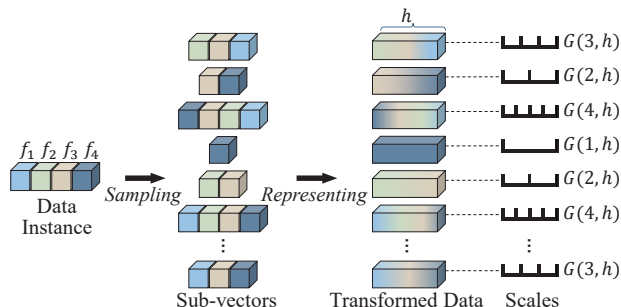
---

[1]National Key Laboratory of Parallel and Distributed Computing [2]College of Computer, National University of Defense Technology [3]College of Science, National University of Defense Technology. Correspondence to: Yijie Wang <wangyijie@nudt.edu.cn>.

Figure 1. A toy example of scales in tabular data. For a tabular data instance described by four features, sub-vectors with varied feature subspaces are randomly sampled from the original space and then transformed into h-dimensional representations. Scale is computed as the mathematical relationship G between sub-vector length and the representation dimensionality h.

scale indicates the ratio between the real size of something and its size on a map, model, or diagram. This naturally inspires us to define the scale concept in tabular data as:

Definition 1.1 (*Scale in Tabular Data*). Sub-vectors of tabular data instances are transformed to representations. Scale is defined as the mathematical relationship between the dimensionality of sub-vectors and that of the representations, which indicates the level of detail in representations.

Figure 1 delineates a toy example. Scales serve as labels attached to these transformed data, thereby supplying labeled data for driving neural networks on tabular data.

However, it is still challenging to harness these labeled data. Due to the feature diversity, some sub-vectors with lower dimensionality are with more details than higher-dimensional sub-vectors. Also, some randomly sampled subspaces might be irrelevant or even noisy. These notorious problems suggest that it is unfeasible to define canonical proxy tasks like classification or simple prediction. Instead, we define scale learning as follows.

Definition 1.2 (*Scale Learning*). Each individual data sample in scale learning is defined as a group of representations transformed from varied sub-vectors of a data instance. The predictions and corresponding labels are converted to two distributions, and scale learning is defined as a distribution alignment task.

Through optimizing distributions, the learning model is forced to focus on the relative ranking of scales rather than raw absolute values, and the increased sampling times also dilute ineffective irrelevant/noisy subspaces. Our model essentially learns the listwise ranking of representations transformed from varied subspaces of each original data instance, during which our model can capture inherent regularities and patterns related to the data structure. This is a novel kind of data-driven supervision that is different from

current point-wise generative methods and discriminative models based on classification, comparison, or mapping.

Based on this supervision, this paper introduces a Scale Learning-based deep Anomaly Detection method (termed SLAD). Concretely, SLAD first specifies a transformation function to represent sub-vectors and a labeling function to calculate data scales. After creating scale-based labeled data, SLAD performs scale learning, optimized by a distributional divergence-based loss function. Through this proxy task, SLAD embeds high-level information (i.e., underlying regularities and patterns related to the data structure) into the learned scale-based ranking mechanism. It is similar to self-supervised learning models in vision and natural language domains that embed data semantics into representations. Such high-level information helps our model to tame data complexity and reveal hard anomalies. We extend the inlier-priority property proposed in (Wang et al., 2022) to our model. That is, due to the imbalanced nature of anomaly detection, the learning process can prioritize inliers, and the learned regularities reflect "normality" shared in inliers. Anomalies, by definition, show deviated behaviors, and they cannot comply with these learned models. Hence, for identifying anomalies, errors computed by the loss function are directly exploited to indicate abnormal degrees.

Our contributions are summarized as follows:

- Conceptually, we introduce the scale concept in tabular data. By defining scale learning as a distribution alignment task, we appropriately harness scale-based labeled data to actuate neural networks for tabular data. Essentially, we devise a novel kind of data-driven supervision, and neural networks can model intrinsic regularities pertaining to the data structure.

- Methodologically, we propose SLAD, a scale learning-based deep anomaly detection method. The loss values are directly exploited to indicate abnormal degrees, thus allowing SLAD to produce anomaly scores in an end-to-end fashion. Our method also contributes a novel self-supervised strategy to other tasks like representation learning in tabular data.

- Theoretically, we analyze the shape of the created data sample in scale learning to ensure its effectiveness in revealing anomalies. To back up the application of scale learning in anomaly detection, we examine gradient magnitude to illustrate the inlier-priority property.

- Empirically, extensive experiments validate both the contributions in detection accuracy and the superiority in handling complicated training data. For example, SLAD raised the state-of-the-art AUC-PR from 0.82 to 0.92 (+10 points) on a popular *Thyroid* benchmark.

2. Related work

Deep learning-empowered anomaly detection has garnered much interest recently (Pang et al., 2021; Ruff et al., 2021). Due to the unsupervised nature of anomaly detection, without readily accessible labeled training data, finding supervisory signals becomes one crucial step to fuel deep learning models for anomaly detection. This section reviews how existing studies define their learning tasks.

One typical pipeline is based on generative methods. They take reconstruction as the learning objective to construct various kinds of autoencoders, generative adversarial networks, or prediction models and treat reconstruction errors as anomaly scores (Chen et al., 2017; Zhou & Paffenroth, 2017; Liu et al., 2019; 2021; Wang et al., 2021). Albeit intuitive, these methods overemphasize fine-grained reconstruction errors at the point-wise level, and they may fail to access high-level semantic information.

An alternative manner is to use one-class classification to obtain a model (e.g., hypersphere or hyperplane) that can accurately describe the "normality". Many anomaly detectors (Ruff et al., 2018; Goyal et al., 2020; Zhang & Deng, 2021; Liznerski et al., 2021) train neural networks to learn a new representation space by posing one-class constraints. However, the underlying one-class assumption might be vulnerable since there is often more than one prototype in inliers. Besides, some methods resort to additional label information. Self-training models exploit iteratively predicted results of training data as supervisory signals while updating the model parameters (Pang et al., 2020; Qiu et al., 2022), whereas this process might be disturbed by mislabeled data. It is also noteworthy that a recent method named outlier exposure introduces labeled data from other datasets, thus forming synthetically labeled data (Hendrycks et al., 2019).

The success of contrastive self-supervised learning in vision and natural language domains sheds light on the potential of discriminative models for embedding rich semantic information into representations. By using various transformation operations in image data (e.g., rotations, cropping, flip, cutout, and interpolation), many insightful approaches (Golan & El-Yaniv, 2018; Tack et al., 2020; Sehwag et al., 2021; Li et al., 2021; Wang et al., 2022; Ristea et al., 2022) create different views of initial data and employ classification, comparison, or mapping as pretext tasks. To identify anomalies, these methods perform independent abnormality measurements upon the learned representations or directly leverage the loss function. However, it is still non-trivial to define transformation operations for non-image data.

There are very limited attempts that generalize the above contrastive strategy to tabular data. GOAD (Bergman & Hoshen, 2020) is the pioneer transformation-based method that can handle non-image data, which generalizes the spatial transformation to random affine transformation. NeuTraL (Qiu et al., 2021) employs learnable neural transformations and proposes a noise-free, deterministic contrastive loss. The literature (Shenkar & Wolf, 2022) learns mappings that maximize the mutual information between each sampled sub-vector and the part that is masked out.

We finally review a related field, i.e., self-supervised pre-training for tabular data. These models also perform contrastive learning upon different views created by corrupting random feature sub-spaces based on respective empirical marginal distribution (Bahri et al., 2022) or feature correlations (Yao et al., 2021). In addition to the corruption, the study (Yoon et al., 2020) proposes "mask estimator" and "feature estimator" heads on top of the encoder state.

3. Scale Learning for Anomaly Detection

Problem Formulation. Let $\mathcal{X}$ be the input tabular data described by a D-dimensional feature space. Each data instance $\mathbf{x} \in \mathcal{X}$ is a vector $\mathbf{x} \in \mathbb{R}^D$. By training on $\mathcal{X}$, a deep anomaly detection model builds a scoring function $\tau : \mathbb{R}^D \mapsto \mathbb{R}$ to quantitatively measure abnormal degrees of incoming data instances.

Overview. Figure 2 depicts the overall framework of SLAD. We take one data instance $\mathbf{x}$ as an example to illustrate the procedure of SLAD. There are two main components in SLAD. *The creation of scale-based supervisory signals* consists of a transformation function T and a labeling function G, which respectively define how to transform sub-vectors of an original data instance $\mathbf{x}$ to representations $\mathbf{U}$ and how to compute scales as labels $\mathbf{y}$. SLAD treats each $\mathbf{U}$ matrix that contains c transformed vectors as an individual training sample, and labeled data $\mathcal{O} \times \mathcal{Y} = \{(\mathbf{U}_j, \mathbf{y}_j)\}_{j=1}^r$ offer supervisory signals for neural network training. In terms of *scale learning*, we construct a neural network Φ, and network parameters are optimized via a distribution alignment loss function L.

3.1. The Creation of Scale-based Supervisory Signals

Transformation Function T. T yields representations of sub-vectors. A unified h-dimensional representation frame is set since these transformed data serve as training samples for downstream scale learning. T can be also understood as a data preprocessing step. Some popular padding methods or dimensionality reduction techniques may change information contained in original sub-vectors. Instead, SLAD employs neural transformation to define T. Complicated neural transformations with non-linear activation may also modify the intrinsic data structure of the input. Random linear projection is a simple yet effective feature mapping technique, which can achieve dimensionality modification. Thus, T is defined as simple feed-forward layers that are
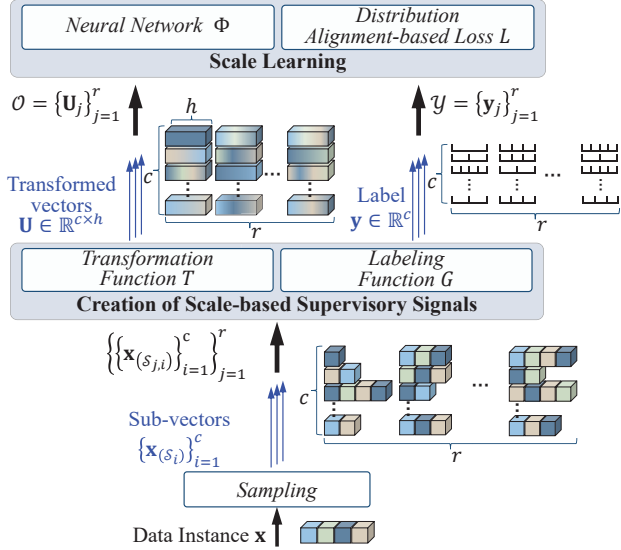
Figure 2. Overall framework of SLAD. For an original data instance $\mathbf{x}$, SLAD first generates a group of c sub-vectors $\{\mathbf{x}_{(\mathcal{S}_i)}\}_{i=1}^c$ via random sampling, where $\mathbf{x}_{(\mathcal{S}_i)}$ is the sub-vector of $\mathbf{x}$ on the subspace $\mathcal{S}_i \subseteq \{1, \cdots, D\}$. These sub-vectors are then transformed to a unified h-dimensional representation frame by a *Transformation function T*, yielding a matrix $\mathbf{U} \in \mathbb{R}^{c \times h}$. *Labeling function G* measures scales as data labels $\mathbf{y} \in \mathbb{R}^c$ of transformed data in $\mathbf{U}$. Each $\mathbf{U}$ and corresponding $\mathbf{y}$ are treated as one training sample, and the above process is repeated r times, which produces $\mathcal{O} \in \mathbb{R}^{r \times c \times h}$ attached with labels $\mathcal{Y} \in \mathbb{R}^{r \times c}$. A neural network $\Phi : \mathbb{R}^{r \times c \times h} \mapsto \mathbb{R}^{r \times c}$ is trained via the loss function L to predict scale-based distributions of transformed data.

randomly initialized, and each feature subspace corresponds to a transformation layer. For a ν-dimensional sub-vector $\mathbf{x}_{(\mathcal{S}_i)} \in \mathbb{R}^\nu$, its representation is obtained via a weight matrix $\mathbf{W}_\nu \in \mathbb{R}^{h \times \nu}$ and bias $b \in \mathbb{R}^h$, i.e.,

$$T(\mathbf{x}_{(\mathcal{S}_i)}) = \mathbf{W}_\nu \mathbf{x}_{(\mathcal{S}_i)} + b. \tag{1}$$

For c randomly sampled sub-vectors of a data instance $\mathbf{x}$, their transformations are denoted in a matrix $\mathbf{U} \in \mathbb{R}^{c \times h}$. $\mathbf{U}$ is treated as an individual data sample for scale learning.

