OM4AnI: A Novel Overlap Measure for Anomaly Identification in Multi-Class Scatterplots

Liqun Liu, Leonid Bogachev, Mahdi Rezaei, Nishant Ravikumar, Arjun Khara, Mohsen Azarmi, and Roy A. Ruddle

Abstract—Scatterplots are widely used across various domains to identify anomalies in datasets, particularly in multi-class settings, such as detecting misclassified or mislabeled data. However, scatterplot effectiveness often declines with large datasets due to limited display resolution. This paper introduces a novel Visual Quality Measure (VQM) – OM4AnI (Overlap Measure for Anomaly Identification) – which quantifies the degree of overlap for identifying anomalies, helping users estimate how effectively anomalies can be observed in multi-class scatterplots. OM4AnI begins by computing anomaly index based on each data point's position relative to its class cluster. The scatterplot is then discretized into a matrix representation by binning the display space into cell-level (pixel-level) grids and computing the coverage for each pixel. It takes into account the anomaly index of data points covering these pixels and visual features (marker shapes, marker sizes, and rendering orders). Building on this foundation, we sum all the coverage information in each cell (pixel) of matrix representation to obtain the final quality score with respect to anomaly identification. We conducted an evaluation to analyze the efficiency, effectiveness, sensitivity of OM4AnI in comparison with six representative baseline methods that are based on different computation granularity levels: data level, marker level, and pixel level. The results show that OM4AnI outperforms baseline methods by exhibiting more monotonic trends against the ground truth and greater sensitivity to rendering order, unlike the baseline methods. It confirms that OM4AnI can inform users about how effectively their scatterplots support anomaly identification. Overall, OM4AnI shows strong potential as an evaluation metric and for optimizing scatterplots through automatic adjustment of visual parameters.

Index Terms—Anomaly identification, Visual quality measure, Multi-class scatterplot, Explainable AI

1 Introduction

Anomaly identification (e.g., identifying outliers or misclassified data points) is a key analytical task in multiple-class scatterplots (each data point belongs to one of multiple distinct classes. These classes are usually represented using different colors, shapes, or marker styles to help distinguish them) [34]. This task helps users understand how instances are distributed and determine whether any have been incorrectly projected into clusters to which they do not belong, which supports the detection of misclassified instances (see Figure 1). Identifying anomalies using scatterplots is widely used across various domains, such as the medical domain [26], Explainable Artificial Intelligence (XAI) [19, 23, 31, 38, 44, 45], and dimensionality reduction [16, 20].

A key issue in identifying anomalies from multi-class scatterplots is overplotting, especially when dealing with large datasets. Overplotting hinders anomaly detection because markers can obscure one another, particularly when overlapped by markers from different groups or classes. Therefore, designing a VQM to measure how much anomaly information is hidden in a multi-class scatterplot remains valuable.

However, most VQMs for multi-class scatterplots do not consider the anomaly identification task. Existing scatterplot VQMs for measuring overlap are either task-based or task-agnostic. Among task-based measures, cluster separation is a commonly used type of visual effectiveness measure, which includes algorithmic approaches that consider neighborhood types and class-purity evaluation techniques [2], as well as data-driven approaches, such as those by Aupetit and Sedlmair [2], who utilized proximity graphs and machine learning methods to model human perception of class separation. Even when scatterplots are well-separated, anomalies may still be hidden (see Figure 1); thus, these measures are not effective for evaluating anomaly identification.

In task-agnostic measures, researchers have primarily focused on computing the visibility of data points, either by calculating the proportion of occluded points [40] or by counting the number of data points in specific regions [6, 12]. These approaches have mostly considered only the data coordinates. Some incorporate certain visual features, such as marker size; however, no researchers have considered a critical visual feature—rendering order—which determines the sequence in which each data item is drawn in a scatterplot.

Our research designs a novel VQM, named OM4AnI, which provides users with a value representing the quality of a multi-class scatterplot in terms of its support for anomaly identification under overlap conditions (We assume that the markers in scatterplots are opaque). OM4AnI first calculates the anomaly index based on each data point's relative position to other data points belonging to the same class. Then, we discretize the scatterplot into a matrix representation by creating a grid and mapping markers into the corresponding grid cells. The coverage information is then transformed into matrix form, taking into account marker shape, marker size, and rendering order. Finally, we calculate the quality score by summarizing the coverage information encoded in the matrix representation.

