Journal of Computer Science & Technology

Indexed in: SCIE, EI, INSPEC, CBST, DBLP, etc.

Sponsored by: ICT, CAS & CCF

JCST Papers

Only for academic and non-commercial use

Thanks for reading!

[Survey]

[Computer Architecture and Systems]

[Artificial Intelligence and Pattern Recognition]

[Computer Graphics and Multimedia]

[Data Management and Data Mining]

[Software Systems]

[Computer Networks and Distributed Computing]

[Theory and Algorithms]

[Emerging Areas]

JCST WeChat Subscription Account

JCST URL: https://jcst.ict.ac.cn

SPRINGER URL: https://www.springer.com/journal/11390

E-mail: jcst@ict.ac.cn

Online Submission: https://mc03.manuscriptcentral.com/jcst

Twitter: JCST_Journal

LinkedIn: Journal of Computer Science and Technology

Liu SZ, Ma S, Chen HQ *et al.* Combining *K*NN with AutoEncoder for outlier detection. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 39(5): 1153-1166 Sept. 2024. DOI: 10.1007/s11390-023-2403-y

Combining *K*NN with AutoEncoder for Outlier Detection

Shu-Zheng Liu[1] (刘叔正), Shuai Ma[1,*] (马　帅), *Senior Member, CCF, IEEE, Member, ACM*
Han-Qing Chen[1] (陈瀚清), Li-Zhen Cui[2,3] (崔立真), *Senior Member, CCF*, and Jie Ding[4] (丁　杰)

[1] *State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China*

[2] *School of Software, Shandong University, Jinan 250100, China*

[3] *Joint SDU-NTU Centre for Artificial Intelligence Research, Shandong University, Jinan 250100, China*

[4] *School of Computer Science, Jiangsu University of Science and Technology, Zhenjiang 212003, China*

E-mail: liusz@buaa.edu.cn; mashuai@buaa.edu.cn; chenhanqing@buaa.edu.cn; clz@sdu.edu.cn; jding2022@just.edu.cn

Received April 12, 2022; accepted September 13, 2023.

Abstract　K-nearest neighbor (KNN) is one of the most fundamental methods for unsupervised outlier detection because of its various advantages, e.g., ease of use and relatively high accuracy. Currently, most data analytic tasks need to deal with high-dimensional data, and the KNN-based methods often fail due to "the curse of dimensionality". AutoEncoder-based methods have recently been introduced to use reconstruction errors for outlier detection on high-dimensional data, but the direct use of AutoEncoder typically does not preserve the data proximity relationships well for outlier detection. In this study, we propose to combine KNN with AutoEncoder for outlier detection. First, we propose the Nearest Neighbor AutoEncoder (NNAE) by persevering the original data proximity in a much lower dimension that is more suitable for performing KNN. Second, we propose the K-nearest reconstruction neighbors (KNRNs) by incorporating the reconstruction errors of NNAE with the K-distances of KNN to detect outliers. Third, we develop a method to automatically choose better parameters for optimizing the structure of NNAE. Finally, using five real-world datasets, we experimentally show that our proposed approach NNAE+KNRN is much better than existing methods, i.e., KNN, Isolation Forest, a traditional AutoEncoder using reconstruction errors (AutoEncoder-RE), and Robust AutoEncoder.

Keywords　outlier detection, AutoEncoder, K-nearest neighbor (KNN), unsupervised learning

1　Introduction

Outlier detection refers to the problem of finding patterns in data that do not confirm to the expected behaviors, and is a fundamental data mining task for various data analytics applications, such as fraud detection in credit-card transactions, intrusion detection in computer networks, and disease discovering in medical diagnoses[1, 2]. As data labels are often unavailable or too expensive to acquire when the data scale is large or outlier patterns are complicated, unsupervised outlier detection has become a popular and practical solution due to its ease of deployment[1, 3–26].

Although outliers can be defined differently in different tasks, the common characteristic of an outlier is that it deviates so much from the other observations as to arise suspicions that may be generated by a different mechanism[1, 27], which typically presents an interesting event implied in the data. There are several common unsupervised techniques to distinguish outliers from inliers[2], such as association rule mining based methods[28, 29], cluster-based methods[5, 6], nearest neighbor based methods[8, 30, 31], information theoretic methods[32], spectral methods[9], and ensemble methods[24, 25], among which K-nearest neighbor (KNN) is one of the most common methods due to its various advantages[1]. Given the hypothesis that outliers are far away from the other data points while inliers stack closely, KNN takes the K-distance of a point (i.e., the distance to its K-th nearest

Regular Paper

This work was supported in part by the National Natural Science Foundation of China under Grant Nos. 61925203 and U22B2021.

*Corresponding Author

neighbor) as the outlier score that is expected to be large when the point is more likely an outlier.

The merits of KNN lie in that: 1) it obtains a more detailed granularity of the analysis to distinguish outliers from noises, which is much better than the other methods like cluster-based methods[1] ; 2) it does not require any assumptions about the generative distribution of data; 3) by providing an appropriate distance measure, its implementation is straightforward[2]. These make KNN effective for various data analytic tasks. However, KNN still has troubles when it faces with high-dimensional data due to "the curse of dimensionality"[33], and its performance heavily depends on the choice of K[1].

Several techniques have been proposed for handling the curse of dimensionality, such as principal components analysis[34, 35], multi-dimensional scaling[36], matrix factorization[37], factorization machines[11], word embedding[16], and network embedding[10]. AutoEncoder is another alternative to obtain a representation of the data in a low-dimensional space for unsupervised outlier detection[3, 7, 12–14, 18–23, 38–41]. It is commonly used for embedding complex patterns in high-dimensional data, and can embed almost any data distribution well with appropriate settings. Variations of AutoEncoder have also been proposed to learn a better representation for outlier detection, such as sparse AutoEncoder[12], denoising AutoEncoder[41], robust AutoEncoder[3], variational AutoEncoder[7, 22], memory-augmented AutoEncoder[18] and adversarial AutoEncoder[19, 20]. The underlying rationale behind AutoEncoder for outlier detection roots in commonly agreed facts that outliers are the minority and inliers are the majority, and that AutoEncoder learns a better representation for inliers and a worse one for outliers with a proper training. Thus, reconstruction errors of the representation for inliers are typically smaller, while those for outliers are larger, which can be used to distinguish outliers from inliers.

On the one hand, KNN-based methods have the weakness for handling high-dimensional data, while AutoEncoder-based methods essentially work for this situation. On the other hand, AutoEncoder-based methods typically do not preserve well data proximity relationships for outlier detection as shown by Fig.1, while KNN-based methods prove the usefulness of data proximity for outlier detection. Thus, a natural question arises: is there a way to seamlessly combine KNN and AutoEncoder to derive a stronger approach for outlier detection? If "yes", there remain