Labeling Function G. SLAD further computes scales. Each dimension derived via neural transformation T is with equal status, so the representation dimensionality can be directly exploited. However, as original tabular features are varied, we intend to weigh each feature to capture this kind of difference. For ease of learning, we also increase the spacing of each scale value via a magnification factor. Therefore, given the representation dimension h and the feature subspace $\mathcal{S}_i$ of a sub-vector, G is defined as

$$G(\mathcal{S}_i, h) = \gamma \frac{\sum_{k \in \mathcal{S}_i} \omega_k}{h}, \tag{2}$$

where ω_k is the weight of the kth feature and γ is a magnification factor. The γ factor is a hyper-parameter. In terms of the feature weight, a feature is more informative if it has strong interactions with other features. Thus, Pearson product-moment correlation coefficient is employed. Let $\mathbf{u}_k = \{x_{(i,k)}\}_{i=1}^{|\mathcal{X}|}$ denote the values of the kth feature. The weight of kth feature is computed as $\omega_k = \frac{1}{|\mathcal{F}|} \sum_{k'=1}^{|\mathcal{F}|} \left| \frac{\text{cov}(\mathbf{u}_k, \mathbf{u}_{k'})}{\text{dev}(\mathbf{u}_k)\text{dev}(\mathbf{u}_{k'})} \right|$, where $\text{cov}(\cdot, \cdot)$ and $\text{dev}(\cdot)$ denote the covariance and the standard deviation. ω ranges from 0 to 1. High-dimensional feature space is often contaminated by noisy/irrelevant features, and this calculation function might be biased. This function also induces considerably heavy computational overhead when handling high-dimensional data. Therefore, SLAD omits this step and sets $\omega = 1$ for all features when the dimensionality of the original feature space is high.

For the $\mathbf{U}$ matrix deriving from a group of sub-vectors with feature subspaces $\{\mathcal{S}_1, \cdots, \mathcal{S}_c\}$, its label is defined as a list of scales, i.e., $\mathbf{y} = \{G(\mathcal{S}_1, h), \cdots, G(\mathcal{S}_c, h)\}$.

3.2. Scale Learning

The above process is repeated r times for an original data instance, creating labeled data $\mathcal{O} \times \mathcal{Y} = \{(\mathbf{U}_j, \mathbf{y}_j)\}_{j=1}^r$. SLAD constructs a neural network $\Phi : \mathbb{R}^{c \times h} \mapsto \mathbb{R}^c$ that maps each newly created data sample to a scale list, i.e., $\mathbf{p} = \Phi(\mathbf{U})$. Scale learning is defined as a distribution alignment task to handle the feature diversity and irrelevant/noisy sampled subspaces. Specifically, the predictions are processed by a softmax layer σ, i.e., $\sigma(\mathbf{p}) = \{ \frac{\exp(p_i)}{\sum_j \exp(p_j)} \}_{i=1}^c$, which generates a probability distribution. $\mathbf{y}$ is also processed by a softmax function σ to produce target distribution. Listwise prediction of c transformed vectors in $\mathbf{U}$ can be optimized uniformly. This way allows the optimization to be supervised by the relative values. The distribution alignment task substantially teaches the network to rank representations transformed from different feature subspaces of the original data instance via the predicted scale values.

After obtaining the prediction $\tilde{\mathbf{p}} = \sigma(\Phi(\mathbf{U}))$ and the target $\tilde{\mathbf{y}} = \sigma(\mathbf{y})$, loss value ℓ is defined by a distributional divergence measure. We employ Jensen–Shannon divergence in our implementation, i.e.,

$$\ell(\tilde{\mathbf{p}}\|\tilde{\mathbf{y}}) = \frac{1}{2} \sum_{i=1}^c \tilde{p}_i \log(\frac{\tilde{p}_i}{\frac{1}{2}(\tilde{p}_i + \tilde{y}_i)}) + \frac{1}{2} \sum_{i=1}^c \tilde{y}_i \log(\frac{\tilde{y}_i}{\frac{1}{2}(\tilde{p}_i + \tilde{y}_i)}). \tag{3}$$

The overall loss function of SLAD can be further defined as

$$L = \mathbb{E}_{\mathbf{x} \sim \mathcal{X}} \mathbb{E}_{(\mathbf{U}, \mathbf{y}) \sim \mathcal{O}_\mathbf{x} \times \mathcal{Y}_\mathbf{x}} \left[\ell\left(\sigma(\Phi(\mathbf{U})) \| \sigma(\mathbf{y}) \right) \right], \tag{4}$$

where $\mathcal{O}_\mathbf{x}$ and $\mathcal{Y}_\mathbf{x}$ denote the supervisory signals created by an original data instance $\mathbf{x}$.

3.3. Anomaly Detection

Scale learning is not directly related to anomaly detection, and this is essentially a surrogate learning task to drive neural network training. SLAD embeds feature interactions, patterns, and inherent regularities related to the data structure into the learned scale-based ranking mechanism. What SLAD leverages for anomaly detection are these high-level data abstractions.

We further present an *inlier-priority property*. It suggests that the update of the neural network is inclined to prioritize inliers due to the imbalanced nature of anomaly detection, i.e., what SLAD derives are normal, common regularities in inliers. Anomalies, by definition, are rare events and behave differently, thereby showing deviation from these learned regularities. This property is first proposed by (Wang et al., 2022) in which a classification-based pretext task is defined. We extend this property from the cross-entropy loss to our distributional divergence loss (theoretical analysis and empirical study in Section 3.4 and 4.2), which further supports the application of scale learning in anomaly detection.

Therefore, errors computed through the loss function can indicate abnormal degrees of incoming data. For a testing data instance $\mathbf{x}$, SLAD also creates transformed data samples $\mathcal{O}$ and corresponding supervisory labels $\mathcal{Y}$, and its anomaly score is obtained via

$$\tau(\mathbf{x}) = \sum_{(\mathbf{U},\mathbf{y}) \in \mathcal{O} \times \mathcal{Y}} \ell\big(\sigma(\Phi(\mathbf{U})) \big\| \sigma(\mathbf{y})\big). \quad (5)$$

3.4. Theoretical Analysis

We explore two questions: (**Q1**) How to ensure the effectiveness of the created data sample of scale learning in revealing anomalies? (**Q2**) Does scale learning model normal regularities of inliers, thus exposing anomalies? The training sample in scale learning (the $\mathbf{U}$ matrix) is generated from randomly sampled feature subspaces. Thus, to solve Q1, we consider how to determine *the shape of each transformed data sample* (the sampling times c) such that real anomalies can still stand out in the transformed form. As for Q2, we examine the inlier-priority property by analyzing the gradients that determine the neural network optimization.

The Shape of the Data Sample of Scale Learning and its Effectiveness in Revealing Anomalies. Let $\mathbf{x}_a$ be an anomaly, and $\mathbf{U}$ indicates its transformed matrix. We below derive the relationship between the size of $\mathbf{U}$ and the probability that $\mathbf{U}$ is useful to reveal $\mathbf{x}_a$ as an anomaly. We assume the abnormality of $\mathbf{x}_a$ is reflected in a subspace $\mathcal{G}$ of the whole feature space $\mathcal{F}$, i.e., $\mathcal{G} \subseteq \mathcal{F}$, and $|\mathcal{G}| = \beta|\mathcal{F}|$, $\beta \in (0,1]$. The elements in $\mathcal{G}$ are effective features. Generally, a subset of $\mathcal{G}$ is sufficient to discover the anomaly, and we denote this minimum size as $\alpha|\mathcal{G}|$, $\alpha \in (0,1]$. Let

$\mathcal{S}$ be one of the sampled subspaces when creating $\mathbf{U}$. The dimensionality of $\mathcal{S}$ is uniformly sampled from 1 to $|\mathcal{F}|$.

We first give the following Lemma (proof in Appendix A) to show the probability of $\mathcal{S}$ being effective.

Lemma 3.1. *The probability of $\mathcal{S}$ containing at least $\alpha|\mathcal{G}|$ effective features of $\mathcal{G}$ (i.e., $\mathcal{S}$ is effective) is:*

$$Pr(\mathcal{S} \text{ is useful}) = \frac{1}{|\mathcal{F}|} \sum_{j=\alpha|\mathcal{G}|}^{|\mathcal{F}|} \sum_{k=\alpha|\mathcal{G}|}^{j} \binom{j}{k} (\frac{|\mathcal{G}|}{|\mathcal{F}|})^k (1 - \frac{|\mathcal{G}|}{|\mathcal{F}|})^{j-k}. \tag{6}$$

Based on the above Lemma, we further present an intriguing fact in the following Theorem (proof in Appendix B), which bounds the above probability.

Theorem 3.2. *Given the effective feature space $\mathcal{G}$ and the minimum size $\alpha|\mathcal{G}|$ to reveal the anomaly, the lower bound of the probability of the randomly sampled subspace $\mathcal{S}$ being useful is* $\inf\big(Pr(\mathcal{S} \text{ is useful})\big) = 1 - \alpha$.

Let the success probability of an individual sampling be the lower bound, i.e., $1 - \alpha$. We assume $\mathbf{U}$ is useful to disclose the anomaly if it has at least one element that is transformed from the effective sub-vectors. Consequently, similar to Lemma 3.1, the probability of $\mathbf{U}$ being useful is:

$$Pr(\mathbf{U} \text{ is useful}) = \sum_{k=1}^{c} \binom{c}{k} (1 - \alpha)^k (\alpha)^{c-k}. \quad (7)$$

$Pr(\mathbf{U} \text{ is useful})$ and c are positively related, whereas a large number of useless elements in $\mathbf{U}$ may also disrupt the identification of $\mathbf{x}_a$. Thus, we use a size that is as small as possible while ensuring $Pr(\mathbf{U} \text{ is useful})$ is large enough. In our default setting, we use $c = 10$, which makes the probability exceed 0.999 when $\alpha = 0.5$ and 0.99 when $\alpha = 0.6$.