The evaluation shows that OM4AnI outperforms six baseline methods in predicting hidden anomalies (data points that are occluded by other data points belonging to the different class to them). In terms of time cost, OM4AnI does not incur a significantly higher computational burden compared to the baseline methods. Regarding effectiveness, OM4AnI demonstrates stronger monotonic trends in quality scores with respect to anomaly information. Additionally, OM4AnI is sensitive to both marker size and rendering order, whereas none of the baseline methods are sensitive to both visual features.

Our main contributions are as follows. First, we develop a novel VQM for multi-class scatterplots to measure overlap in the context of anomaly identification tasks (see Section 3). OM4AnI incorporates the anomaly index score of each data point, computed based on data

- *Liqun Liu is with University of Leeds. E-mail: L.Liu6@leeds.ac.uk*
- *Leonid Bogachev is with University of Leeds. E-mail: L.V.Bogachev@leeds.ac.uk*
- *Mahdi Rezaei is with University of Leeds. E-mail: M.Rezaei@leeds.ac.uk*
- *Nishant Ravikumar is with University of Leeds. E-mail: N.Ravikumar@leeds.ac.uk*
- *Arjun Khara is with University of Leeds. E-mail: A.Khara@leeds.ac.uk*
- *Mohsen Azarmi is with University of Leeds. E-mail: tsmaz@leeds.ac.uk*
- *Roy Ruddle is with University of Leeds. E-mail: R.A.Ruddle@leeds.ac.uk*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

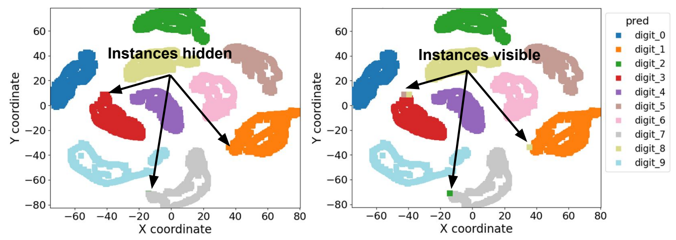

Fig. 1: Scatterplot examples for the anomaly identification task. Scatterplots are employed to assess the performance of a pretrained Convolutional Neural Network (CNN) model [25] in classifying the MNIST dataset [13]. (a) and (b) present scatterplots of instances used for classification, where the x-axis and y-axis represent the coordinates of each instance, computed through t-SNE algorithm [41]. After altering the rendering order, the visibility of misclassified instances changes. They are expected to have different VQM values.

coordinates and visual features (e.g., marker size and rendering order), as key elements in its design. Second, we evaluate the VQM using both synthetic and real-world datasets to analyze the performance of OM4AnI (see Section 4) through comparing it with 6 baseline methods.

2 Related Work

Our paper focuses on VQM for multi-class scatterplots. Therefore, we structure our literature review around two key aspects: scatterplots, and VQM techniques specifically related to scatterplots—an area that has been central to the information visualization community since its inception [3].

2.1 Scatterplots

A scatterplot is a type of data visualization used to display the relationship between two numerical variables. Each point on the scatterplot represents an observation in the dataset, with its position determined by its values on the x-axis and y-axis. However, traditional scatterplots are limited in handling large datasets due to display constraints, especially with the rapid development of AI, which generates vast amounts of data that are difficult to visualize.

To address this issue, many improved designs for scatterplots have been proposed within the visualization community. As introduced by Ellis and Dix [15], these methods can be categorized into three groups: (1) appearance improvement (e.g., sampling [4, 6] or aggregation [9]), (2) spatial distortion (e.g., point/line displacement [28] or topological distortion [27]), and (3) temporal methods (e.g., animation [43]).

Even though many variations of scatterplots have been developed based on the traditional design to address large datasets, traditional scatterplots remain widely used across various domains. For example, in XAI, Chatzimparmpas et al. [10, 11] reported that 115 out of 199 papers related to XAI employed scatterplots, and our review confirmed that most of these used traditional scatterplots. Therefore, conducting research on traditional scatterplots remains valuable for domain experts.