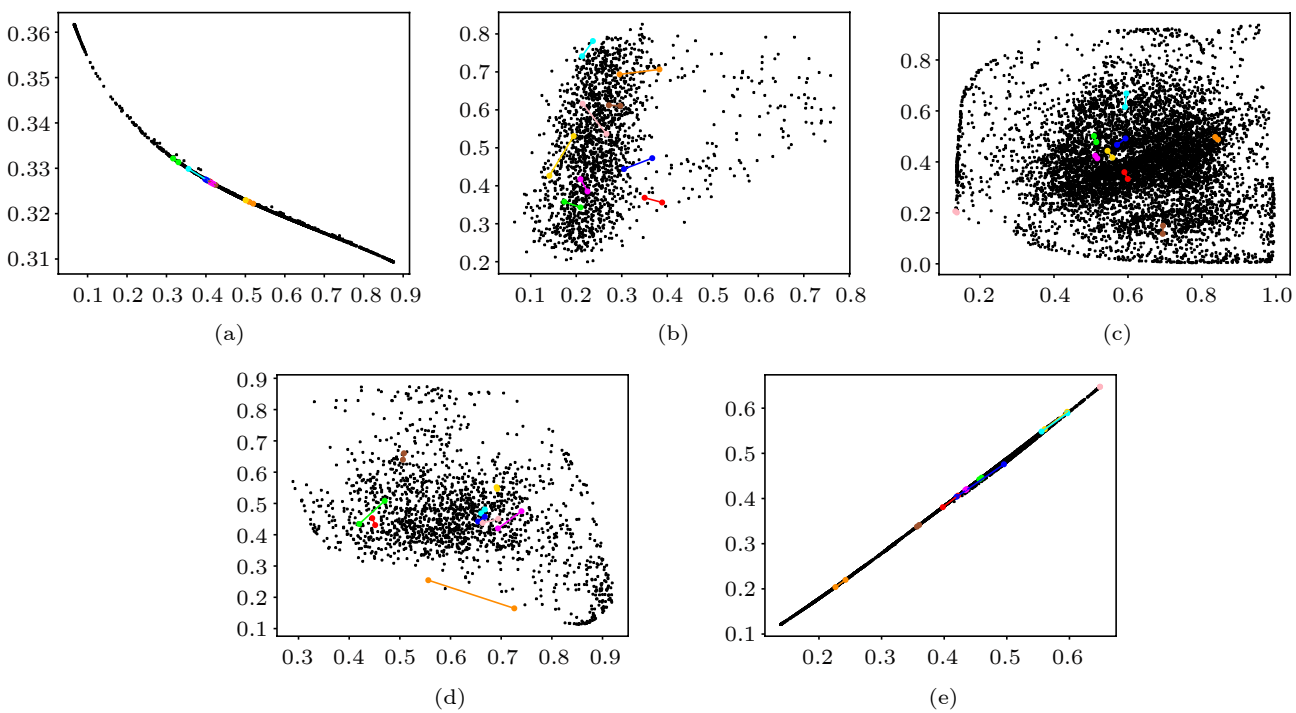

Fig.1. Example of nearest neighbor proximity in the original and embedding spaces with AutoEncoder on five real-world datasets: (a) Cardio, (b) Waveform_noise, (c) Fashion-MNIST, (d) USPS, (e) STL 10 (see details in Section 4). We plot the data points in a two-dimensional latent space, embedded with a fully-connected traditional AutoEncoder. Here colored data point pairs are the nearest neighbors in the original data space, while they are obviously not in the latent space. Black data points can be simply ignored, as they are not for consideration. This shows that the direct use of AutoEncoder does not preserve well the data proximity relationships for outlier detection.

two issues to resolve: 1) how to preserve the data proximity relationships like KNN during the embedding of AutoEncoder, and 2) how to combine reconstruction errors of AutoEncoder with K-nearest neighbor distances of KNN to design the outlier score.

To this end, we present an effective approach, referred to as NNAE+KNRN, for outlier detection by combining KNN with AutoEncoder, i.e., a nearest neighbor AutoEncoder (NNAE) and a K-nearest reconstruction neighbors (KNRN) outlier score. NNAE employs the non-linear dimension reduction deep network AutoEncoder to alleviate the curse of dimensionality, to empower the ability of obtaining the complex non-linear data patterns, and to preserve the data proximity with a newly designed loss function dedicated to KNN, instead of using the commonly used sparse regularization term on AutoEncoder; KNRN combines the K-nearest neighbor distances with the reconstruction errors generated by AutoEncoder when detecting outliers, instead of simply using the K-th distances.

The main contributions of the paper are as follows.

1) To deal with the curse of dimensionality and to alleviate the K choice problem, we propose NNAE to learn an embedding in a low-dimensional space by preserving the data proximity with a newly designed loss function dedicated to KNN, instead of using the commonly used sparse regularization, which also improves the robustness with respect to K.

2) To further alleviate the K choice problem, we develop a new outlier score KNRN, which combines the K-distance used in KNN-based methods and the reconstruction errors used in AutoEncoder-based methods by treating reconstruction errors as the confidence of embedding.

3) To improve the usability of our approach, we also develop an indicator Z for choosing better structure parameters of NNAE, by making use of its reconstruction errors, instead of setting by experienced users.

4) Using five real-world datasets, we experimentally show that our approach NNAE+KNRN is much better than KNN[1], Isolation Forest[4], a traditional AutoEncoder using reconstruction errors (AutoEncoder-RE)[38], and Robust AutoEncoder[3].

This paper is organized as follows. Section 2 introduces related work. Section 3 presents our approach NNAE+KNRN in detail, followed by experimental study in Section 4 and conclusions in Section 5.

2 Related Work

An outlier is "an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism"[27]. There are typically three types of outliers: 1) a point outlier is a data point that can be considered an outlier with respect to the other data points, 2) contextual outliers can only be considered as outliers in a specific context, and 3) collective outliers are those groups of data considered as outliers with respect to the entire data[2]. Two types of outcomes can be generated for outlier detection algorithms: outlier scores that indicate the extent of outliers, and labels that indicate the data instance is an outlier or not[1]. By the aspect of training, outlier detection techniques can be categorized into three types: 1) supervised outlier detection, which needs training data having both outlier and inlier labels; 2) semi-supervised outlier detection, which needs inliers and trains upon inliers, or assumes that the training data only has inliers; 3) unsupervised outlier detection, which does not need training data at all, assuming that inliers are far more frequent than outliers[2]. We focus on unsupervised outlier detection in this study, as it does not rely on training data, and is easy to deploy.

Outlier Detection with KNN. Unsupervised outlier detection methods can be categorized into various types, e.g., association rule mining based methods, clustering-based methods, nearest neighbor based methods, information theoretic methods, and spectral methods[2], among which KNN is one of the most common method. Given the hypothesis that outliers are far away from the other points, KNN detects outliers as follows. For each data record (or data point) in a dataset, its K-nearest neighbors are found, and then KNN calculates its K-distance (the distance to its K-th nearest neighbors) as the outlier score. There are basically three types of KNN variations. 1) The exact K-nearest neighbor[30] finds the exact K-th nearest point of each data point, and takes the distance of the point itself to its K-th nearest point as the outlier score that is typically large when the point is in a point sparse region, which means the point is more likely an outlier. 2) The average K-nearest neighbor[8] uses the average distance to all its K nearest neighbors, which increases the robustness regarding to the parameter K. However, it slightly decreas-

es the accuracy. 3) The reverse nearest neighbor (outlier detection using in-degree number (ODIN)) [31] uses the number of reverse K-nearest neighbors to define the outlier scores, where a data point p is a reverse K-nearest neighbor of another data point q if and only if q is among the K-nearest neighbors of p. KNN is straightforward to implement, and effective on various tasks[42, 43]. However, it performs poorly on high-dimensional data due to the curse of dimensionality, in which situation the noises and irrelevant features make the detection extremely hard. Moreover, the performance of KNN heavily depends on the choice of K, which is also hard to determine in an unsupervised context[44]. These limitations hinder the applicability of KNN in practice.

Outlier Detection with Embedding. Various embedding methods for outlier detection have been developed, such as multidimensional scaling[36], matrix factorization[37], factorization machines[11], word embedding[16], and network embedding[10]. As a tool designed for non-linear embedding, AutoEncoder has wide applications for outlier detection[3, 7, 12–14, 18–23, 38–41]. There are two types of applications. 1) As a detector, AutoEncoder obtains larger reconstruction errors for outliers while smaller ones for inliers. Under the hypothesis that outliers are much less than inliers, AutoEncoder can learn a representation fitting inliers better. Using the deviations of reconstruction errors, AutoEncoder can distinguish outliers from inliers[3, 7, 12, 14, 18–23, 41]. Among them, [19, 20, 23] adopt adversarial training into AutoEncoder. [23] proposes an architecture composed of two AutoEncoders AE_1 and AE_2 sharing the same encoder. The model is trained in two phases, including usual AutoEncoder training and extra adversarial training, when AE_1 is trained to "fool" AE_2 and AE_2 is trained to distinguish the real data from the reconstructed data coming from AE_1. [19, 20] use AutoEncoder as a generator, and append a discriminator after the decoder to differentiate the real (input) and fake (reconstructed) samples. In contrast to [23], AutoEncoder and the discriminator in [19, 20] are trained simultaneously. 2) AutoEncoder can also work as an embedding tool for outlier detection. [35] uses AutoEncoder to reduce the dimension of the data. However, embedding and detecting are two independent procedures.

