ADOD: Adaptive Density Outlier Detection

Li Qian*†, Jing Qian‡, Xin Sun§, Wengang Guo¶, Christian Böhm†

*Institute of Informatics, Ludwig Maximilian University of Munich, Munich, Germany, li.qian@dbs.ifi.lmu.de
†Faculty of Computer Science, University of Vienna, Vienna, Austria, christian.boehm@univie.ac.at
‡Carbon Neutral R&D Center, China Railway Hi-Tech Industry Corporation Limited, Beijing, China, jing.qian@crhic.cn
§Faculty of Data Science, City University of Macau, Macau, China, sunxin1984@ieee.org
¶ College of Electronic and Information Engineering, Tongji University, Shanghai, China, guowg@tongji.edu.cn

Abstract—Outlier detection plays a dual role in data analysis: cleansing data to optimize the performance of downstream tasks and identifying potentially rare valuable events or patterns. Proximity-based methods, which are independent of data distribution assumptions, are plagued by parameter selection and performance challenges when handling data with varying densities. This study proposed a novel unsupervised algorithm named Adaptive Density Outlier Detection (ADOD) to address these challenges. The core innovation of ADOD involves two main aspects: adaptive neighborhood boundaries and density consistency scoring. First, instead of relying on a predefined fixed radius, ADOD employs perplexity to calculate the local scale of each data point. It then dynamically adjusts the neighborhood boundaries according to this scale to adapt to data with varying densities. Second, ADOD estimates local density using a mutual neighbor graph and combines the density differences between data points and their neighbors to compute outlier scores, effectively distinguishing outliers that significantly deviate from their surroundings. This study evaluated ADOD on one synthetic and 32 real datasets, and compared it with 14 classical and state-of-the-art algorithms from different categories. Extensive experimental results demonstrated the superior performance of ADOD, achieving the highest average accuracy across ROC, P@N, and AP metrics. This study promotes the development of outlier detection techniques and expands their potential for real-time applications.

Index Terms—Outlier Detection, Unsupervised Learning, Adaptive Density, Mutual Neighbors Graph

I. INTRODUCTION

Outlier detection identifies rare and suspicious data points that significantly differ from the majority [1]. On the one hand, for the majority fitting the expected pattern, outlier detection serves as data cleaning. By effectively identifying and removing outliers, data quality can be improved, thus ensuring the accuracy and reliability of downstream tasks such as predictive modeling and statistical analysis [2]. On the other hand, the few outliers often reveal valuable events or observations [3]. For example, in finance, outliers may indicate fraudulent transactions, leading to significant economic losses; in cybersecurity, outliers may imply network intrusion, resulting in private data leakage; in healthcare, outliers may indicate atypical diagnostic observations, resulting in missing the best time for treatment, or even endangering the patient's life [3]. Therefore, accurate and efficient outlier detection improves the quality of data analysis and aids industries in proactively responding to potential threats and seizing opportunities [4].
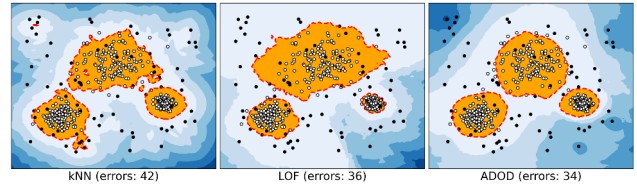

Fig. 1: Decision boundaries comparison on the ThreeBlob Outlier dataset using kNN, LOF, and ADOD: Analysis on a synthetic dataset of 500 points from three Gaussian blobs ($\sigma = [0.6, 1.2, 0.3]$) with 15% uniformly distributed outliers, where white dots represent true inliers and black dots represent true outliers. Decision boundaries are depicted with predicted inliers (yellow), predicted outliers (blue), and boundaries (red dashed lines). Error counts are recorded for each algorithm.

Outlier detection techniques can generally be divided into five categories [3]: proximity-based, probabilistic-based, linear model-based, ensemble-based, and neural network-based methods [5]. Among them, proximity-based methods [6] have received considerable attention due to their high interpretability and independence from data distribution assumptions. Proximity-based methods can be further divided into distance and density methods. Distance methods such as k Nearest Neighbors (kNN) [7] determine outliers by analyzing the distance to the k-th nearest neighbor, assuming that points farther away are more likely to be outliers. However, kNN uses the same k for all points, which limits its adaptability to complex density variations. As shown in Fig. 1, kNN incorrectly detects certain outliers as inliers, leading to more errors and irregular, scattered decision boundaries.

To alleviate this limitation, density methods such as Local Outlier Factor (LOF) [8] propose the concept of reachable distance, which is defined as the maximum distance between a point and its neighbor, and the k-distance of the neighbor. This distance metric adjusts for the difference in distances between points in regions with different densities and reduces the misclassification to a certain extent. LOF assesses outliers by calculating the local density of each point and its neighbors. However, its accuracy remains a concern in regions with significant density variations [9]. As shown in Fig. 1, for boundary points with significant density variations, their nearest k neighbors are located in both high- and low-

density regions. However, LOF incorrectly estimates their local densities, resulting in irregular decision boundaries.

To effectively handle data with varying densities, this study proposed the algorithm ADOD (Adaptive Density Outlier Detection). Inspired by the concept of perplexity in t-SNE [10], ADOD adopts perplexity as a smoothing mechanism to measure the effective number of neighbors, thereby avoiding the constraints of predefined fixed k for each point. ADOD calculates the adaptive local scale of each data point by perplexity, and dynamically adjusts the neighborhood boundary according to this scale. Based on these boundaries, it constructs a mutual neighbor graph to estimate the local density and ensures inliers in regions with different densities have similar local densities. Finally, density consistency scoring is performed by combining the local densities of a data point and the density differences with its mutual neighbors, thus distinguishing the outliers that significantly deviate from their surroundings, as shown in Fig. 1. This simple and robust algorithm avoids tedious parameter tuning and exhibits excellent performance on data from various domains.

The primary contributions include:

1) **Adaptive density outlier detection**: ADOD, one of the first outlier detection algorithms, applies perplexity for density estimation to adapt to varying densities.
2) **Efficiency and scalability**: ADOD, optimized with nearest neighbor search, enhances efficiency and scalability for handling large datasets.
3) **Generalization to unknown data**: ADOD generalizes to unknown data through comparisons with known data, extending its utility for real-time applications.

II. RELATED WORK

Probabilistic-based methods identify outliers by building statistical models of data and then calculating the probabilistic deviation of data points [3]. For example, Angle-Based Outlier Detection (ABOD) [11] calculates the angular difference between each data point and all other points, whereas FastABOD [11] simplifies the computation through nearest neighbor search (NNS) [12]. Stochastic Outlier Selection (SOS) [13] uses affinity and binding probabilities between data points to generate random neighbor graphs but relies on the perplexity parameter and has high memory requirements. Copula-based Outlier Detection (COPOD) [14] and Empirical-Cumulative-distribution-based Outlier Detection (ECOD) [15] assess the outlierness of data points using an empirical cumulative distribution function. COPOD models variable dependencies using Copula models, whereas ECOD focuses on the tail probabilities of each dimension. Both methods may be limited in real-time environments due to the lack of relationship exploitation between known and unknown data.

Linear model-based methods identify outliers by projecting data into a new space and using linear transformations [3]. For example, Principal Component Analysis (PCA) [16] assumes linear relationships and projects data to a low-dimensional space by analyzing the principal and secondary components, identifying points that deviate from the principal components

as outliers. Kernel PCA (KPCA) [17] projects data to a high-dimensional feature space through kernel functions, extracting principal components for outlier detection. However, KPCA is computationally expensive and unsuitable for large data. One-Class Support Vector Machines (OCSVM) [18] construct a hyperplane in a high-dimensional space, assuming that inliers form a tight group and outliers lie outside this group. However, OCSVM is sensitive to parameter selection and has a high computational cost.

Ensemble-based methods improve robustness and accuracy via the combination of results from multiple base models [3]. For example, Feature Bagging (FB) [19] combines multiple detections based on random feature subsets but is sensitive to subset selection. Isolation Forest (IF) [20] constructs random tree structures to isolate data points using average path lengths. Deep Isolation Forest (DIF) [21] uses deep learning techniques to generate random representations and axis-parallel partitions. However, these representations may not efficiently capture outlier patterns in high-dimensional and complex data, resulting in unstable performance. Locally Selective Combination of Parallel Outlier Ensembles (LSCP) [22] dynamically selects the optimal detector in a local region, with its effectiveness depending on the local region definition and detector selection.

Neural network-based methods employ deep learning models to capture complex nonlinear relationships and identify outliers [5]. For example, Single-Objective Generative Adversarial Active Learning (SO-GAAL) [23] generates potential outliers through a single generator and improves data discrimination boundaries through iterative learning with generative adversarial network (GAN), but it may encounter the mode collapse problem. Multiple-Objective GAAL (MO-GAAL) [23] extends SO-GAAL by using multiple generators to enhance data diversity and reduce mode collapse risk, but at a higher computational cost. Adversarially Learned Anomaly Detection (ALAD) [24] combines reconstruction errors with features obtained through adversarial learning using a bidirectional GAN. Unifying Local Outlier Detection Methods via Graph Neural Networks (LUNAR) [25] employs graph neural networks for message passing and aggregation within local neighborhoods to generate learnable outlier scores. However, its performance depends on hyperparameter selection.