Tasks play a crucial role in both the design of scatterplots and the development of VQMs. Sarikaya and Gleicher [34] summarized 12 scatterplot-related tasks, which include both low-level tasks and high-level tasks (e.g., identifying objects, locating objects, exploring neighborhoods or identifying anomalies). Many of these tasks can be significantly affected by overplotting. Our work focuses on the task of identifying anomalies, which is one of the most commonly used tasks in different domains.

2.2 VQMs for Scatterplots

VQMs can be categorized into three aspects [7]. The first aspect involves size metrics, which form the basis for other computations, such as quantifying data items. The second aspect concerns feature preservation, which evaluates whether the abstracted data or dimensions retain their original characteristics. The third aspect includes visual effectiveness metrics, which assess image degradation and perceptual quality. In this work, we focus on reviewing the feature preservation and visual effectiveness measures relevant to scatterplots.

2.2.1 Feature Preservation Metrics

Feature preservation metrics are primarily focused on evaluating whether a method performs well for generating a scatterplot. In the context of scatterplot VQMs, the most commonly used are two types of feature preservation VQMs. The first type measures the process of projecting high-dimensional data to two-dimensional space, aiming to evaluate the effectiveness of dimensionality reduction algorithms. An example is the "Classes Are Not Cluster" metric, which assesses how well the structure of high-dimensional data is preserved in the low-dimensional projection [20].

The second type evaluates the performance of sampling methods by estimating how well data density is preserved after sampling. For instance, Bertini and Santucci [5] proposed a metric to assess how well a sampling method preserves relative data density, considering the represented density as the number of active pixels within the same area. Johansson and Cooper [22] also introduced a method based on distance transforms to create graphical representations and compare raw data with abstract visualizations.

All of these VQMs evaluate methods that either generate scatterplots by projecting multi-dimensional data into two dimensions or by sampling the original data, thereby focusing on the scatterplot generation process. They measure how well a scatterplot retains the same features as the original scatterplot or data, without evaluating whether the scatterplot reflects the actual insights in the data. In contrast, our work focuses directly on the scatterplot itself, aiming to determine whether it contains hidden anomalies.

2.2.2 Visual Effectiveness Metrics

Visual effectiveness metrics can be categorized into two groups: task-based and task-agnostic VQMs. Task-based VQMs include tasks such as identifying cluster separation and identifying correlation. Task-agnostic VQMs aim to estimate scatterplot quality without focusing on specific tasks, such as by measuring the visibility index of data points.

One popular type of task-based VQM focuses on cluster separation. Traditional approaches to this task primarily rely on algorithmic methods. Aupetit and Sedlmair categorized cluster separation VQMs from two perspectives: the type of neighborhood and the method of class-purity evaluation [2]. Then, they proposed a data-driven evaluation framework for assessing class separation in scatterplots based on human judgment [36]. Using this framework, they evaluated a set of 15 separation measures and found that the best-performing measure was the Distance Consistency Measure (**DSC**) [39], which is defined as the fraction of points whose distance to their own-class centroid in the 2D view is not strictly smaller than their distance to any other class centroid.

Apart from algorithmic methods for measuring cluster separation in scatterplots, data-driven approaches are also viable for designing VQMs that assess how well clusters are separated. Abbas et al. [1] introduced ClustMe, a VQM based on the Gaussian Mixture Model and human judgment, to evaluate the complexity of grouping patterns for monochrome scatterplots. Aupetit and Sedlmair [2] proposed visual separation measures that focus on neighborhood-based and component-based class purity evaluations, employing different proximity graphs and machine learning methods to model human perception of class separation. Jeon et al. [21] designed CLAMS, a perceptual regression model trained on a user study dataset, to determine whether a scatterplot exhibits good separation.

All of these task-based VQM methods focus solely on cluster separation in multi-class scatterplots and consider only the coordinate data. These cluster-separation-focused VQMs are not well-suited for measuring anomaly identification. For example, in Figure 1, anomalies may still be hidden even when clusters are well-separated. Moreover, all of these approaches ignore visual factors such as marker size and rendering order, which can significantly affect anomaly visibility. In contrast, our method addresses this gap by focusing on the design of a VQM specifically for anomaly identification in multi-class scatterplots.