KNN-based methods have the weakness for handling high-dimensional data, while AutoEncoder-based methods essentially work for outlier detection in this situation. AutoEncoder-based methods do not preserve well the data proximity relationships for outlier detection, while KNN-based methods have proven the usefulness of data proximity for outlier detection. To the best of our knowledge, this study is the first try to combine KNN and AutoEncoder to obtain a stronger approach to outlier detection on high-dimensional data.

3 Combining KNN with AutoEncoder

In this section, we first formalize the problem to be investigated, and then introduce our approach to outlier detection by seamlessly combining KNN and AutoEncoder, which learns an embedding with NNAE that empowers the ability of obtaining the complex non-linear data patterns and preserving the data proximity, and detects outliers with KNRN that combines the K-nearest neighbor distances with the reconstruction errors generated by NNAE.

Problem Statement. We aim to detect outliers for high-dimensional data points, where a point $\boldsymbol{x}$ is a d-dimensional vector, i.e., $\boldsymbol{x} \in \mathbb{R}^d$. A point $\boldsymbol{x}$ is an outlier if its anomaly score $S(\boldsymbol{x}) > \lambda$, where S is a function to be defined in Subsection 3.3 and λ is a threshold. Given a set of data points $D = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, where n is the number of data points, the outlier detection problem is to find all the outliers in D.

3.1 Overview

We first introduce the design principle and the overall structure of our approach for outlier detection.

Design Principle. Detecting outliers is hard for KNN in the original space for high-dimensional data, since the original data not only has features irrelevant to outlier detection, but also contains noises that could overwhelm true information. Hence, we adopt to learn an embedding of the data in a low dimension space with AutoEncoder. However, existing methods using AutoEncoder do not preserve well the data proximity relationships for outlier detection although they can handle noises well. Thus, we propose to learn an embedding with NNAE to preserve the data proximity with a newly designed loss function. Moreover, both reconstruction errors of the learned embedding and the K-nearest neighbor distances show their usefulness for AutoEncoder and KNN-based outlier detection methods, respectively. This is the reason why we propose KNRN to combine K-nearest neighbor distances with reconstruction errors to define outlier scores, where reconstruction errors are considered as the confidence probability for the embedding of a

data point such that higher reconstruction errors enlarge outlier scores and lower reconstruction errors reduce outlier scores, respectively. In this way, KNN and AutoEncoder are seamlessly unified to enjoy the mutual benefits from each other.

Overall Framework. The overall framework of our approach is illustrated in Fig.2. First, it finds the K-nearest neighbors for all data records with a K-D tree index. Second, it trains NNAE, which requires the K-nearest neighbors for each record, to learn the proximity focused embedding and the reconstruction error of each data record. Finally, it uses KNRN to detect outliers. Our approach aims to establish a close collaboration between the embedding component and the detector component, to feed the proximity-based detector with a better perseveration of the original data proximity.

3.2 Nearest Neighbor AutoEncoder

We introduce NNAE in detail, whose structure is shown in Fig.3, where the encoder and the decoder are symmetric and have different parameters, and the structure and parameters for points and neighbor pairs are the same.

Given a set D of data points (or records) handled as a matrix $\boldsymbol{X}$, the n-layer encoder is defined as,

$$E(\boldsymbol{X}) = f_n(f_{n-1}(\dots f_1(\boldsymbol{X}))),$$

such that the i-th layer of the encoder transformation $f_i(\boldsymbol{X}_{i-1})$ is

$$f_i(\boldsymbol{X}_{i-1}) = \begin{cases} \text{Sigmoid}(\boldsymbol{X}_{i-1} + \boldsymbol{B}), & i \neq n, \\ \boldsymbol{W} \cdot \boldsymbol{X}_{i-1} + \boldsymbol{B}, & i = n, \end{cases}$$

where $\boldsymbol{X}_{i-1} = f_{i-1}(\dots f_1(\boldsymbol{X}))$ with $n \geqslant i \geqslant 1$ and $\boldsymbol{X}_0 = \boldsymbol{X}$, and $\boldsymbol{W}$ and $\boldsymbol{B}$ are the weight matrix and bias vector of the i-th layer, respectively.

The n-layer decoder is defined similarly as,

$$D(E(\boldsymbol{X})) = g_n(g_{n-1}(\dots g_1(E(\boldsymbol{X})))),$$

such that the i-th layer of the decoder transformation $g_i(\boldsymbol{Y}_{i-1})$ is

$$g_i(\boldsymbol{Y}_{i-1}) = \text{Sigmoid}(\text{W} \cdot \text{Y}_{\text{i}-1} + \text{B}), \quad 1 \leqslant \text{i} \leqslant \text{n},$$

where $\boldsymbol{Y}_{i-1} = g_{i-1}(\dots g_1(E(\boldsymbol{X})))$, and $\boldsymbol{W}$ and $\boldsymbol{B}$ are

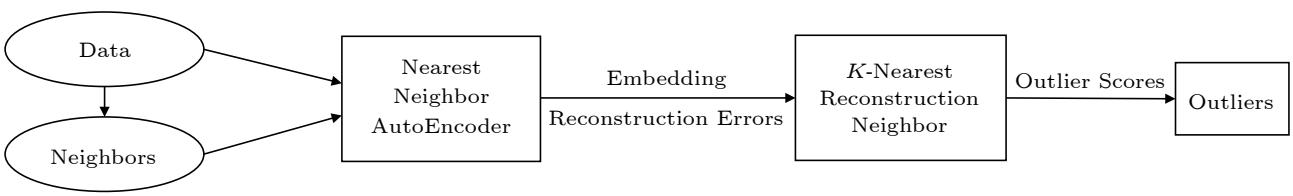

Fig.2. Overall framework of proposed approach NNAE + KNRN.

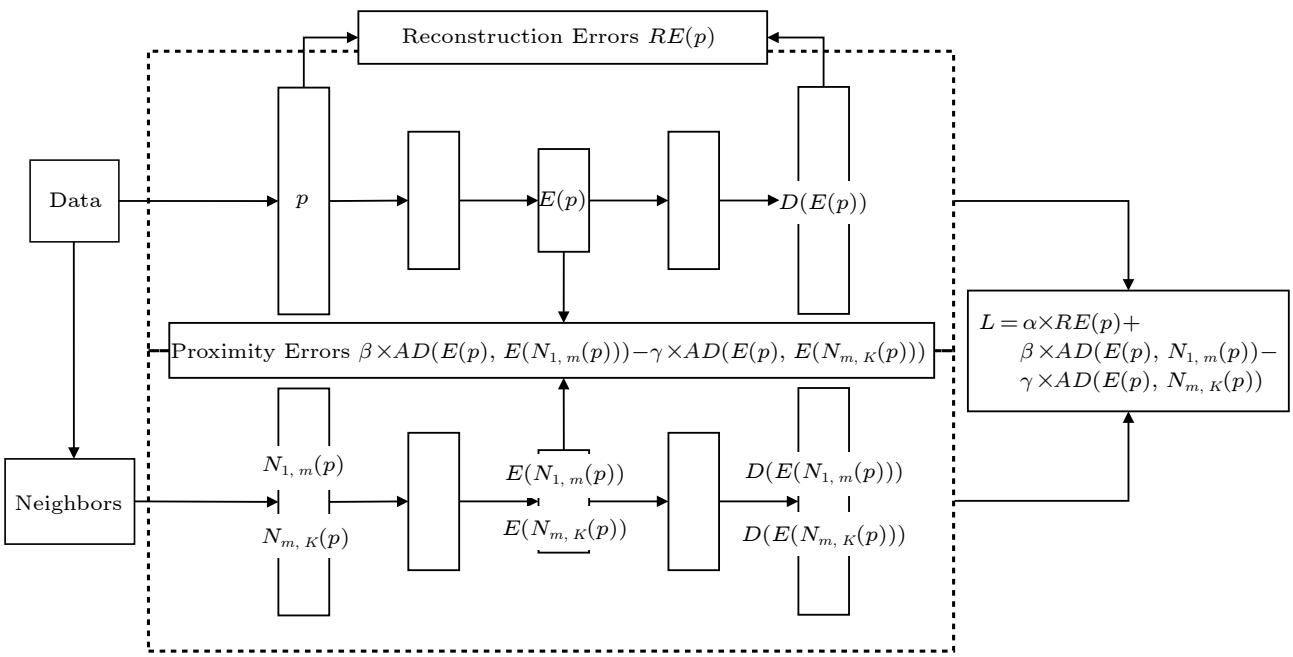

Fig.3. Structure of NNAE.

the same as the encoder.