Inlier-priority Property in Scale Learning. The inlier-priority property indicates that the network optimization is inclined to prioritize inliers. Since the theoretical analysis of DNNs is still intractable, we consider the same analyzable case that has been used in (Wang et al., 2022), i.e., a feedforward structure with sigmoid activation. The penultimate layer outputs u units, and the final softmax layer contains c nodes. Network weights are initialized by a uniform distribution $[-1,1]$. Considering the kth element ($1 \le k \le c$) of the prediction, the gradients w.r.t. the weights (denoted as $\mathbf{w}_k = \{w_{(s,k)}\}_{s=1}^{u}$) are directly responsible for this output. Let L_k be the kth position of the loss function of N training data objects, and we can derive the expectation of gradient magnitude of updating $\mathbf{w}_k$ as follows:

$$\begin{aligned}
\mathbb{E}\Big[\big\|\nabla_{\mathbf{w}_k} L_k\big\|_2^2\Big] &= \sum_{s=1}^{u} \mathbb{E}\Big[\big(\sum_{i=1}^{N} \nabla_{w_{(s,k)}} \ell_k^{(i)}\big)^2\Big] \\
&= \sum_{s=1}^{u} \sum_{i=1}^{N} \sum_{j=1}^{N} \mathbb{E}\Big[\nabla_{w_{(s,k)}} \ell_k^{(i)} \nabla_{w_{(s,k)}} \ell_k^{(j)}\Big].
\end{aligned} \tag{8}$$

Table 1. Detection accuracy (AUC-ROC/AUC-PR $\pm$ standard deviation) of SLAD and its competing methods. The best detector per dataset is boldfaced. ICL and GAAL run out of memory (OOM) on the ultra-high-dimensional dataset *Thrombin*.

	DATA	SLAD	ICL	NeuTraL	GOAD	RCA	GAAL	DSVDD	iForest
AUC-ROC	Thyroid	**0.995 ± 0.001**	0.974 ± 0.015	0.985 ± 0.002	0.952 ± 0.005	0.934 ± 0.005	0.768 ± 0.096	0.930 ± 0.032	0.988 ± 0.002
	Arrthymia	**0.825 ± 0.007**	0.784 ± 0.048	0.805 ± 0.025	0.806 ± 0.008	0.767 ± 0.009	0.704 ± 0.082	0.807 ± 0.008	0.814 ± 0.007
	Waveform	**0.812 ± 0.047**	0.649 ± 0.048	0.621 ± 0.023	0.604 ± 0.022	0.626 ± 0.019	0.732 ± 0.074	0.516 ± 0.012	0.718 ± 0.019
	UNSW-NB15	**0.941 ± 0.004**	0.918 ± 0.010	0.916 ± 0.017	0.903 ± 0.003	0.935 ± 0.001	0.796 ± 0.060	0.902 ± 0.028	0.758 ± 0.016
	Bank	**0.730 ± 0.004**	0.724 ± 0.014	0.720 ± 0.018	0.587 ± 0.006	0.699 ± 0.003	0.655 ± 0.032	0.608 ± 0.057	0.723 ± 0.008
	Thrombin	**0.939 ± 0.007**	OOM	0.460 ± 0.033	0.839 ± 0.011	0.916 ± 0.000	OOM	0.520 ± 0.046	0.898 ± 0.008
	PageBlocks	**0.972 ± 0.004**	0.909 ± 0.025	0.961 ± 0.002	0.670 ± 0.006	0.864 ± 0.002	0.765 ± 0.032	0.904 ± 0.009	0.927 ± 0.005
	Amazon (tab)	**0.605 ± 0.007**	0.592 ± 0.005	0.570 ± 0.036	0.500 ± 0.000	0.538 ± 0.008	0.495 ± 0.032	0.539 ± 0.013	0.565 ± 0.008
	Yelp (tab)	0.658 ± 0.014	**0.664 ± 0.009**	0.627 ± 0.027	0.501 ± 0.000	0.585 ± 0.008	0.584 ± 0.039	0.593 ± 0.032	0.609 ± 0.007
	MVTec (tab)	**0.812 ± 0.009**	0.778 ± 0.010	0.788 ± 0.009	0.666 ± 0.030	0.663 ± 0.022	0.675 ± 0.026	0.806 ± 0.014	0.757 ± 0.011
AUC-PR	Thyroid	**0.921 ± 0.012**	0.726 ± 0.070	0.824 ± 0.018	0.778 ± 0.008	0.654 ± 0.012	0.429 ± 0.133	0.470 ± 0.030	0.783 ± 0.037
	Arrthymia	0.604 ± 0.006	0.572 ± 0.038	0.589 ± 0.022	0.631 ± 0.005	0.562 ± 0.009	0.505 ± 0.071	0.646 ± 0.008	**0.633 ± 0.021**
	Waveform	**0.432 ± 0.132**	0.123 ± 0.040	0.095 ± 0.014	0.079 ± 0.004	0.088 ± 0.008	0.148 ± 0.060	0.059 ± 0.002	0.111 ± 0.005
	UNSW-NB15	0.858 ± 0.003	**0.859 ± 0.005**	0.811 ± 0.014	0.813 ± 0.005	0.542 ± 0.009	0.470 ± 0.230	0.794 ± 0.028	0.111 ± 0.006
	Bank	**0.470 ± 0.003**	0.468 ± 0.015	0.445 ± 0.018	0.300 ± 0.006	0.423 ± 0.002	0.370 ± 0.050	0.315 ± 0.059	0.449 ± 0.013
	Thrombin	**0.625 ± 0.014**	OOM	0.038 ± 0.002	0.648 ± 0.013	0.587 ± 0.003	OOM	0.074 ± 0.023	0.421 ± 0.017
	PageBlocks	**0.872 ± 0.016**	0.799 ± 0.033	0.871 ± 0.008	0.449 ± 0.010	0.739 ± 0.004	0.500 ± 0.034	0.746 ± 0.017	0.705 ± 0.015
	Amazon (tab)	**0.120 ± 0.002**	0.117 ± 0.001	0.114 ± 0.011	0.095 ± 0.000	0.105 ± 0.003	0.099 ± 0.008	0.107 ± 0.005	0.112 ± 0.002
	Yelp (tab)	**0.153 ± 0.005**	**0.153 ± 0.003**	**0.153 ± 0.015**	0.097 ± 0.000	0.127 ± 0.005	0.125 ± 0.012	0.135 ± 0.013	0.132 ± 0.003
	MVTec (tab)	**0.778 ± 0.009**	0.740 ± 0.009	0.751 ± 0.011	0.606 ± 0.032	0.604 ± 0.022	0.618 ± 0.028	0.771 ± 0.017	0.698 ± 0.011

$\mathbb{E}\left[\|\nabla_{\mathbf{w}_k} L_k\|_2^2\right]$ essentially quantifies the influence of training data on network optimization. We respectively denote the gradients induced by inliers and anomalies as $\nabla_{\mathbf{w}_k}^{\text{inlier}} L_k$ and $\nabla_{\mathbf{w}_k}^{\text{anom}} L_k$. Based on Taylor series expansion and gradients computation, we derive the following approximation (detailed derivation in Appendix D):

$$\frac{\mathbb{E}\left[\|\nabla_{\mathbf{w}_k}^{\text{inlier}} L_k\|_2^2\right]}{\mathbb{E}\left[\|\nabla_{\mathbf{w}_k}^{\text{anom}} L_k\|_2^2\right]} \approx \frac{N_{\text{inlier}}^2}{N_{\text{anom}}^2}. \quad (9)$$

Due to the imbalanced nature (i.e., $N_{\text{inlier}} \gg N_{\text{anom}}$), inliers govern the optimization process by inducing a significantly larger gradient magnitude. Therefore, the neural network can learn *normal* regularities and patterns in inliers, thereby exposing anomalies in the inference stage.

4. Experiments

Datasets. Ten datasets are employed in our experiments. *Thyroid* and *Arrhythmia* are two medical datasets out of four popular benchmarks used in existing studies of this research line (Bergman & Hoshen, 2020; Qiu et al., 2021). The other two datasets in this suite are from KDD99, while KDD99 is broadly abandoned as virtually all anomalies can be detected via one-dimensional marginal distributions. Instead, a modern intrusion detection dataset, UNSW-NB15, is exploited. Besides, our experiments also base on several datasets from different domains including *Waveform* (physics), *Bank* (marketing), *Thrombin* (biology), and *PageBlocks* (web). These datasets are commonly used in anomaly detection literature (Pang et al., 2021; Campos et al., 2016). We employ tabular version of three vision/NLP datasets including *MVTec (tab)*, *Amazon (tab)*, and *Yelp (tab)*, which are provided by a recent anomaly detection benchmark study (Han et al., 2022).

Evaluation Protocol. We follow the mainstream experimental setting of this research line (Bergman & Hoshen, 2020; Qiu et al., 2021; Shenkar & Wolf, 2022) by using 50% of normal samples for training, while the testing set contains the other half of normal samples as well as all the anomalies Following (Campos et al., 2016; Pang et al., 2019; Wang et al., 2022; Han et al., 2022; Xu et al., 2019), two evaluation metrics, the area under the Receiver-Operating-Characteristic curve (AUC-ROC) and the area under the Precision-Recall curve (AUC-PR), are employed. These two metrics can impartially evaluate the detection performance, while not posing any assumption on the anomaly threshold. Unless otherwise stated, the reported metrics are averaged results over five independent runs.

Baseline Methods. Seven state-of-the-art baselines are utilized. ICL (Shenkar & Wolf, 2022), NeuTraL (Qiu et al., 2021), and GOAD (Bergman & Hoshen, 2020) are contrastive self-supervised methods. RCA (Liu et al., 2021) and GAAL (Liu et al., 2019) are reconstruction-based generative methods. We also utilize DSVDD (Ruff et al., 2018), which is a deep anomaly detection method based on one-class classification. iForest (Liu et al., 2008) is a popular traditional (non-deep) anomaly detection baseline.

4.1. Anomaly Detection Performance

Effectiveness in Real-world Datasets. Table 1 illustrates the detection performance in terms of AUC-ROC and AUC-PR, of our model SLAD and the competing methods. SLAD outperforms its state-of-the-art competing methods on eight out of ten datasets according to both two evaluation metrics. SLAD averagely obtains 7%-21% AUC-ROC improvement and 15%-61% AUC-PR gain over its seven contenders. Par-
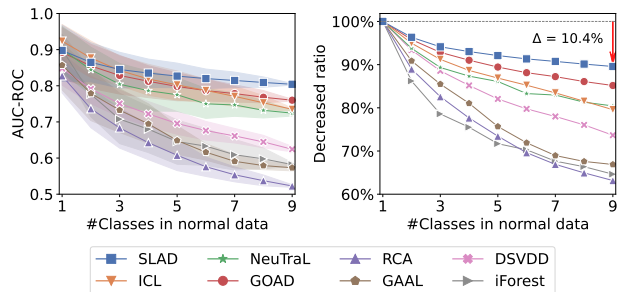
Figure 3. (*Left*) AUC-ROC with 95% confidence intervals on datasets with different numbers of classes considered to be inliers. (*Right*) The proportion of decline compared to only one class appeared in normal data.

Figure 4. Loss values of testing inliers/anomalies during training.

ticularly, on the popular benchmark *Thyroid*, SLAD raises the state-of-the-art AUC-PR by 10 points from 0.82 to 0.92. We also achieve over 190% AUC-PR leap (from 0.15 to 0.43) on *Waveform*. Contrastive self-supervised counterparts also show more competitive performance than reconstruction- or one-class-based methods. The superiority of SLAD validates the effectiveness of our scale learning task in accurately modeling normal regularities of the inherent data structure. Note that SLAD performs less effectively on *Arrhythmia* that has limited data instances (less than 500) since the success of neural networks generally relies on sufficient training data. Nevertheless, SLAD still obtains the best AUC-ROC performance on *Arrhythmia*.