Proximity-based methods identify outliers by measuring the distance or density difference between a data point and its neighbors [3]. For example, kNN [7] and its variants AvgKNN and MedKNN [26] identify outliers by analyzing the k-th nearest distances: kNN considers the exact k-th distance, AvgKNN averages these distances, and MedKNN takes the median. All methods are sensitive to the parameter k and are affected by density variations. LOF [8] computes the local density of each data point and compares it with its neighbors, which is also sensitive to parameter selection. Connectivity-based Outlier Factor (COF) [27], a variant of LOF, evaluates outlier scores by calculating the average link distance between the data point and its neighbors but requires high memory.

The proposed ADOD algorithm is a proximity-based method. It determines the adaptive local density of each data
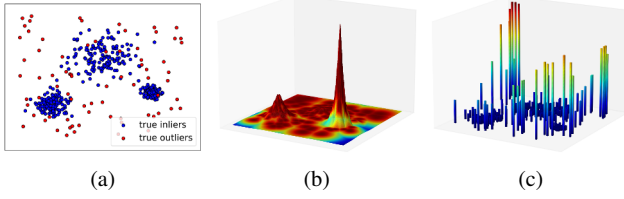
Fig. 2: Illustration of key ADOD steps on ThreeBlob Outlier dataset. (a) Ground truth. (b) Gaussian probability density via adaptive local scales (higher peaks indicate denser regions). (c) Outlier scores (higher bars indicate a greater likelihood of being an outlier).

point by perplexity and makes adjustments according to the local densities of its mutual neighbors. ADOD avoids dependence on a fixed k, improves robustness and applicability, performs well with varying densities and large data, and quickly adapts to unknown data for real-time processing.

III. METHODOLOGY

A. Problem Definition

We aim to perform unsupervised outlier detection in a multidimensional dataset. The dataset is defined as $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where each data point $\mathbf{x}_i \in \mathbb{R}^d$ is a vector in d-dimensional space, and n denotes the number of data points. The objective is to compute an outlier score for each data point, denoted as s_i for $\mathbf{x}_i$, forming the outlier score set $S = \{s_1, s_2, \ldots, s_n\}$. A higher score indicates a higher probability of being an outlier.

B. Algorithm Description

1) Adaptive Local Scale Estimation: We use the probability density function of the Gaussian distribution [28] to define the influence of the neighboring data point $\mathbf{x}_j$ on the central data point $\mathbf{x}_i$ and quantify it using the conditional probability $p_{j|i}$ with the following expression:

$$p_{j|i} = \frac{\exp(-D_{ij}^2/2\sigma_i^2)}{\sum_{l \neq i} \exp(-D_{il}^2/2\sigma_i^2)} \tag{1}$$

where $D_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$ denotes the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$, forming an $n \times n$ dissimilarity matrix D in d-dimensional space. Here, σ_i represents the local scale of $\mathbf{x}_i$. Based on σ_i, we can adaptively adjust the neighborhood boundary of $\mathbf{x}_i$ to reflect changes in local density. The bell-shaped curve of the Gaussian distribution provides a natural weight decay mechanism [28]. This causes data points further from the centroid to exhibit a gradually decreasing effect, which is suitable for quantifying the strength of interactions between data points, as shown in Fig. 2(b).

In data analysis, perplexity is commonly used as a smoothing mechanism to measure the effective number of neighbors, excelling in density estimation. It has been widely applied in techniques such as t-SNE [10] for dimensionality reduction, DBADV [29] for clustering, and SOS [13] for outlier

detection. In contrast to SOS, which uses perplexity to assess the affinity between data points and requires careful tuning, ADOD uses perplexity to obtain the adaptive local scale σ_i of $\mathbf{x}_i$, improving the adaptability to datasets with varying densities. Perplexity is defined as follows:

$$Perp(P_i) = 2^{H(P_i)} \tag{2}$$

where $H(P_i)$, the entropy of the probability distribution P_i centered on $\mathbf{x}_i$, is used to quantify the uncertainty of the distribution [28]. It is calculated as $H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}$, where $p_{j|i}$ is the conditional probability of selecting point $\mathbf{x}_j$ given $\mathbf{x}_i$. A higher $H(P_i)$ indicates a more uniform probability distribution within the neighborhood of $\mathbf{x}_i$, implying minimal density differences. Conversely, a lower entropy signifies significant density differences within the neighborhood.

The local scale σ_i is continuously adjusted by fixing the perplexity and using a binary search [10] to ensure that the neighborhood complexity (i.e., the number of effective neighbors) is balanced for each data point. Thus, it adaptively reflects the true relationships and density variations of the data points in their local surroundings. Although no explicit guidance is provided for setting the perplexity, inspired by the rule of thumb in kNN [30], it is often recommended to use an order of magnitude of $\sqrt{n}$ for k [31]. Similarly, we could set the perplexity to correlate with the order of magnitude of $\sqrt{n}$ to ensure reliable density estimation over sufficient neighbors while avoiding unnecessary computational costs associated with excessively large neighborhoods. Experiments on parameter sensitivity analysis in Section IV verified the effectiveness of this setting, simplifying parameter selection and enhancing adaptability to data size.

2) Adaptive Neighborhood Boundary Determination: ADOD employs a quantile function [32] to determine an adaptive neighborhood boundary for each data point, ensuring that neighbors have similar densities. As the inverse of the cumulative distribution function, the quantile function calculates a value such that the probability of a random variable being less than or equal to this value matches the specified cumulative probability. This property of the quantile function renders it suitable for determining the neighborhood boundaries because it dynamically adjusts to the distributional characteristics of data. The formula is expressed as follows:

$$r_i = \mu + \sqrt{2}\sigma_i \cdot \text{erf}^{-1}(2 \cdot p_{\text{cum}} - 1) \tag{3}$$

where $\text{erf}^{-1}(\cdot)$ is the inverse of the error function that determines the corresponding critical value based on cumulative probability. Because the neighborhood is centered on each data point $\mathbf{x}_i$, the mean μ is considered 0, and σ_i is the local scale of $\mathbf{x}_i$. The parameter p_{cum} represents the specified cumulative probability that a random variable falls within the calculated boundary. To ensure that the neighborhood boundaries are meaningful, p_{cum} should be $(0.5, 1)$. Here, we set p_{cum} to 0.999, which refers to the three-sigma rule [32]. This ensures that almost all data points are included in the neighborhood

of each data point while effectively identifying and isolating potential outliers that deviate significantly. Experiments on parameter sensitivity analysis in Section IV verified the effectiveness of this setting.

3) Mutual Neighbor Graph Construction: In the ADOD algorithm, the mutual neighbor graph depicts the adjacency between data points. This mutual connection reflects the interactions between data points and helps identify potential outliers. The mutual neighbor graph G is an undirected graph consisting of a node set V and an edge set E. The node set V contains all data points in the dataset, $V = \{\mathbf{x}_i \mid \mathbf{x}_i \in X\}$. The edge set E includes all pairs of nodes $(\mathbf{x}_i, \mathbf{x}_j)$ that satisfy the mutual neighbor condition: the data point $\mathbf{x}_j$ is within the neighborhood boundary r_i of $\mathbf{x}_i$, and simultaneously, $\mathbf{x}_i$ is within the neighborhood boundary r_j of $\mathbf{x}_j$. To ensure that each node pair is considered only once and to avoid self-loops, we define the set of mutual neighborhood edges as follows, where $i < j$ ensures that each node pair is unique:

$$E = \{(\mathbf{x}_i, \mathbf{x}_j) \mid D_{ij} \leq \min(r_i, r_j), i < j\} \tag{4}$$

4) Local Density Estimation: The local density d_i of $\mathbf{x}_i$ quantifies the number of neighbors within its adaptive neighborhood boundary r_i, reflecting the density distribution in that area. It is defined as follows:

$$d_i = \frac{\deg(\mathbf{x}_i) + 1}{r_i} \tag{5}$$

where r_i is the adaptive neighborhood boundary of $\mathbf{x}_i$, and $\deg(\mathbf{x}_i)$ denotes the degree of $\mathbf{x}_i$ in the mutual neighbor graph G, which is the number of mutual neighbors directly connected to $\mathbf{x}_i$. The "1" indicates that the data point itself is included, ensuring that each point includes at least itself in estimating its local density, providing a baseline density for isolated points. In dense regions, inliers have a relatively small neighborhood boundary with fewer mutual neighbors, whereas in sparse regions, they exhibit a larger neighborhood boundary with more mutual neighbors. Therefore, even if these inliers are in regions with different densities, their local density results remain comparable, effectively preventing misclassification due to density differences. For outliers, which typically have larger neighborhood boundaries but fewer mutual neighbors, the local density is significantly lower than that of inliers.