Based on the traditional fully-connected layer AutoEncoder[38], we introduce two new terms for the loss function of NNAE. We define two sets of neighbors for each data point p: set $N_{1,m}(p)$ consists of m points from its 1-nearest to m-nearest neighbors in the original data space, and set $N_{m,K}(p)$ consists of m points from its $(K-m+1)$-nearest to K-nearest neighbors in the original data space, where $m<K$ is a hyperparameter indicating the considered number of nearest neighbors, and K is the number of nearest neighbors for KNRN. 1) The first term $AD(p,N_{1,m}(p))$ is the average Euclidian distance in the latent embedding space between point p and the set of points in $N_{1,m}(p)$. We expect that a point remains close to its first m-nearest neighbors in the latent space, which is done by minimizing the first term in the loss function. 2) The second term $AD(p,N_{m,K}(p))$ is the average Euclidian distance in the latent embedding space between point p and the set of points in $N_{m,K}(p)$. To facilitate the detection of outliers with the K-distance, we expect to enlarge the distances of a point to its last m-nearest neighbors in its K-nearest neighbors, which is achieved by minimizing the negative of the second term in the loss function.

The loss function is finally defined as follows.

$$L=\frac{1}{|X|}\sum_{p\in X}(\alpha\times RE(p)+\beta\times AD(p,N_{1,m}(p))-\gamma\times AD(p,N_{m,K}(p))),$$

where $RE(p)=(p-D(E(p)))^2$ is the reconstruction error of a data point p, and α, β, and γ are fixed constants (set to 200, 200, and 10 by default, respectively). That is, both reconstruction errors and nearest neighbors are considered in the loss function.

NNAE is implemented in the following way. Instead of acquiring a batch of data records only, NNAE acquires a batch of data records together with their m-nearest neighbors and their $(K-m+1)$-nearest to K-nearest neighbors. After the greedy layer-by-layer pre-training[35] and iterative global training, NNAE generates two kinds of information for each data point p in D: embedding $E(p)$ and reconstruction error $RE(p)$. In this way, NNAE not only learns an embedding in a low-dimensional space with the better preserved data proximity, but also improves the robustness with respect to the key parameter K.

3.3 *K*-Nearest Reconstruction Neighbors

We next introduce how to detect outliers with KNRN that combines the K-nearest neighbor distances with the reconstruction errors of NNAE, instead of using the K-distances only as outlier scores.

As observed by existing AutoEncoder-based outlier detection[3, 38], AutoEncoder is capable of fitting inliers better while outliers worse, and, hence, the reconstruction errors contain valuable information for identifying outliers. So does the K-distance of KNN. These motivate us to propose KNRN as the outlier scores. Let $KD(p)$ be the distance between point p and its K-nearest neighbor, $RE_{\max}$ and $RE_{\min}$ be the maximum and minimum reconstruction errors, respectively, among the reconstruction errors $RE(p)$ of all data points p. The KNRN outlier score of a data point p is defined as follows:

$$KD(p)\times \mathrm{e}^{(RE(p)-RE_{\min})/(RE_{\max}-RE_{\min})}.$$

With the help of the reconstruction error of a neighbor, which approximately represents the confidence of the K-distance, KNRN reduces the dependence on parameter K. Hence, the combination of the K-distance with the reconstruction errors also further alleviates the aforementioned K choice problem of KNN.

3.4 Improving Model Usability

We finally present a method to improve the usability of our model. We adopt the structure parameters L and θ from [13], where L stands for the number of layers in the encoder and θ stands for the compression rate between two adjacent layers in the encoder, and the decoder is symmetric to the encoder.

We develop a method to help the user to decide L and θ to reach a reasonably good performance of NNAE for unsupervised outlier detection. Let N_l be the set of data points whose reconstruction errors belong to the 50% lowest ones, and N_h be the set of points whose reconstruction errors belong to the 5% highest ones. We define a structure indicator Z to represent the ability of NNAE to distinguish inliers and outliers as follows:

$$Z=\left(\frac{1}{|N_l|}\sum_{p\in N_l}RE(p)\right)\Big/\left(\frac{1}{|N_h|}\sum_{p\in N_h}RE(p)\right).$$

It is expected that NNAE learns an embedding fitting inliers better while outliers worse. Hence, the smaller the reconstruction errors of points in N_l, the

better, and the lager the reconstruction errors of points in N_h, the better. That is, a smaller Z means a better embedding from the aspect of outlier detection, and we can use Z to choose structure parameters L and θ. In practice, one needs to process multiple combinations of L and θ, and choose the setting of L and θ with the smallest Z. As will be seen in the experiment study, this method is always better than random choices, and can find parameters with a good performance.

Remarks. Our proposed approach NNAE+KNRN has several unique characteristics. First, it utilizes the power of AutoEncoder to deal with the curse of dimensionality. Second, it provides the ability to partially preserve data proximity for outlier detection with a newly designed loss function. Third, it detects outliers with KNRN that combines the K-nearest neighbor distances with the reconstruction errors of NNAE to alleviate the K choice problem of KNN and to improve performance. Fourth, a structure indicator is developed to improve the usability by utilizing reconstruction errors.

4 Experimental Study

In this section, we conduct an experimental study of our unsupervised outlier detection approach NNAE+KNRN. Using five real-world datasets, we conduct four sets of experiments to evaluate: 1) the overall accuracy of NNAE+KNRN compared with KNN[1], Isolation Forest[4], AutoEncoder-RE[38], Robust AutoEncoder (RAE)[3], and variations AE+KNN and NNAE+KNN of combining KNN with AutoEncoder, 2) the robustness of NNAE+KNRN with respect to K, compared with KNN, AE+KNN, and NNAE+KNN using parameter K, 3) the impacts of the number of focused nearest neighbors m on NNAE, and 4) the effectiveness of structure indicator Z to choose appropriate structure parameters L and θ for NNAE+KNRN. Codes of our methods and datasets are available at GitHub①.

4.1 Experimental Settings

4.1.1 Datasets

We use five real-world datasets (summarized in Table 1) to test our unsupervised outlier detection approach, and we use the min-max scalar to convert all their data values into the range of $[0, 1]$.

Table 1. Statistics of Real-World Datasets

Dataset	Number of Samples	Number of Features	Outlier Rate (%)
Cardio②	1 831	22	10.6
Waveform_noise③	1 835	40	9.9
Fashion-MNIST[45]	7 777	784	10.0
USPS④	1 725	256	10.0
STL10[46]	3 000	4 096	10.0

1) *Cardio.* It is a cardiotocography dataset from UCI Machine Learning Repository②, which contains the measurements of fetal heart rate signals. There are three classes of data: normal, suspect, and pathological. The suspect class is discarded, and the records belonging to the normal class are marked as inliers, while the ones belonging to the pathological class are marked as outliers.

2) *Waveform_noise.* It is a waveform database generator dataset from UCI Machine Learning Repository③, which contains three classes of waves that all contain noises, referred to as $W0$, $W1$, and $W2$, respectively, where $W0$ waves are marked as inliers, and $W1$ and $W2$ waves are sampled equally as outliers.

3) *Fashion-MNIST.* It is a grayscale image dataset[45], consisting of images of 28×28 pixels. The images are associated with labels from ten classes, including T-shirts/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot. We modify this dataset to adapt the need for outlier detections, where T-shirts are the majority as inliers, and all the other nine categories of images are sampled equally to form outliers.

4) *USPS④.* It is a 16×16 grayscale pixel handwritten image dataset from Kaggle, which contains numbers from 0 to 9. Similar to Fashion-MNIST, we modify this dataset such that the images of number 0 is the majority as inliers, and all the other nine categories of images are sampled equally to form outliers.