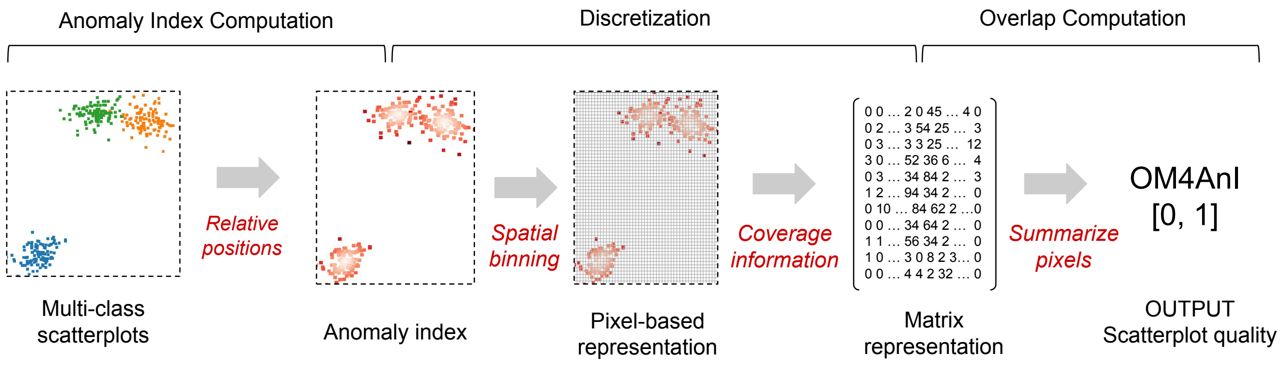

Fig. 2: The pipeline of our method involves three key steps. First, we compute the anomaly index for each data point, which serves as an input parameter for the next step. Then, we discretize the scatterplot into a matrix representation by binning it onto cell-based (pixel-based) grids and calculating the coverage information for each pixel, taking into account the anomaly index of each data point covering these pixels and visual features such as marker shapes, marker sizes, and rendering orders. Finally, we compute the overlap degree by summing the coverage information of cells (pixels) in matrix representation as a value between 0 and 1, where 0 indicates poor quality and 1 represents perfect quality with no data points occluded.

Task-agnostic VQMs are also important for assessing the quality of scatterplots. Although they are not specifically focused on a single task, they assess how much information is hidden in a scatterplot. Some measures are computed solely based on coordinate data (data level). Among these, basic methods calculate the local density of a scatterplot in a 2D view, such as Nearest Neighbor Distance [12] (**NND**) and Kernel Density Overlap Degree [32] (**KDOD**). Beyond basic density estimation, some measures compute the size of the overlapping region (e.g., convex hull) between groups and sum the densities of objects within the overlapping areas [35].

Some measures operate at the marker level and consider the overlap among multiple markers. For example, the Visibility Index (**VI**) [40] quantifies the percentage of glyphs that are not completely overlapped by other glyphs. A related approach, Pairwise Bounding Box Overlap Degree (**PBBOD**), calculates the ratio of the intersection area between each pair of markers to the smaller area of the two markers [18].

Some measures are designed VQMs at the pixel level. The fundamental idea is to compute information density based on the number of pixels in specific regions to assess overplotting [6]. Three basic measures proposed by Ellis and Dix [15] calculate the overlap: (1) the percentage of pixels plotted more than once; (2) the percentage of plotted points that fall within pixels containing more than one point; and (3) the percentage of plotted points that are hidden from view due to overplotting [14]. Based on basic measures, Grid Density Overlap Degree (**GDOD**), as proposed in [24], determines whether multiple data points are mapped to a single pixel. Similarly, M_{pix} (**MP**) counts the number of pixels occupied by more than two markers [42].

Whether at the data level, marker level, or pixel level, these task-agnostic VQMs often overlook important visual features such as rendering order, and relatively few studies consider marker size. In contrast, our work not only considers the density of data items but also incorporates visual features such as marker size, marker shape, and rendering order. Furthermore, to the best of our knowledge, our work is the first to explicitly account for rendering order in a VQM.

In order to evaluate our methods, we selected six representative methods for comparison from both task-based and task-agnostic perspectives (see Section 4.3). For the task-based aspect, we selected the best-performing algorithm-based method, **DSC**. The other five baseline methods fall under the task-agnostic category: one method, **KDOD**, is based on data level; two methods, **VI** and **PBBOD**, operate at the marker level; and two methods, **GDOD** and **MP**, operate at the pixel level (see Table 2).