5) Density Consistency Outlier Scoring: The outlier score is calculated based on the combination of the local density of the data point and the local density differences with its mutual neighbors to identify potential outliers. The outlier score s_i of $\mathbf{x}_i$ is defined as follows:

$$s_i = d_i^{-1} + \sum_{j \in N_i} w'_{ij}(d_j^{-1} - d_i^{-1}) \tag{6}$$

where d_i^{-1} and d_j^{-1} denote the inverse of the local density of x_i and its neighbor x_j, respectively. The inverse of local density reflects the relative sparsity of a data point, where a higher value implies lower density. Consequently, such points typically gain higher outlier scores than inliers. Furthermore,

---

Algorithm 1: ADOD

Input: Dataset $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathbb{R}^{n \times d}$ with n data points and d features

Output: Outlier scores $S = \{s_1, \ldots, s_n\} \in \mathbb{R}^n$

1 Initialize $perplexity = 2 \cdot \lfloor \sqrt{n} \rfloor$, $p_{\text{cum}} = 0.999$

2 Calculate dissimilarity matrix D, where $D_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$

3 **foreach** $\mathbf{x}_i \in X$ **do**

4   Calculate adaptive local scale σ_i: $\sigma_i \leftarrow \text{binary\_search}(D_i, perplexity)$

5   Determine adaptive neighborhood boundary r_i: $r_i \leftarrow \sqrt{2}\sigma_i \cdot \text{erf}^{-1}(2 \cdot p_{\text{cum}} - 1)$

6 Initialize $G = (V, E)$ with $V = X$ and $E = \emptyset$

7 **for** $i, j \in \{1, \ldots, n\}, i < j$ **do**

8   **if** $D_{ij} \leq \min(r_i, r_j)$ **then**

9          $E \leftarrow E \cup \{(\mathbf{x}_i, \mathbf{x}_j)\}$

10 **foreach** $\mathbf{x}_i \in V$ **do**

11   Estimate local density $d_i \leftarrow \frac{\deg(\mathbf{x}_i) + 1}{r_i}$

12 **foreach** $\mathbf{x}_i \in V$ **do**

13      $N_i = \{\mathbf{x}_j \mid (\mathbf{x}_i, \mathbf{x}_j) \in E\}$

14   **foreach** $j \in N_i$ **do**

15          $w_{ij} \leftarrow \frac{1}{D_{ij}}$

16   Normalize weights: $w'_{ij} \leftarrow \frac{w_{ij}}{\sum_{l \in N_i} w_{il}}$

17      $s_i \leftarrow d_i^{-1} + \sum_{j \in N_i} w'_{ij}(d_j^{-1} - d_i^{-1})$

18 **return** $S = \{s_1, \ldots, s_n\}$

---

by calculating $d_j^{-1} - d_i^{-1}$, ADOD quantifies the density difference between each data point and its mutual neighbors, thereby identifying those points that deviate from their surroundings. If the neighbor of x_i is in a sparser surrounding compared to x_i, $d_j^{-1} - d_i^{-1} > 0$, then its outlier score further increases. Whereas, if it is in a denser surrounding, $d_j^{-1} - d_i^{-1} < 0$, the outlier score further decreases. Finally, if x_i and its neighbors are in regions of similar density, the outlier score remains unchanged.

The mutual neighbor set N_i of $\mathbf{x}_i$ is obtained from the edge set E as follows: $N_i = \{\mathbf{x}_j \mid (\mathbf{x}_i, \mathbf{x}_j) \in E\}$. The weights w_{ij} are calculated based on the Euclidean distance D_{ij} between x_i and x_j, defined by the formula $w_{ij} = \frac{1}{D_{ij}}$. These weights are then normalized to ensure that the total sum of all neighboring weights equals 1, using the formula $w'_{ij} = \frac{w_{ij}}{\sum_{l \in N_i} w_{il}}$. This process reflects the distance relationship between each neighbor point and the current point (i.e., the closer the distance, the greater the influence), and balances the contribution of each neighbor to the outlier score, as shown in Fig. 2(c).

C. Algorithm Overview

The ADOD algorithm is designed to identify potential outliers in the dataset, and its pseudocode is presented in Algorithm 1. The algorithm accepts a dataset $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathbb{R}^{n \times d}$ containing n data points where each data point has

403

Algorithm 2: ADOD for Unknown Data

Input: Unknown dataset
$X^{\mathrm{unk}} = \{\mathbf{x}_1^{\mathrm{unk}}, \dots, \mathbf{x}_m^{\mathrm{unk}}\} \in \mathbb{R}^{m \times d}$ with m data points and d features, Known dataset
$X^{\mathrm{kn}} = \{\mathbf{x}_1^{\mathrm{kn}}, \dots, \mathbf{x}_n^{\mathrm{kn}}\} \in \mathbb{R}^{n \times d}$

Output: Outlier scores $S^{\mathrm{unk}} = \{s_1^{\mathrm{unk}}, \dots, s_m^{\mathrm{unk}}\} \in \mathbb{R}^m$

1 Initialize $perplexity = 2 \cdot \lfloor \sqrt{n} \rfloor$, $p_{\mathrm{cum}} = 0.999$

2 **foreach** $\mathbf{x}_i^{unk} \in X^{unk}$ **do**

3   Calculate dissimilarity vector D_i^{unk} for $\mathbf{x}_i^{\mathrm{unk}}$ with each $\mathbf{x}_j^{\mathrm{kn}}$ in X^{kn}: $D_{ij}^{\mathrm{unk}} = \|\mathbf{x}_i^{\mathrm{unk}} - \mathbf{x}_j^{\mathrm{kn}}\|_2$ for $j = 1, \dots, n$

4   Calculate adaptive local scale σ_i^{unk}:
$\sigma_i^{\mathrm{unk}} \leftarrow \mathrm{binary\_search}(D_i^{\mathrm{unk}}, perplexity)$

5   Determine adaptive neighborhood boundary r_i^{unk}:
$r_i^{\mathrm{unk}} \leftarrow \sqrt{2}\sigma_i^{\mathrm{unk}} \cdot \mathrm{erf}^{-1}(2 \cdot p_{\mathrm{cum}} - 1)$

6   Initialize neighbor set $N_i^{\mathrm{unk}} \leftarrow \emptyset$

7   **for** $j \in \{1, \dots, n\}$ **do**

8   **if** $D_{ij}^{unk} \leq \min(r_i^{unk}, r_j^{kn})$ **then**

9          $N_i^{\mathrm{unk}} \leftarrow N_i^{\mathrm{unk}} \cup \{\mathbf{x}_j^{\mathrm{kn}}\}$

10   Estimate local density $d_i^{\mathrm{unk}} \leftarrow \frac{|N_i^{\mathrm{unk}}| + 1}{r_i^{\mathrm{unk}}}$

11   **foreach** $\mathbf{x}_j^{kn} \in N_i^{unk}$ **do**

12       $w_{ij}^{\mathrm{unk}} \leftarrow \frac{1}{D_{ij}^{\mathrm{unk}}}$

13   Normalize weights: $w_{ij}^{\mathrm{unk}\prime} \leftarrow \frac{w_{ij}^{\mathrm{unk}}}{\sum_{\mathbf{x}_l^{\mathrm{kn}} \in N_i^{\mathrm{unk}}} w_{il}^{\mathrm{unk}}}$

14    $s_i^{\mathrm{unk}} \leftarrow \frac{1}{d_i^{\mathrm{unk}}} + \sum_{\mathbf{x}_j^{\mathrm{kn}} \in N_i^{\mathrm{unk}}} w_{ij}^{\mathrm{unk}\prime} \left(\frac{1}{d_j^{\mathrm{kn}}} - \frac{1}{d_i^{\mathrm{unk}}} \right)$

15 **return** $S^{unk} = \{s_1^{unk}, \dots, s_m^{unk}\}$

---

accurately identify known data patterns, but also be able to flexibly adapt to unknown patterns [3]. An efficient outlier detection algorithm should leverage the known model and data structure to comprehensively assess the degree of outlierness, including comparing the unknown data points with known ones regarding distance or dissimilarity, and computing the corresponding outlier scores.

Algorithm 2 details how ADOD evaluates potential outliers in an unknown dataset. It receives an unknown dataset $X^{\mathrm{unk}} = \{\mathbf{x}_1^{\mathrm{unk}}, \dots, \mathbf{x}_m^{\mathrm{unk}}\} \in \mathbb{R}^{m \times d}$ with m data points, and a known dataset $X^{\mathrm{kn}} = \{\mathbf{x}_1^{\mathrm{kn}}, \dots, \mathbf{x}_n^{\mathrm{kn}}\} \in \mathbb{R}^{n \times d}$ with n data points. For each unknown data point $\mathbf{x}_i^{\mathrm{unk}}$, the algorithm first computes the Euclidean distance between $\mathbf{x}_i^{\mathrm{unk}}$ and all points in the known dataset, forming the dissimilarity vector D_i^{unk} (line 3). Next, based on D_i^{unk} and the same perplexity as the known dataset, the adaptive local scale σ_i^{unk} for $\mathbf{x}_i^{\mathrm{unk}}$ is determined by a binary search, and then adaptive neighborhood boundaries r_i^{unk} are calculated based on σ_i^{unk} and the predetermined cumulative probability p_{cum} (lines 4-5). Subsequently, a set of mutual neighbors N_i^{unk} of $\mathbf{x}_i^{\mathrm{unk}}$ is constructed to include the known data points that satisfy the mutual neighbor conditions (lines 6-9). The local density d_i^{unk} of $\mathbf{x}_i^{\mathrm{unk}}$ is determined by the number of neighbors within its adaptive neighborhood boundary r_i^{unk} (line 10). The outlier score s_i^{unk} is computed using the normalized neighbor weights and density differences, which quantify the degree of outlierness of the unknown data points with the known data points (lines 11-14).