5) *STL10.* It is an image dataset[46] containing ten classes: airplane, bird, car, cat, deer, dog, horse, monkey, ship, and truck. We categorize them into animals (bird, cat, deer, dog, horse, and monkey) and ve-

①https://github.com/CodesandDataL/NNAE-KNRN, Mar. 2024.
②http://archive.ics.uci.edu/dataset/193/cardiotocography, Mar. 2024.
③http://archive.ics.uci.edu/dataset/108/waveform+database+generator+version+2, Mar. 2024.
④https://www.kaggle.com/datasets/bistaumanga/usps-dataset, Mar. 2024.

hicles (airplane, car, ship, and truck). The dataset considers animals as inliers, and the vehicles are sampled as outliers. Before min-max scaling, we use vgg16[47] to preprocess 96×96 pixels of images to 4 096 vectors.

4.1.2 Methods for Comparison

To demonstrate the effectiveness of our method, we carefully choose several methods for comparison.

1) KNN[1] is certainly a baseline of our method, as our approach is to combine KNN with AutoEncoder. KNN is a classic method for outlier detection, which is a proximity-based method, and uses K-distances as the outlier scores.

2) Isolation Forest[4] is an effective ensemble method for outlier detection on high-dimensional data. It first constructs several isolation trees, and splits the dataset into individual points. It then uses the average depth of a point in the forest as the outlier score. It is one of the state-of-the-art methods for unsupervised outlier detection, and works well for high-dimensional data.

3) A traditional AutoEncoder using reconstruction errors (AutoEncoder-RE)[38] uses reconstruction errors only as outlier scores, as another baseline.

4) Robust AutoEncoder (RAE)[3], similar to the traditional AutoEncoder[38], uses reconstruction errors as the outlier scores as well. Their difference lies in that RAE refines the training set by rolling records having larger reconstruction errors out during the training process. With this optimization, RAE learns the nature of inliers better.

5) Variations of combining KNN with AutoEncoder are also chosen for comparison: 1) AE+KNN using the AutoEncoder in [3] for embedding and the original KNN for detecting, and 2) NNAE+KNN using NNAE for embedding and the original KNN for detecting.

4.1.3 Evaluation Metric

To test the effectiveness of our method, we adopt the commonly used AUC (area under ROC (receiver operating characteristic) curve) as the metric. An ROC curve is a graph showing the performance of a classification model at all classification thresholds, and it plots two parameters: TPR (true positive rate) and FPR (false positive rate). AUC measures the entire two-dimensional area underneath the entire ROC curve from (0, 0) to (1, 1). For the methods that generate outlier scores as the output, we calculate TPR and FPR pairs varying the threshold of outlier scores. For methods that generate labels of outliers, i.e., RAE, we vary the parameter λ, which determines the reconstruction error threshold to roll out data, to generate pairs of TPR and FPR to form an ROC curve for AUC.

All the tests are repeated over three times, and the average is reported to avoid the impacts of randomness.

4.2 Experimental Results

The important parameters and their recommended ranges of our method are listed in Table 2.

Table 2. Parameters of Proposed Approach NNAE+KNRN

Parameter	Description	Range
K	K-nearest neighbors	[25, 200]
L	Number of layers of EnCoder	[3, 9]
θ	Compression rate between two adjacent layers	[0.2, 0.8]
m	Number of focused nearest neighbors in NNAE	[2, 5]

4.2.1 AUC Accuracy Evaluation

In the first set of tests, we evaluate the AUC accuracy on all datasets for all methods, including KNN[1], Isolation Forest[4], AutoEncoder-RE[38], RAE[3], variations of combining KNN with AutoEncoder AE+KNN and NNAE+KNN, and our proposed method NNAE+KNRN. Note that we develop a method to choose the structure parameters L and θ, here the AUC results of NNAE+KNRN are obtained under the setting with this parameter selection method, instead of the setting of a fine-tuned AUC results. The AUC scores would be much higher with a fine-tuned setting. All the AutoEncoder structure parameters use the same setting as NNAE+KNRN. Moreover, K is set to 100, and the focus parameter m is set to 4. The results are reported in Table 3, where the highest AUC score for each dataset is highlighted.

The results show that our approach NNAE+KNRN performs better than all the other comparison methods on all datasets, except that it is slightly worse than NNAE+KNN on Waveform_noise. Even if the AUC results of NNAE+KNRN are obtained under the setting with the structure indicator Z, it remains much better than KNN and AutoEncoder-RE on all datasets. That is, without any prior knowl-

Table 3. AUC Accuracy Comparison of All Methods

Method	Cardio	Waveform_noise	USPS	Fashion-MNIST	STL10
KNN ($K = 100$)[1]	0.949	0.854	0.967	0.899	0.533
Isolation Forest[4]	0.927	0.859	0.924	0.900	0.617
AutoEncoder-RE[38]	0.796	0.529	0.710	0.837	0.587
RAE[3]	0.838	0.540	0.834	0.797	0.545
AE+KNN	0.921	0.901	0.889	0.904	0.655
NNAE+KNN	0.966	**0.927**	0.958	0.939	0.748
NNAE+KNRN	**0.967**	0.923	**0.984**	**0.940**	**0.750**

edge on AutoEncoder, our approach can easily surpass the other methods. It can also be observed that NNAE+KNRN works well on high-dimensional datasets (e.g., STL10 with 4 096 features), while KNN only reaches an accuracy with AUC $= 0.533$. These justify the benefits of combining KNN and AutoEncoder for outlier detection.

Further, the results show that the performance in general follows a rule of KNN $<$ AE+KNN $<$ NNAE+KNN $<$ NNAE+KNRN, which shows that each component of NNAE+KNRN enriches the capability of outlier detection.

4.2.2 *Impacts of K on Accuracy*

In the second set of tests, we evaluate the robustness of NNAE+KNRN with respect to K, compared with methods KNN, AE+KNN, and NNAE+KNN that use parameter K. As mentioned before, the K choice problem has always been an issue for KNN-based methods. In the tests, m is fixed to 5, while K varies from 25 to 200. The results are reported in Fig.4.

The results show that our method NNAE+KNRN works well on all five datasets with K falling in $\{25, 50, 100, 150, 200\}$. On datasets Cardio, Fashion-MNIST, and USPS, its AUC results are very consistent and insensitive to parameter K compared with the original KNN. On datasets Waveform_noise and STL10, parameter K does have an impact on the AUC results. However, even the worst AUC result is still much better than the one of KNN. Indeed, KNN performs consistently low on STL10, around AUC $= 0.53$, indicating that KNN essentially does not work on STL10. However, our method significantly improves the accuracy, up to AUC $= 0.887$, which also partially alleviates the impact of parameter K to a large extent. These show that our method NNAE+KNRN significantly alleviates the K choice problem, and it indeed improves KNN with the aspects of both its accuracy and usability.

4.2.3 *Impacts of m on Accuracy*

In the third set of tests, we evaluate the impacts of m the number of focused nearest neighbors on NNAE. We vary m from 1 to 5, and the other settings are along the same lines as Subsection 4.2.1. The results of NNAE+KNN and NNAE+KNRN are reported in Fig.5.

The results show that NNAE is in general not very sensitive to m, and a small m already suffices to reach a good performance. Moreover, NNAE+KNN and NNAE+KNRN are in general better than KNN and AE+KNN, which verifies the rationale of introducing the focused nearest neighbors in NNAE. Further, the performance of NNAE+KNRN is essentially better than that of NNAE+KNN, which again verifies the benefit of combining the K-distance used in KNN-based methods and the reconstruction errors used in AutoEncoder-based methods.

