ODIM: Outlier Detection via Likelihood of Under-Fitted Generative Models

Dongha Kim* [1] **Jaesung Hwang*** [2] **Jongjin Lee** [3] **Kunwoong Kim** [4] **Yongdai Kim** [4]

Abstract

The unsupervised outlier detection (UOD) problem refers to a task to identify inliers given training data which contain outliers as well as inliers, without any labeled information about inliers and outliers. It has been widely recognized that using fully-trained likelihood-based deep generative models (DGMs) often results in poor performance in distinguishing inliers from outliers. In this study, we claim that *the likelihood itself could serve as powerful evidence for identifying inliers in UOD tasks, provided that DGMs are carefully under-fitted.* Our approach begins with a novel observation called the *inlier-memorization (IM) effect*–when training a deep generative model with data including outliers, the model initially memorizes inliers before outliers. Based on this finding, we develop a new method called the *outlier detection via the IM effect (ODIM)*. Remarkably, the ODIM requires only a few updates, making it computationally efficient–*at least tens of times faster* than other deep-learning-based algorithms. Also, the ODIM filters out outliers excellently, regardless of the data type, including tabular, image, and text data. To validate the superiority and efficiency of our method, we provide extensive empirical analyses on close to 60 datasets.

1. Introduction

Outlier detection Outlier (also anomaly) is an observation that differs significantly from other observations, and outlier detection (OD) is the task of identifying outliers in a given dataset. OD has wide applications such as fraud detection, fault detection, and defect detection in images. OD is also used as a pre-processing step in supervised learning to filter out anomalous training samples, which may degrade the performance of a predictive model.

*Equal contribution [1]Department of Statistics and Data Science Center, Sungshin Women's University, Seoul, Republic of Korea [2]SK Telecom, Seoul, Republic of Korea [3]Samsung Research, Seoul, Republic of Korea [4]Department of Statistics, Seoul National University, Seoul, Republic of Korea. Correspondence to: Yongdai Kim <ydkim0903@gmail.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

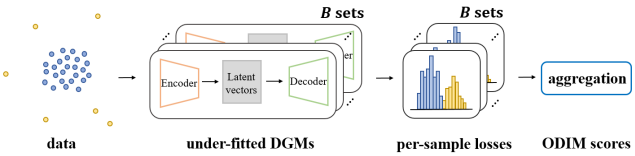

Figure 1. An illustration of the ODIM method.

OD problems can be categorized into three areas in general: 1) Supervised outlier detection (SOD) requires label information about whether each training sample is inlier (also normal) or outlier and solves the two-class classification task. 2) Semi-supervised outlier detection (SSOD) refers to methods that assume all training data being inliers and construct patterns or models based only on the inliers. 3) Unsupervised outlier detection (UOD) deals with the most realistic situations where training data include some outliers but no label information about anomalousness is available. Most anomaly detection tasks in practice are involved in UOD since the information of outliers in massive data is hardly known in advance.

Likelihood-based approaches in OD To detect outliers from data, a fundamental approach might involve using a deep generative model (DGM) and regarding each sample as either an inlier or not based on its likelihood value. However, it is widely recognized that *the likelihood value itself using fully trained DGMs is by no means a reasonable indicator to identify outliers*, particularly in out-of-distribution (OOD) tasks (or SSOD tasks) where none of the outliers are present in training data. The likelihood values of outliers with a fully trained DGM are often higher than those of inliers (Nalisnick et al., 2019b; Nalisnick et al.; Lan & Dinh, 2021).

Overview of our method In this study, we claim that *likelihood could serve as an effective score in UOD tasks, provided we employ carefully trained under-fitting DGMs.*

Our algorithm is motivated by the so called *memorization effect* that is observed in noisy label problems (Arpit et al., 2017; Jiang et al., 2018). The goal of noisy label problems is

to learn an accurate classifier when some of the class labels in the training data are contaminated.

When standard supervised learning algorithms are applied to such mislabeled data, an interesting phenomenon known as the memorization effect is observed. In this effect, correctly labeled data are learned earlier than mislabeled ones in the training phase of deep neural networks. The memorization effect makes it possible to detect mislabeled data by comparing per-sample losses in the early training phase.

Building upon the effect, our primary goal is to apply this concept to the field of UOD. We start with finding a new and interesting observation that *the memorization effect is also observed in learning DGMs*. That is, when we train a deep generative model with training data that include outliers, the inliers' loss values reduce prior to those of outliers at early updates. We call this observation the *inlier-memorization (IM) effect*. The IM effect occurs because, in the early training phase, decreasing the loss values of inliers rather than outliers is a more beneficial direction to reduce the overall loss, which will be discussed in Section 3.

Based on the IM effect, we propose a simple yet powerful UOD solver called the *outlier detection via the IM effect* (ODIM). We train a DGM with a log-likelihood-based approach such as the VAE (Kingma & Welling, 2013) or IWAE (Burda et al., 2016) for a few updates, and we regard data with large loss values compared to the per-sample loss distribution as outliers. Figure 1 provides a visual representation of our method.

A previous study has explored the use of the memorization effect for UOD problems (Wang et al., 2019). In their work, the authors found that the memorization effect can also be observed in the self-supervised learning framework when using artificially and carefully designed pseudo-labels. They successfully applied this approach to accurately detect outliers in image domains. However, it is challenging to extend their method to other data domains such as tabular data and sequential data due to the unavailability of a suitable pseudo-labeling strategy for these domains.

On the other hand, the ODIM is *domain-agnostic* and thus can be applied to various data domains, including tabular, image, and sequential data, as it does not rely on specific pseudo-labeling strategies. By analyzing nearly 60 datasets from various domains, we demonstrate that the ODIM consistently yields state-of-the-art or competitive results in identifying outliers across a wide range of data types.

Additionally, the ODIM offers significant *computational efficiency*. In fact, the ODIM requires only a few training updates, often less than a single epoch, in the training phase to detect outliers. Thus, the ODIM is at least tens of times faster than other recent deep-learning-based UOD solvers, such as Ruff et al. (2018b), requiring a larger number of training updates, for example, 200 epochs.

The remainder of this paper is organized as follows. Section 2 provides a brief review of related research on OD problems. Detailed descriptions of the ODIM algorithm with discussions of the IM effect are given in Section 3. Results of various experiments including performance tests and ablation studies follow in Section 4. Finally, further discussions and concluding remarks are respectively presented in Section 5 and 6. The key contributions of this work are:

- We find a new phenomenon called the IM effect that DGMs memorize inliers prior to outliers at early training updates.
- We develop a simple and powerful likelihood-based UOD solver called the ODIM based on the IM effect and other improving techniques.
- We empirically demonstrate the superiority and efficiency of our method by analyzing extensive benchmark datasets.
- We conduct additional experiments to cover a couple of extensions where our method can be applied.

2. Related works

We review algorithms for both SSOD and UOD since the former algorithms are often used in UOD tasks as well.

Semi-supervised outlier detection A popular technique for SSOD is the one class classification approach which transforms data into a feature space and distinguishes outliers from inliers by their radii from the center on the feature space. The OCSVM (Schölkopf et al., 2001) and SVDD (Tax & Duin, 2004) are two representative algorithms, which use kernel techniques to construct the feature space.

Succeeding their ideas, plenty of SSOD algorithms using deep neural networks have been developed. The DeepSVDD (Ruff et al., 2018a) extends the SVDD by utilizing a deep autoencoder (AE) for learning a feature map, and the DeepSAD (Ruff et al., 2020) modifies the DeepSVDD to incorporate labeled outliers to training data. Modifications of the DeepSVDD have been developed by Zong et al. (2018b); Mahmood et al. (2021); Xia et al. (2015). In addition to AE, deep generative models are also popularly used for SSOD (Ryu et al., 2018; Nalisnick et al., 2019a; Jiang et al., 2022).

There are methods for SSOD other than the one class classification approach. The SimCLR (Chen et al., 2020b) and BERT (Devlin et al., 2019) utilize self-supervised learning to obtain a desirable feature map, and various algorithms based on this idea have been developed (Golan & El-Yaniv, 2018; Bergman & Hoshen, 2020; Tack et al., 2020; Sehwag et al., 2021). When some labels (not related to inliers or outliers) are available, several studies have found that feature

maps for classification of those labels can improve outlier detection (Hendrycks & Gimpel, 2017; Liang et al., 2018; Gomes et al., 2022).

There have been attempts to use the likelihood to detect outliers. As mentioned in Section 1, the likelihood itself performs poorly in the SSOD, thus, certain transformations or indirect uses of the likelihood have been studied (Nalisnick et al., 2019b; Nalisnick et al.; Lan & Dinh, 2021).

Unsupervised outlier detection As for traditional approaches, the LOF (Breunig et al., 2000a) compares the density of a given datum compared to the densities of its neighborhoods, and the IF (Liu et al., 2008a) utilizes the fact that outliers can be separated out by random trees with relatively small sizes. The UOCL (Liu et al., 2014) solves UOD problems by employing pseudo soft labels and training them jointly with the one-class classification model.

There are various methods to solve UOD problems with deep learning models. The RDA (Zhou & Paffenroth, 2017) combines the robust PCA and AE to detect outliers. The DSEBM (Zhai et al., 2016) utilizes the energy-based model for density estimation and uses the energy score or reconstruction error to identify outliers. The RSRAE (Lai et al., 2020) devises a new hidden layer called RSR, inserting it between encoder and decoder of a deep AE to separate inliers and outliers effectively. The E^3-Outlier (Wang et al., 2019) trains a deep neural network by self-supervised learning and identifies outliers based on how fast the loss decreases as the training proceeds. Recently, diffusion models (Ho et al., 2020) have also been leveraged to detect outliers in both SSOD and UOD tasks (Livernoche et al., 2023).

3. Proposed method

3.1. Notations and definitions

For a given input vector $\mathbf{x} \in \mathbb{R}^D$, we denote its anomalousness by $y^o \in \{0, 1\}$, that is, $y^o = 0$ if $\mathbf{x}$ is an inlier and $y^o = 1$ otherwise. Note that only $\mathbf{x}$ is observable but y^o is not under the UOD regime. Let $\mathcal{U}^{tr} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be unlabeled training data. Our goal is to detect outlier samples, i.e. $\mathbf{x}$ with $y^o = 1$, from $\mathcal{U}^{tr}$ as accurately as possible.

Let $p(\mathbf{x}|\mathbf{z}; \theta)$ and $q(\mathbf{z}|\mathbf{x}; \phi)$ be given decoder and encoder parameterized by θ and ϕ, respectively, where $\mathbf{z} \in \mathbb{R}^d$ (generally assuming $d < D$) is a latent vector. We construct the distribution of the input random vector $\boldsymbol{X} \in \mathbb{R}^D$ as follows:

$$\boldsymbol{X} \sim p(\mathbf{x}|\boldsymbol{Z}; \theta),$$

where $\boldsymbol{Z} \sim \mathcal{N}(0_d, I_d)$ denotes a latent random vector.

For a given $p \in \mathbb{N}$, we denote the l_p-norm of a vector $\mathbf{a}$ by $\|\mathbf{a}\|_p$. For two real-valued functions defined on $\mathbb{R}_{>0}$, $f(t)$ and $g(t)$, $f(t)$ is said to be $\Theta(g(t))$ if there exist positive constants C_1, C_2, and T such that $C_1 \cdot g(t) \leq f(t) \leq C_2 \cdot g(t)$ holds for all $t \geq T$.

3.2. Motivation: inlier-memorization effect

Suppose that we are training a likelihood-based DGM with a certain learning framework where the training data contain both inliers and outliers. To illustrate the IM effect, we analyze `Cardio` dataset and train a DGM using the VAE method (Kingma & Welling, 2013).

To prepare the data, we normalize each variable of `Cardio` to a range between 0 and 1. The encoder and decoder architectures are 2-layered deep neural networks (DNNs) with $d = 5$ and 50 hidden nodes for each hidden layer. Our focus is to analyze the distribution of per-sample VAE loss as updates proceed. We train the decoder and encoder by minimizing the VAE loss function for up to 500 updates.

The panels in Figure 2, excluding the last one, display the empirical distribution of per-sample loss values for the training data at different updates. We can observe that the discrepancy in loss distributions between inliers and outliers becomes clearer as updates progress in the early training phase. However, as the DGM is sufficiently trained, the two distributions become overlapped, making it almost impossible to distinguish them based solely on their loss values. We call this phenomenon the *inlier-memorization (IM) effect*.

The IM effect is not surprising conceptually. When the per-sample loss function is continuous, reducing the loss in dense regions is beneficial for overall loss reduction (e.g. the negative log-likelihood). As inliers tend to be located in *dense* regions and outliers in *sparse* regions, reasonable learning algorithms prioritize dense regions in the early training phase, leading to the IM effect. It is important to note that the IM effect is observed only in the early training phase, as the learned model memorizes both inliers and outliers later.

3.3. Theoretical analysis

We provide a theoretical explanation of the occurence of the IM effect with a simple example where we train a linear factor model using the VAE. That is, $p(\mathbf{x}|\mathbf{z}; \theta)$ is the density function of $W\mathbf{z} + b + \epsilon$, where $W \in \mathbb{R}^{D \times d}, b \in \mathbb{R}^D$ are the loading matrix and bias vector and $\epsilon \sim N(0_D, \sigma^2 I_D)$ is a noise random vector. And we set $q(\mathbf{z}|\mathbf{x}; \phi)$ as the density function of $U\mathbf{x} + v + \tau$, where $U \in \mathbb{R}^{d \times D}$, $v \in \mathbb{R}^d$, and $\tau \sim \mathcal{N}(0_d, \eta^2 I_d)$. For simplicity, we fix σ and η and only train W, b, U, and v. That is, the learnable parameters θ and ϕ become (W, b) and (U, v), respectively. Note that the objective function of the VAE for a given input vector $\mathbf{x}$ is

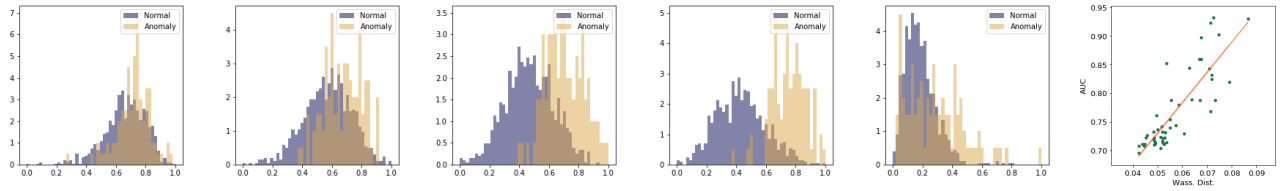

Figure 2. (**1st to 5th**) The distributions of the per-sample (normalized) VAE loss values of `Cardio` after 10, 20, 30, 40, and 500 training updates, respectively. For each panel, we depict the histograms of inliers and outliers separately. (**Last**) The positive relationship between the Wasserstein distance and identifying performance (AUC) on `Cardio`.

given as

$$L^{\text{VAE}}(\theta, \phi; \mathbf{x}) := \mathop{\mathbb{E}}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x};\phi)} \left[\log \left(\frac{p(\mathbf{x}|\mathbf{z};\theta)p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x};\phi)} \right) \right],$$

where $p(\mathbf{z})$ is the density function of $\mathcal{N}(0_d, 1_d)$. We assume that each element in W, b, U, and v is randomly initialized by the i.i.d. uniform distribution on $[-1, 1]$. Then we have the following result whose proof is given in Appendix A.