E. Complexity Analysis

The time complexity of the ADOD algorithm without optimization is analyzed as follows: First, the adaptive local scale estimation is mainly affected by the dissimilarity matrix computation, calculating the Euclidean distance between all pairs of data points. The time complexity is $O(n^2)$, where n denotes the number of data points. Next, the time complexity of the adaptive neighborhood boundary determination step is $O(n)$. Mutual neighborhood graph construction necessitates the comparison of every pair of data points with a time complexity of $O(n^2)$. The local density estimation iterates over all the data points with a time complexity of $O(n)$. Finally, the density consistency outlier scoring step exhibits a time complexity of $O(n \cdot k_{avg})$, where k_{avg} is the average number of mutual neighbors used to compute the density difference between each data point and its mutual neighbors.

In summary, the overall time complexity of the ADOD algorithm without optimization is $O(n^2)$. For the evaluation of unknown data, the time complexity is similar to that of the above steps, and the overall time complexity is $O(m \cdot n)$, where m denotes the number of unknown data points.

F. Efficiency Optimization using NNS

We optimize the efficiency of the ADOD algorithm using nearest neighbor search (NNS) [12]. This method enables the algorithm to focus on the primary neighbors of each data point rather than calculating all pairwise distances, significantly increasing processing speed and reducing memory usage. As

d features and outputs outlier scores for all data points $S = \{s_1, \dots, s_n\} \in \mathbb{R}^n$. It begins by calculating the Euclidean distances between all pairs of data points to form a dissimilarity matrix D (line 2). For each data point $\mathbf{x}_i$, an adaptive local scale σ_i is determined via a binary search based on perplexity and D_i, and the adaptive neighborhood boundaries r_i are computed based on σ_i and the predetermined cumulative probability p_{cum} (lines 3-5). A mutual neighbor graph G is constructed, where the edge set E includes only pairs of data points that lie within the neighborhood boundary of each other (lines 6-9). The local density d_i of $\mathbf{x}_i$ is determined by the number of neighbors within its neighborhood boundary (lines 10-11). Finally, the outlier score for each data point is calculated based on the local density and normalized weights of the neighbors w'_{ij} (lines 12-17). Through this density-consistent scoring mechanism, ADOD can effectively identify and quantify outlier patterns in data, thus rendering it adaptive and robust across various data properties.

D. Generalization to Unknown Data

In data analysis, unknown data refers to data points added after the initial training or model construction, which differ from the known dataset observed during model initialization. To effectively detect outliers among these unknown data points, an outlier detection algorithm should not only

previously discussed, the number of neighbors is typically related to the order of magnitude of $\sqrt{n}$ [30]. In this study, we follow the approach used in t-SNE [10], which applies a multiplier of 3 to the perplexity, and set the number of neighbors for NNS to $3\sqrt{n}$. This ensures that sufficient local information is captured while maintaining high computational efficiency. When evaluating unknown data points, NNS simplifies the querying of neighbors between the unknown and known data points. This strategy accelerates the outlier detection process and enables the algorithm to adapt to real-time applications.

IV. EXPERIMENTS

A. Experimental Setup

1) Experimental Environment: All experiments were conducted on a machine equipped with a 14-core Intel Core i9 2.50 GHz CPU, 64GB of RAM, and an NVIDIA GeForce RTX 3080 Ti GPU. The proposed algorithm ADOD is implemented in Python, utilizing the `faiss-gpu` library [33] for NNS. The code repository is available at https://github.com/Qian-Lily/ADOD.

2) Datasets: Table I summarizes the 32 commonly used real datasets for outlier detection. These datasets are sourced from two repositories: 20 from ODDS [34] and 12 from AD-Bench [35]. They cover various domains, including healthcare, chemistry, physics, linguistics, finance, astronomy, web, image processing, and sociology. The total number of samples ranges from 80 to 567,498, the number of samples after deduplication ranges from 80 to 286,048, the number of dimensions ranges from 3 to 500, and the percentage of outliers after deduplication ranges from 0.03% to 34.90%. To ensure the accuracy and validity of the experiments, we removed duplicate entries from the datasets to maintain data uniqueness. As the outlier detection algorithms discussed in this study are unsupervised, we used the entire dataset, applying standardized preprocessing to ensure consistency when evaluating their performance on real datasets. The labels "1" denote outliers, and "0" denote inliers.

3) Baselines: To comprehensively evaluate the performance of ADOD, we selected 14 state-of-the-art and representative outlier detection algorithms for comparison. These include probabilistic-based ECOD [15], FastABOD [11], and SOS [13]; linear model-based KPCA [17] and OCSVM [18]; proximity-based LOF [8], COF [27], and kNN [7]; ensemble-based DIF [21], FB [19], and LSCP [22]; and neural network-based MO-GAAL [23], ALAD [24], and LUNAR [25]. Detailed descriptions of these algorithms are provided in Section II. The implementation code for these algorithms can be found in the PyOD library [36], a Python toolbox specifically designed for outlier detection. All comparison algorithms were configured with default parameters from the PyOD library, with the LSCP algorithm using the settings provided in the PyOD examples, where LOF was employed as the $detector\_list$ with $n\_neighbors$ set to $\{15, 20, 25, 30\}$.

4) Evaluation Metric: We evaluated the performance by averaging the scores from 10 independent trials with different random values, using the area under the receiver operating

TABLE I: Characteristics of the 32 real-world datasets. Datasets with "_OD" and "_AD" suffixes are from ODDS and ADBench repositories, respectively.

Dataset	Total	Unique	Dim.	%Out.	Category
Hepatitis_AD	80	80	19	16.25	Healthcare
wine_OD	129	129	13	7.75	Chemistry
lympho_OD	148	148	18	4.05	Healthcare
WPBC_AD	198	198	33	23.74	Healthcare
Stamps_AD	340	340	9	9.12	Document
WDBC_AD	367	367	30	2.72	Healthcare
wbc_OD	378	377	30	5.31	Healthcare
arrhythmia_OD	452	452	274	14.60	Healthcare
pima_OD	768	768	8	34.90	Healthcare
vowels_OD	1456	1452	12	3.17	Linguistics
cardio_OD	1831	1822	21	9.60	Healthcare
musk_OD	3062	3062	166	3.17	Chemistry
Waveform_AD	3443	3443	21	2.90	Physics
speech_OD	3686	3686	400	1.65	Linguistics
thyroid_OD	3772	3656	6	2.54	Healthcare
PageBlocks_AD	5393	5393	10	9.46	Document
satimage-2_OD	5803	5801	36	1.19	Astronautics
satellite_OD	6435	6435	36	31.64	Astronautics
pendigits_OD	6870	6870	16	2.27	Image
annthyroid_OD	7200	7062	6	7.56	Healthcare
mnist_OD	7603	7603	100	9.21	Image
mammography_OD	11183	7848	6	3.22	Healthcare
magic_gamma_AD	19020	18905	10	34.77	Physics
campaign_AD	41188	41176	62	11.27	Finance
shuttle_OD	49097	49097	9	7.15	Astronautics
smtp_OD	95156	71230	3	0.03	Web
backdoor_AD	95329	87020	196	2.16	Network
celeba_AD	202599	113983	39	2.55	Image
fraud_AD	284807	275661	29	0.17	Finance
cover_OD	286048	286048	10	0.96	Botany
census_AD	299285	223223	500	8.25	Sociology
http_OD	567498	221900	3	0.03	Web

characteristic (ROC) [36], Precision@rank n (P@N) [36], and average precision (AP) [15]. These metrics, which compare the ground truth with the predicted scores, indicate better performance with higher values. Due to space limitations, AP results are provided in our code repository. Furthermore, we used the critical difference (CD) diagram [37] to illustrate the statistical differences, where this diagram visualizes the statistical comparisons using the Wilcoxon signed-rank test with Holm's correction, with a default p-value of 0.05.

B. Parameter Sensitivity Analysis

Fig. 3(a) shows the average ROC score performance of ADOD on 32 real datasets with different cumulative probability p_{cum} and perplexity settings. The performance was optimal for $p_{\text{cum}} = 0.999$, with the ROC score peaking at 0.852 (marked with a yellow star) when perplexity was $2\sqrt{n}$, before gradually decreasing. Fig. 3(b) illustrates the average P@N score, also optimal for $p_{\text{cum}} = 0.999$, reaching 0.467 at $3\sqrt{n}$, with only a slight decrease of 0.01 at $2\sqrt{n}$. P@N scores remained stable between $2\sqrt{n}$ and $4\sqrt{n}$, with minor variations.

Overall, the cumulative probability setting significantly impacted performance. With $p_{\text{cum}} = 0.999$, the ROC score changed by 0.022, and the P@N score changed by 0.059. These results indicate that perplexity variations have a smaller
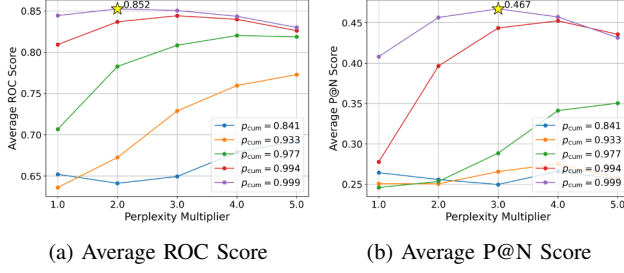
(a) Average ROC Score

(b) Average P@N Score

Fig. 3: Impact of perplexity and cumulative probability on ADOD performance. Sensitivity analysis was performed with cumulative probability values $p_{\text{cum}} \in \{0.841, 0.933, 0.977, 0.994, 0.999\}$ (corresponding to 1, 1.5, 2, 2.5, and 3 sigma) and perplexity values $\{1, 2, 3, 4, 5\} \times \sqrt{n}$.

impact on performance, demonstrating high robustness. Based on this analysis, we recommend setting the perplexity to $2\sqrt{n}$ and the cumulative probability p_{cum} to 0.999. Throughout the experiments, we adopted these parameter settings to ensure the optimal performance of the ADOD algorithm in outlier detection tasks.