4.2.4 *Effectiveness of Structure Indicator Z*

In the last set of tests, we evaluate the effectiveness of the structure indicator parameter Z on NNAE+KNRN, which is used to choose the appropriate NNAE structure parameters L and θ for NNAE+KNRN, instead of being set by experienced users. For each of the five real-world datasets, we test 16 sets of combinations of L and θ, and choose the smallest parameter Z. We set L to $\{3, 5, 7, 9\}$ and θ to $\{0.2, 0.4, 0.6, 0.8\}$, respectively, to form 16 different structures of NNAE, and set all the other parameters along the same lines as Subsection 4.2.1. The results are reported in Table 4, where all the AUC results of the 16 combinations are listed that are sorted in descending order, and we only present the top 7 results in the table as Z never selects the structure parameters that have AUC results worse than the median. The AUC results with the smallest indicator pa-

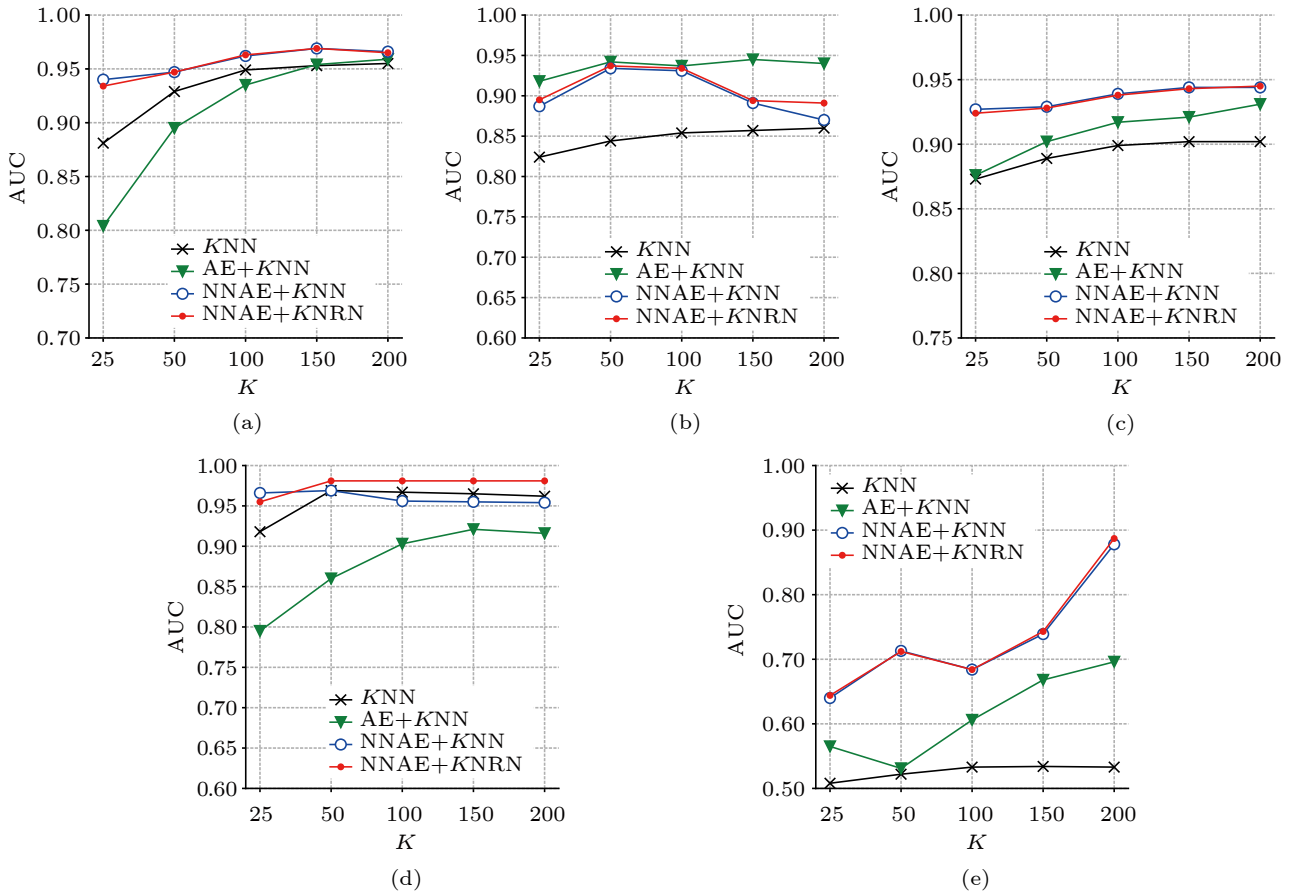

Fig.4. AUC results with respect to K. (a) Cardio. (b) Waveform_noise. (c) Fashion-MNIST. (d) USPS. (e) STL10.

rameter Z for each dataset are highlighted.

The results tell us that the smallest indicator parameter Z has reached reasonably good performances for choosing the parameters for NNAE, although it does not indicate the best performance among combinations. Further, the performance is definitely better than a random choice, such as the median AUC results among the 16 combinations. In practice, this is pragmatic when users have no prior knowledge on the structure of AutoEncoder. Hence, the indicator Z essentially improves the usability of our approach NNAE+KNRN in practice.

4.2.5 *Summary*

From these experiments we find the followings.

1) NNAE+KNRN shows a significant accuracy improvement over KNN, AutoEncoder-RE, RAE, and Isolation Forest by on average 11.40%, 34.93%, 31.97%, and 8.85% for AUC, respectively. This shows the benefit of combining KNN with AutoEncoder-based methods for outlier detection on high-dimensional data.

2) NNAE+KNRN alleviates the K choice problem by making use of the newly designed loss function and the reconstruction errors, where a small number of the focused nearest neighbors suffices for NNAE.

3) The structure indicator Z is reasonably good for choosing the structure parameters of NNAE to improve the usability.

5 Conclusions

In this paper, we introduced a novel approach NNAE+KNRN by seamlessly combining KNN with AutoEncoder for outlier detection on high-dimensional data, where the nearest neighbor AutoEncoder (NNAE) deals with the curse of dimensionality and alleviates the K choice problem, and the K-nearest reconstruction neighbors (KNRN) further alleviate the K choice problem. We also developed a structure indicator Z to improve the usability of NNAE+KNRN in practice by choosing reasonably good structure parameters for NNAE. We finally experimentally showed that our proposed approach NNAE+KNRN makes significant accuracy improve-

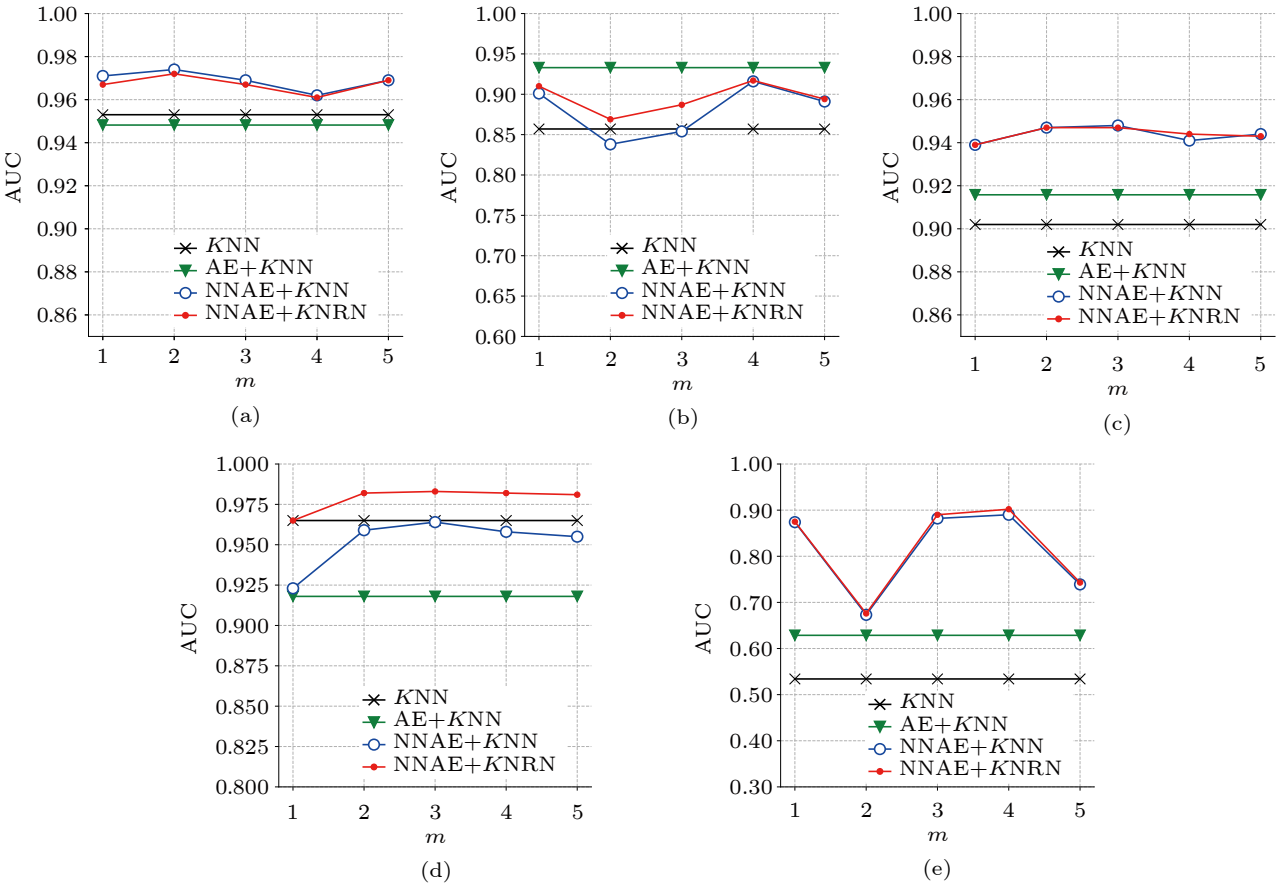

Fig.5. AUC results with respect to m. (a) Cardio. (b) Waveform_noise. (c) Fashion-MNIST. (d) USPS. (e) STL10.