Proposition 3.1. [1] *For an input vector* $\mathbf{x}$*, the following holds:*

$$\mathbb{E}_{\theta,\phi} \left\| \frac{\partial}{\partial \theta} L^{VAE}(\theta, \phi; \mathbf{x}) \right\|_2^2 = \Theta \left(\|\mathbf{x}\|_1^4 \right).$$

Proposition 3.1 indicates that in the early phases of learning, the magnitude of the gradient of the VAE is proportional to the l_1-norm of the input vector on average. This implies that when the norms of inliers and outliers are similar, the initial update direction of θ is influenced by the inliers, as they are much more prevalent than outliers in the training dataset. Consequently, during the initial training phase, the generative model is trained towards memorizing the inliers before the outliers, resulting in the IM effect.

Of course, Proposition 3.1 may not hold *after* initial updates. However, we empirically found that this tendency persists for a while, and the loss distributions between inliers and outliers becomes more distinguishable (See Figure 2).

Remark 3.2. As mentioned in the overview of our method in Section 1, the behavior of parameter update in early training steps is frequently utilized, particularly in addressing noisy label problems (Arpit et al., 2017; Jiang et al., 2018), yet it lacks rigorous theoretical validation. To the best of our knowledge, our study offers the first theoretical insights about parameter updates during the initial learning step, particularly in relation to the norms of data. Exploring this aspect further could be a promising research avenue.

Remark 3.3. Besides the ELBO, such as VAE, there is another widely-used likelihood-based approach, normalizing flows (Kobyzev et al., 2020). Among the normalizing flows methods, we consider GLOW (Kingma & Dhariwal, 2018) to investigate whether the IM effect also occurs in other likelihood-based DGMs. We observe that the IM effect clearly appears during the GLOW training. This finding supports the claim that *the IM effect is a universal phenomenon in likelihood-based models.* Detailed descriptions of this experiment are provided in Appendix B.

[1] Since the generative model $p(\mathbf{x};\theta)$ is related only with the parameter θ, we only consider the gradient with respect to θ.

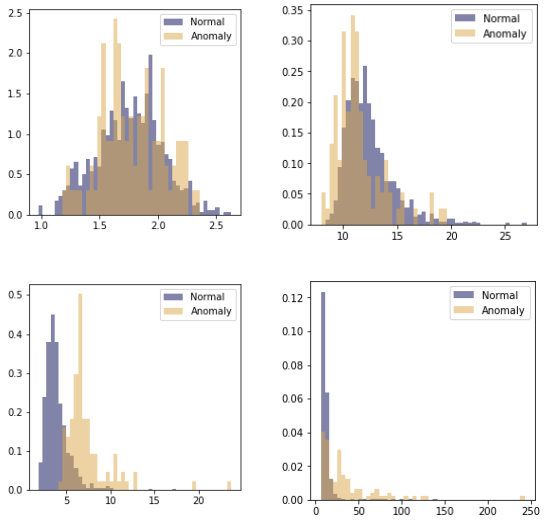

Figure 3. Distributions of per-sample (**Left**) input l_1-norm values and (**Right**) gradient l_2-norm values of VAE loss on `Cardio`. We consider two pre-processing schemes to normalize each feature: 1) (**Upper**) min-max scaling, and 2) (**Lower**) standardization.

3.4. Choice of pre-processing technique

Proposition 3.1 suggests that ODIM performs well when the norms of inliers and outliers are similar. However, it should be noted that the norm of a datum depends on the choice of a pre-processing. Therefore, careful selection of pre-processing is essential for the success of ODIM. A pre-processing which makes the norms of inliers and outleris be

similar would be a good choice.

As potential pre-processing candidates, we compare two widely used techniques, 1) min-max scaling and 2) standardization. The min-max scaling transforms each variable to have a distribution ranged between zero and one, and the standardization makes each variable to have zero mean and unit variance.

We explore how these techniques affect the performance of the ODIM. Intuitively, we can infer that, since the majority of data are inliers, the process of shifting the entire data towards the origin in the standardization technique would likely result in small norms for the inliers, potentially weakening the IM effect. This conjecture can be supported by the following simple proposition, whose proof is provided in Appendix A.

Proposition 3.4. *Let $\mathbf{X}^{in}$ and $\mathbf{X}^{out}$ be inlier and outlier random vectors with zero mean, i.e., $\mathbb{E}(X^{in}) = \mathbb{E}(X^{out}) = 0$. Suppose that their respective supports are $Supp(X^{in}) = A^{in}$ and $Supp(X^{out}) = A^{out}$, where A^{in} is a bounded convex set and A^{out} is a set wrapping A^{in}, i.e., $A^{in} \cap A^{out} = \emptyset$ and $conv(A^{out}) \supsetneq A^{in}$. Define $\mathbf{X}^{in}_{mm}$ and $\mathbf{X}^{out}_{mm}$ as pre-processed inlier and outlier random vectors using the min-max scaling. Similarly, we define $\mathbf{X}^{in}_{st}$ and $\mathbf{X}^{out}_{st}$ obtained by the standardization. Then, we have $\mathbb{E}\|\mathbf{X}^{in}_{mm}\|_1 = \mathbb{E}\|\mathbf{X}^{out}_{mm}\|_1$, while $\mathbb{E}\|\mathbf{X}^{in}_{st}\|_1 < \mathbb{E}\|\mathbf{X}^{out}_{st}\|_1$.*

We note that the condition that the supports of inliers and outliers do not overlap in the above proposition is not strictly necessary for the proof. But we maintain this, as the definition of an outlier is an observation *significantly different from inliers*.

Figure 3 visually validates our theoretical result. We again note the implication of Proposition 3.1 that if the norm of a particular sample is large, its gradient would have a more significant impact on the parameter update. Thus, when the standardization is applied, the input norms of inliers are larger than those of outliers, which leads to potentially biased parameter update towards memorizaing outliers, even if inliers are more prevalent than outliers. This phenomenon hinders a DGM from training inliers at early learning updates, and thus, diminishes the strength of the IM effect.

From these findings, we use the min-max scaling throughout our experiments, instead of the standardization. We empirically observe that the ODIM with the min-max scaled data has better results on most of the tabular datasets analyzed in the experiment. More results about comparing the min-max and standardization are provided in the ablation studies and Appendix C.

While the min-max scaling is better than the standardization, we do not claim that it is the optimal choice for the IM effect. We leave the optimal choice of pre-processing as a future research topic.

3.5. Algorithm description

The IM effect can be used for outlier detection by utilizing the per-sample loss value of a deep generative model during early training phases. In this section, we propose a new UOD solver called the *outlier detection via IM effect (ODIM)*. The ODIM consists of three steps: 1) applying the min-max scaling to the data, 2) training a DGM for a specified number of updates, and 3) identifying a sample as an outlier if its corresponding loss value is relatively large. To implement this approach, additional considerations are required, which are explained below.

Choice of the learning algorithm for a DGM Selecting a learning algorithm for a DGM carefully is crucial to make the IM effect appear more clearly. There exist numerous algorithms to train likelihood-based DGMs, which can be roughly divided into two approaches: 1) the ELBO approach, which calculates the lower bound of log-likelihood, (Kingma & Welling, 2013; Burda et al., 2016; Ho et al., 2020) and 2) normalizing flows approach, which calculates the exact log-likelihood (Dinh et al., 2015; 2017; Kingma & Dhariwal, 2018). In this study, we conclude to employ the importance weighted autoencoder (IWAE, Burda et al. (2016)), one of the ELBO approach. Detailed discussions can be found in Appendix B.

The objective function of the IWAE is given as:

$$L^{\text{IWAE}}(\theta, \phi; \mathbf{x}) := -\mathop{\mathbb{E}}_{\mathbf{z}_1,\ldots,\mathbf{z}_K \sim q(\mathbf{z}|\mathbf{x};\phi)} \left[\log \left(\frac{1}{K} \sum_{k=1}^{K} \frac{p(\mathbf{x}|\mathbf{z}_k;\theta)p(\mathbf{z}_k)}{q(\mathbf{z}_k|\mathbf{x};\phi)} \right) \right],$$

where $p(\mathbf{z})$ is the density of the standard multivariate Gaussian distribution and K is the number of samples. Note that the IWAE reduces to the VAE when $K = 1$. We train the encoder and decoder simultaneously by minimizing the negative empirical expectation, given as

$$\mathbb{E}_{\mathbf{x}\sim\mathcal{U}^{tr}} L^{\text{IWAE}}(\theta, \phi; \mathbf{x}) \tag{1}$$

with respect to θ and ϕ using an SGD-based optimizer.

Selection of the optimal number of updates We empirically found out that the IM effect often emerges very early in the training phase, often even within a single epoch, and its magnitude (i.e., the difference of the loss distributions between inliers and outliers) is highly sensitive to the number of model updates. And as the model memorizes outliers as well as inliers gradually, the model becomes no longer capable of distinguishing inliers from outliers (See the 5th panel in Figure 2). Thus, it would be a key for the success of the ODIM algorithm to choose the optimal number of updates data adaptively.

We devise a heuristic but powerful strategy to decide the number of updates where the IM effect is maximized. At each update of the model, we evaluate the *degree of bimodality* of the per-sample loss distribution and select the optimal number of updates where the degree of bimodality is maximized.

We observe that the two per-sample loss distributions, one for inliers and the other for outliers, seemingly follow Gaussian distributions (See the first four panels in Figure 2). Therefore, to quantify the degree of bimodality, we fit a two-component Gaussian mixture model (GMM-2), denoted as $\pi_1\mathcal{N}(\mu_1, \sigma_1^2) + \pi_2\mathcal{N}(\mu_2, \sigma_2^2)$, to the (normalized) per-sample loss values using the currently estimated generative model. We then measure the discrepancy between the two normal distributions in the fitted GMM-2 using the Wasserstein distance. This discrepancy measure serves as an indicator of the degree of bimodality.

The rightmost panel in Figure 2 illustrates the values of AUC on the training data of `Cardio` at the first $10 \times m$ updates for $m = 1, \ldots, 50$ and their corresponding Wasserstein distances. We can clearly see that the Wasserstein distance is a useful measure for selecting the optimal number of updates.

In practice, we calculate the Wasserstein distance at every N_{u} update and stop the update process if the largest Wasserstein distance has not been improved for N_{pat} consecutive times. And the optimal number of updates is determined as the one that maximizes the Wasserstein distance. We set N_{u} and N_{pat} to 10 in all numerical experiments, unless otherwise specified.

Incorporating multiple ODIM scores Our method includes several random components, such as parameter initialization and mini-batch arrangement, resulting in stochastic outcomes. To stabilize and enhance our method, we employ an ensemble strategy. Multiple models with different initial values are trained in parallel to obtain the multiple best models by use of the ODIM.

Let $(\theta^{*(b)}, \phi^{*(b)}), b = 1, \ldots, B$ be B pairs of estimated parameters, each of which is trained independently using the IWAE and our early stopping rule, where B is the number of ensembled models. Then, for a given datum $\mathbf{x}$, the formulation of the ensembled ODIM score becomes $\sum_{b=1}^{B} L^{\text{IWAE}}(\theta^{*(b)}, \phi^{*(b)}; \mathbf{x})/B$. We consider the input $\mathbf{x}$ as an inlier when its score is low and vise versa. In our experiments, the number of multiple models is fixed at 10, i.e., $B = 10$, unless otherwise specified. It is worth noting that the ODIM algorithm runs quickly, so implementing an ensemble is still computationally efficient. We provide the ODIM's pseudo algorithm in Algorithm 1.

Algorithm 1 ODIM

In practice, we set $(K, N_{\text{u}}, N_{\text{pat}}) = (50, 10, 10)$.

Input: Training dataset $\mathcal{U}^{tr} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$

Require: : Decoder and encoder: $p(\mathbf{x}|\mathbf{z}; \theta)$ and $q(\mathbf{z}|\mathbf{x}; \phi)$, GMM-2 model: $\pi_1\mathcal{N}(\mu_1, \sigma_1^2) + \pi_2\mathcal{N}(\mu_2, \sigma_2^2)$, Mini-batch size: n_{mb}, Optimizer: $\mathcal{O}$, Number of samples in IWAE: K, Update unit number: N_{u}, Maximum patience: N_{pat}

for b in $(1 : B)$ **do**
  Initialize $(\theta^{(b)}, \phi^{(b)})$ and set d_{WD}^{max} to 0.
  while $n_{\text{pat}} < N_{\text{pat}}$ **do**
    for k in $(1 : N_{\text{u}})$ **do**
      Drawn n_{mb} samples, $\{\mathbf{x}_i\}_{i=1}^{n_{\text{mb}}}$, from $\mathcal{U}^{tr}$.
      Apply the min-max scaling to $\{\mathbf{x}_i\}_{i=1}^{n_{\text{mb}}}$.
      Update $(\theta^{(b)}, \phi^{(b)})$ using the IWAE with $\{\mathbf{x}_i\}_{i=1}^{n_{\text{mb}}}$ and $\mathcal{O}$.
      $\{\tilde{l}_i\}_{i=1}^{n_{\text{mb}}} \leftarrow \text{MinMax}(\{L^{\text{IWAE}}(\mathbf{x}_i)\}_{i=1}^{n_{\text{mb}}})$.
      Fit the parameters in GMM-2 using $\{\tilde{l}_i\}_{i=1}^{n_{\text{mb}}}$ and calculate the WD distance d_{WD}.
      if $d_{\text{WD}} > D_{\text{WD}}^{\text{max}}$ **then**
        $d_{\text{WD}}^{\text{max}} \leftarrow d_{\text{WD}}$
        $(\theta^{*(b)}, \phi^{*(b)}) \leftarrow (\theta^{(b)}, \phi^{(b)})$
        $n_{\text{pat}} \leftarrow 0$
      else
        $n_{\text{pat}} \leftarrow n_{\text{pat}} + 1$
      end if
    end for
  end while
end for
Calculate ODIM scores:

$$l_i^* \leftarrow \frac{1}{B}\sum_{b=1}^{B} L^{\text{IWAE}}(\theta^{*(b)}, \phi^{*(b)}; \mathbf{x}_i), i = 1, \ldots, n$$

Output: ODIM scores $\{l_i^*\}_{i=1}^{n}$

4. Numerical experiments

We demonstrate the superiority of our proposed method through extensive experiments. We analyze a wide range of datasets across tabular, image, and text types. Across all data types, we show that ODIM outperforms other competitors, including state-of-the-art methods, in terms of outlier detection performance and computational cost. Additionally, we discuss an extension of ODIM in two situations: 1) when a small amount of anomalous information is available and 2) when we consider a differential privacy regime.

For all the experiments, we report the averaged results based on five implementations with randomly initialized parameters. We utilize the `Pytorch` framework to run our algorithm using a single NVIDIA TITAN XP GPU. The implementation code for our method is publicly available at

https://github.com/jshwang0311/ODIM.

Dataset description As aforementioned above, we analyze 57 benchmark datasets for OD covering tabular, images, and texts, all of which are sourced from ADBench[2] (Han et al., 2022a).

We consider 36 tabular datasets that are frequently analyzed in the OD literature. These datasets originate from diverse domains, such as healthcare, finance, and astronomy.

For images, we analyze six datasets from ADBench: MNIST, MNIST-C, FMNIST, CIFAR10, SVHN, and MVTec-AD. We utilize the feature vectors extracted by the ViT model (Dosovitskiy et al., 2021), which are available in ADBench.

We additionally include five more benchmark datasets commonly employed in natural language processing (NLP) domains: Amazon, 20news, Agnews, Imdb, and Yelp. For these datasets, we employ our method using the embedding features provided by either BERT (Devlin et al., 2019) or RoBERTa (Liu et al., 2019), also available in ADBench.

For each dataset, we perform the min-max scaling. We refer to Appendix C and Han et al. (2022b) for the detailed descriptions of all the datasets.