Capability to Handle Complicated Normal Data. This experiment investigates whether discriminative models can better handle complicated normal data than reconstruction- or one-class-based baselines. Following (Qiu et al., 2021), this question is empirically studied by increasing the variability of inliers. We use F-MNIST, a popular multi-class dataset, by treating each pixel as one feature. A suite of datasets is created by sampling data from one class as anomalies and increasing the number of classes considered to be inliers. For each case, we use nine different combinations of selected normal classes and five random seeds per class combination, producing 405 ($9\times9\times5$) datasets in total.

Figure 3 illustrates the AUC-ROC results and the decline proportion w.r.t. the increasing of class numbers in normal data. As each case corresponds to a group of datasets, we also report the 95% confidence interval in addition to the average performance in the left panel. Each detector has a comparably good performance when only one original class appears in inliers. The increased variety in the normal class makes the task more challenging. SLAD downgrades by about 10% and still achieves over 0.8 AUC-ROC when the normal class contains nine prototypes, while over 30% decline is shown in generative models. Reducing errors in point-wise details is hard to converge when the normal class
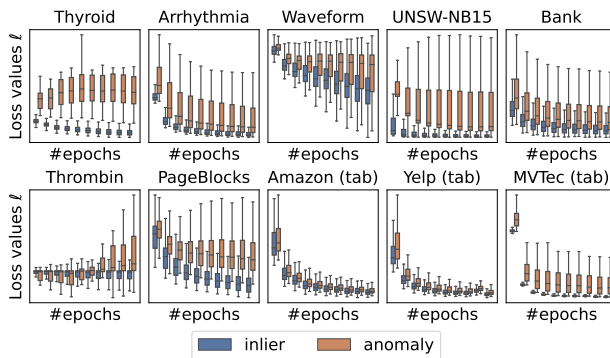
is complicated. The one-class assumption also does not hold when there are more than two original classes considered to be inliers. By contrast, SLAD, ICL, NeuTraL, and GOAD better handle the increased complexity of the normal class. The superiority over contrastive self-supervised counterparts validates the technical advantages of learning to rank subspace-based transformed data compared to only learning the discrimination between transformations.

4.2. A Closer Look at Scale Learning

This experiment further investigates why scale learning can be used for anomaly detection by specifically validating the inlier-priority property and examining whether the learned regularities and patterns are class-dependent.

Validating the Inlier-priority Property. To look into the optimization process of scale learning, we illustrate loss values of testing inliers and anomalies per training epoch, which empirically examines the inlier-priority property. Training data are contaminated by 2% anomalies. Loss values ℓ of testing inliers and anomalies are respectively calculated, and Figure 4 shows box plots of loss values. Neural network inclines to model the inlier class, and testing inliers generally yield lower errors. Compared to inliers, anomalies present significantly higher distributional divergence between derived predictions and targets, and thus two classes can be gradually separated. On datasets *Amazon (tab)* and *Yelp (tab)*, anomalies also yield clearly reduced loss values. It might be because we employ the adapted tabular version of these two datasets, and their tabular representations are not embedded with informative features to distinguish anomalies. Other state-of-the-art competitors also fail to produce good detection results on these two datasets as shown in Table 1.

Validating the Class-dependency. The anomaly detection performance on the used ten datasets validates that the learned regularities cannot apply to anomalies. We further delve into this question by employing the multi-class F-
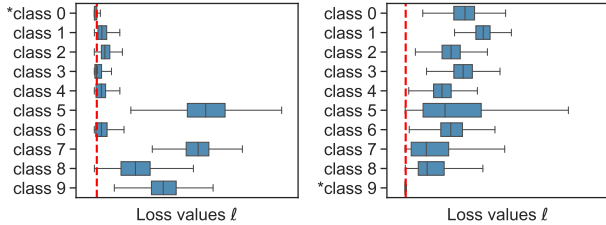
Figure 5. Loss values of data from the trained class and other new classes that only appear in the testing stage. * indicates the trained class. The red dashed line marks the upper quartile loss values of data instances from the trained class.

MNIST dataset. After training on one class, if data instances from new classes do not comply with the learned regularities and patterns embedded in the scale-ranking mechanism, the learned network cannot make expected predictions for data instances from new classes. Therefore, this experiment tests whether the learned network generalizes to other classes by calculating loss values of data instances in new classes compared to the trained class. Two cases are set by employing different original classes, i.e., class 0 (T-shirts) and class 1 (trousers), for training. As shown in Figure 5, loss values ℓ per class are denoted in box plots. The data instances from the trained class can well fit the learned network, deriving obviously lower errors. By contrast, the loss values of other classes are higher. The lower quartile of new classes is much higher than or comparable to the upper quartile of the trained class. These results further validate that our scale learning is class-dependent, thus further supporting its application in anomaly detection. Please note that, in the left panel of Figure 5, loss values in class 3 are much lower than that in other new classes since T-shirts in class 0 are semantically similar to pullovers in class 3. It is more challenging to distinguish this class. Nonetheless, data instances from this new class still show observable higher divergence than the trained class.

4.3. Ablation Study

This experiment answers two questions: (**Q1**) Can several designs in the transformation function T and the labeling function G be replaced with alternatives? (**Q2**) Is it necessary to define scale learning as a distribution alignment task? We first validate the choice of our random affine transformation function T by replacing it with a zero padding function in **w/** T_{Zero} and deeper feed-forward network structure in **w/** T_{MLP}, and the feature weight of the labeling function is removed in **w/o** G_ω. We design another three ablated versions (**w/** L_{ce}, **w/** L_{mse}, and **w/** L_{dcl}), which define scale learning as classification, regression, and contrastive learning using the cross-entropy loss, the mean-squared error, and the deterministic contrastive loss (Qiu et al., 2021), respectively. SLAD is compared with the above six ablated versions. Table 2 reports the AUC-ROC results. SLAD outperforms

Table 2. AUC-ROC performance with improvement rates of SLAD over its ablation variants per dataset. Positive rates are boldfaced. **w/** T_{Zero} cannot handle the ultra-high-dimensional data *Thrombin*. As SLAD only calculates feature weights on low-dimensional data, **w/o** G_ω is performed on three datasets.

		Ablation on the Creation of Supervisory Signals		
Data	**SLAD**	**w/** T_{Zero}	**w/** T_{MLP}	**w/o** G_ω
Thyroid	0.995	0.992 (**0.3%**)	0.995 (0.0%)	0.950 (**4.7%**)
Arrthymia	0.825	0.814 (**1.4%**)	0.821 (**0.5%**)	-
Waveform	0.812	0.759 (**7.0%**)	0.767 (**5.9%**)	0.800 (**1.5%**)
UNSW-NB15	0.937	0.933 (**0.4%**)	0.907 (**3.3%**)	-
Bank	0.730	0.724 (**0.8%**)	0.717 (**1.8%**)	-
Thrombin	0.941		0.698 (**34.8%**)	-
PageBlocks	0.972	0.971 (**0.1%**)	0.967 (**0.5%**)	0.966 (**0.6%**)
Amazon (tab)	0.608	0.552 (**10.1%**)	0.599 (**1.5%**)	-
Yelp (tab)	0.661	0.612 (**8.0%**)	0.654 (**1.1%**)	-
MVTec (tab)	0.812	0.775 (**4.8%**)	0.787 (**3.2%**)	-

		Ablation on Scale Learning		
Data	**SLAD**	**w/** L_{ce}	**w/** L_{mse}	**w/** L_{dcl}
Thyroid	0.995	0.674 (**47.6%**)	0.983 (**1.2%**)	0.978 (**1.7%**)
Arrthymia	0.825	0.728 (**13.3%**)	0.813 (**1.5%**)	0.805 (**2.5%**)
Waveform	0.812	0.527 (**54.1%**)	0.473 (**71.7%**)	0.770 (**5.5%**)
UNSW-NB15	0.937	0.914 (**2.5%**)	0.900 (**4.1%**)	0.922 (**1.6%**)
Bank	0.730	0.517 (**41.2%**)	0.732 (-0.3%)	0.714 (**2.2%**)
Thrombin	0.941	0.704 (**33.7%**)	0.493 (**90.9%**)	0.626 (**50.3%**)
PageBlocks	0.972	0.742 (**31.0%**)	0.979 (-0.7%)	0.976 (-0.4%)
Amazon (tab)	0.608	0.536 (**13.4%**)	0.602 (**1.0%**)	0.610 (-0.3%)
Yelp (tab)	0.661	0.556 (**18.9%**)	0.666 (-0.8%)	0.676 (-2.2%)
MVTec (tab)	0.812	0.646 (**25.7%**)	0.764 (**6.3%**)	0.776 (**4.6%**)

w/ T_{Zero} and **w/** T_{MLP} on most of the used datasets, which validates the choice of random affine transformation when creating representations. Feature weights bring an approximate 5% improvement on the popular *Thyroid* benchmark. Besides, our distribution alignment-based scale learning illustrates better results than other canonical proxy tasks, which illustrates its superiority. It is interesting to note that **w/** L_{mse} obtains superior average performance than **w/** L_{ce}, which implies that using clear quantitative labels may better teach the network than qualitative learning in classification. **w/** L_{dcl} achieves relatively better performance since it also treats a group of transferred data as one training sample and uses the contrastive loss, i.e., it also optimizes predictions in a relative manner.

5. Conclusions

This paper introduces SLAD, a deep anomaly detection method for tabular data. The core novelty of our work includes the scale concept in tabular data and a new kind of data-driven supervisory signals based on scales. This supervision essentially learns the ranking of representations transformed from varied feature subspaces. It is different from current point-wise generative models and classification-, comparison-, and mapping-based discriminative models, presenting a new manner of self-supervised learning. By harnessing this supervision, our model learns inherent regularities and patterns related to the data structure, which of-

fers valuable high-level information for identifying anomalies. Theoretically, we analyze how to ensure the effectiveness of the created data sample in revealing anomalies by determining its shape, and we also examine the inlier-priority property to support the application of scale learning in anomaly detection. Extensive experiments manifest that SLAD significantly outperforms various kinds of state-of-the-art anomaly detectors (including generative, contrastive, and one-class methods) and shows clear superiority when handling complicated data with highly varied inliers.

Acknowledgements

References

Aggarwal, C. C. *Outlier analysis*. Springer, 2017. doi: https://doi.org/10.1007/978-1-4614-6396-2.

Anand, R., Mehrotra, K. G., Mohan, C. K., and Ranka, S. An improved algorithm for neural network classification of imbalanced training sets. *IEEE Transactions on Neural Networks*, 4(6):962–969, 1993.

Bahri, D., Jiang, H., Tay, Y., and Metzler, D. Scarf: Self-supervised contrastive learning using random feature corruption. In *International Conference on Learning Representations*, 2022.

Bergman, L. and Hoshen, Y. Classification-based anomaly detection for general data. In *International Conference on Learning Representations*, 2020.

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

Chen, J., Sathe, S., Aggarwal, C., and Turaga, D. Outlier detection with autoencoder ensembles. In *SIAM International Conference on Data Mining*, pp. 90–98. SIAM, 2017.

Golan, I. and El-Yaniv, R. Deep anomaly detection using geometric transformations. In *Advances in Neural Information Processing Systems*, pp. 9758–9769, 2018.

Goyal, S., Raghunathan, A., Jain, M., Simhadri, H. V., and Jain, P. DROCC: Deep robust one-class classification. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 3711–3721. PMLR, 2020.