C. Decision Boundaries Comparison

Fig. 4 shows the decision boundary and number of errors for various algorithms on ThreeBlob Outlier dataset. In this experiment, ADOD for unknown data was used to generate decision boundaries, where the dataset represents known data, and the entire two-dimensional plane represents unknown data. The decision boundaries are depicted with yellow regions for inliers and blue regions for outliers, with the gradient of blue indicating the outlier score, where darker blue represents a higher outlier score. In summary, ADOD exhibited the lowest number of errors (34) and outperformed all the baselines by accurately identifying three Gaussian clusters of different densities with clear decision boundaries. LOF, LSCP, and FastABOD exhibited relatively few errors (36) but featured irregular decision boundaries. LSCP presented almost the same results as LOF because LSCP is an ensemble-based method based on LOF detector. OCSVM, with 38 errors, displayed smooth decision boundaries but merged blobs of different densities into a single region. COF, ECOD, DIF, MO-GAAL, SOS and ALAD failed to obtain reasonable decision boundaries. SOS, ALAD, and KPCA reported error numbers of 106, 120, and 148, respectively.

D. Results and Analysis on Real Dataset

Table II shows the ROC scores on real datasets. ROC scores, or Areas Under the Curve (AUC), are calculated by plotting ROC curves that contrast the true positive rate with the false positive rate. ADOD ranked first on 19 out of 32 datasets, achieving an overall average score of 0.852 and outperforming all baseline algorithms. ADOD and OCSVM both achieved a perfect score of 1.0 on the musk dataset, while their scores on the http dataset were rounded to 1.0. Despite leading only two datasets, OCSVM ranked second overall with an
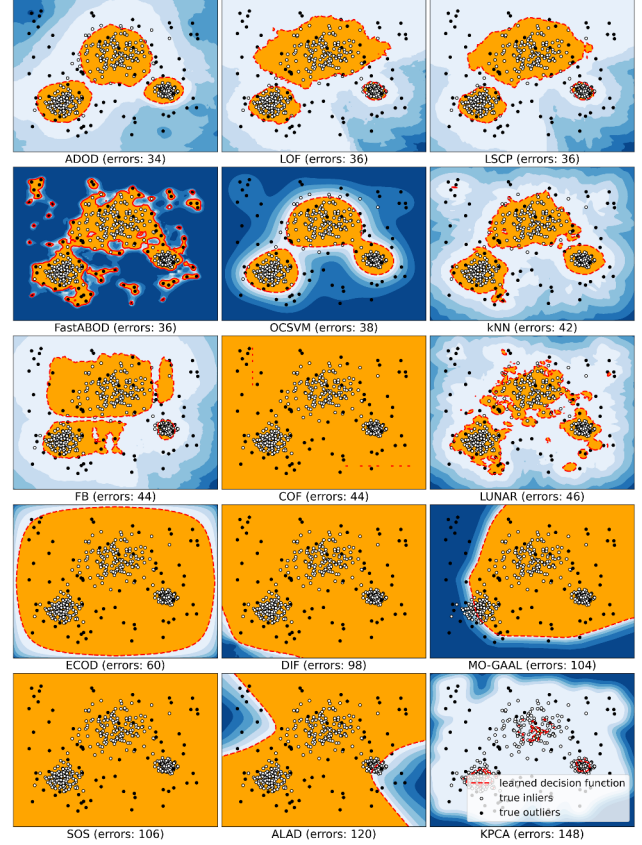

Fig. 4: Decision boundaries comparison on the ThreeBlob Outlier dataset using various outlier detection algorithms, ranked by error counts from lowest to highest.

average score of 0.817. ECOD led in six datasets, ranking third overall with a score of 0.803. In contrast, SOS and MO-GAAL underperformed, with average scores of 0.540 and 0.532, respectively. KPCA performed the worst, with an average score of 0.483, ranking 15th. It is worth noting that certain algorithms failed to complete the computation because of out-of-memory (O/M) or out-of-time (O/T) issues on specific large or high-dimensional datasets, with a single runtime limit of 12 hours. For instance, SOS, KPCA, and COF encountered out-of-memory issues on certain datasets, whereas LSCP and MO-GAAL were plagued by timeouts.

Table III shows the P@N scores on real datasets. P@N measures the proportion of true outliers among the first N predictions of the model, where N denotes the number of true outliers in the dataset. ADOD ranked first on 14 out of 32 datasets, with an overall average score of 0.457, significantly outperforming all the baselines. ADOD achieved a perfect score of 1.0 on the musk dataset. OCSVM followed in terms of overall performance, ranking first on four datasets, with an average score of 0.385. The scores of ADOD improved by approximately 18.70% compared to the runner-up OCSVM. In contrast, SOS performed the worst, with an average score of

TABLE II: ROC performance on real datasets: highest scores in bold; ranking in parentheses (lower is better). Notations: N/A (No Results), O/M (Out-of-Memory, >64GB), O/T (Out-of-Time, >12h).