Baseline For baselines to be compared with the ODIM, we refer to Livernoche et al. (2023) and the baselines they considered. To be more detailed, we first consider all the UOD solvers, 16 in total, including deep-learning-based ones, described in ADBench: PCA (Shyu et al., 2003), OCSVM, (Schölkopf et al., 2001), LOF (Breunig et al., 2000b), CBLOF (He et al., 2003), COF (Tang et al., 2002), HBOS (Goldstein & Dengel, 2012), kNN (Ramaswamy et al., 2000), SOD (Kriegel et al., 2009), COPOD (Li et al., 2020), ECOD (Li et al., 2022), IF (Liu et al., 2008b), LODA (Pevný, 2016), FeatureBagging (Lazarevic & Kumar, 2005), MCD (Fauconnier & Haesbroeck, 2009), DeepSVDD (Ruff et al., 2018b), and DAGMM (Zong et al., 2018a).

Furthermore, we consider four recent UOD methods outside of ADBench: DROCC (Goyal et al., 2020), GOAD (Bergman & Hoshen, 2020), ICL (Shenkar & Wolf, 2022), and DTE (Livernoche et al., 2023).

We exclude likelihood-based methods specifically developed for SSOD, such as Nalisnick et al. (2019b), as we have observed their poor performance in UOD tasks (Please read Appendix D for empirical evidence.).

Architecture & learning schedule We use two hidden layered DNN architectures for building the encoder and decoder and set K, the number of samples drawn from the encoder used for constructing the IWAE objective function, to 50. We minimize the IWAE objective function in (1) with the Adam optimizer (Kingma & Ba, 2014) with a mini-batch size of 128 and a learning rate of 5e-4. To run the ODIM, we fix the two hyper-parameters, N_{u} and N_{pat}, to 10. For ensemble learning, we train 10 pairs of encoder and decoder, each of which is trained independently.

[2] https://github.com/Minqi824/ADBench

Table 1. Averaged AUC and PR scores over 46 tabular datasets.

Method	OCSVM	COPOD	ECOD	DeepSVDD	ICL	DDPM	DTE	ODIM
AUC	0.740	0.730	0.729	0.543	0.652	0.712	0.730	**0.757**
PR	0.360	0.339	0.349	0.182	0.201	0.332	0.321	**0.366**

Table 2. Averaged AUC and PR scores over 6 image datasets.

Method	OCSVM	COPOD	ECOD	DeepSVDD	ICL	DDPM	DTE	ODIM
AUC	0.744	0.508	0.511	0.580	0.655	0.738	0.757	**0.813**
PR	0.271	0.090	0.091	0.176	0.172	0.267	0.282	**0.429**

4.1. Performance for outlier identification

We begin by comparing ODIM with baseline methods to assess its performance in identifying outliers within a training dataset. To do this, we examine the area under receiver operating characteristic (AUC), a standard measure that most other studies have used. Additionally, to provide solid evidence of our method's superior performance, we evaluate the area under precision-recall (PR), which summarizes the precision-recall curve.

As we compare tens of baselines with ODIM, we provide selected results in our main manuscript, with detailed results of all methods for each dataset in Appendix C. We note that all the baseline results are referenced from Appendix in Livernoche et al. (2023).

Results for tabular data Table 1 provides a summary of the averaged AUC and PR scores for 46 tabular datasets. The ODIM achieves the highest averaged scores in terms of both AUC and PR, indicating its superior performance in outlier detection for tabular data. These superior results imply that the ODIM can be readily used as a reliable method for outlier detection in tabular data.

Results for image data Table 2 showcases the averaged AUC and PR scores for 6 image datasets. The results clearly show that ODIM provides outstanding results with large margins compared to other competitors in most cases.

Results for text data The performance results for 5 text datasets are also summarized in Table 3. Once again, the ODIM stands out as the best-performing method across text datasets on average. It is worth highlighting that the ODIM has demonstrated superior and consistent performance across different data types throughout the aforementioned empirical experiments, making it an effective method

Table 3. Averaged AUC and PR scores over 5 text datasets.

Method	OCSVM	COPOD	ECOD	DeepSVDD	ICL	DDPM	DTE	ODIM
AUC	0.566	0.554	0.537	0.504	0.546	0.548	0.598	**0.659**
PR	0.062	0.060	0.057	0.054	0.058	0.059	0.070	**0.097**

Table 4. Running time comparison for the ODIM and other competitors. All records are measured in seconds.

Method	OCSVM	LoF	IF	DeepSVDD	ODIM
Cover	2164.270	24.959	6.192	3463.821	7.735
Mammography	2.482	0.221	0.408	135.958	4.057
Pendigits	1.247	1.899	0.350	83.944	5.595
Satellite	1.152	1.735	0.369	78.876	4.949
Shuttle	63.146	4.426	0.987	594.589	6.306
FMNIST	52.114	15.446	4.298	744.652	12.293
WM-811K	11498.385	910.600	89.499	4561.028	9.498
Agnews	25.877	3.348	3.156	138.114	7.506

for outlier detection tasks. Therefore, the ODIM can be used as an off-the-shelf tool for outlier detection.

Implementation time comparison Recall our claim mentioned in Section 1 that our method is computationally efficient compared to other deep-learning-based baselines and even other conventional competitors. To validate this claim, we conduct a comparative analysis of the running times on multiple datasets, as summarized in Table 4.

As expected, the two deep learning methods are considerably slower compared to other methods. On the other hand, the ODIM demonstrates remarkable efficiency. This significant improvement in computational efficiency makes ODIM a highly practical and scalable method for OD tasks.

4.2. Ablation study

We conduct additional experiments to cover how the ODIM behaves with respect to the choice of the hyper-parameters, whose results are given in Figure 4. The followings are the summary of our ablation studies, whose detailed explanations are in Appendix C.

1. Increasing the value of K leads to better performance, and the improvement saturates when $K \geq 50$.
2. Large value of N_{pat} is beneficial, though the extent of improvement diminishes when $N_{\text{pat}} \geq 10$.
3. Using a larger value of B generally improves the outlier identification performance.
4. The ODIM is stable with respect to reasonable learning rates, highlighting the ease of applicability in practice.

Furthermore, we apply two pre-processing techniques, min-max scaling and standardization, and compare their corresponding results for the ODIMs on 30 tabular datasets. We observe that the standardization yields the averaged AUC score of 0.760, which is worse than that when the min-max scaling is used, 0.790. The result for each dataset can be found in Appendix C.

Table 5. Averaged results of training AUC (and PR) scores with various values of l. We consider l, $l = 0.0, 0.3, 0.5$.

l	0.0	0.3	0.5
AUC (PR)	0.885 (0.647)	0.947 (0.871)	0.958 (0.891)

Table 6. Averaged results of training AUC (and PR) scores when applying the DP-SGD algorithm. We iterate the DP-SGD until $\epsilon = 10$ while fixing $\delta = 10^{-5}$.

Method	DeepSVDD	ODIM
AUC (PR)	0.614 (0.152)	**0.710** (**0.234**)

5. Further discussions

ODIM with partially labeled outliers Our method can be extended to a scenario where partially labeled data are additionally given. That is, we assume that besides $\mathcal{U}^{tr}$, a few labeled outlier dataset $\mathcal{L}^{tr} = \{(\mathbf{x}_1^l, 1), \ldots, (\mathbf{x}_m^l, 1)\}$ is also available.

We simply adopt the idea of Daniel et al. (2019), which encourages the log-likelihood of known outliers to decrease with the variational *upper* bound, called χ upper bound (CUBO). And we modify the loss function of the ODIM by adding the expected CUBO on $\mathcal{L}^{tr}$ from the original IWAE.

Table 5 summarizes the averaged training AUC and PR results of the modified method across various tabular datasets for different proportions of labeled outliers. Please refer to Appendix D for detailed results of each dataset. It is clearly seen that using label information helps to enhance identifying performance by a large margin. Rigorous descriptions of this modification and further discussions are provided in Appendix D.

Differentially private ODIM As the ODIM typically requires far fewer updates, combining our method with differentially private algorithms is expected to yield a synergic benefits. We impose privacy protection to our method by simply applying DP-SGD (Abadi et al., 2016), instead of conventional SGD-based methods, when minimizing the loss function (1). To guarantee differential privacy (DP), DP-SGD deforms gradient of each sample by clipping and adding noise, and parameters are trained based on this modified quantity.

As a measure of DP, we adopt (ϵ, δ)-DP (Dwork, 2006), which is the de facto standard in this field. We iterate DP-SGD to train DGMs for obtaining the ODIM scores, until the privacy budget ϵ first exceeds a pre-specified value while fixing $\delta = 10^{-5}$. Detailed explanations for (ϵ, δ)-DP and DP-SGD, and the calculation of privacy budget when utilizing DP-SGD is provided in Appendix D.

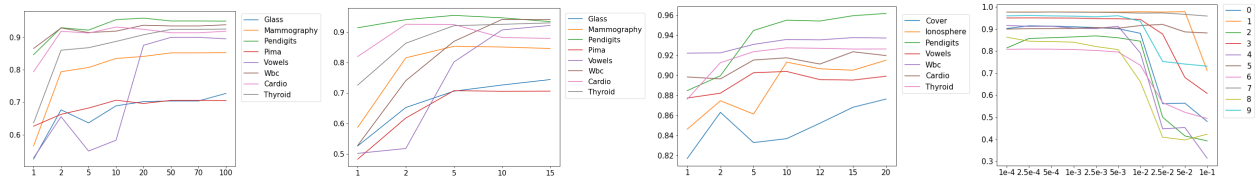

Figure 4. (**From left to right**) **1)** AUC results on tabular datasets with various values of K. **2)** AUC results on tabular datasets with various values of N_{pat}. **3)** AUC results on tabular datasets with various values of B. **4)** AUC results on FMNIST for each class with various learning rates. We vary the learning rate from 1e-4 to 1e-1.

Table 6 shows the averaged AUC and PR scores of our method and DeepSVDD across several tabular dataset, both trained until $\epsilon = 10$. Detailed results for each dataset can be found in Appendix D. We note that we exclude non-SGD-based methods such as OCSVM and IF since they DP-SGD is not applicable to them. Large margins in AUC and PR between our method and DeepSVDD indicates the strong potential of the ODIM when publishing OD algorithms with privacy guarantee.

6. Concluding remarks

This paper proposed a powerful yet efficient UOD method called the ODIM. The ODIM is inspired by a new observation called the IM effect, that deep generative models tend to memorize inliers first during early training. Combined with the technique to select the optimal number of training updates and the ensemble method, we showed that the ODIM provides consistently superior results in identifying outliers, regardless of data types, with significantly faster running times. Exploring the optimal pre-processing technique for the ODIM and developing new methods discussed in Section 5 would be promising future works.

Impact Statement

Our proposed method called ODIM enhances the accuracy and efficiency of anomaly detection across various domains, including healthcare, finance, and astronomy. ODIM's effectiveness and versatility can extend its applicability, benefiting a wide range of sectors and contributing to societal advancements in safety, security, and operational efficiency. Additionally, its computational efficiency also makes it accessible for real-time applications, promoting its use in dynamic environments such as industrial monitoring.

Acknowledgements

DK was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2022R1G1A1010894 and No. RS-2023-00218231). YK was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2020R1A2C3A01003550) and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) [NO. RS-2021-II211343, Artificial Intelligence Graduate School Program (Seoul National University)].

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., et al. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pp. 233–242. PMLR, 2017.

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

Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, may 2000a. ISSN 0163-5808. doi: 10.1145/335191.335388.

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

Burda, Y., Grosse, R. B., and Salakhutdinov, R. Importance weighted autoencoders. In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Chen, D., Orekondy, T., and Fritz, M. Gs-wgan: A gradient-sanitized approach for learning differentially private generators. *Advances in Neural Information Processing Systems*, 33:12673–12684, 2020a.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 13–18 Jul 2020b.

Daniel, T., Kurutach, T., and Tamar, A. Deep variational semi-supervised novelty detection. *arXiv preprint arXiv:1911.04971*, 2019.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423.

Dinh, L., Krueger, D., and Bengio, Y. NICE: non-linear independent components estimation. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Dong, J., Roth, A., and Su, W. J. Gaussian differential privacy. *arXiv preprint arXiv:1905.02383*, 2019.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Dwork, C. Differential privacy. In Bugliesi, M., Preneel, B., Sassone, V., and Wegener, I. (eds.), *Automata, Languages and Programming*, pp. 1–12, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-35908-1.

Fauconnier, C. and Haesbroeck, G. Outliers detection with the minimum covariance determinant estimator in practice. *Statistical Methodology*, 6(4):363–379, 2009.

Golan, I. and El-Yaniv, R. Deep anomaly detection using geometric transformations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 9781–9791, Red Hook, NY, USA, 2018. Curran Associates Inc.

Goldstein, M. and Dengel, A. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI-2012: poster and demo track*, 1:59–63, 2012.

Gomes, E. D. C., Alberge, F., Duhamel, P., and Piantanida, P. Igeood: An information geometry approach to out-of-distribution detection. In *International Conference on Learning Representations*, 2022.

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, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3711–3721. PMLR, 2020.

Han, S., Hu, X., Huang, H., Jiang, M., and Zhao, Y. Adbench: Anomaly detection benchmark. In *NeurIPS*, 2022a.

Han, S., Hu, X., Huang, H., Jiang, M., and Zhao, Y. Adbench: Anomaly detection benchmark. In *Neural Information Processing Systems (NeurIPS)*, 2022b.

He, Z., Xu, X., and Deng, S. Discovering cluster-based local outliers. *Pattern recognition letters*, 24(9-10):1641–1650, 2003.

Hendrycks, D. and Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *Proceedings of International Conference on Learning Representations*, 2017.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Jiang, D., Sun, S., and Yu, Y. Revisiting flow generative models for out-of-distribution detection. In *International Conference on Learning Representations*, 2022.

Jiang, L., Zhou, Z., Leung, T., Li, L.-J., and Fei-Fei, L. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, pp. 2304–2313. PMLR, 2018.

Kim, D., Hwang, J., and Kim, Y. On casting importance weighted autoencoder to an em algorithm to learn deep generative models. In *International Conference on Artificial Intelligence and Statistics*, pp. 2153–2163. PMLR, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 10236–10245, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kobyzev, I., Prince, S. J., and Brubaker, M. A. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979, 2020.

Kriegel, H.-P., Kröger, P., Schubert, E., and Zimek, A. Outlier detection in axis-parallel subspaces of high dimensional data. In *Advances in Knowledge Discovery and Data Mining: 13th Pacific-Asia Conference, PAKDD 2009 Bangkok, Thailand, April 27-30, 2009 Proceedings 13*, pp. 831–838. Springer, 2009.

Lai, C.-H., Zou, D., and Lerman, G. Robust subspace recovery layer for unsupervised anomaly detection. In *International Conference on Learning Representations*, 2020.

Lan, C. L. and Dinh, L. Perfect density models cannot guarantee anomaly detection. *Entropy*, 23(12):1690, 2021. doi: 10.3390/E23121690.

Lazarevic, A. and Kumar, V. Feature bagging for outlier detection. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pp. 157–166, 2005.

Li, Z., Zhao, Y., Botta, N., Ionescu, C., and Hu, X. Copod: copula-based outlier detection. In *2020 IEEE international conference on data mining (ICDM)*, pp. 1118–1123. IEEE, 2020.

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

Liang, S., Li, Y., and Srikant, R. Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations*, 2018.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, 2008a. doi: 10.1109/ICDM.2008.17.