Han, S., Hu, X., Huang, H., Jiang, M., and Zhao, Y. Adbench: Anomaly detection benchmark. In *Advances in Neural Information Processing Systems: Datasets and Benchmarks Track*, 2022.

Hendrycks, D., Mazeika, M., and Dietterich, T. Deep anomaly detection with outlier exposure. In *International Conference on Learning Representations*, 2019.

Larsen, A. B. L., Sønderby, S. K., Larochelle, H., and Winther, O. Autoencoding beyond pixels using a learned similarity metric. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48, pp. 1558–1566. PMLR, 2016.

Li, C.-L., Sohn, K., Yoon, J., and Pfister, T. Cutpaste: Self-supervised learning for anomaly detection and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9664–9674, 2021.

Liu, B., Wang, D., Lin, K., Tan, P.-N., and Zhou, J. Rca: A deep collaborative autoencoder approach for anomaly detection. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pp. 1505–1511, 2021.

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

Liu, Y., Li, Z., Zhou, C., Jiang, Y., Sun, J., Wang, M., and He, X. Generative adversarial active learning for unsupervised outlier detection. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1517–1528, 2019.

Liznerski, P., Ruff, L., Vandermeulen, R. A., Franks, B. J., Kloft, M., and Muller, K. R. Explainable deep one-class classification. In *International Conference on Learning Representations*, 2021.

Pang, G., Shen, C., and van den Hengel, A. Deep anomaly detection with deviation networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 353–362, 2019.

Pang, G., Yan, C., Shen, C., Hengel, A. v. d., and Bai, X. Self-trained deep ordinal regression for end-to-end video anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12173–12182, 2020.

Pang, G., Shen, C., Cao, L., and Hengel, A. V. D. Deep learning for anomaly detection: A review. *ACM Computing Surveys*, 54(2), 2021.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

Qiu, C., Pfrommer, T., Kloft, M., Mandt, S., and Rudolph, M. Neural transformation learning for deep anomaly detection beyond images. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 8703–8714. PMLR, 2021.

Qiu, C., Li, A., Kloft, M., Rudolph, M., and Mandt, S. Latent outlier exposure for anomaly detection with contaminated data. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 18153–18167. PMLR, 2022.

Ristea, N.-C., Madan, N., Ionescu, R. T., Nasrollahi, K., Khan, F. S., Moeslund, T. B., and Shah, M. Self-supervised predictive convolutional attentive block for anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13576–13586, 2022.

Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Müller, E., and Kloft, M. Deep one-class classification. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 4393–4402, 2018.

Ruff, L., Kauffmann, J. R., Vandermeulen, R. A., Montavon, G., Samek, W., Kloft, M., Dietterich, T. G., and Müller, K.-R. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795, 2021.

Sehwag, V., Chiang, M., and Mittal, P. Ssd: A unified framework for self-supervised outlier detection. In *International Conference on Learning Representations*, 2021.

Shenkar, T. and Wolf, L. Anomaly detection for tabular data with internal contrastive learning. In *International Conference on Learning Representations*, 2022.

Tack, J., Mo, S., Jeong, J., and Shin, J. Csi: novelty detection via contrastive learning on distributionally shifted instances. In *Advances in Neural Information Processing Systems*, pp. 11839–11852, 2020.

Wang, S., Zeng, Y., Yu, G., Cheng, Z., Liu, X., Zhou, S., Zhu, E., Kloft, M., Yin, J., and Liao, Q. E 3 outlier: A self-supervised framework for unsupervised deep outlier detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):2952–2969, 2022.

Wang, Z., Wang, Y., Xu, H., and Wang, Y. Effective anomaly detection based on reinforcement learning in network traffic data. In *Proceedings of the IEEE 27th International Conference on Parallel and Distributed Systems*, pp. 299–306. IEEE, 2021.

Wolpert, D. H. and Macready, W. G. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

Xia, Y., Cao, X., Wen, F., Hua, G., and Sun, J. Learning discriminative reconstructions for unsupervised outlier removal. In *International Conference on Computer Vision*, pp. 1511–1519, 2015.

Xu, H., Wang, Y., Wang, Y., and Wu, Z. Mix: A joint learning framework for detecting both clustered and scattered outliers in mixed-type data. In *International Conference on Data Mining*, pp. 1408–1413. IEEE, 2019.

Xu, H., Wang, Y., Jian, S., Huang, Z., Wang, Y., Liu, N., and Li, F. Beyond outlier detection: Outlier interpretation by attention-guided triplet deviation network. In *Proceedings of the Web Conference*, pp. 1328–1339, 2021.

Xu, H., Pang, G., Wang, Y., and Wang, Y. Deep isolation forest for anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–14, 2023. doi: 10.1109/TKDE.2023.3270293.

Yao, T., Yi, X., Cheng, D. Z., Yu, F., Chen, T., Menon, A., Hong, L., Chi, E. H., Tjoa, S., Kang, J., et al. Self-supervised learning for large-scale item recommendations. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 4321–4330, 2021.

Yoon, J., Zhang, Y., Jordon, J., and van der Schaar, M. Vime: Extending the success of self-and semi-supervised learning to tabular domain. *Advances in Neural Information Processing Systems*, 33:11033–11043, 2020.

Zhang, Z. and Deng, X. Anomaly detection using improved deep svdd model with data structure preservation. *Pattern Recognition Letters*, 148:1–6, 2021.

Zhao, Y., Nasrullah, Z., and Li, Z. Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research*, 20:1–7, 2019.

Zhou, C. and Paffenroth, R. C. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 665–674, 2017.

A. Proof of Lemma 3.1

Proof. The subspace $\mathcal{S}$ is uniformly sampled from the full feature space $\mathcal{F}$. We assume the sampling process is with replacement, and the probability of sampling one effective feature is $\frac{|\mathcal{G}|}{|\mathcal{F}|}$. The number of effective features in a subspace with j elements is a variable that follows a binominal distribution, i.e., $X \sim b(j, \frac{|\mathcal{G}|}{|\mathcal{F}|})$, and the probability function of X is:

$$Pr_j(X\!=\!k) = \binom{j}{k}(\frac{|\mathcal{G}|}{|\mathcal{F}|})^k(1 - \frac{|\mathcal{G}|}{|\mathcal{F}|})^{j-k}, \tag{10}$$

The length of the sampled subspace follows a discrete uniform distribution of $\{1, \cdots, |\mathcal{F}|\}$. Therefore, the probability of the subspace $\mathcal{S}$ being useful (i.e., $\mathcal{S}$ contains at least $\alpha|\mathcal{G}|$ effective features of $\mathcal{G}$) can be derived as follows.

$$Pr(\mathcal{S} \text{ is useful}) = \frac{1}{|\mathcal{F}|}\sum_{j=1}^{|\mathcal{F}|} Pr_j(X \geq \alpha|\mathcal{G}|). \tag{11}$$

As $Pr_j(X \geq \alpha|\mathcal{G}|)$ is zero when $j < \alpha|\mathcal{G}|$, based on Equation 10, the probability of the sampled subspace $\mathcal{S}$ containing at least $\alpha|\mathcal{G}|$ effective features is:

$$\begin{aligned}
Pr(\mathcal{S} \text{ is useful}) &= \frac{1}{|\mathcal{F}|}\sum_{j=\alpha|\mathcal{G}|}^{|\mathcal{F}|}\sum_{k=\alpha|\mathcal{G}|}^{j} Pr_j(X = k) \\
&= \frac{1}{|\mathcal{F}|}\sum_{j=\alpha|\mathcal{G}|}^{|\mathcal{F}|}\sum_{k=\alpha|\mathcal{G}|}^{j} \binom{j}{k}(\frac{|\mathcal{G}|}{|\mathcal{F}|})^k(1 - \frac{|\mathcal{G}|}{|\mathcal{F}|})^{j-k}
\end{aligned}. \tag{12}$$

$\square$

B. Proof of Theorem 3.2

Proof. In this proof, for the simplicity of notation, we write $|\mathcal{F}|, |\mathcal{G}|, \alpha|\mathcal{G}|$ as n, m, and q, repetitively, and $Pr(\mathcal{S} \text{ is useful})$ is abbreviated as Pr.

We first show that Pr monotonically decreases with the increase of n.

$$\begin{aligned}
Pr_n - Pr_{n+1} =& \frac{1}{n}\sum_{j=q}^{n}\sum_{k=q}^{j}\binom{j}{k}(\frac{m}{n})^k(1-\frac{m}{n})^{j-k} - \frac{1}{n+1}\sum_{j=q}^{n+1}\sum_{k=q}^{j}\binom{j}{k}(\frac{m}{n+1})^k(1-\frac{m}{n+1})^{j-k} \\
=& \frac{1}{n}\sum_{j=q}^{n}\sum_{k=q}^{j}\binom{j}{k}(\frac{m}{n})^k(1-\frac{m}{n})^{j-k} - \frac{1}{n+1}\sum_{j=q}^{n}\sum_{k=q}^{j}\binom{j}{k}(\frac{m}{n+1})^k(1-\frac{m}{n+1})^{j-k} \\
& - \frac{1}{n+1}\sum_{k=q}^{n+1}\binom{n+1}{k}(\frac{m}{n+1})^k(1-\frac{m}{n+1})^{n+1-k} \\
\geq& \frac{1}{n}\sum_{j=q}^{n}\sum_{k=q}^{j}\binom{j}{k}\frac{1}{n}(\frac{m}{n})^k(1-\frac{m}{n})^{n-k} - \sum_{k=q}^{n}\binom{n}{k}\frac{1}{n+1}(\frac{m}{n+1})^k(1-\frac{m}{n+1})^{n-k} \\
& - \frac{1}{n+1}\sum_{k=q}^{n}\binom{n+1}{k}(\frac{m}{n+1})^k(1-\frac{m}{n+1})^{n+1-k} - \frac{1}{n+1}(\frac{m}{n+1})^{n+1} \\
\geq& \frac{1}{n}(\frac{m}{n})^n(1-\frac{m}{n})^{n-n} - \frac{1}{n+1}(\frac{m}{n+1})^n(1-\frac{m}{n+1})^{n-n} \\
& - \frac{n+1}{n+1-n}\frac{1}{n+1}(\frac{m}{n+1})^n(1-\frac{m}{n+1})^{n+1-n} - \frac{1}{n+1}(\frac{m}{n+1})^{n+1} \\
\geq& 0
\end{aligned} \tag{13}$$

Therefore, we have

$$\frac{1}{n}\sum_{j=q}^{n}\sum_{k=q}^{j}\binom{j}{k}(\frac{m}{n})^k(1-\frac{m}{n})^{j-k} \geq \lim_{n\to\infty}\frac{1}{n}\sum_{j=q}^{n}\sum_{k=q}^{j}\binom{j}{k}(\frac{m}{n})^k(1-\frac{m}{n})^{j-k}. \tag{14}$$

We note that

$$\lim_{n\to\infty}\frac{1}{n}\sum_{j=q}^{n}\sum_{k=0}^{q-1}\binom{j}{k}(\frac{m}{n})^k(1-\frac{m}{n})^{j-k} = \lim_{n\to\infty}\sum_{j=q}^{n}\sum_{k=0}^{q-1}\frac{j!m^k}{(j-k)!k!n^{k+1}}(1-\frac{m}{n})^j. \tag{15}$$