Table 4. Effectiveness of Structure Indicator Z (AUC Result of NNAE-KNRN)

Dataset	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Rank 6	Rank 7	···	Rank
Cardio[⑤]	**0.967**	0.967	0.965	0.963	0.962	0.954	0.953	···	1/16
Waveform_noise[⑥]	0.952	0.943	0.939	0.936	0.923	**0.922**	0.918	···	6/16
Fashion-MNIST[45]	0.944	0.943	0.941	**0.940**	0.936	0.935	0.932	···	4/16
USPS[⑦]	0.992	0.992	0.990	0.988	0.988	0.987	**0.984**	···	7/16
STL10[46]	0.858	0.847	0.783	0.783	0.754	0.754	**0.750**	···	7/16

ment over KNN, AutoEncoder-RE, RAE, and Isolation Forest by on average 11.40%, 34.93%, 31.97%, and 8.85% for AUC, respectively.

A couple of issues need a further study. First, we are to develop better methods for choosing the structure parameters. Second, as AutoEncoder has proven its usefulness for KNN and spectral clustering[48, 49], we are to utilize it for other data mining algorithms.

Conflict of Interest The authors declare that they have no conflict of interest.

References

[1] Aggarwal C C. Outlier analysis. In *Data Mining*, Aggarwal C C (ed.), Springer, 2015. DOI: 10.1007/978-3-319-14142-8_8.

[2] Chandola V, Banerjee A, Kumar V. Anomaly detection: A survey. *ACM Computing Surveys*, 2009, 41(3): Article No. 15. DOI: 10.1145/1541880.1541882.

[3] Zhou C, Paffenroth R C. Anomaly detection with robust deep autoencoders. In *Proc. the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2017, pp.665–674. DOI: 10.1145/3097983.3098052.

⑤http://archive.ics.uci.edu/dataset/193/cardiotocography, Mar. 2024.
⑥http://archive.ics.uci.edu/dataset/108/waveform+database+generator+version+2, Mar. 2024.
⑦https://www.kaggle.com/datasets/bistaumanga/usps-dataset, Mar. 2024.

[4] Liu F T, Ting K M, Zhou Z H. Isolation forest. In *Proc. the 8th IEEE International Conference on Data Mining*, Dec. 2008, pp.413–422. DOI: 10.1109/ICDM.2008.17.

[5] Sequeira K, Zaki M. ADMIT: Anomaly-based data mining for intrusions. In *Proc. the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Jul. 2002, pp.386–395. DOI: 10.1145/775047.775103.

[6] Eskin E, Arnold A, Prerau M, Portnoy L, Stolfo S. A geometric framework for unsupervised anomaly detection. In *Applications of Data Mining in Computer Security*, Barbará D, Jajodia S (eds.), Springer, 2002, pp.77–101. DOI: 10.1007/978-1-4615-0953-0_4.

[7] An J, Cho S. Variational autoencoder based anomaly detection using reconstruction probability. Technical Report, Data Mining Center of Seoul National University, 2015. https://paperswithcode.com/paper/variational-autoencoder-based-anomaly, Sept. 2024.

[8] Angiulli F, Pizzuti C. Fast outlier detection in high dimensional spaces. In *Proc. the 6th European Conference on Principles of Data Ming and Knowledge Discovery*, Aug. 2002, pp.15–26. DOI: 10.1007/3-540-45681-3_2.

[9] Idé T, Kashima H. Eigenspace-based anomaly detection in computer systems. In *Proc. the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2004, pp.440–449. DOI: 10.1145/1014052.1014102.

[10] Hu R J, Aggarwal C C, Ma S, Huai J P. An embedding approach to anomaly detection. In *Proc. the 32nd IEEE International Conference on Data Engineering*, May 2016, pp.385–396. DOI: 10.1109/ICDE.2016.7498256.

[11] Zhu M X, Aggarwal C C, Ma S, Zhang H, Huai J P. Outlier detection in sparse data with factorization machines. In *Proc. the 2017 ACM Conference on Information and Knowledge Management*, Nov. 2017, pp.817–826. DOI: 10.1145/3132847.3132987.

[12] Ng A. Sparse autoencoder. *CS294A Lecture Notes*, 2011, 72(2011): 1–19. https://graphics.stanford.edu/courses/cs233-21-spring/ReferencedPapers/SAE.pdf/, Sept. 2024.

[13] Chen J H, Sathe S, Aggarwal C, Turaga D. Outlier detection with autoencoder ensembles. In *Proc. the 2017 SIAM International Conference on Data Mining*, Apr. 2017, pp.90–98. DOI: 10.1137/1.9781611974973.11.

[14] Zhang C X, Song D J, Chen Y C, Feng X Y, Lumezanu C, Cheng W, Ni J C, Zong B, Chen H F, Chawla N V. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In *Proc. the 33rd AAAI Conference on Artificial Intelligence*, Jan. 27–Feb. 1, 2019, pp.1409–1416. DOI: 10.1609/aaai.v33i01.33011409.

[15] Pang G S, Cao L B, Aggarwal C. Deep learning for anomaly detection: Challenges, methods, and opportunities. In *Proc. the 14th ACM International Conference on Web Search and Data Mining*, Mar. 2021, pp.1127–1130. DOI: 10.1145/3437963.3441659.

[16] Ruff L, Zemlyanskiy Y, Vandermeulen R, Schnake T, Kloft M. Self-attentive, multi-context one-class classification for unsupervised anomaly detection on text. In *Proc. the 57th Annual Meeting of the Association for Computational Linguistics*, Jul. 2019, pp.4061–4071. DOI: 10.18653/v1/p19-1398.

[17] Bergmann P, Fauser M, Sattlegger D, Steger C. MVTec AD—A comprehensive real-world dataset for unsupervised anomaly detection. In *Proc. the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2019, pp.9592–9600. DOI: 10.1109/CVPR.2019.00982.

[18] Gong D, Liu L Q, Le V, Saha B, Mansour M R, Venkatesh S, Van Den Hengel A. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In *Proc. the 2019 IEEE/CVF International Conference on Computer Vision*, Oct. 27–Nov. 2, 2019, pp.1705–1714. DOI: 10.1109/ICCV.2019.00179.

[19] Hou J L, Zhang Y Y, Zhong Q Y, Xie D, Pu S L, Zhou H. Divide-and-assemble: Learning block-wise memory for unsupervised anomaly detection. In *Proc. the 2021 IEEE/CVF International Conference on Computer Vision*, Oct. 2021, pp.8771–8780. DOI: 10.1109/ICCV48922.2021.00867.

[20] Chen X H, Deng L W, Huang F T, Zhang C W, Zhang Z Q, Zhao Y, Zheng K. DAEMON: Unsupervised anomaly detection and interpretation for multivariate time series. In *Proc. the 37th IEEE International Conference on Data Engineering*, Apr. 2021, pp.2225–2230. DOI: 10.1109/ICDE51399.2021.00228.