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

Liu, W., Hua, G., and Smith, J. R. Unsupervised one-class learning for automatic outlier removal. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3826–3833, 2014. doi: 10.1109/CVPR.2014.483.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

Livernoche, V., Jain, V., Hezaveh, Y., and Ravanbakhsh, S. On diffusion modeling for anomaly detection. *CoRR*, abs/2305.18593, 2023. doi: 10.48550/ARXIV.2305.18593.

Mahmood, A., Oliva, J., and Styner, M. A. Multiscale score matching for out-of-distribution detection. In *International Conference on Learning Representations*, 2021.

Mironov, I. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*, pp. 263–275. IEEE, 2017.

Nalisnick, E., Matsukawa, A., Teh, Y. W., and Lakshminarayanan, B. Detecting out-of-distribution inputs to deep generative models using typicality, 2019a.

Nalisnick, E. T., Matsukawa, A., Teh, Y. W., and Lakshminarayanan, B. Detecting out-of-distribution inputs to deep generative models using a test for typicality. *CoRR*, abs/1906.02994.

Nalisnick, E. T., Matsukawa, A., Teh, Y. W., Görür, D., and Lakshminarayanan, B. Do deep generative models know what they don't know? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019b.

Pevnỳ, T. Loda: Lightweight on-line detector of anomalies. *Machine Learning*, 102:275–304, 2016.

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

Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S. A., Binder, A., Müller, E., and Kloft, M. Deep one-class classification. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4393–4402. PMLR, 10–15 Jul 2018a.

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

Ruff, L., Vandermeulen, R. A., Görnitz, N., Binder, A., Müller, E., Müller, K.-R., and Kloft, M. Deep semi-supervised anomaly detection. In *International Conference on Learning Representations*, 2020.

Ryu, S., Koo, S., Yu, H., and Lee, G. G. Out-of-domain detection based on generative adversarial network. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 714–718, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1077.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Schölkopf, B., Platt, J., Shawe-Taylor, J., Smola, A., and Williamson, R. Estimating support of a high-dimensional distribution. *Neural Computation*, 13:1443–1471, 07 2001. doi: 10.1162/089976601750264965.

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 *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Shyu, M.-L., Chen, S.-C., Sarinnapakorn, K., and Chang, L. A novel anomaly detection scheme based on principal component classifier. In *Proceedings of the IEEE foundations and new directions of data mining workshop*, pp. 172–179. IEEE Press, 2003.

Tack, J., Mo, S., Jeong, J., and Shin, J. Csi: Novelty detection via contrastive learning on distributionally shifted instances. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 11839–11852. Curran Associates, Inc., 2020.

Tang, J., Chen, Z., Fu, A. W.-C., and Cheung, D. W. Enhancing effectiveness of outlier detections for low density patterns. In *Advances in Knowledge Discovery and Data Mining: 6th Pacific-Asia Conference, PAKDD 2002 Taipei, Taiwan, May 6–8, 2002 Proceedings 6*, pp. 535–548. Springer, 2002.

Tax, D. M. and Duin, R. P. Support vector data description. *Machine learning*, 54(1):45–66, 2004.

Tomczak, J. and Welling, M. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pp. 1214–1223. PMLR, 2018.

van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. In Balcan, M. and Weinberger, K. Q. (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1747–1756. JMLR.org, 2016.

Wang, S., Zeng, Y., Liu, X., Zhu, E., Yin, J., Xu, C., and Kloft, M. Effective end-to-end unsupervised outlier detection via inlier priority of discriminative network. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Xia, Y., Cao, X., Wen, F., Hua, G., and Sun, J. Learning discriminative reconstructions for unsupervised outlier removal. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1511–1519, 2015. doi: 10.1109/ICCV.2015.177.

Zhai, S., Cheng, Y., Lu, W., and Zhang, Z. Deep structured energy based models for anomaly detection. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 1100–1109. JMLR.org, 2016.

Zhao, Z., Kunar, A., Birke, R., Van der Scheer, H., and Chen, L. Y. Ctab-gan+: Enhancing tabular data synthesis. *Frontiers in big Data*, 6, 2023.

Zhou, C. and Paffenroth, R. C. Anomaly detection with robust deep autoencoders. KDD '17, pp. 665–674, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450348874. doi: 10.1145/3097983.3098052.

Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D., and Chen, H. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018a.

Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D., and Chen, H. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*, 2018b.

A. Proof of Propositions

A.1. Proof of Proposition 1

Note that the objective function of the VAE is given as:

$$L^{\text{VAE}}(\theta, \phi; \mathbf{x}) := \int_{\mathbf{z}} \log \left(\frac{p(\mathbf{x}|\mathbf{z}; \theta) p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x}; \phi)} \right) \cdot q(\mathbf{z}|\mathbf{x}; \phi) d\mathbf{z}.$$

Thus, we have the equations:

$$\frac{\partial}{\partial w_{ij}} L^{\text{VAE}}(\theta, \phi; \mathbf{x}) = \int_{\mathbf{z}} \frac{\partial}{\partial w_{ij}} \log \left(p(\mathbf{x}|\mathbf{z}; \theta) \right) \cdot q(\mathbf{z}|\mathbf{x}; \phi) d\mathbf{z},$$

and

$$\frac{\partial}{\partial b_i} L^{\text{VAE}}(\theta, \phi; \mathbf{x}) = \int_{\mathbf{z}} \frac{\partial}{\partial b_i} \log \left(p(\mathbf{x}|\mathbf{z}; \theta) \right) \cdot q(\mathbf{z}|\mathbf{x}; \phi) d\mathbf{z},$$

where w_{ij} and b_i for $i \in [D]$ and $j \in [d]$ are the (i, j) element of W and the i-th element of b, respectively. Here, we define $[L] := \{1, \ldots, L\}$ for $L \in \mathbb{N}$. Note that

$$\mathbb{E}_{\theta,\phi} \left\| \frac{\partial}{\partial \theta} L^{\text{VAE}}(\theta, \phi; \mathbf{x}) \right\|_2^2 = \sum_i \sum_j \mathbb{E}_{\theta,\phi} \left[\frac{\partial}{\partial w_{ij}} L^{\text{VAE}}(\theta, \phi; \mathbf{x}) \right]^2 + \sum_i \mathbb{E}_{\theta,\phi} \left[\frac{\partial}{\partial b_i} L^{\text{VAE}}(\theta, \phi; \mathbf{x}) \right]^2.$$

We are going to characterize the two terms, $\mathbb{E}_{\theta,\phi} \left[\frac{\partial}{\partial w_{ij}} L^{\text{VAE}}(\theta, \phi; \mathbf{x}) \right]^2$ and $\mathbb{E}_{\theta,\phi} \left[\frac{\partial}{\partial b_i} L^{\text{VAE}}(\theta, \phi; \mathbf{x}) \right]^2$, and combine them to make the final conclusion.

w.r.t. w_{ij}

Since $X|(Z = \mathbf{z}) \sim \mathcal{N}\left(W\mathbf{z} + b, \sigma^2\right)$ holds, we have

$$\log p(\mathbf{x}|\mathbf{z}; \theta) = -\frac{1}{2\sigma^2} \sum_{i=1}^{D} \left(x_i - \mathbf{w}_i' \mathbf{z} - b_i \right)^2 + const = -\frac{1}{2\sigma^2} \sum_{i=1}^{D} \left(x_i - \sum_{j=1}^{d} w_{ij} z_j - b_i \right)^2 + const,$$

where $\mathbf{w}_i$ is the i-th row of W and $const$ is a constant not depending on θ. Therefore, we can obtain the following result of the log-likelihood with respect to w_{ij}:

$$\frac{\partial}{\partial w_{ij}} \log \left(p(\mathbf{x}|\mathbf{z}; \theta) \right) = \frac{1}{\sigma^2} \left(x_i - \mathbf{w}_i' \mathbf{z} - b_i \right) \cdot z_j = \frac{1}{\sigma^2} \left[x_i z_j - w_{ij} z_j^2 - \sum_{j' \neq j} w_{ij'} z_j z_{j'} - b_i z_j \right].$$

Hence, the first derivative of the VAE objective function with respect to w_{ij} becomes:

$$\frac{\partial}{\partial w_{ij}} L^{\text{VAE}}(\theta, \phi; \mathbf{x}) = \int_{\mathbf{z}} \frac{1}{\sigma^2} \left[(x_i - b_i) z_j - w_{ij} z_j^2 - \sum_{j' \neq j} w_{ij'} z_j z_{j'} - b_i z_j \right] \cdot q(\mathbf{z}|\mathbf{x}; \phi) d\mathbf{z}$$

$$= \frac{1}{\sigma^2} \left[(x_i - b_i)(\mathbf{u}_j' \mathbf{x} + v_j) - w_{ij} \left(\left(\mathbf{u}_j' \mathbf{x} + v_j \right)^2 + \eta^2 \right) - \left(\mathbf{u}_j' \mathbf{x} + v_j \right) \sum_{j' \neq j} w_{ij'} \left(\mathbf{u}_{j'}' \mathbf{x} + v_{j'} \right) \right]$$

$$= \frac{1}{\sigma^2} \left[(x_i - b_i)(\mathbf{u}_j' \mathbf{x} + v_j) - \left(\mathbf{u}_j' \mathbf{x} + v_j \right) \sum_{j'} w_{ij'} \left(\mathbf{u}_{j'}' \mathbf{x} + v_{j'} \right) - w_{ij} \eta^2 \right],$$

where $\mathbf{u}_j$ is the j-th row of U. By squaring the above term, we have

$$\begin{aligned}\left[\frac{\partial}{\partial w_{ij}} L^{\text{VAE}}(\theta,\phi;\mathbf{x})\right]^2 = \quad & \frac{1}{\sigma^4}\Bigg[(x_i-b_i)^2\left(\mathbf{u}_j'\mathbf{x}+v_j\right)^2 + \left(\mathbf{u}_j'\mathbf{x}+v_j\right)^2\sum_{j'} w_{ij'}^2\left(\mathbf{u}_{j'}'\mathbf{x}+v_{j'}\right)^2 + w_{ij}^4\eta^4 \\ & +2\left(\mathbf{u}_j'\mathbf{x}+v_j\right)^2\sum_{j''>j'} w_{ij'}w_{ij''}(\mathbf{u}_{j'}'\mathbf{x}+v_{j'})(\mathbf{u}_{j''}'\mathbf{x}+v_{j''}) \\ & +2w_{ij}\eta^2\left(\mathbf{u}_j'\mathbf{x}+v_j\right)\sum_{j'} w_{ij'}\left(\mathbf{u}_{j'}'\mathbf{x}+v_{j'}\right) - 2w_{ij}\eta^2(x_i-b_i)\left(\mathbf{u}_j'\mathbf{x}+v_j\right) \\ & -2(x_i-b_i)\left(\mathbf{u}_j'\mathbf{x}+v_j\right)^2\sum_{j'} w_{ij'}\left(\mathbf{u}_{j'}'\mathbf{x}+v_{j'}\right)\Bigg].\end{aligned}$$

Now, we will calculate the expected value of the above equation with respect to θ and ϕ. To do this, we will take the expectation for each term in the RHS of the above equation. Note that, for a random variable $X \sim Unif[-1,1]$, $\mathbb{E}[X]=0$, $\mathbb{E}\left[X^2\right]=1/3$, and $\mathbb{E}\left[X^4\right]=1/5$. Thus, we have

$$\begin{aligned}&\mathbb{E}_{\theta,\phi}\left[(x_i-b_i)^2\left(\mathbf{u}_j'\mathbf{x}+v_j\right)^2\right] = \mathbb{E}_{\theta,\phi}\left[(x_i^2-2x_ib_i+b_i^2)\left((\mathbf{u}_j'\mathbf{x})^2+v_j^2+2v_j\mathbf{u}_j'\mathbf{x}\right)\right] \\ &= \mathbb{E}_{\theta,\phi}\left[(x_i^2+b_i^2)\left(\sum_{i'} u_{ji'}^2x_{i'}^2+v_j^2\right)\right] = \left(x_i^2+\frac{1}{3}\right)\cdot\left(\frac{1}{3}\|\mathbf{x}\|_2^2+\frac{1}{3}\right),\end{aligned}$$

$$\begin{aligned}&\mathbb{E}_{\theta,\phi}\left[\left(\mathbf{u}_j'\mathbf{x}+v_j\right)^2\sum_{j'} w_{ij'}^2\left(\mathbf{u}_{j'}'\mathbf{x}+v_{j'}\right)^2\right] = \mathbb{E}_{\theta,\phi}\left[\left(\sum_{i'} u_{ji'}x_{i'}+v_j\right)^2\sum_{j'} w_{ij'}^2\left(\sum_{i'} u_{j'i'}x_{i'}+v_{j'}\right)^2\right] \\ &= \frac{1}{3}\mathbb{E}_{\theta,\phi}\left[\left(\sum_{i'} u_{ji'}x_{i'}+v_j\right)^4+\left(\sum_{i'} u_{ji'}x_{i'}+v_j\right)^2\sum_{j'\neq j}\left(\sum_{i'} u_{j'i'}x_{i'}+v_{j'}\right)^2\right] \\ &= \frac{1}{3}\mathbb{E}_{\theta,\phi}\left[\left(\sum_{i'} u_{ji'}x_{i'}\right)^4+6\left(\sum_{i'} u_{ji'}x_{i'}\right)^2v_j^2+v_j^4+\left(\sum_{i'} u_{ji'}x_{i'}+v_j\right)^2\sum_{j'\neq j}\left(\sum_{i'} u_{j'i'}x_{i'}+v_{j'}\right)^2\right] \\ &= \frac{1}{3}\mathbb{E}_{\theta,\phi}\left[\sum_{i'} u_{ji'}^4x_{i'}^4+6\sum_{i''>i'} u_{ji'}^2u_{ji''}^2x_{i'}^2x_{i''}^2+6\left(\sum_{i'} u_{ji'}^2x_{i'}^2\right)v_j^2+v_j^4+\left(\sum_{i'} u_{ji'}^2x_{i'}^2+v_j^2\right)\sum_{j'\neq j}\left(\sum_{i'} u_{j'i'}^2x_{i'}^2+v_{j'}^2\right)\right] \\ &= \frac{1}{3}\left[\frac{1}{5}\sum_{i'} x_i^4+\frac{2}{3}\sum_{i''>i'} x_{i'}^2x_{i''}^2+\frac{2}{3}\sum_{i'} x_{i'}^2+\frac{1}{5}+\left(\frac{1}{3}\sum_{i'} x_{i'}^2\right)\sum_{j'\neq j}\left(\frac{1}{3}\sum_{i'} x_{i'}^2\right)\right] \\ &= \frac{1}{3}\left[\frac{1}{3}\|\mathbf{x}\|_2^4-\frac{2}{15}\|\mathbf{x}\|_4^4+\frac{2}{3}\|\mathbf{x}\|_2^2+\frac{1}{5}+(d-1)\left(\frac{1}{3}\|\mathbf{x}\|_2^2+\frac{1}{3}\right)\right],\end{aligned}$$

$$\mathbb{E}_{\theta,\phi} w_{ij}^2\eta^4 = \eta^4\mathbb{E}_\theta w_{ij}^2 = \frac{1}{3}\eta^4,$$

$$\mathbb{E}_{\theta,\phi}\left[2\left(\mathbf{u}_j'\mathbf{x}+v_j\right)^2\sum_{j''>j'} w_{ij'}w_{ij''}(\mathbf{u}_{j'}'\mathbf{x}+v_{j'})(\mathbf{u}_{j''}'\mathbf{x}+v_{j''})\right] = 0,$$

$$\begin{aligned}&\mathbb{E}_{\theta,\phi}\left[2w_{ij}\eta^2\sum_{j'} w_{ij'}\left(\mathbf{u}_j'\mathbf{x}+v_j\right)\left(\mathbf{u}_{j'}'\mathbf{x}+v_{j'}\right)\right] = 2\mathbb{E}_{\theta,\phi}\left[w_{ij}^2\eta^2\left(\mathbf{u}_j'\mathbf{x}+v_j\right)^2\right] = \frac{2}{3}\eta^2\mathbb{E}_\phi\left[\left(\sum_{i'} u_{ji'}x_{i'}+v_j\right)^2\right] \\ = \quad & \frac{2}{3}\eta^2\mathbb{E}_\phi\left[\sum_{i'} u_{ji'}^2x_{i'}^2+v_j^2\right] = \frac{2}{9}\eta^2\left(\|\mathbf{x}\|_2^2+1\right),\end{aligned}$$

$$\mathbb{E}_{\theta,\phi}\left[-2w_{ij}\eta^2(x_i-b_i)\left(\mathbf{u}_j'\mathbf{x}+v_j\right)\right]=0,$$

and

$$\mathbb{E}_{\theta,\phi}\left[-2(x_i-b_i)\left(\mathbf{u}_j'\mathbf{x}+v_j\right)^2\sum_{j'}w_{ij'}\left(\mathbf{u}_{j'}'\mathbf{x}+v_{j'}\right)\right]=0.$$