In addition,

$$
\begin{aligned}
\lim_{n\to\infty}\sum_{j=q}^{n}\frac{j!m^k}{(j-k)!k!n^{k+1}}(1-\frac{m}{n})^j &= \frac{1}{k!m}\lim_{n\to\infty}\sum_{j=1}^{n}\frac{j!}{(j-k)!}(\frac{m}{n})^{k+1}(1-\frac{m}{n})^j \\
&= \frac{1}{k!m}\lim_{n\to\infty}\sum_{j=1}^{n}j^k(\frac{m}{n})^{k+1}(1-\frac{m}{n})^j \\
&= \frac{e^{-m}}{k!m}\left(k!e^m - k! - k!m - m^k - O(m^{k-1})\right) \\
&\leq \frac{1}{m}.
\end{aligned}
\tag{16}
$$

According to the above inequality, we can derive the following results from Equation (15):

$$\lim_{n\to\infty}\frac{1}{n}\sum_{j=q}^{n}\sum_{k=0}^{q-1}\binom{j}{k}(\frac{m}{n})^k(1-\frac{m}{n})^{j-k} \leq \sum_{k=0}^{q-1}\frac{1}{m} \leq \frac{q}{m}. \tag{17}$$

Hence,

$$
\begin{aligned}
&\lim_{n\to\infty}-\frac{1}{n}\sum_{j=q}^{n}\sum_{k=0}^{q-1}\binom{j}{k}(\frac{m}{n})^k(1-\frac{m}{n})^{j-k} \geq -\frac{q}{m} \\
&\lim_{n\to\infty}\frac{1}{n}\sum_{j=q}^{n}\left(1-\sum_{k=0}^{q-1}\binom{j}{k}(\frac{m}{n})^k(1-\frac{m}{n})^{j-k}\right) \geq 1-\frac{q}{m} \\
&\lim_{n\to\infty}\frac{1}{n}\sum_{j=q}^{n}\sum_{k=q}^{j}\binom{j}{k}(\frac{m}{n})^k(1-\frac{m}{n})^{j-k} \geq 1-\frac{q}{m},
\end{aligned}
\tag{18}
$$

As

$$\lim_{n\to\infty}\frac{1}{n}\sum_{j=q}^{n}\sum_{k=q}^{j}\binom{j}{k}(\frac{m}{n})^k(1-\frac{m}{n})^{j-k} - (1-\frac{q}{m}) = \frac{e^{-m}(q! + m^{q-1} + O(m^{q-1}))}{(q-1)!m}, \tag{19}$$

i.e., Equation (19) monotonically decreases when m is a large number.

Also,

$$\lim_{m\to\infty}\left(\lim_{n\to\infty}\frac{1}{n}\sum_{j=q}^{n}\sum_{k=q}^{j}\binom{j}{k}(\frac{m}{n})^k(1-\frac{m}{n})^{j-k} - (1-\frac{q}{m})\right) = 0. \tag{20}$$

Based on Equation (18)(19)(20), we finally show the lower bound of the probability is

$$\inf(Pr) = 1 - \frac{q}{m} = 1 - \alpha. \tag{21}$$

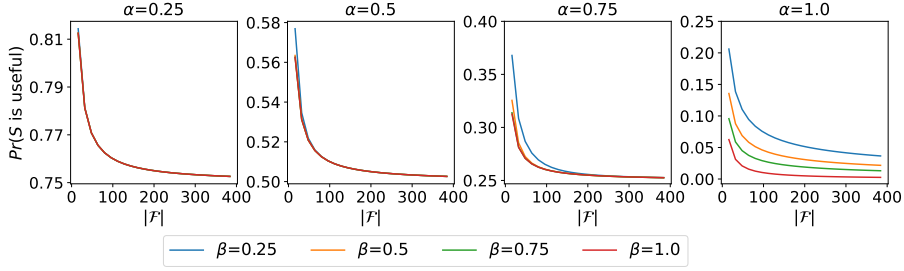$\square$

Figure 6. $Pr(\mathcal{S}$ is useful) of different α and β settings w.r.t. the dimensionality of feature space.

C. Empirical Validation of Theorem 3.2

We further empirically inspect the lower bound of $Pr(\mathcal{S}$ is useful) in Theorem 3.2. Two ratios α and β are chosen from $\{0.25, 0.5, 0.75, 1.0\}$, and the data dimensionality $|\mathcal{F}|$ ranges from 0 to 400. Figure 6 shows how probability Pr changes w.r.t. different α, β, and $|\mathcal{F}|$. The probability Pr monotonically decreases with the increase of $|\mathcal{F}|$, which is also proved in Appendix B. The function curves of Pr show a clear lower bound that is determined by the α value. As shown in four represented α cases, the lower bound is shown to be $1 - \alpha$, which is the same result as proved in Appendix B

D. Theoretical Derivation of the Inlier-Priority Property in Scale Learning

We first consider the gradient $\nabla_{w_{(s,k)}} \ell_k$ in Equation (8). ℓ_k is the kth part of the summation in Equation (3), i.e.,

$$\ell_k = \frac{1}{2}\tilde{p}_k \log\left(\frac{\tilde{p}_k}{\frac{1}{2}(\tilde{p}_k + \tilde{y}_k)}\right) + \frac{1}{2}\tilde{y}_k \log\left(\frac{\tilde{y}_k}{\frac{1}{2}(\tilde{p}_k + \tilde{y}_k)}\right). \tag{22}$$

Thus, $\nabla_{w_{(s,k)}} \ell_k$ is given by

$$\begin{aligned}
\nabla_{w_{(s,k)}} \ell_k &= \frac{\partial \ell_k}{\partial \tilde{p}_k} \frac{\partial \tilde{p}_k}{\partial p_k} \frac{\partial p_k}{\partial w_{(s,k)}} \\
&= \frac{1}{2}\left(\log 2 + \log \tilde{p}_k - \log(\tilde{y}_k + \tilde{p}_k)\right) \cdot \tilde{p}_k (1 - \tilde{p}_k) \cdot h_s,
\end{aligned} \tag{23}$$

where p_k is the prediction before the softmax layer, and h_s is the output of the sth node in the penultimate layer.

We then consider $\mathbb{E}\left[\nabla_{w_{(s,k)}} \ell_k^{(i)} \nabla_{w_{(s,k)}} \ell_k^{(j)}\right]$. Let $g_{i,j}^{(s,k)}(\mathbf{w}_k) = \nabla_{w_{(s,k)}} \ell_k^{(i)} \nabla_{w_{(s,k)}} \ell_k^{(j)}$. To simplify computation, we drive the following function according to the second-order Taylor series expansion:

$$g_{i,j}^{(s,k)}(\mathbf{w}_k) \approx g_{i,j}^{(s,k)}(\boldsymbol{\mu}) + \nabla_{\mathbf{w}_k} g_{i,j}^{(s,k)}(\boldsymbol{\mu}) \cdot (\mathbf{w}_k - \boldsymbol{\mu}) + \frac{1}{2}(\mathbf{w}_k - \boldsymbol{\mu})^T \cdot \nabla_{\mathbf{w}_k}^2 g_{i,j}^{(s,k)}(\boldsymbol{\mu}) \cdot (\mathbf{w}_k - \boldsymbol{\mu}). \tag{24}$$

where $\boldsymbol{\mu}$ is the expectation of $\mathbf{w}_k$.

Given that the weights are initialized by a uniform distribution $[-1, 1]$, we have $\mu_{(s,k)} = 0$ and $\boldsymbol{\mu} = \mathbf{0}$, Hence,

$$\mathbb{E}\left[g_{i,j}^{(s,k)}(\mathbf{w}_k)\right] \approx \mathbb{E}\left[g_{i,j}^{(s,k)}(\mathbf{0})\right] + \mathbb{E}\left[\nabla_{\mathbf{w}_k} g_{i,j}^{(s,k)}(\mathbf{0})\mathbf{w}_k\right] + \mathbb{E}\left[\frac{1}{2}\mathbf{w}_k^T \nabla_{\mathbf{w}_k}^2 g_{i,j}^{(s,k)}(\mathbf{0})\mathbf{w}_k\right]. \tag{25}$$

The prediction p_k is zero when network weights $\mathbf{w}_k = \mathbf{0}$, and after a softmax layer, $\tilde{p}_k = \frac{1}{c}$. As $\tilde{y}_k$ is a constant number, we assume $\tilde{y}_k = \frac{1}{c}$ here. In addition, $\mathbb{E}(w_{(s,k)}^2) = \frac{1}{3}$ and $\mathbb{E}(w_{(s,k)}w_{(t,k)}) = 0$ when $s \neq t$. Hence, we have

$$\mathbb{E}\left[g_{i,j}^{(s,k)}(\mathbf{w}_k)\right] \approx \frac{1}{6}\sum_{t=1}^{u}\sum_{l=1}^{c} \nabla_{w_{t,l}}^2 g_{i,j}^{(s,k)}(\mathbf{0}). \tag{26}$$

13

To compute $\nabla^2_{\mathbf{w}_k} g^{(s,k)}_{i,j}(\mathbf{0})$, we first compute the following derivatives:

$$
\begin{aligned}
\nabla_{w_{(t,l)}} \tilde{p}_k &= \tilde{p}_k(\delta(k,l) - \tilde{p}_k) \cdot h_t, \\
\nabla_{w_{(t,l)}} \tilde{p}_l &= \tilde{p}_l(1 - \tilde{p}_l) \cdot h_t, \\
\nabla^2_{w_{(t,l)}} \tilde{p}_k &= h_t\big((\delta(k,l) - \tilde{p}_l) \cdot \nabla_{w_{(t,l)}} \tilde{p}_k - \tilde{p}_k \cdot \nabla_{w_{(t,l)}} \tilde{p}_l\big).
\end{aligned}
\tag{27}
$$

where δ outputs whether two inputs are the same, i.e., $\delta(k,l) = 1$ if $k = l$ and $\delta(k,l) = 0$ otherwise.

Therefore,

$$
\begin{aligned}
&\nabla_{w_{(t,l)}} \Big(\tilde{p}_k(1 - \tilde{p}_k)(\log 2 + \log \tilde{p}_k - \log(\tfrac{1}{c} + \tilde{p}_k)) \Big) \\
&= \Big((1 - 2\tilde{p}_k)\big(\log 2 + \log \tilde{p}_k - \log(\tfrac{1}{c} + \tilde{p}_k) \big) + \frac{1 - \tilde{p}_k}{1 + c \cdot \tilde{p}_k} \Big) \nabla_{w_{(t,l)}} \tilde{p}_k,
\end{aligned}
\tag{28}
$$

and

$$
\begin{aligned}
&\nabla^2_{w_{(t,l)}} \Big(\tilde{p}_k(1 - \tilde{p}_k)(\log 2 + \log \tilde{p}_k - \log(\tfrac{1}{c} + \tilde{p}_k)) \Big) \\
&= \nabla_{w_{(t,l)}} \Big(\Big((1 - 2\tilde{p}_k)\big(\log 2 + \log \tilde{p}_k - \log(\tfrac{1}{c} + \tilde{p}_k) \big) + \frac{1 - \tilde{p}_k}{1 + c \cdot \tilde{p}_k} \Big) \nabla_{w_{(t,l)}} \tilde{p}_k \Big) \\
&= -2\big(\log 2 + \log(\tilde{p}_k) - \log(\tfrac{1}{c} + \tilde{p}_k) \big)\big(\nabla_{w_{(t,l)}} \tilde{p}_k\big)^2 + (1 - 2\tilde{p}_k)(\frac{1}{1 + c \cdot \tilde{p}_k})\big(\nabla_{w_{(t,l)}} \tilde{p}_k\big)^2 \\
&\quad - \frac{(1 + c + c \cdot \tilde{p}_k)\big(\nabla_{w_{(t,l)}} \tilde{p}_k\big)^2 - c \cdot \big(\nabla_{w_{(t,l)}} \tilde{p}_k\big)^3}{(1 + c \cdot \tilde{p}_k)^2} + \Big((1 - 2\tilde{p}_k)\big(\log 2 + \log(\tilde{p}_k) - \log(\tfrac{1}{c} + \tilde{p}_k) \big) + \frac{1 - \tilde{p}_k}{1 + c \cdot \tilde{p}_k} \Big) \nabla^2_{w_{(t,l)}} \tilde{p}_k.
\end{aligned}
\tag{29}
$$