Datasets	ECOD	FastABOD	SOS	KPCA	OCSVM	LOF	COF	kNN	DIF	FB	LSCP	MO-GAAL	ALAD	LUNAR	ADOD
Hepatitis	0.739(4)	0.620(10)	0.480(12)	0.431(15)	0.721(5)	0.710(7)	0.479(13)	0.740(3)	0.720(6)	0.709(8)	0.762(2)	0.700(9)	0.479(13)	0.606(11)	**0.824(1)**
wine	0.733(5)	0.433(12)	0.477(9)	0.291(14)	0.696(6)	0.886(3)	0.302(13)	0.546(8)	0.634(7)	0.883(4)	0.898(2)	0.077(15)	0.476(10)	0.475(11)	**0.967(1)**
lympho	**0.997(1)**	0.876(9)	0.617(14)	0.816(11)	0.975(3)	0.970(4)	0.864(10)	0.967(7)	0.793(12)	0.969(6)	0.970(4)	0.583(15)	0.696(13)	0.921(8)	0.981(2)
WPBC	0.481(9)	0.466(14)	0.463(15)	0.471(12)	0.485(8)	**0.519(1)**	0.474(11)	0.501(7)	0.504(6)	0.514(3)	0.517(2)	0.478(10)	0.465(14)	0.514(3)	0.507(5)
Stamps	0.876(2)	0.731(6)	0.455(15)	0.571(12)	0.872(3)	0.604(11)	0.540(14)	0.833(5)	0.869(4)	0.635(9)	0.698(7)	0.628(10)	0.558(13)	0.684(8)	**0.912(1)**
WDBC	0.971(7)	0.921(11)	0.501(14)	0.722(12)	0.983(3)	0.982(4)	0.947(8)	0.977(6)	0.922(10)	0.984(2)	0.981(5)	0.068(15)	0.704(13)	0.937(9)	**0.992(1)**
wbc	0.915(8)	0.915(8)	0.622(13)	0.466(14)	0.949(2)	0.945(5)	0.898(10)	**0.951(1)**	0.877(11)	0.949(2)	0.945(5)	0.070(15)	0.659(12)	0.939(7)	0.946(4)
arrhythmia	**0.805(1)**	0.748(11)	0.628(12)	0.589(14)	0.774(3)	0.760(9)	0.772(4)	0.768(6)	0.786(2)	0.757(10)	0.762(7)	0.612(13)	0.533(15)	0.772(4)	0.761(8)
pima	0.594(10)	0.675(4)	0.531(13)	0.492(14)	0.624(6)	0.603(7)	0.591(11)	0.714(2)	0.673(5)	0.602(8)	0.602(8)	0.306(15)	0.537(12)	0.687(3)	**0.735(1)**
vowels	0.605(11)	**0.985(1)**	0.652(10)	0.562(12)	0.793(9)	0.943(4)	0.975(2)	0.975(2)	0.536(13)	0.942(5)	0.942(5)	0.100(15)	0.536(13)	0.942(5)	0.860(8)
cardio	0.938(2)	0.548(10)	0.538(13)	0.537(14)	0.935(3)	0.544(12)	0.567(9)	0.710(5)	0.905(4)	0.578(8)	0.548(10)	0.379(15)	0.600(7)	0.618(6)	**0.946(1)**
musk	0.956(3)	0.054(15)	0.454(13)	0.498(12)	**1.000(1)**	0.637(6)	0.560(9)	0.759(4)	0.540(11)	0.619(7)	0.612(8)	0.674(5)	0.550(10)	0.236(14)	**1.000(1)**
Waveform	0.604(11)	0.654(10)	0.595(12)	0.507(14)	0.672(9)	0.708(6)	0.697(7)	0.734(2)	0.693(8)	0.731(3)	0.718(4)	0.436(15)	0.509(13)	0.710(5)	**0.786(1)**
speech	0.470(12)	**0.751(1)**	0.648(2)	0.442(15)	0.466(13)	0.508(7)	0.532(5)	0.485(10)	0.472(11)	0.507(8)	0.500(9)	0.455(14)	0.517(6)	0.536(4)	0.620(3)
thyroid	**0.976(1)**	0.943(5)	0.529(14)	0.060(15)	0.957(4)	0.704(12)	0.609(13)	0.959(3)	0.712(11)	0.750(9)	0.751(8)	0.869(7)	0.717(10)	0.935(6)	0.970(2)
PageBlocks	0.914(6)	0.750(8)	0.512(14)	0.357(15)	0.915(2)	0.725(10)	0.623(13)	0.855(4)	0.814(5)	0.782(6)	0.740(9)	0.719(11)	0.704(12)	0.761(7)	**0.937(1)**
satimage-2	0.965(4)	0.812(8)	0.493(15)	0.561(11)	0.997(2)	0.534(13)	0.565(10)	0.936(5)	0.995(3)	0.528(14)	0.545(12)	0.899(6)	0.694(9)	0.878(7)	**0.998(1)**
satellite	0.583(7)	0.555(9)	0.466(15)	0.572(8)	0.664(4)	0.546(11)	0.536(13)	0.672(2)	0.650(5)	0.542(12)	0.554(10)	0.669(3)	0.532(14)	0.639(6)	**0.745(1)**
pendigits	0.927(3)	0.673(8)	0.505(12)	0.603(9)	0.931(2)	0.499(13)	0.522(11)	0.744(5)	0.912(4)	0.489(15)	0.497(14)	0.713(6)	0.523(10)	0.688(7)	**0.940(1)**
annthyroid	0.784(4)	**0.830(1)**	0.624(11)	0.318(15)	0.675(10)	0.743(6)	0.727(8)	0.803(3)	0.546(13)	0.804(2)	0.750(5)	0.624(11)	0.536(14)	0.729(7)	0.697(9)
mnist	0.746(6)	0.774(4)	N/A	0.517(14)	0.850(2)	0.679(10)	0.635(11)	0.841(3)	0.730(7)	0.681(9)	0.696(8)	0.559(13)	0.560(12)	0.766(5)	**0.877(1)**
mammography	**0.886(1)**	0.756(6)	0.535(14)	0.405(15)	0.844(2)	0.659(10)	0.604(12)	0.797(4)	0.542(13)	0.669(9)	0.677(8)	0.712(7)	0.606(11)	0.788(5)	0.807(3)
magic_gamma	0.639(11)	0.798(3)	0.575(12)	0.222(15)	0.675(9)	0.697(8)	0.648(10)	0.815(2)	0.703(6)	0.715(5)	0.700(7)	0.531(14)	0.557(13)	**0.819(1)**	0.728(4)
campaign	**0.770(1)**	0.732(4)	0.579(12)	0.529(14)	0.737(3)	0.621(9)	0.595(11)	0.741(2)	0.559(13)	0.605(10)	0.631(7)	0.622(8)	0.529(14)	0.660(6)	0.705(5)
shuttle	**0.993(1)**	0.617(10)	0.498(15)	0.843(6)	0.992(2)	0.523(12)	0.522(13)	0.640(8)	0.984(3)	0.509(14)	0.525(11)	0.733(7)	0.847(5)	0.623(9)	0.949(4)
smtp	0.907(7)	0.945(3)	0.510(14)	0.181(15)	0.923(5)	0.902(8)	0.730(12)	0.951(2)	0.737(11)	0.894(9)	0.919(6)	0.615(13)	0.787(10)	0.939(4)	**0.961(1)**
backdoor	0.886(2)	0.717(10)	O/M	O/M	0.886(2)	0.787(6)	0.739(9)	0.783(7)	0.745(8)	0.796(5)	0.807(4)	O/T	0.702(11)	0.566(12)	**0.912(1)**
celeba	0.701(3)	0.415(11)	O/M	O/M	0.719(2)	0.489(9)	O/M	0.630(4)	0.491(8)	0.507(7)	0.485(10)	O/T	0.528(5)	0.527(6)	**0.756(1)**
fraud	0.947(3)	0.886(7)	O/M	O/M	0.952(2)	0.506(11)	O/M	0.945(4)	0.912(6)	0.508(10)	0.534(9)	O/T	0.679(8)	0.930(5)	**0.955(1)**
cover	0.920(3)	0.750(5)	O/M	O/M	0.952(2)	0.521(11)	O/M	0.779(4)	0.589(7)	0.538(9)	0.527(10)	O/T	0.589(7)	0.713(6)	**0.970(1)**
census	0.488(9)	**0.565(1)**	O/M	O/M	0.542(3)	0.532(5)	O/M	0.551(2)	0.335(10)	0.518(6)	O/T	O/T	0.489(8)	0.517(7)	0.535(4)
http	0.995(4)	0.981(6)	O/M	O/M	**1.000(1)**	0.638(11)	O/M	0.996(3)	0.928(7)	0.806(9)	0.709(10)	O/T	0.888(8)	0.985(5)	**1.000(1)**
average	0.803(3)	0.715(6)	0.540(13)	0.483(15)	0.817(2)	0.676(10)	0.639(11)	0.784(4)	0.713(7)	0.688(9)	0.694(8)	0.532(14)	0.603(12)	0.720(5)	**0.852(1)**

TABLE III: P@N performance on real datasets: highest scores in bold; ranking in parentheses (lower is better). Notations: N/A (No Results), O/M (Out-of-Memory, >64GB), O/T (Out-of-Time, >12h).