By integrating all the above expected values and using the property $\|\mathbf{x}\|_2 \geq \|\mathbf{x}\|_4$, we arrive at the following result:

$$\mathbb{E}_{\theta,\phi}\left[\frac{\partial}{\partial w_{ij}}L^{\text{VAE}}(\theta,\phi;\mathbf{x})\right]^2=\Theta\left(\|\mathbf{x}\|_2^4\right).$$

w.r.t. b_i

We have

$$\frac{\partial}{\partial b_i}\log p(\mathbf{x}|\mathbf{z};\theta)=\frac{1}{\sigma^2}\left[x_i-\sum_{j=1}^{d}w_{ij}z_j-b_i\right],$$

thus,

$$\frac{\partial}{\partial b_i}L^{\text{VAE}}(\theta,\phi;\mathbf{x})=\int_{\mathbf{z}}\frac{1}{\sigma^2}\left[x_i-\sum_{j=1}^{d}w_{ij}z_j-b_i\right]\cdot q(\mathbf{z}|\mathbf{x};\phi)d\mathbf{z}=\frac{1}{\sigma^2}\left[(x_i-b_i)-\sum_j w_{ij}(\mathbf{u}_j'\mathbf{x}+v_j)\right].$$

By squaring the above term,

$$\begin{aligned}&\left[\frac{\partial}{\partial b_i}L^{\text{VAE}}(\theta,\phi;\mathbf{x})\right]^2\\&=\frac{1}{\sigma^4}\left[(x_i-b_i)^2+\sum_j w_{ij}^2(\mathbf{u}_j'\mathbf{x}+v_j)^2+2\sum_{j'>j}w_{ij}w_{ij'}(\mathbf{u}_j'\mathbf{x}+v_j)(\mathbf{u}_{j'}'\mathbf{x}+v_{j'})-2(x_i-b_i)\sum_j w_{ij}(\mathbf{u}_j'\mathbf{x}+v_j)\right].\end{aligned}$$

Here, we calculate the expected value of each term in the above RHS. We have

$$\mathbb{E}_{\theta,\phi}\left[(x_i-b_i)^2\right]=x_i^2+\frac{1}{3},$$

$$\mathbb{E}_{\theta,\phi}\left[\sum_j w_{ij}^2(\mathbf{u}_j'\mathbf{x}+v_j)^2\right]=\frac{1}{3}\sum_j\mathbb{E}_{\theta,\phi}\left[\sum_{i'}u_{ji'}^2x_{i'}^2+v_j^2\right]=\frac{d}{9}\|\mathbf{x}\|_2^2+\frac{d}{9},$$

$$\mathbb{E}_{\theta,\phi}\left[2\sum_{j'>j}w_{ij}w_{ij'}(\mathbf{u}_j'\mathbf{x}+v_j)(\mathbf{u}_{j'}'\mathbf{x}+v_{j'})\right]=0,$$

and

$$\mathbb{E}_{\theta,\phi}\left[2(x_i-b_i)\sum_j w_{ij}(\mathbf{u}_j'\mathbf{x}+v_j)\right]=0.$$

Combining the above expectations, we have

$$\mathbb{E}_{\theta,\phi}\left[\frac{\partial}{\partial b_i}L^{\text{VAE}}(\theta,\phi;\mathbf{x})\right]^2=\Theta\left(\|\mathbf{x}\|_2^2\right).$$

Final conclusion Combining the above results, we have

$$\mathbb{E}_{\theta,\phi}\left\|\frac{\partial}{\partial\theta}L^{\text{VAE}}(\theta,\phi;\mathbf{x})\right\|_2^2=\sum_i\sum_j\mathbb{E}_{\theta,\phi}\left[\frac{\partial}{\partial w_{ij}}L^{\text{VAE}}(\theta,\phi;\mathbf{x})\right]^2+\sum_i\mathbb{E}_{\theta,\phi}\left[\frac{\partial}{\partial b_i}L^{\text{VAE}}(\theta,\phi;\mathbf{x})\right]^2$$
$$=\sum_i\sum_j\Theta\left(\|\mathbf{x}\|_2^4\right)+\sum_i\Theta\left(\|\mathbf{x}\|_2^2\right)=\Theta\left(\|\mathbf{x}\|_2^4\right).$$

Thus, the proof is completed by using the inequality $D^{-1/2}\|\mathbf{x}\|_1\leq\|\mathbf{x}\|_2\leq\|\mathbf{x}\|_1$ □

A.2. Proof of Proposition 2

Due to the second condition of the Proposition 2, there exist four real numbers $-\infty<a_j<b_j<c_j<d_j<\infty$ and a small positive number $\epsilon>0$ such that $X_j^{in}\in[b_j,c_j]$ and $X_j^{out}\in[a_j,b_j-\epsilon]\cup[c_j+\epsilon,d_j]$ for all $j\in[D]$. Suppose that $\inf X_j^{in}=a_j$ and $\sup X_j^{in}=d_j$ for $j\in[D]$. And we define $X_{j,+}^{in}=\max\{0,X_j^{in}\}$ and $X_{j,-}^{in}=\max\{0,-X_j^{in}\}$ so that $X_j^{in}=X_{j,+}^{in}-X_{j,-}^{in}$. Similarly we define $X_{j,+}^{out}$ and $X_{j,-}^{out}$.

About min-max Since $\mathbb{E}X_j^{in}=\mathbb{E}X_j^{out}=0$ for $j\in[D]$, it is trivial to show that $\mathbb{E}|X_{mm,j}^{in}|=\mathbb{E}|X_{mm,j}^{out}|=-a_j/(d_j-a_j)$.

About standardization It is sufficient to show that $\mathbb{E}|X_j^{in}|<\mathbb{E}|X_j^{out}|$ since $\mathbb{E}X_j^{in}=\mathbb{E}X_j^{out}=0$. Note that $0\leq X_{j,+}^{in}\geq c_j<c_j+\epsilon\geq X_{j,+}^{in}<d_j$, so we can have $\mathbb{E}X_{j,+}^{in}<\mathbb{E}X_{j,+}^{out}$. In a similar manner, we can also derive $\mathbb{E}X_{j,-}^{in}<\mathbb{E}X_{j,-}^{out}$. By considering that $|X_j^{in}|=X_{j,-}^{in}+X_{j,+}^{in}$ and $|X_j^{out}|=X_{j,-}^{out}+X_{j,+}^{out}$, the proof is completed. □

B. Why do we choose the IWAE method as a learning framework?

In training DGMs with likelihood regimes, two primary approaches are commonly used: 1) employing the lower bound of the log-likelihood, often called ELBO, (Kingma & Welling, 2013; Burda et al., 2016; Tomczak & Welling, 2018; Kim et al., 2020), and 2) normalizing flows which calculate the exact log-likelihood (Dinh et al., 2015; 2017; Kingma & Dhariwal, 2018).

Two things need to be considered when selecting a DGM framework for exploiting the IM effect: 1) the clarity of the IM effect during learning and 2) computational efficiency. Among the normalizing flows methods, we considered the GLOW (Kingma & Dhariwal, 2018). We trained GLOW on the `FMNIST` dataset and monitored the per-sample loss values, i.e., negative log-likelihood, of inliers and outliers during the early learning phases. The results, illustrated in Figure B.1, shows that the IM effect clearly appears during GLOW training.

Although the IM effect occurs when using GLOW, we ultimately chose the IWAE method, one of the ELBO methods, because the performance of ODIM using GLOW was insufficient compared to IWAE. Additionally, ELBO-based methods are relatively flexible in choosing the encoder and decoder, while normalizing flows are somewhat limited in this regard. And the advantage of IWAE in comparison with VAE is described in Section 4.2 of the main manuscript.

In fact, there is another line of work, called auto-regressive models (van den Oord et al., 2016; Salimans et al., 2017), that utilizes an exact log-likelihood. We note that we excluded auto-regressive models from the exact likelihood approach since they are generally very computationally burdensome.

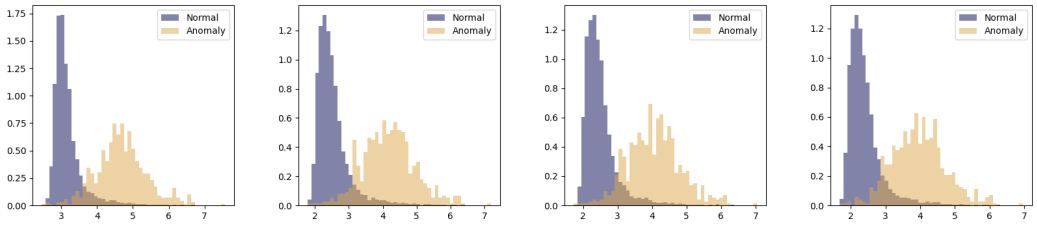

Figure B.1. (**Left to Right**) The distributions of the per-sample (normalized) negative log-likelihood values of GLOW on `FMNIST` after 10, 20, 30, and 40 updates, respectively. For each panel, we depict the histograms of inliers and outliers separately.

C. Numerical experiments

C.1. Data description

We examine a total of 46 tabular datasets, 6 image datasets, and 5 sequential datasets. All of these datasets are taken from a source called ADBench (Han et al., 2022b). The basic information of all datasets we analyze is summarized in Table C.1.

Table C.1. Description of ADBench datasets

Number	Data	# Samples	# Features	# Anomaly	% Anomaly
1	ALOI	49534	27	1508	3.04
2	Annthyroid	7200	6	534	7.42
3	Backdoor	95329	196	2329	2.44
4	Breastw	683	9	239	34.99
5	Campaign	41188	62	4640	11.27
6	Cardio	1831	21	176	9.61
7	Cardiotocography	2114	21	466	22.04
8	Celeba	202599	39	4547	2.24
9	Census	299285	500	18568	6.20
10	Cover	286048	10	2747	0.96
11	Donors	619326	10	36710	5.93
12	Fault	1941	27	673	34.67
13	Fraud	284807	29	492	0.17
14	Glass	214	7	9	4.21
15	Hepatitis	80	19	13	16.25
16	Http	567498	3	2211	0.39
17	InternetAds	1966	1555	368	18.72
18	Ionosphere	351	32	126	35.90
19	Landsat	6435	36	1333	20.71
20	Letter	1600	32	100	6.25
21	Lymphography	148	18	6	4.05
22	Magic.gamma	19020	10	6688	35.16
23	Mammography	11183	6	260	2.32
24	MNIST	7603	100	700	9.21
25	Musk	3062	166	97	3.17
26	Optdigits	5216	64	150	2.88
27	PageBlocks	5393	10	510	9.46
28	Pendigits	6870	16	156	2.27
29	Pima	768	8	268	34.90
30	Satellite	6435	36	2036	31.64
31	Satimage-2	5803	36	71	1.22
32	Shuttle	49097	9	3511	7.15
33	Skin	245057	3	50859	20.75
34	Smtp	95156	3	30	0.03
35	SpamBase	4207	57	1679	39.91
36	Speech	3686	400	61	1.65
37	Stamps	340	9	31	9.12
38	Thyroid	3772	6	93	2.47
39	Vertebral	240	6	30	12.50
40	Vowels	1456	12	50	3.43
41	Waveform	3443	21	100	2.90
42	WBC	223	9	10	4.48
43	WDBC	367	30	10	2.72
44	Wilt	4819	5	257	5.33
45	Wine	129	13	10	7.75
46	WPBC	198	33	47	23.74
47	Yeast	1484	8	507	34.16
48	CIFAR10	5263	512	263	5.00
49	FMNIST	6315	512	315	5.00
50	MNIST-C	10000	512	500	5.00
51	MVTec-AD	5354	512	1258	23.5
52	SVHN	5208	512	260	5.00
53	Agnews	10000	768	500	5.00
54	Amazon	10000	768	500	5.00
55	Imdb	10000	768	500	5.00
56	Yelp	10000	768	500	5.00
57	20news	10000	768	500	5.00

C.2. Detailed AUC and PR results over tabular datasets (training data)

Tables C.2.1-C.2.2 and C.3.1-C.3.2 provide a comparison of inlier identification performance for each method and dataset in terms of AUC and PR.