If $k = l$ and $\mathbf{w} = 0$, we have

$$
\begin{aligned}
\nabla_{w_{(t,l)}} \Big(\tilde{p}_k(1 - \tilde{p}_k)(\log 2 + \log \tilde{p}_k - \log(\tfrac{1}{c} + \tilde{p}_k)) \Big) &= \frac{(1 - c)^2}{2c^3} h_t, \\
\nabla^2_{w_{(t,l)}} \Big(\tilde{p}_k(1 - \tilde{p}_k)(\log 2 + \log \tilde{p}_k - \log(\tfrac{1}{c} + \tilde{p}_k)) \Big) &= \frac{3(c - 3)(c - 1)^2}{4c^4}(h_t)^2.
\end{aligned}
\tag{30}
$$

If $k \neq l$ and $\mathbf{w} = 0$, we have

$$
\begin{aligned}
\nabla_{w_{(t,l)}} \Big(\tilde{p}_k(1 - \tilde{p}_k)(\log 2 + \log \tilde{p}_k - \log(\tfrac{1}{c} + \tilde{p}_k)) \Big) &= \frac{1 - c}{2c^3} h_t, \\
\nabla^2_{w_{(t,l)}} \Big(\tilde{p}_k(1 - \tilde{p}_k)(\log 2 + \log \tilde{p}_k - \log(\tfrac{1}{c} + \tilde{p}_k)) \Big) &= \frac{-9 + 7c - 2c^2}{4c^4}(h_t)^2.
\end{aligned}
\tag{31}
$$

Hence,

$$
\begin{aligned}
&\frac{1}{6} \sum_{t=1}^{u} \sum_{l=1}^{c} \nabla^2_{w_{t,l}} g^{(s,k)}_{i,j}(\mathbf{0}) \\
&= \frac{1}{6} \frac{1}{4} h^{(i)}_s h^{(j)}_s \sum_{t=1}^{u} \Big(\frac{(c^2 - c + 1)(c - 1)^3}{4c^6} h^{(i)}_t h^{(j)}_t \Big) + \frac{(c - 1)(3c^3 - 14c^2 + 16c - 9)}{4c^4}((h^{(i)}_t)^2 + (h^{(j)}_t)^2)).
\end{aligned}
\tag{32}
$$

The literature (Anand et al., 1993) has proved that the randomly initialized network have: $\mathbb{E}\big[h^{(i)}_s h^{(j)}_s\big] \approx \frac{1}{4}$, $\mathbb{E}\big[(h^{(i)}_s)^2(h^{(j)}_s)^2\big] \approx \frac{1}{16}$, and $\mathbb{E}\big[(h^{(i)}_s)^3(h^{(j)}_s)\big] \approx \frac{1}{16}$.

Therefore,

$$
\begin{aligned}
\mathbb{E}\big[\|\nabla_{\mathbf{w}_k} L_k\|^2_2\big] &\approx N^2 u\Big(\frac{(c^3 + 1)(c - 1)^3}{1536c^6(c + 1)} + \frac{(c - 1)(3c^3 - 14c^2 + 16c - 9)}{768c^4} \Big) \\
&\triangleq Q N^2,
\end{aligned}
\tag{33}
$$

where Q is a constant related to c. The gradient magnitude is proportional to the size of training data.

We can then compare the magnitude induced by inliers and anomalies, i.e.,

$$\frac{\mathbb{E}\big[\|\nabla_{\mathbf{w}_k}^{\text{inlier}} L_k\|_2^2\big]}{\mathbb{E}\big[\|\nabla_{\mathbf{w}_k}^{\text{anom}} L_k\|_2^2\big]} \approx \frac{N_{\text{inlier}}}{N_{\text{anom}}}. \tag{34}$$

E. Algorithms Details

We outline the detailed training procedure of SLAD in Algorithm 1. SLAD uses interactions between features to calculate feature weights for the subsequent labeling function in Steps 4-5. This process is omitted by using a threshold δ in Steps 6-7 when handling high-dimensional data due to the huge computational overhead and the inaccuracy caused by irrelevant/noisy features. SLAD creates scale-based supervisory signals via the transformation function T and labeling function G in Steps 9-20. Every c transformed vectors are contained in a $\mathbf{U}$ matrix in Steps 14-16, and finally creates r matrices in $\mathcal{O}$ attached with labels in $\mathcal{Y}$ via Step 18. SLAD further trains a neural network Φ to rank scale values via the loss values ℓ and the loss function L in Steps 21-28.

Algorithm 1 Training Procedure of SLAD

1: **Input:** Dataset $\mathcal{X}$.
2: **Output:** Trained neural network Φ^*
3: Initialize $\omega \in \mathbb{R}^D$ as the a feature weight list.
4: **if** $D < \delta$ **then**
5:      $\omega \leftarrow \left\{ \frac{1}{|\mathcal{F}|} \sum_{k'=1}^{|\mathcal{F}|} \left| \frac{\text{cov}(\mathbf{u}_k, \mathbf{u}_{k'})}{\text{dev}(\mathbf{u}_k)\text{dev}(\mathbf{u}_{k'})} \right| \right\}_{k=1}^{D}$
6: **else**
7:      $\omega \leftarrow \mathbf{1}$
8: **end if**
9: Initialize $\mathcal{O} \leftarrow \{\}, \mathcal{Y} \leftarrow \{\}$
10: **for** $j = 1$ **to** r **do**
11:      **for** $\mathbf{x} \in \mathcal{X}$ **do**
12:          Initialize a $c \times h$ matrix $\mathbf{U}$ for transferred data.
13:          Initialize a c-dimensional vector $\mathbf{y}$ for scale-based labels.
14:          **for** $i = 1$ **to** c **do**
15:              Sample a feature subset from $\mathcal{S}_i \subseteq \mathcal{F}$.
16:              $\mathbf{U}_{(i,\cdot)} \leftarrow T(\mathbf{x}_{(\mathcal{S}_i)}), \mathbf{y}_{(i)} \leftarrow G(\mathcal{S}_i, h)$
17:          **end for**
18:          $\mathcal{O} \leftarrow \mathcal{O} \cup \mathbf{U}, \mathcal{Y} \leftarrow \mathcal{Y} \cup \mathbf{y}$
19:      **end for**
20: **end for**
21: Initialize parameters in neural network Φ.
22: **repeat**
23:      **repeat**
24:          Sample mini-batch training data $\mathcal{B}_x \sim \mathcal{O}, \mathcal{B}_y \sim \mathcal{Y}$.
25:          $L \leftarrow \mathbb{E}_{\mathbf{U} \sim \mathcal{B}_x, \mathbf{y} \sim \mathcal{B}_y} \Big[\ell\big(\sigma(\Phi(\mathbf{U})) \| \sigma(\mathbf{y})\big) \Big]$
26:          Update network parameters according to L
27:      **until** Reach maximum number of mini-batches
28: **until** Reach maximum training epochs
29: **return:** Φ^*

F. Datasets

Table 3 reports the domains of the used datasets and their statistical information including data size, dimensionality, the number of anomalies (#anom), and anomaly percentage to the whole data size (ratio). These datasets are publicly available and they are broadly used in related literature (Han et al., 2022; Shenkar & Wolf, 2022; Bergman & Hoshen, 2020; Qiu et al., 2021; Xu et al., 2023). In *UNSW-NB15*, we use the network traffic of "DoS" attack as anomalies.

Table 3. Dataset information. Data size and dimensionality indicate the number of data instances, and the number of features. #anom and ratio denote the number of anomalies and the abnormal percentage over all the data instances. The statistical information of MVTec (tab) is the average value over its fifteen sub-datasets.

Dataset	Domain	Data size	Dimensionality	#anom	ratio
Thyroid	Healthcare	3,772	6	93	2.5%
Arrhythmia	Healthcare	452	274	66	14.6%
Waveform	Physics	3,443	21	100	2.9%
UNSW-NB15	Intrusion detection	96,000	196	3,000	3.1%
Bank	Marketing	41,188	62	4,640	11.3%
Thrombin	Biology	1,909	139,351	42	2.2%
PageBlocks	Web	5,393	10	510	9.5%
Amazon (tab)	NLP	10,000	768	500	5.0%
Yelp (tab)	NLP	10,000	768	500	5.0%
MVTec (tab)	CV	357	512	84	23.5%

G. Implementation Details

For SLAD, we use $h = 128$ in the transformation function to represent sub-vectors to a unified 128-dimensional frame. Our transformation function uses c and r to respectively control the number of transferred data in each training sample and the repeat times. We use $c = 10$ and $r = 20$ by default. In the labeling function G, we use $\delta = 50$ as the threshold to omit the correlation-based weighting function. The magnification factor γ in G is set as 200 by default. All the deep anomaly detectors use a 1e-3 learning rate, and the mini-batch size is 128. Multi-layer perceptron network is exploited for these deep detectors except GOAD which uses the default convolutional network. We use the LeakyReLU function as the non-linear activation layer, and the hidden layer contains 100 neural units. For ICL, NeuTraL, GOAD, and RCA, the representation dimensionality is set as 128, which is the same as h in SLAD. These datasets are split into three groups, i.e., datasets with smaller sizes including *Thyroid*, *Arrhythmia*, and *Waveform*, transferred CV dataset *MVTec (tab)*, and other datasets. We vary the number of training epochs from {10, 20, 50, 100} for three categories and report empirically good results.

Our experiments are conducted on a workstation with Intel Xeon Silver 4210R CPU, a single NVIDIA TITAN RTX GPU, and 64 GB RAM. All the anomaly detection methods used in our experiments are implemented in Python. Deep detectors (SLAD, ICL, NeuTraL, GOAD, RCA) use the PyTorch framework (Paszke et al., 2019). We implement them in the DeepOD package (`https://github.com/xuhongzuo/deepod`). The implementation of GAAL is taken from the PyOD package (Zhao et al., 2019), and the non-deep baseline iForest is implemented in the Scikit-learn package. The source code of SLAD is available at `https://github.com/xuhongzuo/scale-learning`.

H. Additional Empirical Results

H.1. Effectiveness on ADBench

We further employ ADBench (Han et al., 2022), the latest anomaly detection benchmark collection of tabular data, which consists of 47 classical tabular datasets. These datasets are from various domains and contain varied sizes, dimensionality, anomaly types, and noise levels. It is very challenging to be the universal winner in all cases, as has been suggested in the no-free-lunch theorem (Wolpert & Macready, 1997). Therefore, we focus on the overall performance on this large-scale benchmark. We employ the average rank and the number/percentage of cases that each anomaly detector ranks within the Top2/Top4 positions among 8 anomaly detector candidates (our model and seven contenders). Table 4 shows the experiment results. Our method SLAD successfully outperforms all of its seven state-of-the-art competing methods according to six evaluation metrics. It is also noteworthy that iForest is a powerful baseline, even if its workflow is very simple. In addition, the advanced self-supervised methods ICL and NeuTraL show competitive AUC-PR performance. Nevertheless, our method still leads to new state-of-the-art detection accuracy across this comprehensive benchmark collection.