Datasets	ECOD	FastABOD	SOS	KPCA	OCSVM	LOF	COF	kNN	DIF	FB	LSCP	MO-GAAL	ALAD	LUNAR	ADOD
Hepatitis	0.308(4)	0.000(14)	0.077(12)	0.083(11)	0.231(7)	0.231(7)	0.000(14)	0.308(4)	0.215(9)	0.254(6)	0.354(2)	0.338(3)	0.154(10)	0.069(13)	**0.385(1)**
wine	0.100(5)	0.000(8)	0.000(8)	0.000(8)	0.000(8)	0.100(5)	0.000(8)	0.000(8)	0.130(4)	0.150(3)	0.200(2)	0.000(8)	0.060(7)	0.000(8)	**0.600(1)**
lympho	**0.833(1)**	0.500(8)	0.000(15)	0.167(14)	0.667(2)	0.667(2)	0.500(8)	0.667(2)	0.433(11)	0.667(2)	0.667(2)	0.267(12)	0.217(13)	0.500(8)	0.667(2)
WPBC	0.128(14)	0.128(14)	0.234(2)	**0.238(1)**	0.170(7)	0.170(7)	0.170(7)	0.170(7)	0.162(12)	0.179(6)	0.181(5)	0.223(3)	0.217(4)	0.160(13)	0.170(7)
Stamps	0.290(2)	0.161(13)	0.032(15)	0.241(3)	0.194(6)	0.194(6)	0.194(6)	0.194(6)	**0.374(1)**	0.190(12)	0.194(6)	0.229(4)	0.097(14)	0.203(5)	0.194(6)
WDBC	0.400(5)	0.200(10)	0.100(14)	0.400(5)	0.500(2)	0.400(5)	0.200(10)	0.400(5)	0.120(13)	0.470(3)	0.470(3)	0.000(15)	0.180(12)	0.320(9)	**0.800(1)**
wbc	0.450(8)	0.300(11)	0.100(14)	0.200(12)	0.500(5)	0.550(2)	0.350(10)	0.500(5)	0.360(9)	**0.570(1)**	0.500(5)	0.000(15)	0.145(13)	0.535(4)	0.550(2)
arrhythmia	**0.485(1)**	0.379(9)	0.288(13)	0.390(8)	0.424(3)	0.379(9)	0.409(5)	0.409(5)	0.468(2)	0.370(12)	0.379(9)	0.280(14)	0.194(15)	0.417(4)	0.394(7)
pima	0.455(7)	0.530(3)	0.347(13)	0.329(14)	0.478(6)	0.440(8)	0.422(11)	0.545(2)	0.485(5)	0.439(9)	0.434(10)	0.195(15)	0.385(12)	0.526(4)	**0.556(1)**
vowels	0.174(11)	**0.804(1)**	0.283(9)	0.100(12)	0.261(10)	0.304(8)	0.609(2)	0.478(4)	0.024(14)	0.322(6)	0.311(7)	0.000(15)	0.035(13)	0.480(3)	0.370(5)
cardio	0.531(3)	0.229(7)	0.109(14)	0.179(10)	0.503(4)	0.160(12)	0.217(8)	0.337(5)	0.535(2)	0.157(13)	0.167(11)	0.082(15)	0.200(9)	0.265(6)	**0.651(1)**
musk	0.495(3)	0.000(15)	0.062(11)	0.302(4)	**1.000(1)**	0.258(6)	0.206(9)	0.237(8)	0.012(14)	0.243(7)	0.268(5)	0.068(10)	0.044(12)	0.037(13)	**1.000(1)**
Waveform	0.040(13)	0.050(10)	0.138(3)	0.090(9)	0.100(7)	0.100(7)	0.150(2)	0.042(12)	0.128(5)	0.118(6)	0.035(14)	0.032(15)	0.131(4)		**0.320(1)**
speech	0.033(6)	**0.147(1)**	0.098(2)	0.049(4)	0.033(6)	0.033(6)	0.016(13)	0.016(13)	0.028(11)	0.061(3)	0.033(6)	0.011(15)	0.021(12)	0.038(5)	0.033(6)
thyroid	0.548(2)	0.237(6)	0.021(14)	0.000(15)	0.387(3)	0.118(9)	0.065(13)	0.269(5)	0.065(12)	0.073(11)	0.139(8)	0.362(4)	0.117(10)	0.222(7)	**0.570(1)**
PageBlocks	0.431(4)	0.400(5)	0.090(14)	0.071(15)	0.494(2)	0.349(11)	0.304(12)	0.486(3)	0.350(10)	0.374(8)	0.352(9)	0.391(6)	0.303(13)	0.376(7)	**0.612(1)**
satimage-2	0.609(4)	0.174(7)	0.029(14)	0.129(10)	**0.927(1)**	0.087(13)	0.159(8)	0.377(5)	0.912(3)	0.096(12)	0.101(11)	0.000(15)	0.201(6)	0.151(9)	0.913(2)
satellite	0.449(7)	0.372(13)	0.262(15)	0.384(8)	0.538(2)	0.376(11)	0.380(9)	0.492(4)	0.482(5)	0.374(12)	0.377(10)	0.493(3)	0.349(14)	0.463(6)	**0.604(1)**
pendigits	**0.359(1)**	0.077(11)	0.019(15)	0.136(5)	0.340(2)	0.083(8)	0.103(6)	0.153(3)	0.085(7)	0.085(7)	0.083(8)	0.071(12)	0.053(14)	0.078(10)	0.147(4)
annthyroid	0.305(3)	**0.313(1)**	0.219(9)	0.073(15)	0.245(7)	0.242(9)	0.229(10)	**0.313(1)**	0.092(14)	0.254(5)	0.247(6)	0.113(12)	0.107(11)	0.243(8)	0.298(4)
mnist	0.180(11)	0.350(5)	N/A	0.167(12)	0.387(3)	0.294(8)	0.264(9)	0.424(2)	0.232(10)	0.297(7)	0.315(6)	0.154(14)	0.155(13)	0.374(4)	**0.449(1)**
mammography	**0.371(1)**	0.190(7)	0.047(12)	0.043(13)	0.269(3)	0.182(8)	0.130(10)	0.221(4)	0.003(14)	0.136(9)	0.202(5)	0.000(15)	0.110(11)	0.198(6)	0.285(2)
magic_gamma	0.460(11)	0.638(3)	0.414(12)	0.122(15)	0.525(8)	0.523(9)	0.483(10)	0.648(2)	0.550(4)	0.547(6)	0.527(7)	0.409(13)	0.404(14)	**0.659(1)**	0.549(5)
campaign	**0.393(1)**	0.296(5)	0.192(10)	0.016(15)	0.367(2)	0.209(9)	0.182(11)	0.328(3)	0.145(14)	0.156(12)	0.222(8)	0.270(6)	0.147(13)	0.255(7)	0.328(3)
shuttle	0.868(3)	0.193(9)	0.076(15)	0.336(6)	**0.956(1)**	0.125(11)	0.112(13)	0.214(8)	0.952(2)	0.097(14)	0.125(11)	0.280(7)	0.503(4)	0.192(10)	0.408(5)
smtp	**0.714(1)**	0.095(7)	0.000(10)	0.000(10)	0.571(3)	0.000(10)	0.000(10)	0.286(4)	0.133(6)	0.000(10)	0.143(5)	0.000(10)	0.038(8)	0.029(9)	0.667(2)
backdoor	0.137(11)	0.318(8)	O/M	O/M	0.548(2)	0.475(3)	0.456(5)	0.436(6)	0.030(12)	0.374(7)	0.475(3)	O/T	0.227(9)	0.185(10)	**0.549(1)**
celeba	0.134(2)	0.000(11)	O/M	O/M	**0.138(1)**	0.021(9)	O/M	0.051(4)	0.028(6)	0.028(6)	0.021(9)	O/T	0.041(5)	0.026(8)	0.069(3)
fraud	0.294(2)	0.046(7)	O/M	O/M	0.076(6)	0.000(9)	O/M	0.093(5)	**0.316(1)**	0.000(9)	0.000(9)	O/T	0.035(8)	0.197(4)	0.271(3)
cover	0.164(2)	0.053(6)	O/M	O/M	0.076(4)	0.034(8)	O/M	0.085(3)	0.013(10)	0.030(9)	0.037(7)	O/T	0.005(11)	0.060(5)	**0.277(1)**
census	0.046(9)	**0.097(1)**	O/M	O/M	0.064(6)	0.087(3)	O/M	0.095(2)	0.025(10)	0.053(8)	O/T	O/T	0.081(4)	0.070(5)	0.054(7)
http	0.278(4)	0.250(5)	O/M	O/M	0.361(2)	0.139(7)	O/M	0.292(3)	0.006(11)	0.019(10)	0.111(8)	O/T	0.156(6)	0.060(9)	**0.875(1)**
average	0.358(3)	0.235(7)	0.126(15)	0.165(12)	0.385(2)	0.229(10)	0.231(9)	0.301(4)	0.246(6)	0.225(11)	0.250(5)	0.164(13)	0.157(14)	0.235(7)	**0.457(1)**

only 0.126, ranking 15th. Notably, several algorithms scored zero on specific datasets, which may be attributed to two main factors: challenging characteristics of datasets (such as wine, smtp, and fraud) and inadequate adaptation of some algorithms (such as FastABOD, SOS, KPCA, COF, and MO-GAAL). These results further highlight the robustness and excellent performance of ADOD in handling diverse types of real datasets.
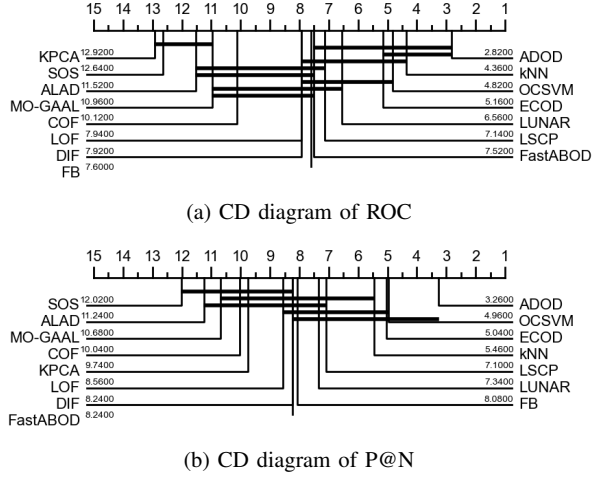
(a) CD diagram of ROC

(b) CD diagram of P@N

Fig. 5: CD diagram illustrating pairwise statistical difference comparison: ADOD outperforms baselines on ROC and P@N, with statistically significant results on P@N.
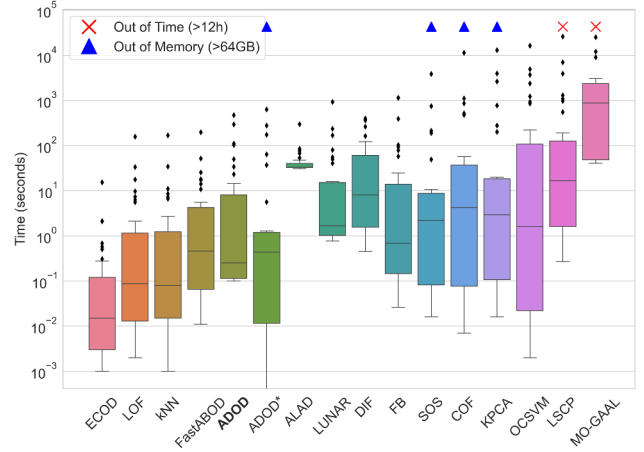

Fig. 6: Runtime comparison of 16 algorithms on 32 datasets. Boxplot illustrates median, quartiles, and outliers of log-scaled runtime for each algorithm, sorted by their average runtime.

Fig. 5 shows the CD between the 15 outlier detection algorithms regarding ROC and P@N scores on 25 datasets following the removal of those containing N/A, O/T, and O/M. ADOD ranked first in 13 out of 25 datasets based on ROC scores and in 10 out of 25 datasets according to P@N scores, achieving the highest average rank across both metrics. ADOD demonstrated statistically significant differences in P@N scores compared to other algorithms, indicated by the lack of connections between ADOD and the other algorithms in the CD diagram. kNN, OCSVM, and ECOD also performed well in both evaluations, usually ranking high. The algorithms with poor performance included KPCA, COF, MO-GAAL, ALAD, and SOS, which ranked low on both metrics.

E. Runtime Analysis

As shown in Fig. 6, ECOD exhibited the shortest runtime. In all experiments, ADOD refers to the optimized version with NNS. ADOD*, the original algorithm without NNS optimization, ranked 6th with an average time of 42s but suffered from out-of-memory errors on datasets exceeding 200k samples. In contrast, the optimized ADOD ranked 5th with an average time of 33s, maintaining a low and stable runtime across most datasets and demonstrating its efficiency and reliability in handling large datasets. For example, ADOD (49s) was approximately 12.8 times faster than ADOD* (629s) on the celeba dataset (114k). SOS, COF, and KPCA encountered out-of-memory errors when processing the backdoor (87k) and celeba (114k) datasets. LSCP (average time: 1288s) and MO-GAAL (average time: 2628s) ran more than 12 hours on large or high-dimensional datasets, revealing performance bottlenecks in complex tasks.