Table C.2.1. Training AUC value comparisons on tabular datasets

Data	CBLOF	FeatureBagging	HBOS	IForest	kNN	LODA	LOF	MCD	PCA	DAGMM	DROCC	GOAD	PlanarFlow
Aloi	0.556	0.792	0.531	0.542	0.613	0.495	0.767	0.520	0.549	0.517	0.500	0.497	0.520
Annthyroid	0.676	0.788	0.608	0.816	0.761	0.453	0.710	0.918	0.676	0.548	0.631	0.453	0.966
Backdoor	0.897	0.790	0.740	0.725	0.826	0.515	0.764	0.848	0.888	0.752	0.500	0.587	0.787
Breastw	0.961	0.408	0.984	0.983	0.980	0.970	0.446	0.985	0.946	0.811	0.847	0.845	0.965
Campaign	0.738	0.594	0.768	0.704	0.750	0.493	0.614	0.775	0.734	0.580	0.500	0.443	0.566
Cardio	0.832	0.579	0.839	0.922	0.830	0.856	0.551	0.815	0.949	0.625	0.655	0.908	0.796
Cardiotocography	0.561	0.538	0.595	0.681	0.503	0.708	0.527	0.500	0.747	0.546	0.449	0.624	0.643
Celeba	0.753	0.514	0.754	0.707	0.736	0.600	0.432	0.803	0.792	0.627	0.726	0.432	0.703
Census	0.664	0.538	0.611	0.607	0.671	0.454	0.562	0.731	0.662	0.491	0.443	0.488	0.604
Cover	0.922	0.571	0.707	0.873	0.866	0.922	0.568	0.696	0.934	0.742	0.747	0.124	0.417
Donors	0.808	0.691	0.743	0.771	0.829	0.566	0.629	0.765	0.825	0.558	0.747	0.225	0.899
Fault	0.665	0.591	0.506	0.544	0.715	0.478	0.579	0.505	0.480	0.495	0.668	0.546	0.469
Fraud	0.954	0.616	0.945	0.950	0.955	0.856	0.548	0.911	0.952	0.857	0.500	0.724	0.895
Glass	0.855	0.659	0.820	0.790	0.870	0.624	0.618	0.795	0.715	0.630	0.743	0.545	0.766
Hepatitis	0.635	0.469	0.768	0.683	0.669	0.557	0.468	0.721	0.748	0.600	0.582	0.637	0.654
Http	0.996	0.288	0.991	0.999	0.051	0.060	0.338	0.999	0.997	0.838	0.500	0.996	0.994
Internetads	0.616	0.494	0.696	0.686	0.616	0.541	0.587	0.660	0.609	0.515	0.500	0.614	0.608
Ionosphere	0.892	0.876	0.544	0.833	0.922	0.788	0.864	0.951	0.777	0.641	0.766	0.829	0.884
Landsat	0.548	0.540	0.575	0.474	0.614	0.382	0.549	0.607	0.366	0.533	0.626	0.506	0.464
Letter	0.763	0.886	0.589	0.616	0.812	0.537	0.878	0.804	0.524	0.503	0.780	0.598	0.689
Lymphography	0.994	0.523	0.995	0.999	0.995	0.900	0.636	0.989	0.997	0.840	0.878	0.995	0.940
Magic.gamma	0.725	0.700	0.709	0.721	0.795	0.655	0.678	0.699	0.667	0.584	0.728	0.442	0.742
Mammography	0.795	0.726	0.838	0.860	0.852	0.867	0.702	0.690	0.888	0.719	0.779	0.414	0.782
Musk	1.000	0.575	1.000	0.998	0.964	0.993	0.581	0.999	1.000	0.912	0.575	1.000	0.748
Optdigits	0.785	0.539	0.868	0.696	0.395	0.493	0.538	0.413	0.518	0.408	0.565	0.657	0.492
Pageblocks	0.893	0.758	0.779	0.897	0.919	0.712	0.703	0.923	0.907	0.753	0.914	0.609	0.908
Pendigits	0.864	0.518	0.925	0.947	0.828	0.895	0.534	0.834	0.936	0.548	0.520	0.592	0.780
Pima	0.655	0.573	0.704	0.674	0.723	0.595	0.563	0.686	0.651	0.522	0.542	0.606	0.615
Satellite	0.742	0.545	0.762	0.695	0.721	0.614	0.550	0.804	0.601	0.675	0.608	0.702	0.671
Satimage-2	0.999	0.526	0.976	0.993	0.992	0.981	0.539	0.995	0.977	0.911	0.579	0.996	0.970
Shuttle	0.621	0.493	0.986	0.997	0.732	0.389	0.526	0.990	0.990	0.898	0.500	0.208	0.852
Skin	0.675	0.534	0.588	0.670	0.720	0.442	0.550	0.892	0.447	0.554	0.708	0.579	0.773
Smtp	0.863	0.794	0.809	0.905	0.933	0.819	0.899	0.948	0.856	0.868	0.500	0.915	0.784
Spambase	0.541	0.424	0.664	0.637	0.566	0.480	0.453	0.446	0.548	0.488	0.490	0.496	0.528
Speech	0.471	0.509	0.473	0.476	0.480	0.466	0.512	0.494	0.469	0.522	0.483	0.458	0.496
Stamps	0.660	0.502	0.904	0.907	0.870	0.831	0.512	0.838	0.909	0.719	0.760	0.774	0.838
Thyroid	0.909	0.707	0.948	0.979	0.965	0.819	0.657	0.986	0.955	0.719	0.889	0.574	0.992
Vertebral	0.463	0.473	0.317	0.362	0.379	0.294	0.487	0.389	0.378	0.470	0.425	0.468	0.409
Vowels	0.884	0.933	0.679	0.763	0.951	0.705	0.932	0.732	0.604	0.464	0.738	0.791	0.888
Waveform	0.701	0.715	0.694	0.707	0.750	0.594	0.693	0.572	0.635	0.523	0.674	0.592	0.640
Wbc	0.977	0.388	0.987	0.996	0.982	0.992	0.607	0.988	0.993	0.821	0.821	0.949	0.934
Wdbc	0.990	0.867	0.989	0.988	0.980	0.980	0.849	0.969	0.988	0.715	0.347	0.983	0.985
Wilt	0.396	0.666	0.348	0.451	0.511	0.313	0.678	0.859	0.239	0.432	0.400	0.555	0.794
Wine	0.453	0.323	0.907	0.786	0.470	0.822	0.330	0.975	0.819	0.513	0.621	0.734	0.390
Wpbc	0.487	0.436	0.548	0.516	0.512	0.501	0.447	0.534	0.486	0.449	0.483	0.467	0.483
Yeast	0.461	0.465	0.402	0.394	0.396	0.461	0.453	0.406	0.418	0.503	0.396	0.503	0.442
Average	0.746	0.596	0.742	0.759	0.738	0.641	0.600	0.769	0.734	0.629	0.616	0.623	0.721

Table C.2.2. Training AUC value comparisons on tabular datasets

Data	OCSVM	COPOD	ECOD	DeepSVDD	ICL	DDPM	DTE	ODIM
ALOI	0.549	0.515	0.531	0.514	0.548	0.532	0.645	0.531
Annthyroid	0.682	0.777	0.789	0.739	0.599	0.814	0.781	0.588
Backdoor	0.889	0.500	0.500	0.735	0.936	0.892	0.806	0.885
Breastw	0.935	0.994	0.990	0.625	0.807	0.766	0.976	0.991
Campaign	0.737	0.783	0.769	0.508	0.766	0.724	0.746	0.727
Cardio	0.934	0.921	0.935	0.498	0.461	0.723	0.777	0.907
Cardiotocography	0.691	0.664	0.784	0.488	0.372	0.579	0.493	0.610
Celeba	0.781	0.757	0.763	0.491	0.684	0.796	0.699	0.842
Census	0.655	0.500	0.500	0.527	0.668	0.659	0.672	0.662
Cover	0.952	0.882	0.919	0.580	0.681	0.808	0.838	0.899
Donors	0.770	0.815	0.888	0.511	0.739	0.806	0.832	0.808
Fault	0.537	0.455	0.468	0.522	0.661	0.562	0.726	0.567
Fraud	0.954	0.943	0.949	0.769	0.931	0.924	0.956	0.944
Glass	0.661	0.760	0.710	0.517	0.729	0.560	0.881	0.785
Hepatitis	0.704	0.807	0.737	0.361	0.616	0.461	0.631	0.764
Http	0.994	0.991	0.980	0.249	0.921	0.998	0.051	0.995
Internetads	0.615	0.676	0.677	0.583	0.592	0.614	0.634	0.625
Ionosphere	0.838	0.783	0.717	0.514	0.629	0.758	0.924	0.848
Landsat	0.423	0.422	0.368	0.631	0.649	0.496	0.602	0.462
Letter	0.598	0.560	0.573	0.517	0.737	0.847	0.850	0.636
Lymphography	0.996	0.996	0.995	0.681	0.884	0.958	0.989	0.998
Magic.gamma	0.673	0.681	0.638	0.604	0.676	0.763	0.801	0.745
Mammography	0.871	0.905	0.906	0.451	0.658	0.749	0.849	0.846
Musk	1.000	0.948	0.953	0.538	0.790	0.999	0.882	1.000
Optdigits	0.507	0.500	0.500	0.519	0.533	0.402	0.386	0.586
Pageblocks	0.914	0.875	0.914	0.592	0.768	0.820	0.906	0.882
Pendigits	0.929	0.906	0.927	0.383	0.650	0.700	0.786	0.951
Pima	0.631	0.662	0.604	0.510	0.524	0.537	0.707	0.719
Satellite	0.662	0.633	0.583	0.562	0.627	0.715	0.702	0.698
Satimage-2	0.997	0.975	0.965	0.551	0.898	0.996	0.980	0.998
Shuttle	0.992	0.995	0.993	0.576	0.642	0.975	0.698	0.986
Skin	0.547	0.471	0.490	0.548	0.265	0.461	0.718	0.662
Smtp	0.845	0.912	0.882	0.895	0.656	0.956	0.930	0.822
Spambase	0.534	0.688	0.656	0.584	0.459	0.510	0.545	0.550
Speech	0.466	0.489	0.470	0.512	0.512	0.466	0.487	0.475
Stamps	0.882	0.929	0.877	0.465	0.505	0.556	0.820	0.897
Thyroid	0.958	0.939	0.977	0.505	0.693	0.871	0.964	0.916
Vertebral	0.426	0.263	0.417	0.394	0.449	0.563	0.400	0.345
Vowels	0.779	0.496	0.593	0.514	0.784	0.903	0.964	0.823
Waveform	0.669	0.739	0.603	0.609	0.661	0.617	0.729	0.722
Wbc	0.987	0.994	0.994	0.503	0.853	0.948	0.979	0.992
Wdbc	0.984	0.993	0.971	0.602	0.738	0.965	0.975	0.977
Wilt	0.317	0.345	0.394	0.465	0.649	0.659	0.552	0.329
Wine	0.671	0.865	0.738	0.507	0.455	0.374	0.425	0.906
Wpbc	0.485	0.519	0.489	0.493	0.488	0.493	0.502	0.510
Yeast	0.420	0.380	0.443	0.520	0.466	0.463	0.400	0.394
Average	0.740	0.730	0.729	0.543	0.652	0.712	0.730	**0.757**

Table C.3.1. Training PR value comparisons on tabular datasets

Data	CBLOF	FeatureBagging	HBOS	IForest	kNN	LODA	LOF	MCD	PCA	DAGMM	DROCC	GOAD	PlanarFlow
Aloi	0.037	0.104	0.034	0.034	0.048	0.033	0.097	0.032	0.037	0.033	0.030	0.033	0.032
Annthyroid	0.169	0.206	0.228	0.312	0.224	0.098	0.163	0.503	0.196	0.109	0.185	0.131	0.654
Backdoor	0.546	0.217	0.052	0.045	0.479	0.101	0.358	0.121	0.531	0.250	0.025	0.347	0.336
Breastw	0.890	0.284	0.954	0.956	0.932	0.955	0.296	0.962	0.946	0.660	0.776	0.826	0.908
Campaign	0.287	0.145	0.352	0.279	0.289	0.131	0.158	0.325	0.284	0.163	0.113	0.105	0.191
Cardio	0.482	0.161	0.458	0.559	0.402	0.428	0.159	0.364	0.609	0.193	0.272	0.540	0.471
Cardiotocography	0.335	0.276	0.361	0.436	0.324	0.463	0.271	0.311	0.462	0.271	0.258	0.403	0.348
Celeba	0.069	0.024	0.089	0.063	0.061	0.047	0.018	0.092	0.112	0.044	0.047	0.021	0.066
Census	0.087	0.061	0.073	0.073	0.088	0.065	0.069	0.153	0.087	0.062	0.058	0.072	0.073
Cover	0.070	0.019	0.026	0.052	0.054	0.090	0.019	0.016	0.075	0.044	0.056	0.005	0.010
Donors	0.148	0.120	0.135	0.124	0.182	0.255	0.109	0.141	0.166	0.086	0.123	0.040	0.241
Fault	0.473	0.396	0.360	0.395	0.522	0.336	0.388	0.334	0.332	0.361	0.496	0.381	0.329
Fraud	0.145	0.003	0.209	0.145	0.169	0.146	0.003	0.488	0.149	0.084	0.002	0.257	0.447
Glass	0.144	0.151	0.161	0.144	0.167	0.090	0.144	0.113	0.112	0.111	0.159	0.075	0.113
Hepatitis	0.304	0.225	0.328	0.243	0.252	0.275	0.214	0.363	0.339	0.253	0.221	0.291	0.317
Http	0.464	0.047	0.302	0.886	0.010	0.004	0.050	0.865	0.500	0.368	0.004	0.441	0.363
Internetads	0.296	0.182	0.523	0.486	0.296	0.242	0.232	0.344	0.276	0.207	0.197	0.288	0.262
Ionosphere	0.881	0.821	0.353	0.779	0.911	0.741	0.807	0.947	0.721	0.473	0.728	0.781	0.824
Landsat	0.212	0.246	0.231	0.194	0.258	0.183	0.250	0.253	0.163	0.230	0.272	0.198	0.186
Letter	0.166	0.445	0.078	0.086	0.203	0.083	0.433	0.174	0.076	0.083	0.252	0.098	0.153
Lymphography	0.915	0.090	0.919	0.972	0.894	0.490	0.135	0.767	0.935	0.454	0.463	0.897	0.417
Magic.gamma	0.666	0.539	0.617	0.638	0.723	0.579	0.520	0.631	0.589	0.450	0.627	0.326	0.692
Mammography	0.139	0.070	0.132	0.218	0.181	0.218	0.085	0.036	0.204	0.111	0.114	0.046	0.074
Musk	1.000	0.139	0.999	0.945	0.708	0.842	0.118	0.992	1.000	0.500	0.196	1.000	0.391
Optdigits	0.059	0.036	0.192	0.046	0.022	0.029	0.035	0.022	0.027	0.026	0.032	0.039	0.027
Pageblocks	0.547	0.341	0.319	0.464	0.556	0.410	0.292	0.617	0.525	0.255	0.632	0.373	0.538
Pendigits	0.192	0.048	0.247	0.260	0.099	0.186	0.040	0.069	0.219	0.056	0.027	0.075	0.060
Pima	0.484	0.412	0.577	0.510	0.530	0.404	0.406	0.498	0.492	0.372	0.413	0.476	0.476
Satellite	0.656	0.378	0.688	0.649	0.582	0.613	0.381	0.768	0.606	0.527	0.465	0.658	0.595
Satimage-2	0.972	0.042	0.760	0.917	0.690	0.857	0.041	0.682	0.872	0.289	0.076	0.949	0.484
Shuttle	0.184	0.081	0.965	0.976	0.193	0.168	0.109	0.841	0.913	0.438	0.072	0.136	0.346
Skin	0.289	0.207	0.232	0.254	0.290	0.180	0.221	0.490	0.172	0.226	0.285	0.232	0.335
Smtp	0.403	0.001	0.005	0.005	0.415	0.312	0.022	0.006	0.382	0.179	0.000	0.358	0.006
Spambase	0.402	0.344	0.518	0.487	0.415	0.387	0.360	0.349	0.409	0.389	0.383	0.387	0.433
Speech	0.019	0.022	0.023	0.020	0.019	0.016	0.022	0.019	0.018	0.022	0.020	0.019	0.018
Stamps	0.211	0.143	0.332	0.347	0.317	0.280	0.153	0.257	0.364	0.198	0.241	0.285	0.284
Thyroid	0.272	0.069	0.501	0.562	0.392	0.189	0.077	0.702	0.356	0.126	0.338	0.318	0.734
Vertebral	0.123	0.124	0.091	0.097	0.095	0.089	0.130	0.101	0.099	0.134	0.117	0.123	0.111
Vowels	0.166	0.314	0.078	0.162	0.443	0.127	0.326	0.085	0.069	0.041	0.178	0.154	0.295
Waveform	0.122	0.078	0.048	0.056	0.133	0.040	0.071	0.040	0.044	0.032	0.150	0.042	0.150
Wbc	0.691	0.037	0.728	0.948	0.743	0.898	0.077	0.839	0.913	0.327	0.358	0.736	0.431
Wdbc	0.688	0.154	0.761	0.702	0.521	0.527	0.128	0.395	0.613	0.152	0.039	0.589	0.568
Wilt	0.040	0.081	0.039	0.044	0.049	0.036	0.083	0.153	0.032	0.047	0.041	0.065	0.115
Wine	0.170	0.061	0.412	0.207	0.081	0.250	0.064	0.737	0.264	0.120	0.126	0.229	0.086
Wpbc	0.227	0.206	0.241	0.237	0.234	0.226	0.210	0.257	0.229	0.214	0.234	0.214	0.236
Yeast	0.314	0.325	0.328	0.304	0.294	0.330	0.315	0.298	0.302	0.353	0.284	0.332	0.309
Average	0.351	0.184	0.349	0.377	0.337	0.292	0.188	0.382	0.366	0.220	0.221	0.313	0.316