H.2. Robustness w.r.t. Different Contamination Levels in Training Data

Recall that we follow the commonly used experimental protocol (half of the normal samples for training and the rest of normal data in addition to all the anomalies for testing) in Section 4.1. However, in real-world applications, training data might be contaminated by unknown anomalies, Thus, we investigate the robustness w.r.t. different contamination ratios

16

Table 4. Detection performance on a benchmark collection with 47 datasets. Avg. Rank indicates the average ranking of all the datasets. #Top2 and #Top4 respectively count the times that each anomaly detector rank within the Top2 and Top4 positions, and %Top2 and %Top4 denote the corresponding percentage. ↑ indicates that the higher the indicator, the better the detection performance, while ↓ denotes the lower the better.

| Model	AUC-ROC					AUC-PR				
	Avg. Rank (↓)	#Top2 (↑)	%Top2 (↑)	#Top4 (↑)	%Top4 (↑)	Avg. Rank (↓)	#Top2 (↑)	%Top2 (↑)	#Top4 (↑)	%Top4 (↑)
iForest	3.77	16	34.0%	27	57.4%	4.26	12	25.5%	24	51.1%
DSVDD	4.66	10	21.3%	20	42.6%	4.70	12	25.5%	19	40.4%
GAAL	6.23	4	8.5%	9	19.1%	6.21	4	8.5%	8	17.0%
RCA	4.13	13	27.7%	26	55.3%	4.21	11	23.4%	26	55.3%
GOAD	6.15	5	10.6%	10	21.3%	5.85	5	10.6%	10	21.3%
NeuTraL	3.68	16	34.0%	28	59.6%	3.77	17	36.2%	31	66.0%
ICL	4.09	12	25.5%	27	57.4%	3.64	16	34.0%	33	70.2%
SLAD (ours)	**2.98**	**22**	**46.8%**	**39**	**83.0%**	**2.98**	**21**	**44.7%**	**38**	**80.9%**
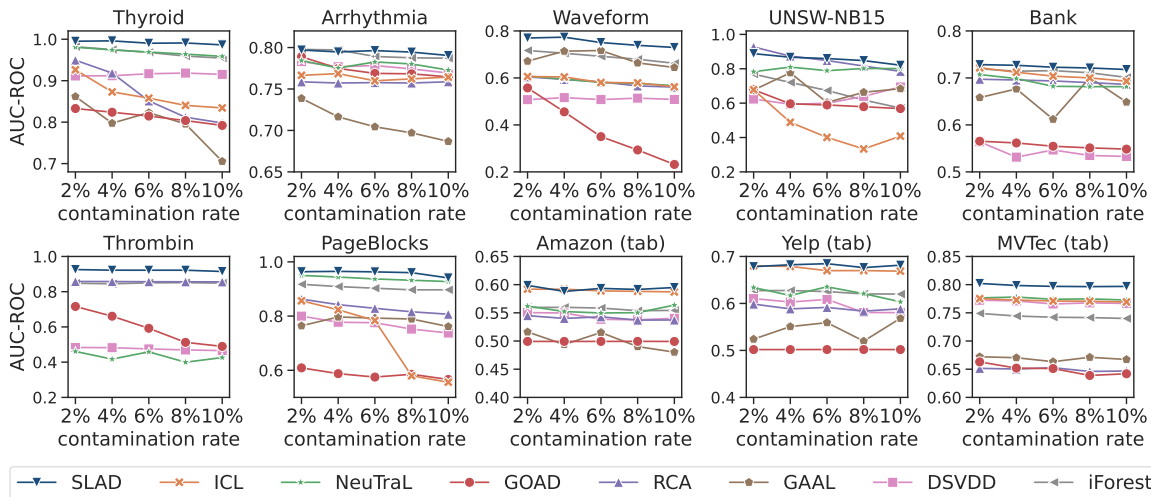
Figure 7. AUC-ROC Performance of SLAD and its competing methods w.r.t. different contamination rate (the percentage of anomalies in training data).

in training data. Half of the anomalies are used for testing, and the other half are candidate anomalies to contaminate the training set. As anomalies are rare events, contamination rates are taken from 2% to 10%. Following (Xu et al., 2023; Pang et al., 2019), if the number of candidate anomalies is insufficient to meet the target contamination ratio, we swap 5% features in two randomly sampled real anomalies to synthesize new anomalies.

Figure 7 depicts the AUC-ROC performance of eight anomaly detectors w.r.t. contamination rate in training data. SLAD shows consistent superiority over its contenders in different scenarios. As these deep anomaly detectors essentially model normal conditions and identify anomalies by measuring the deviation to learned models, contaminated anomalies in training data may mislead the modeling process, leading to the overfitting problem. Generally, detection performance is downgraded with the increase of contamination rate. The increased anomalies may contain diverse behaviors, and thus the performance of some anomaly detectors shows fluctuant trends. In addition, almost all anomaly detectors show stable performance on *Amazon (tab)* and *Yelp (tab)*. These two datasets are challenging, and all the anomaly detectors fail to yield sufficient detection results as reported in Table 1. The anomalies in these data are similar to inliers and hard to distinguish since the transferred tabular features may only describe semantic information and not be informative to identify anomalies. Therefore, anomaly detectors might not suffer from the overfitting problem induced by anomaly contamination.

H.3. Scalability Test

We further conduct a scalability test to examine the time efficiency of SLAD. A group of synthetic datasets is produced containing 5,000 data instances and varied dimensionality (i.e., {64, 128, 256, 1,024, 2,048, 4,096}). We create another
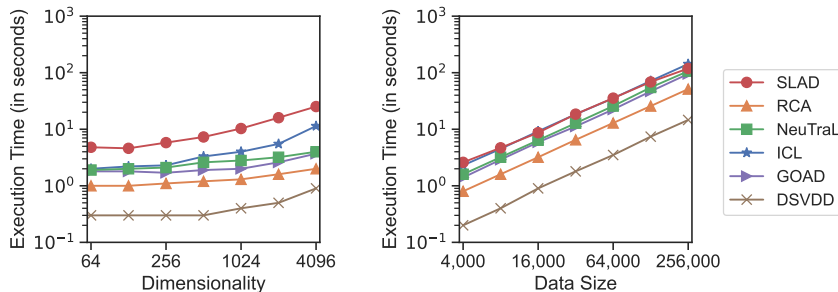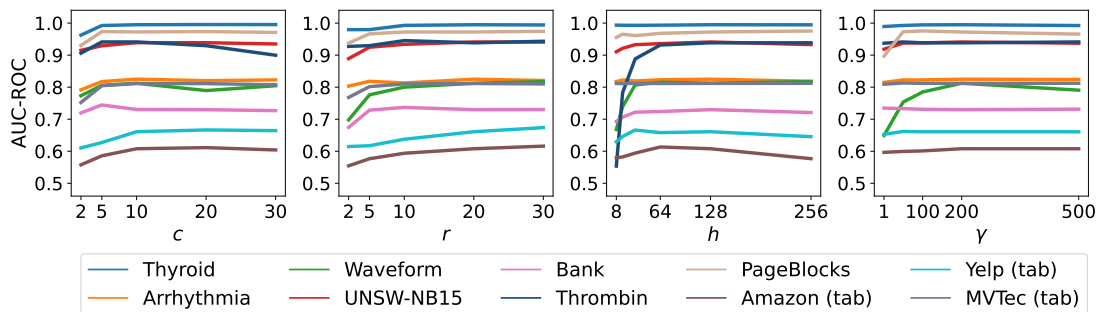
Figure 8. Scalability test results.

Figure 9. Detection performance of SLAD with different hyper-parameter settings.

suite of data with different data sizes ranging from 4,000 to 256,000 and fixed dimensionality (i.e., 32). These two groups of datasets are used to respectively examine the scalability w.r.t. dimensionality and data size. For the sake of fairness, SLAD is compared with deep anomaly detectors implemented in the PyTorch framework. We report the execution time including the training time of 10 epochs and the inference time. Figure 8 demonstrates the scalability test results. SLAD is less efficient than other anomaly detectors on high-dimensional data. In our implementation, we employ subspaces of the original data, and the neural transformation function T needs to prepare a number of neural layers to handle different subspace lengths. This process may induce relatively heavy overhead. Nevertheless, thanks to the GPU acceleration power, all the anomaly detectors take less than 25 seconds to handle 4,096-dimensional data. In addition, recall that we use a dataset *Thrombin* that contains over one hundred thousand features, SLAD can process this ultra-high-dimensional data in 6 minutes. In terms of the scalability w.r.t. data size, SLAD and contrastive models (ICL, NeuTraL, and GOAD) have comparable time efficiency. DSVDD is faster since it maps all the input to a representation and poses a distance-based one-class constraint. However, SLAD and contrastive self-supervised models can lead to superior detection accuracy.

H.4. Parameter Sensitivity

This experiment investigates the sensitivity of our model to four key hyper-parameters including c (the number of elements in each created data sample of scale learning), r (the factor that adjusts the total size of training samples), h (the dimensionality of representations), and γ (the magnification factor in the labeling function G), thereby guiding how to set them during practical usages. In each experiment, the tested parameter takes different values, while others are fixed as default. Candidate values for r and c are taken from $\{2, 5, 10, 20, 30\}$, the dimensionality h is chosen from 8 to 256, and γ uses values in $\{1, 50, 100, 200, 500\}$. Figure 9 reports the AUC-ROC performance of SLAD with different settings. In terms of the parameter c, the increase of c from 2 to 10 brings observable AUC-ROC gain on most of the datasets, while the performance is clearly downgraded on two datasets when c continually raises. SLAD treats a whole list of transferred data as one training sample, but simultaneously optimizing a very long list of predictions (i.e., a large c) may also harm the performance. In terms of r, a larger r generally brings better results, which is also true in many similar ensemble scenarios. Based on the above experimental results, we recommend using $n = 10$ and $r = 20$ in practical usage. Lower dimensionality h cannot contain enough information, which downgrades the detection performance, especially on the high-dimensional dataset *Thrombin*. On the contrary, a large h may be not always feasible. We set $h = 128$ by default in SLAD, which is

18

frequently used as representation dimensionality in many deep models. As for the magnification factor γ, a larger γ can enlarge the spacing between different scale values. We use a unified representation dimensionality h for datasets with varied feature numbers, and thus raw scale values are very small and are hard to be differentiated when handling low-dimensional data (their sub-vectors are short). By using a magnification factor, detection performance shows notable improvement on *Waveform* and *PageBlocks*. SLAD performs stably on other datasets w.r.t. γ.

I. Future Directions

We propose scale learning as a novel self-supervised proxy task for anomaly detection in tabular data. This section notes three future directions that are worth to be explored: (i) SLAD can be generalized to different learning tasks like pre-training representation models on large-scale tabular data, as has been done in (Bahri et al., 2022; Yao et al., 2021; Yoon et al., 2020). (ii) SLAD can be extended to handle other data types like time series, graph data, or even images by defining appropriate sampling operation, the transformation function, and the labeling function. (iii) The intermediate results in SLAD can be further processed for anomaly interpretation (Xu et al., 2021). As empirical errors derived from the loss function indicate the deviation of the target data instance in different feature subspaces, and thus these fine-grained outputs can be further utilized to yield a tailored feature subspace as an interpretation showing the anonymous part of the data instance.