Notably, the performance difference between ADOD and ADOD* was negligible. Comparing their performance on each dataset (ADOD - ADOD*), we obtained an average difference of 0.0035 in ROC with a standard deviation of 0.0171 and an average difference of -0.0054 in P@N with a standard deviation of 0.0394. Thus, NNS optimization improves efficiency and reduces memory overhead while remaining consistent in terms of performance compared with the original algorithm.

F. Visualization on Real Datasets

The first column of Fig. 7 shows the visualization of the musk dataset. ADOD achieved ROC and P@N scores of 1.0, indicating that it accurately identified all the outliers. The visualization results show that the outliers detected by ADOD perfectly matched the distribution of the true labels and were located in separate clusters far from the main clusters. The second column of Fig. 7 shows the visualization of the magic_gamma dataset. ADOD achieved ROC and P@N scores of 0.728 and 0.549, respectively. In the ground truth, outliers were scattered throughout the data distribution, whereas the outliers detected by ADOD were primarily located at the edges of the data distribution. Compared with the outliers in the ground truth, predicted outliers appeared to better represent the characteristics of outliers. The third column of Fig. 7 shows the visualization of the pima dataset. ADOD achieved ROC and P@N scores of 0.735 and 0.556, respectively. Compared with the outliers in the ground truth, the outliers predicted by ADOD better reflected the separation from the inliers, as indicated by two completely red clusters of outliers.

This is an interesting discovery because we often rely on evaluation metrics to assess the performance of outlier detection algorithms. The visualization results revealed certain cases where the results were more valuable even if the metric scores were not high. Through visualization, we can intuitively observe that the outliers detected by ADOD on certain datasets are more consistent with human intuition and are typically located at the edges of the data distributions or form separate small clusters.
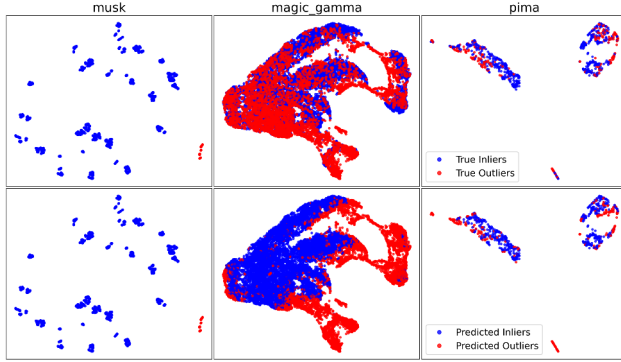
Fig. 7: Visualization of musk, magic_gamma, and pima datasets using UMAP [38], comparing ground truth labels (first row) and ADOD predictions (second row) with inliers in blue and outliers in red.

V. Conclusion

In this paper, we proposed a novel unsupervised Adaptive Density Outlier Detection (ADOD) algorithm for handling data with varying densities. The primary innovations of ADOD included adaptive neighborhood boundaries and density consistency scoring. In addition, we optimized ADOD using nearest neighbor search to improve its efficiency and reduce memory usage. The experimental results demonstrated that ADOD significantly outperformed 14 other outlier detection algorithms from different categories in key performance metrics, such as ROC, P@N, AP and execution time. In the parameter sensitivity analysis, we fixed the values of the optional parameters and demonstrated their robust performance, which enabled ADOD to be regarded as a parameter-free method in practical applications, eliminating the need for users to tune the parameters manually. ADOD is adaptable and scalable while maintaining high accuracy, providing an effective outlier detection solution and expanding its application potential in real-time data processing.

References

[1] Y. Almardeny, N. Boujnah, and F. Cleary, "A novel outlier detection method for multivariate data," *TKDE*, vol. 34, no. 9, pp. 4052–4062, 2022.
[2] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller, "A unifying review of deep and shallow anomaly detection," *Proc. IEEE*, vol. 109, no. 5, pp. 756–795, 2021.
[3] C. C. Aggarwal and C. C. Aggarwal, *An introduction to outlier analysis*. Springer, 2017.
[4] G. Steinbuss and K. Böhm, "Benchmarking unsupervised outlier detection with realistic synthetic data," *TKDD*, vol. 15, no. 4, apr 2021.
[5] X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu, "A comprehensive survey on graph anomaly detection with deep learning," *TKDE*, vol. 35, no. 12, pp. 12 012–12 038, 2023.
[6] D. Samariya and A. Thakkar, "A comprehensive survey of anomaly detection algorithms," *ADS*, vol. 10, no. 3, pp. 829–850, 2023.
[7] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *SIGMOD*, 2000, p. 427–438.
[8] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *SIGMOD*, 2000, p. 93–104.
[9] O. Alghushairy, R. Alsini, T. Soule, and X. Ma, "A review of local outlier factor algorithms for outlier detection in big data streams," *Big Data Cogn. Comput.*, vol. 5, no. 1, 2021.
[10] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *JMLR*, vol. 9, no. 86, pp. 2579–2605, 2008.
[11] H.-P. Kriegel, M. Schubert, and A. Zimek, "Angle-based outlier detection in high-dimensional data," in *KDD*, 2008, p. 444–452.
[12] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *TPAMI*, vol. 33, no. 1, pp. 117–128, 2011.
[13] J. Janssens, F. Huszár, E. Postma, and H. van den Herik, "Stochastic outlier selection," *Technical Report TiCC TR 2012-001*, 2012.
[14] Z. Li, Y. Zhao, N. Botta, C. Ionescu, and X. Hu, "COPOD: Copula-based outlier detection," in *ICDM*, 2020, pp. 1118–1123.
[15] Z. Li, Y. Zhao, X. Hu, N. Botta, C. Ionescu, and G. H. Chen, "ECOD: Unsupervised outlier detection using empirical cumulative distribution functions," *TKDE*, vol. 35, no. 12, pp. 12 181–12 193, 2023.
[16] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, "A novel anomaly detection scheme based on principal component classifier," in *ICDMW*, 2003, pp. 172–179.
[17] H. Hoffmann, "Kernel pca for novelty detection," *PR*, vol. 40, no. 3, pp. 863–874, 2007.
[18] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, no. 7, p. 1443–1471, 2001.
[19] A. Lazarevic and V. Kumar, "Feature bagging for outlier detection," in *KDD*, 2005, p. 157–166.
[20] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *ICDM*, 2008, pp. 413–422.
[21] H. Xu, G. Pang, Y. Wang, and Y. Wang, "Deep isolation forest for anomaly detection," *TKDE*, vol. 35, no. 12, pp. 12 591–12 604, 2023.
[22] Y. Zhao, Z. Nasrullah, M. K. Hryniewicki, and Z. Li, "LSCP: locally selective combination in parallel outlier ensembles," in *SDM*, 2019, pp. 585–593.
[23] Y. Liu, Z. Li, C. Zhou, Y. Jiang, J. Sun, M. Wang, and X. He, "Generative adversarial active learning for unsupervised outlier detection," *TKDE*, vol. 32, no. 8, pp. 1517–1528, 2020.
[24] H. Zenati, M. Romain, C.-S. Foo, B. Lecouat, and V. Chandrasekhar, "Adversarially learned anomaly detection," in *ICDM*, 2018, pp. 727–736.
[25] A. Goodge, B. Hooi, S. K. Ng, and W. S. Ng, "LUNAR: Unifying local outlier detection methods via graph neural networks," in *AAAI*, 2022.
[26] F. Angiulli and C. Pizzuti, "Fast outlier detection in high dimensional spaces," in *PKDD*, 2002, pp. 15–27.
[27] J. Tang, Z. Chen, A. W.-c. Fu, and D. W. Cheung, "Enhancing effectiveness of outlier detections for low density patterns," in *PAKDD*, 2002, pp. 535–548.
[28] K. P. Murphy, *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.
[29] L. Qian, C. Plant, and C. Böhm, "Density-based clustering for adaptive density variation," in *ICDM*, 2021, pp. 1282–1287.
[30] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. John Wiley & Sons, 2012.
[31] T. İnkaya, "Consensus similarity graph construction for clustering," *Pattern Anal. Appl.*, vol. 26, no. 2, p. 703–733, 2023.
[32] W. Gilchrist, *Statistical modelling with quantile functions*. Chapman and Hall/CRC, 2000.
[33] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE TBD*, vol. 7, no. 3, pp. 535–547, 2019.
[34] S. Rayana, "ODDS library," 2016. [Online]. Available: https://odds.cs.stonybrook.edu
[35] S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao, "Adbench: Anomaly detection benchmark," in *NeurIPS*, vol. 35, 2022, pp. 32 142–32 159.
[36] Y. Zhao, Z. Nasrullah, and Z. Li, "Pyod: A python toolbox for scalable outlier detection," *JMLR*, vol. 20, no. 96, pp. 1–7, 2019.
[37] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Min. Knowl. Discov.*, vol. 33, no. 4, pp. 917–963, 2019.
[38] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," *ArXiv e-prints*, 2018.