Table C.3.2. Training PR value comparisons on tabular datasets

Data	OCSVM	COPOD	ECOD	DeepSVDD	ICL	DDPM	DTE	ODIM
Aloi	0.039	0.031	0.033	0.034	0.046	0.036	0.056	0.041
Annthyroid	0.188	0.174	0.272	0.192	0.123	0.297	0.228	0.161
Backdoor	0.534	0.025	0.025	0.372	0.717	0.520	0.473	0.272
Breastw	0.897	0.989	0.982	0.482	0.635	0.537	0.921	0.988
Campaign	0.283	0.368	0.354	0.149	0.267	0.299	0.281	0.297
Cardio	0.536	0.576	0.567	0.177	0.108	0.278	0.376	0.564
Cardiotocography	0.408	0.403	0.502	0.252	0.188	0.338	0.312	0.389
Celeba	0.103	0.093	0.095	0.031	0.045	0.093	0.052	0.118
Census	0.085	0.062	0.062	0.075	0.095	0.086	0.090	0.084
Cover	0.099	0.068	0.113	0.048	0.022	0.046	0.048	0.032
Donors	0.139	0.209	0.265	0.112	0.119	0.143	0.188	0.124
Fault	0.401	0.313	0.326	0.375	0.473	0.392	0.532	0.444
Fraud	0.110	0.252	0.215	0.250	0.127	0.146	0.137	0.257
Glass	0.130	0.111	0.183	0.090	0.122	0.073	0.206	0.119
Hepatitis	0.278	0.389	0.295	0.170	0.231	0.165	0.238	0.315
Http	0.356	0.280	0.145	0.093	0.091	0.642	0.024	0.280
Internetads	0.291	0.505	0.505	0.252	0.237	0.295	0.290	0.234
Ionosphere	0.829	0.663	0.633	0.392	0.472	0.633	0.920	0.867
Landsat	0.175	0.176	0.164	0.362	0.329	0.200	0.255	0.179
Letter	0.113	0.068	0.077	0.099	0.208	0.367	0.256	0.121
Lymphography	0.885	0.907	0.894	0.254	0.264	0.731	0.805	0.967
Magic.gamma	0.625	0.588	0.533	0.499	0.548	0.651	0.730	0.693
Mammography	0.187	0.430	0.435	0.025	0.046	0.099	0.175	0.098
Musk	1.000	0.369	0.475	0.107	0.128	0.984	0.434	1.000
Optdigits	0.027	0.029	0.029	0.039	0.030	0.022	0.021	0.030
Pageblocks	0.531	0.370	0.520	0.288	0.285	0.493	0.530	0.509
Pendigits	0.226	0.177	0.270	0.022	0.045	0.056	0.089	0.302
Pima	0.477	0.536	0.484	0.366	0.385	0.400	0.528	0.491
Satellite	0.654	0.570	0.526	0.406	0.451	0.662	0.563	0.652
Satimage-2	0.965	0.797	0.666	0.052	0.102	0.783	0.507	0.949
Shuttle	0.907	0.962	0.905	0.149	0.135	0.779	0.187	0.947
Skin	0.220	0.179	0.183	0.221	0.173	0.175	0.290	0.239
Smtp	0.383	0.005	0.589	0.240	0.004	0.502	0.411	0.261
Spambase	0.402	0.544	0.518	0.456	0.370	0.384	0.407	0.410
Speech	0.019	0.019	0.020	0.018	0.020	0.020	0.019	0.018
Stamps	0.318	0.398	0.324	0.099	0.117	0.143	0.273	0.336
Thyroid	0.329	0.179	0.472	0.024	0.066	0.325	0.360	0.327
Vertebral	0.107	0.085	0.110	0.107	0.115	0.150	0.098	0.109
Vowels	0.196	0.034	0.083	0.037	0.219	0.311	0.504	0.375
Waveform	0.052	0.057	0.040	0.061	0.063	0.050	0.109	0.053
Wbc	0.813	0.883	0.882	0.069	0.211	0.758	0.722	0.710
Wdbc	0.539	0.760	0.493	0.063	0.065	0.482	0.465	0.612
Wilt	0.035	0.037	0.042	0.046	0.109	0.076	0.054	0.036
Wine	0.135	0.364	0.195	0.116	0.087	0.074	0.074	0.239
Wpbc	0.222	0.234	0.217	0.240	0.234	0.238	0.227	0.236
Yeast	0.303	0.308	0.332	0.350	0.318	0.320	0.294	0.287
Average	0.360	0.339	0.349	0.182	0.201	0.332	0.321	**0.366**

C.3. Detailed AUC and PR results over image datasets (training data)

Tables C.4.1-C.4.2 and C.5.1-C.5.2 present the AUC and PR results on the image datasets.

Table C.4.1. Training AUC value comparisons on image datasets

Data	CBLOF	FeatureBagging	HBOS	IForest	kNN	LODA	LOF	MCD	PCA	DAGMM	DROCC	GOAD	PlanarFlow
MNIST	0.843	0.664	0.574	0.811	0.867	0.564	0.658	0.856	0.848	0.631	0.615	0.698	0.645
MNIST-C	0.757	0.702	0.689	0.733	0.786	0.591	0.699	0.739	0.741	0.581	0.594	0.752	0.705
FMNIST	0.871	0.748	0.748	0.831	0.875	0.672	0.738	0.840	0.853	0.664	0.564	0.860	0.819
CIFAR10	0.663	0.687	0.572	0.629	0.659	0.591	0.686	0.639	0.659	0.530	0.503	0.659	0.621
SVHN	0.601	0.629	0.542	0.580	0.604	0.534	0.628	0.583	0.599	0.528	0.521	0.597	0.580
MVTec-AD	0.754	0.745	0.732	0.747	0.763	0.644	0.742	0.618	0.724	0.596	0.544	0.730	0.637
Average	0.748	0.696	0.643	0.722	0.759	0.599	0.692	0.712	0.737	0.588	0.557	0.716	0.668

Table C.4.2. Training AUC value comparisons on image datasets

Data	OCSVM	COPOD	ECOD	DeepSVDD	ICL	DDPM	DTE	ODIM
MNIST	0.849	0.500	0.500	0.605	0.691	0.816	0.853	0.836
MNIST-C	0.751	0.500	0.500	0.552	0.670	0.751	0.788	0.736
FMNIST	0.860	0.500	0.500	0.647	0.758	0.861	0.873	0.909
CIFAR10	0.663	0.548	0.567	0.555	0.557	0.663	0.660	0.922
SVHN	0.604	0.500	0.500	0.521	0.571	0.605	0.607	0.568
MVTec-AD	0.735	0.500	0.500	0.603	0.683	0.732	0.761	0.906
Average	0.744	0.508	0.511	0.580	0.655	0.738	0.757	**0.813**

Table C.5.1. Training PR value comparisons on image datasets

Data	CBLOF	FeatureBagging	HBOS	IForest	kNN	LODA	LOF	MCD	PCA	DAGMM	DROCC	GOAD	PlanarFlow
MNIST	0.386	0.241	0.109	0.290	0.409	0.170	0.233	0.308	0.381	0.215	0.237	0.297	0.259
MNIST-C	0.173	0.128	0.126	0.178	0.191	0.101	0.127	0.166	0.170	0.092	0.096	0.177	0.154
FMNIST	0.329	0.194	0.269	0.320	0.346	0.180	0.188	0.245	0.319	0.138	0.106	0.328	0.297
CIFAR10	0.103	0.115	0.075	0.089	0.102	0.086	0.114	0.084	0.101	0.062	0.060	0.102	0.085
SVHN	0.079	0.083	0.063	0.073	0.079	0.064	0.083	0.068	0.078	0.059	0.060	0.078	0.074
MVTec-AD	0.570	0.536	0.546	0.570	0.580	0.464	0.532	0.451	0.540	0.362	0.317	0.546	0.454
Average	0.273	0.216	0.198	0.253	0.285	0.177	0.213	0.221	0.265	0.155	0.146	0.255	0.221

Table C.5.2. Training PR value comparisons on image datasets

Data	OCSVM	COPOD	ECOD	DeepSVDD	ICL	DDPM	DTE	ODIM
MNIST	0.385	0.092	0.092	0.253	0.232	0.374	0.400	0.462
MNIST-C	0.179	0.050	0.050	0.097	0.098	0.178	0.192	0.245
FMNIST	0.329	0.050	0.050	0.181	0.158	0.325	0.339	0.650
CIFAR10	0.102	0.065	0.067	0.073	0.070	0.102	0.104	0.530
SVHN	0.078	0.050	0.050	0.063	0.068	0.078	0.080	0.078
MVTec-AD	0.555	0.236	0.236	0.387	0.404	0.546	0.578	0.611
Average	0.271	0.090	0.091	0.176	0.172	0.267	0.282	**0.429**

C.4. Detailed AUC and PR results over text datasets (training data)

Tables C.6.1-C.6.2 and C.7.1-C.7.2 present the AUC and PR results on the text datasets.

Table C.6.1. Training AUC value comparisons on text datasets

Data	CBLOF	FeatureBagging	HBOS	IForest	kNN	LODA	LOF	MCD	PCA	DAGMM	DROCC	GOAD	PlanarFlow
Amazon	0.579	0.572	0.563	0.558	0.603	0.526	0.571	0.597	0.550	0.501	0.500	0.560	0.495
20news	0.564	0.610	0.537	0.550	0.567	0.539	0.610	0.583	0.545	0.518	0.496	0.553	0.513
Agnews	0.619	0.715	0.554	0.584	0.647	0.568	0.714	0.665	0.566	0.508	0.500	0.592	0.497
Imdb	0.496	0.499	0.499	0.489	0.494	0.466	0.500	0.504	0.478	0.487	0.500	0.486	0.493
Yelp	0.635	0.661	0.600	0.602	0.670	0.581	0.661	0.655	0.592	0.498	0.504	0.590	0.527
Average	0.579	0.611	0.550	0.557	0.596	0.536	0.611	0.601	0.546	0.502	0.500	0.556	0.505

Table C.6.2. Training AUC value comparisons on text datasets

Data	OCSVM	COPOD	ECOD	DeepSVDD	ICL	DDPM	DTE	ODIM
Amazon	0.565	0.571	0.541	0.464	0.528	0.551	0.603	0.597
20news	0.559	0.533	0.544	0.515	0.547	0.547	0.570	0.687
Agnews	0.601	0.551	0.552	0.494	0.591	0.571	0.652	0.805
Imdb	0.484	0.512	0.471	0.526	0.521	0.478	0.495	0.522
Yelp	0.621	0.605	0.578	0.524	0.545	0.594	0.671	0.682
Average	0.566	0.554	0.537	0.504	0.546	0.548	0.598	**0.659**

Table C.7.1. Training PR value comparisons on text datasets

Data	CBLOF	FeatureBagging	HBOS	IForest	kNN	LODA	LOF	MCD	PCA	DAGMM	DROCC	GOAD	PlanarFlow
Amazon	0.067	0.087	0.061	0.062	0.069	0.062	0.088	0.072	0.062	0.054	0.055	0.063	0.056
20news	0.072	0.125	0.059	0.064	0.082	0.064	0.125	0.077	0.061	0.053	0.051	0.066	0.050
Agnews	0.061	0.058	0.059	0.058	0.062	0.054	0.058	0.062	0.057	0.049	0.050	0.058	0.050
Imdb	0.047	0.049	0.047	0.047	0.047	0.046	0.049	0.049	0.046	0.049	0.050	0.047	0.051
Yelp	0.073	0.085	0.070	0.070	0.083	0.067	0.085	0.075	0.069	0.049	0.051	0.068	0.056
Average	0.064	0.081	0.059	0.060	0.068	0.059	0.081	0.067	0.059	0.051	0.051	0.060	0.053

Table C.7.2. Training PR value comparisons on text datasets

Data	OCSVM	COPOD	ECOD	DeepSVDD	ICL	DDPM	DTE	ODIM
Amazon	0.064	0.061	0.062	0.058	0.063	0.063	0.072	0.063
20news	0.068	0.059	0.058	0.053	0.069	0.062	0.085	0.111
Agnews	0.059	0.060	0.055	0.046	0.052	0.057	0.062	0.176
Imdb	0.047	0.050	0.045	0.053	0.054	0.046	0.047	0.049
Yelp	0.073	0.072	0.065	0.058	0.054	0.069	0.085	0.086
Average	0.062	0.060	0.057	0.054	0.058	0.059	0.070	**0.097**

C.8. Ablation studies

Number of samples used in the IWAE Table C.40 summarizes the AUC values on several tabular datasets with various Ks from 1 to 100. Note that the IWAE with $K = 1$ equals the original VAE. As expected, a lower bound closer to the log-likelihood tends to provide a more obvious IM effect, leading to better identification performances. We can also observe that when the value of K becomes larger than 50, the enhancement seems saturated. For this reason, we set $K = 50$ in our experiments.

Table C.40. AUC results of the ODIM with various values of K.

Data	$\mathbf{K = 1}$	$\mathbf{K = 2}$	$\mathbf{K = 5}$	$\mathbf{K = 10}$	$\mathbf{K = 20}$	$\mathbf{K = 50}$	$\mathbf{K = 70}$	$\mathbf{K = 100}$
Glass	0.525	0.676	0.636	0.689	0.702	0.704	0.704	0.727
Mammography	0.565	0.794	0.807	0.835	0.841	0.852	0.852	0.853
Pendigits	0.847	0.930	0.922	0.955	0.959	0.950	0.950	0.950
Pima	0.626	0.663	0.682	0.706	0.696	0.706	0.706	0.705
Vowels	0.528	0.655	0.549	0.583	0.875	0.900	0.900	0.896
Wbc	0.866	0.929	0.915	0.919	0.937	0.935	0.935	0.939
Cardio	0.794	0.919	0.913	0.932	0.925	0.914	0.914	0.919
Thyroid	0.636	0.860	0.868	0.887	0.908	0.924	0.924	0.926
Average	0.673	0.803	0.787	0.813	0.855	0.861	0.861	0.864

Number of models to ensemble We vary the number of models used in the ensemble, i.e., B, from one to twenty and compare the performances on several tabular datasets, whose results are presented in Table C.41. There is a general tendency that using more models helps improve the identifying performance. The optimal number of models varies according to datasets, but the performance is not sensitive to the number of models used in the ensemble unless it is too small.

Table C.41. AUC results of the ODIM with various numbers of generative models to take an ensemble.

Data	$B = 1$	$B = 2$	$B = 5$	$B = 10$	$B = 12$	$B = 15$	$B = 20$
Glass	0.707	0.748	0.750	0.712	0.706	0.700	0.707
Mammography	0.825	0.836	0.844	0.856	0.857	0.858	0.859
Pendigits	0.885	0.900	0.945	0.955	0.954	0.960	0.962
Pima	0.686	0.690	0.699	0.703	0.704	0.705	0.705
Vowels	0.877	0.882	0.903	0.904	0.896	0.895	0.899
Wbc	0.922	0.922	0.931	0.936	0.935	0.938	0.937
Cardio	0.898	0.896	0.915	0.918	0.911	0.924	0.920
Thyroid	0.876	0.913	0.924	0.927	0.927	0.926	0.926
Average	0.835	0.848	0.864	0.864	0.861	0.863	0.864

Number of patience We also examine the behavior of the ODIM with various values of the number of patience, N_{pat}, whose results are provided in Table C.42. Similar to the ensemble scenario, increasing the value of this hyper-parameter generally leads to enhanced performance. However, it appears that the extent of improvement levels off when $N_{\text{pat}} \geq 10$. A largeer value of N_{pat} requires more computing time, therefore, we set $N_{\text{pat}} = 10$ throughout all experiments in our manuscript.