[21] Lai C H, Zou D M, Lerman G. Robust subspace recovery layer for unsupervised anomaly detection. In *Proc. the 8th International Conference on Learning Representations*, Apr. 2020.

[22] Chen W X, Xu H W, Li Z Y, Pei D, Chen J, Qiao H L, Feng Y, Wang Z G. Unsupervised anomaly detection for intricate KPIs via adversarial training of VAE. In *Proc. the 2019 IEEE Conference on Computer Communications*, Apr. 29–May 2, 2019, pp.1891–1899. DOI: 10.1109/INFOCOM.2019.8737430.

[23] Audibert J, Michiardi P, Guyard F, Marti S, Zuluaga M A. USAD: UnSupervised anomaly detection on multivariate time series. In *Proc. the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Aug. 2020, pp.3395–3404. DOI: 10.1145/3394486.3403392.

[24] Putina A, Sozio M, Rossi D, Navarro J M. Random histogram forest for unsupervised anomaly detection. In *Proc. the 2020 IEEE International Conference on Data Mining*, Nov. 2020, pp.1226–1231. DOI: 10.1109/ICDM50108.2020.00154.

[25] Mohotti W A, Nayak R. Efficient outlier detection in text corpus using rare frequency and ranking. *ACM Trans. Knowledge Discovery from Data*, 2020, 14(6): Article No. 71. DOI: 10.1145/3399712.

[26] Li J, Di S M, Shen Y Y, Chen L. FluxEV: A fast and effective unsupervised framework for time-series anomaly detection. In *Proc. the 14th ACM International Conference on Web Search and Data Mining*, Mar. 2021,

pp.824–832. DOI: 10.1145/3437963.3441823.

[27] Hawkins D M. Identification of Outliers. Springer, 1980.

[28] Mahoney M V, Chan P K. Learning rules for anomaly detection of hostile network traffic. In *Proc. the 3rd IEEE International Conference on Data Mining*, Nov. 2003, pp.601–604. DOI: 10.1109/ICDM.2003.1250987.

[29] Tandon G, Chan P K. Weighting versus pruning in rule validation for detecting network and host anomalies. In *Proc. the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2007, pp.697–706. DOI: 10.1145/1281192.1281267.

[30] Ramaswamy S, Rastogi R, Shim K. Efficient algorithms for mining outliers from large data sets. In *Proc. the 2000 ACM SIGMOD International Conference on Management of Data*, May 2000, pp.427–438. DOI: 10.1145/342009.335437.

[31] Hautamäki V, Kärkkäinen I, Fränti P. Outlier detection using *k*-nearest neighbour graph. In *Proc. the 17th International Conference on Pattern Recognition*, Aug. 2004, pp.430–433. DOI: 10.1109/ICPR.2004.1334558.

[32] Jagadish H V, Koudas N, Muthukrishnan S. Mining deviants in a time series database. In *Proc. the 25th International Conference on Very Large Data Bases*, Sept. 1999, pp.102–113.

[33] Zimek A, Schubert E, Kriegel H P. A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining*, 2012, 5(5): 363–387. DOI: 10.1002/sam.11161.

[34] Lee Y J, Yeh Y R, Wang Y C F. Anomaly detection via online oversampling principal component analysis. *IEEE Trans. Knowledge and Data Engineering*, 2013, 25(7): 1460–1470. DOI: 10.1109/TKDE.2012.99.

[35] Hinton G E, Salakhutdinov R R. Reducing the dimensionality of data with neural networks. *Science*, 2006, 313(5786): 504–507. DOI: 10.1126/science.1127647.

[36] Borg I, Groenen P J F. Modern Multidimensional Scaling: Theory and Applications (2nd edition). Springer, 2005.

[37] Xiong L, Chen X, Schneider J. Direct robust matrix factorizatoin for anomaly detection. In *Proc. the 11th IEEE International Conference on Data Mining*, Dec. 2011, pp.844–853. DOI: 10.1109/ICDM.2011.52.

[38] Hawkins S, He H X, Williams G, Baxter R. Outlier detection using replicator neural networks. In *Proc. the 4th International Conference*, Sept. 2002, pp.170–180. DOI: 10.1007/3-540-46145-0_17.

[39] Sakurada M, Yairi T. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proc. the 2nd Workshop on Machine Learning for Sensory Data Analysis*, Dec. 2014, pp.4–11. DOI: 10.1145/2689746.2689747.

[40] Aytekin C, Ni X Y, Cricri F, Aksu E. Clustering and unsupervised anomaly detection with l2 normalized deep auto-encoder representations. In *Proc. the 2018 International Joint Conference on Neural Networks*, Jul. 2018, pp.1–6. DOI: 10.1109/IJCNN.2018.8489068.

[41] Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol P A. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 2010, 11: 3371–3408. DOI: 10.5555/1756006.1953039.

[42] Li Y, Fang B X, Guo L, Chen Y. Network anomaly detection based on TCM-KNN algorithm. In *Proc. the 2nd ACM Symposium on Information, Computer and Communications Security*, Mar. 2007, pp.13–19. DOI: 10.1145/1229285.1229292.

[43] Wu G J, Zhao Z H, Fu G, Wang H P, Wang Y, Wang Z Y, Hou J T, Huang L. A fast *k*NN-based approach for time sensitive anomaly detection over data streams. In *Proc. the 19th International Conference on Computational Science*, Jun. 2019, pp.59–74. DOI: 10.1007/978-3-030-22741-8_5.

[44] Goldstein M, Uchida S. A comparative study on outlier removal from a large-scale dataset using unsupervised anomaly detection. In *Proc. the 5th International Conference on Pattern Recognition Applications and Methods*, Feb. 2016, pp.263–269. DOI: 10.5220/0005701302630269.

[45] Xiao H, Rasul K, Vollgraf R. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. arXiv: 1708.07747, 2017. http://arxiv.org/abs/1708.07747, Aug. 2024.

[46] Coates A, Ng A Y, Lee H. An analysis of single-layer networks in unsupervised feature learning. In *Proc. the 14th International Conference on Artificial Intelligence and Statistics*, Apr. 2011, pp.215–223.

[47] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In *Proc. the 3rd International Conference on Learning Representations*, May 2015.

[48] Duan L, Aggarwal C C, Ma S, Sathe S. Improving spectral clustering with deep embedding and cluster estimation. In *Proc. the 2019 IEEE International Conference on Data Mining*, Nov. 2019, pp.170–179. DOI: 10.1109/ICDM.2019.00027.

[49] Duan L, Ma S, Aggarwal C, Sathe S. Improving spectral clustering with deep embedding, cluster estimation and metric learning. *Knowledge and Information Systems*, 2021, 63(3): 675–694. DOI: 10.1007/s10115-020-01530-8.

Shu-Zheng Liu received his M.S. degree in industrial engineering from Rutgers University, New Brunswick, in 2017, and his M.E. degree in computer science from Beihang University, Beijing, in 2021. He is currently working at Alibaba, Beijing. His research interests include data mining and outlier detection.

Shuai Ma received his Ph.D. degrees in computer science from Peking University, Beijing, in 2004, and from The University of Edinburgh, Edinburgh, in 2010, respectively. He is a professor with the School of Computer Science and Engineering, Beihang University, Beijing. He was a postdoctoral research fellow with the Database Group, The University of Edinburgh, a summer intern at Bell Labs, Murray Hill, NJ, and a visiting researcher of MSRA. His current research interests include big data, database, theory and systems, graph and social data analysis, data cleaning and data quality.

Han-Qing Chen received his B.S. degree in software engineering from Beihang University, Beijing, in 2019. He is a Ph.D. candidate in the School of Computer Science and Technology, Beihang University, Beijing. His research interests include big data and graph analytics.

Li-Zhen Cui received his Ph.D. degree in computer science from Shandong University, Jinan, in 2005. He is a professor with the School of Software & C-FAIR, Shandong University, Jinan. He is a president of the School of Software & C-FAIR, Shandong University, Jinan. His current research interests include recommender systems and data mining.

Jie Ding received his Ph.D. degree in computer science from The University of Edinburgh, Edinburgh, in 2010. He is a professor with the School of Computer Science, Jiangsu University of Science and Technology, Zhenjiang. His current research interests include theoretical computer science, big data, and reinforcement learning.