Table C.42. AUC results of the ODIM with various values of N_{pat}.

Data	$\mathbf{N_{pat} = 1}$	$\mathbf{N_{pat} = 2}$	$\mathbf{N_{pat} = 5}$	$\mathbf{N_{pat} = 10}$	$\mathbf{N_{pat} = 15}$
Glass	0.526	0.652	0.706	0.726	0.744
Mammography	0.587	0.815	0.853	0.851	0.846
Pendigits	0.914	0.940	0.954	0.946	0.934
Pima	0.482	0.617	0.707	0.705	0.706
Vowels	0.502	0.517	0.802	0.907	0.921
Wbc	0.528	0.740	0.869	0.941	0.941
Cardio	0.819	0.925	0.924	0.882	0.879
Thyroid	0.726	0.863	0.921	0.925	0.930
Average	0.636	0.759	0.842	0.860	0.863

Learning schedule We evaluate the robustness of the ODIM to the learning schedule. We consider the Adam optimizer with various learning rates from 1e-4 to 1e-1, whose results on FMNIST are summarized in Table C.43. We present the results of the 10 classes separately, where each class is regarded as the inlier class. Note that the identifying performances rarely change until we use a learning rate larger than 1e-2. As we usually do not consider a learning rate much larger than 1e-3 when we apply the Adam optimizer, we can conclude that our method is stable with respect to the learning schedule, which implies that our method can be used in practice without delicate tunnings.

Table C.43. (Train) AUC results of the ODIM with various values of learning rates on FMNIST.

	Learning rate									
Class	**1e-4**	**2.5e-4**	**5e-4**	**1e-3**	**2.5e-3**	**5e-3**	**1e-2**	**2.5e-2**	**5e-2**	**1e-1**
0	0.902	0.915	0.913	0.911	0.907	0.902	0.881	0.561	0.564	0.481
1	0.977	0.978	0.978	0.978	0.977	0.977	0.979	0.977	0.979	0.712
2	0.814	0.857	0.861	0.864	0.869	0.860	0.845	0.500	0.415	0.392
3	0.950	0.951	0.950	0.950	0.948	0.947	0.944	0.877	0.680	0.608
4	0.917	0.912	0.913	0.904	0.902	0.913	0.791	0.447	0.453	0.314
5	0.900	0.902	0.903	0.903	0.906	0.904	0.916	0.921	0.887	0.882
6	0.808	0.809	0.809	0.807	0.803	0.796	0.735	0.566	0.522	0.494
7	0.977	0.976	0.977	0.977	0.976	0.975	0.972	0.973	0.967	0.959
8	0.863	0.845	0.843	0.841	0.821	0.807	0.661	0.409	0.397	0.423
9	0.960	0.962	0.960	0.959	0.956	0.960	0.935	0.753	0.741	0.732
Average	0.907	0.911	0.911	0.909	0.907	0.904	0.866	0.698	0.661	0.660

Min-max scaling v.s. Standardization We investigate which pre-processing technique is more suitable for our method. The two pre-processing techniques are considered, min-max scaling and standardization. We compare their corresponding results for ODIMs on 30 tabular dataset, which are provided in Table C.44. We can see that using the min-max scaling usually gives better results than standardization. This supports our theoretical claim in Proposition 2.

Table C.44. Comparison of the two data pre-processing methods on 30 tabular datasets: 1) min-max and 2) standardization. We report the AUCs.

Data	Min-max	Standardization
Annthyroid	0.591	0.700
Breastw	0.992	0.772
Cover	0.899	0.957
Glass	0.758	0.687
Ionosphere	0.926	0.862
Letter	0.706	0.619
Mammography	0.850	0.765
Musk	1.000	0.963
Pendigits	0.950	0.941
Pima	0.718	0.452
Speech	0.477	0.450
Vertebral	0.385	0.547
Vowels	0.896	0.596
Wbc	0.937	0.840
Arrhythmia	0.782	0.768
Cardio	0.862	0.948
Satellite	0.695	0.739
Satimage-2	0.999	0.971
Shuttle	0.986	0.994
Thyroid	0.935	0.966
ALOI	0.528	0.545
Backdoor	0.877	0.895
Campaign	0.732	0.753
Celeba	0.832	0.769
Fraud	0.921	0.956
Landsat	0.461	0.540
Magic.gamma	0.774	0.638
PageBlocks	0.879	0.919
Skin	0.648	0.597
Waveform	0.692	0.641
Average	0.790	0.760

D. Further discussions

D.1. ODIM with partially labeled outliers

Some studies have utilized the availability of outlier labels to enhance the efficiency of outlier detection tasks (Ruff et al., 2020; Daniel et al., 2019). But as far as we know, these existing works require that all outliers should be labeled, which is equivalent to the SOD setting. The only difference of these studies compared to the conventional SOD solvers is that they cast the problem into an one-class classification problem rather than a two-class one.

While it is very costly to obtain perfectly labeled data, partially labeled data are frequently met in practice. In this section, we claim that the ODIM can be modified easily for such a situation by leveraging these labeled data. Assume that besides $\mathcal{U}^{tr}$, a few labeled outlier dataset $\mathcal{L}^{tr} = \{(\mathbf{x}_1^l, 1), \ldots, (\mathbf{x}_m^l, 1)\}$ is also available. Again note that outliers are still present in $\mathcal{U}^{tr}$

We simply adopt the idea of Daniel et al. (2019), which encourages the log-likelihood of known outliers to decrease with the variational *upper* bound. For $u > 1$, the upper bound, called χ upper bound (CUBO), is given as:

$$L^{\text{CUBO}}(\theta, \phi; \mathbf{x}) := \frac{1}{u} \log \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x};\phi)} \left[\left(\frac{p(\mathbf{x}|\mathbf{z};\theta)p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x};\phi)} \right)^u \right].$$

With the above CUBO term, we modify the loss function of the ODIM by adding the expected CUBO on $\mathcal{L}^{tr}$ from the original IWAE loss function to have:

$$-\mathbb{E}_{\mathbf{x} \sim \mathcal{U}^{tr}} L^{\text{IWAE}}(\theta, \phi; \mathbf{x}) + \gamma \cdot \mathbb{E}_{\mathbf{x} \sim \mathcal{L}^{tr}} L^{\text{CUBO}}(\theta, \phi; \mathbf{x}),$$

where $\gamma > 0$ is a tuning parameter controlling the degree of the CUBO loss. The CUBO loss generally increases the IWAE per-sample loss values of outliers, encouraging the IM effect to occur more clearly. In our paper, we set $u = 2$ and $\gamma = 1$.

Table D.1 summarizes the averaged results of training AUC and PR scores across six tabular datasets for different proportions of labeled outliers, representing the ratio of labeled outliers to the total number of outliers. It is clearly seen that using label information helps to enhance identifying performance by a large margin.

Note that the proposed modification can be improved further. For instance, we could use the labeled information to select the optimal number of updates. There would be other rooms to improve the ODIM with partially labeled data, which we leave as future research directions.

Table D.1. Training AUC (and PR) scores with various values of l. We consider three values for l, $l = 0.0, 0.3, 0.5$.

l	0.0	0.3	0.5
`Arrhythmia`	0.800 (0.443)	0.837 (0.767)	0.888 (0.772)
`Cardio`	0.916 (0.564)	0.991 (0.934)	0.993 (0.943)
`Satellite`	0.690 (0.652)	0.868 (0.849)	0.881 (0.849)
`Satimage-2`	0.997 (0.949)	0.998 (0.954)	0.999 (0.958)
`Shuttle`	0.981 (0.947)	0.990 (0.977)	0.990 (0.979)
`Thyroid`	0.928 (0.327)	0.995 (0.844)	0.995 (0.845)
Average	0.885 (0.647)	0.947 (0.871)	0.958 (0.891)

D.2. Differentially private ODIM

DP-SGD DP-SGD is a variant of SGD applied when updating parameters while imposing DP guarantee to the model. For a per-sample loss for a given sample $\mathbf{x}$, i.e., $l(\theta, \phi; \mathbf{x})$, we calculate gradient vector $\nabla_{\theta,\phi} l(\theta, \phi; \mathbf{x})$. We conduct a clipping operation with a given positive number $C > 0$, then add a noise from the Gaussian distribution $\mathcal{N}(0, \sigma^2 C^2 I)$ to have a deformed gradient $\bar{\nabla}_{\theta,\phi} l(\theta, \phi; \mathbf{x}) = \nabla_{\theta,\phi} l(\theta, \phi; \mathbf{x}) / \max\left(1, \frac{\|\nabla_{\theta,\phi} l(\theta,\phi;\mathbf{x})\|_2}{C}\right) + \mathcal{N}(0, \sigma^2 C^2 I)$, where $\sigma > 0$ is also a pre-specified number. Then we update parameters (θ, ϕ) using this update information $\bar{\nabla}_{\theta,\phi} l(\theta, \phi; \mathbf{x})$ using a conventional SGD or variants such as Adam and RMSProp.

Here, the two hyper-parameters, C and σ need to be specified before implementing DP-SGD. In our experiment, we set $(C, \sigma) = (20, 1.02)$. The algorithm of DP-SGD is summarized in the following.

Algorithm 2 Differentially private SGD (We set $(C, \sigma) = (20, 1.02)$.)

Require: : Training dataset $\mathcal{U}^{tr} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$, parameters: (θ, ϕ), loss function of a given sample: $l(\theta, \phi; \mathbf{x})$, optimizer: $Opt(\theta, \phi, \nabla)$, group size: L, number of updates: T
Initialize (θ_0, ϕ_0)
for t in $1 : T$ **do**
  Drawn a random sample indices A from $[n]$ with sampling probability of L/n
  For each $i \in A$, compute the gradient
  $g_{t-1}(\mathbf{x}_i) \leftarrow \nabla_{\theta,\phi} l(\theta_{t-1}, \phi_{t-1}; \mathbf{x})$
  Clip the gradient and add Gaussian noise
  $\bar{g}_{t-1}(\mathbf{x}_i) \leftarrow g_{t-1}(\mathbf{x}_i) / \max\left(1, \frac{\|g_{t-1}(\mathbf{x}_i)\|_2}{C}\right) + \mathcal{N}(0, \sigma^2 C^2 I)$
  Aggregate the gradients
  $\bar{g}_{t-1} \leftarrow \frac{1}{L} \sum_{i \in A} \bar{g}_{t-1}(\mathbf{x}_i)$
  Update parameters
  $(\theta_t, \phi_t) \leftarrow Opt(\theta_{t-1}, \phi_{t-1}, \bar{g}_{t-1})$
end for
Output: (θ_T, ϕ_T)

(ϵ, δ)-DP $(\epsilon, \delta$-DP is one of the widely used measures to examine the amount of privacy protection for a given random mechanism. The definition of (ϵ, δ)-DP is as follows:

Definition D.1. A randomized mechanism $\mathcal{M}$ with range $\mathcal{R}$ is (ϵ, δ)-DP, if

$$\Pr[\mathcal{M}(\mathcal{S}) \in \mathcal{O}] \leq \exp(\epsilon) \cdot \Pr[\mathcal{M}(\mathcal{S}') \in \mathcal{O}] + \delta$$

holds for any subset of outputs $\mathcal{O} \subseteq \mathcal{R}$ and for any neighboring datasets $\mathcal{S}$ and $\mathcal{S}'$, i.e., $|\mathcal{S} - \mathcal{S}'| = 1$.

It can be easily inferred that a random mechanism with small values of ϵ and δ is believed to provide strict privacy protection.

Calculation of (ϵ, δ)-DP when applying DP-SGD It is well known that a single iteration of DP-SGD satisfies (ϵ, δ)-DP for certain values of ϵ and δ. However, to train a given model, or parameters, we have to iterate DP-SGD multiple times. At this point, the key lies in calculating ϵ and δ properly of *composition* of DP-SGD operations.

There have been numerous techniques to calculate tight ϵ when random mechanisms are sequentially composed (Abadi et al., 2016; Mironov, 2017; Dong et al., 2019). Among them, we adopt the method of Mironov (2017) as other recent methods followed this approach (Chen et al., 2020a; Zhao et al., 2023).

Detailed experimental results We compare our method to DeepSVDD when DP-SGD is applied. As mentioned earlier, we set $\delta = 10^{-5}$ and iterate DP-SGD to train each model and stopping the learning process when the privacy budget first exceed 10, i.e., $\epsilon = 10$. We note that applying DP-SGD to non-deep-learning-based methods is not possible, as they are not trained using SGD-based optimizers.

We analyze four tabular dataset, whose results are presented in Table D.2.

Table D.2. Train AUC (and PR) value comparisons when DP-SGD is applied. We analyze four tabular datasets and set $(\epsilon, \delta) = (10, 10^{-5})$.

Class	**DeepSVDD**	**ODIM**
`Arrhythmia`	0.585 (0.325)	0.626 (0.362)
`Cardio`	0.543 (0.132)	0.743 (0.321)
`Thyroid`	0.745 (0.109)	0.805 (0.155)
`Vowels`	0.582 (0.040)	0.666 (0.096)
Average	0.614 (0.152)	0.738 (0.234)

D.3. Likelihood based methods have poor performances in UOD tasks

In the experimental section of the main manuscript, we have mentioned that the existing likelihood-based SSOD solvers often struggle to distinguish inliers with outliers in UOD tasks. In this section, we evaluate the performance of a likelihood-based SSOD method (Nalisnick et al., 2019b) and compare its effectiveness to our method on `MNIST` and `FMNIST` datasets. It is worth noting that we attempted to implement other methods, such as Nalisnick et al.; Lan & Dinh (2021), but one of them does not have an official GitHub code, and the other is based on an old version of Tensorflow, which is also not available. Therefore, we have excluded them in our experiment.

Tables D.3 and D.4 provide a summary of the comparison results. It is evident that our method outperforms the competitor across all considered scenarios. Notably, the AUC scores of Nalisnick et al.; Lan & Dinh (2021) often fall below 0.5, indicating a failure to identifying inliers from a given dataset. Consequently, we can conclude that likelihood-based SSOD methods are sub-optimal as UOD solvers.

Table D.3. Train AUC (and PR) value comparisons on `MNIST` (image data)

Class	**Nalisnick et al. (2019b)+GLOW**	**ODIM**
0	0.645 (0.951)	0.937 (0.990)
1	0.710 (0.723)	0.997 (0.999)
2	0.555 (0.928)	0.732 (0.957)
3	0.502 (0.916)	0.800 (0.969)
4	0.322 (0.882)	0.854 (0.978)
5	0.484 (0.924)	0.729 (0.964)
6	0.445 (0.910)	0.859 (0.976)
7	0.204 (0.840)	0.928 (0.990)
8	0.671 (0.954)	0.709 (0.953)
9	0.333 (0.881)	0.889 (0.983)
Average	0.387 (0.869)	0.843 (0.976)

Table D.4. Train AUC(and PR) value comparisons on `FMNIST` (image data)

Class	**Nalisnick et al. (2019b)+GLOW**	**ODIM**
0	0.460 (0.904)	0.905 (0.986)
1	0.059 (0.756)	0.976 (0.997)
2	0.589 (0.938)	0.858 (0.981)
3	0.265 (0.821)	0.943 (0.993)
4	0.593 (0.938)	0.890 (0.985)
5	0.222 (0.807)	0.899 (0.988)
6	0.645 (0.944)	0.802 (0.971)
7	0.101 (0.781)	0.980 (0.998)
8	0.633 (0.941)	0.839 (0.973)
9	0.361 (0.887)	0.958 (0.995)
Average	0.398 (0.886)	0.905 (0.987)