3 Methods

We outline a workflow for OM4AnI to measure the overlap of multi-class scatterplots with respect to anomaly identification in three steps: (1) we compute the anomaly index of each data point based on the relative position of each marker to other markers belonging to the same class; (2) we discretize the scatterplot into a matrix-based representation by mapping all markers onto a grid using a spatial binning method, taking into account features such as marker size, shape, and rendering order, and then transform the pixel-based representation into a matrix by computing the coverage information; (3) we compute the overlap by summing the coverage information within each cell of the matrix. Figure 2 illustrates the pipeline for these steps. We introduce each of the three steps in detail and, at the end of this section, analyze the computational complexity of OM4AnI.

3.1 Step 1 - Anomaly Index

This section introduces anomaly index, which quantifies how unusual or atypical a data point is relative to the rest of the data in each class. In this paper, we use it to indicate how anomalous each data point is compared to other data points belonging to the same class, thereby facilitating anomaly identification. The anomaly index value of a data point j is denoted as w_j.

To compute a data point's anomaly index without the influence of varying data coordinate scales (e.g., x-axis is range of 0 to 1 but y-axis is range of 0 to 100), we normalize the data coordinates along both the x-axis and y-axis. Suppose we want to create a matrix representation in Step 2 with dimensions p_x and p_y (the values of which are defined in Section 3.2), where p_x refers to the number of cells along the x-axis and p_y refers to the number of cells along the y-axis. If $p_x > p_y$, we first map the x-coordinates to the range [0, 1], and then map the y-coordinates to the range $[0, \frac{p_y}{p_x}]$, in order to align the x and y scales with the grid dimensions used in Section 3.2. Conversely, if $p_y \geq p_x$, we map the x-coordinates to the range $[0, \frac{p_x}{p_y}]$ and the y-coordinates to the range [0, 1].

The anomaly index of data points in a scatterplot is computed based on their relative positions to their class clusters under different method schemes using normalized coordinates. While certain data points may have minimal influence on insight discovery, others can significantly alter outcomes if obscured. Our computation provides users with a value indicating how anomalous each data point is. Multiple computation methods are applied in this step as candidates, each of which may ultimately produce a VQM value as a sub-method of OM4AnI, and we evaluate these methods in Section 4. We list three candidate methods for anomaly index computation (Please refer to the Section 1 in supplementary material for detailed computation information):

(1) *Mahalanobis Distance* [30] measures how far a data point is from the mean of a distribution while accounting for correlations between variables. It is computed using the mean vector and the inverse of the covariance matrix, effectively normalizing the deviation based on the dataset's variance structure.
(2) *Local Outlier Factor (LOF)* [8] detects local outliers by comparing the density of a data point to that of its neighbors. A point is consid-

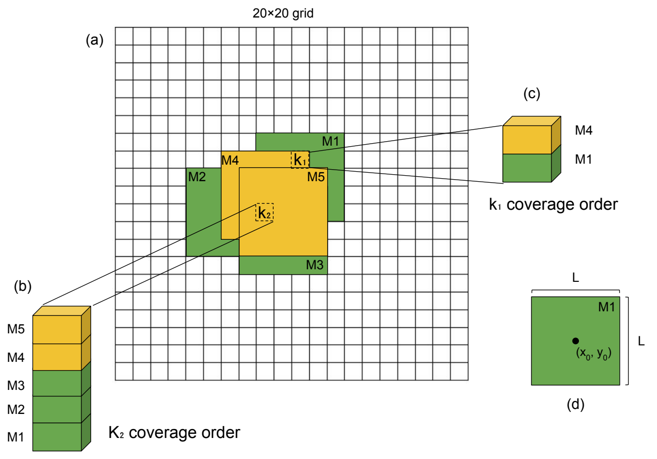

Fig. 3: An illustrative example of discretization. In (a), five data points (rendered as five markers) from two different classes are displayed in a scatterplot. Cells k_1 and k_2 are covered differently: the topmost marker at k_1 is M4, while at k_2 it is M5. The number of cells covered by each marker is determined by their visual parameters, as detailed in (d).

ered an outlier if its local density is significantly lower than that of surrounding points.
(3) *Average-Linkage* computes the average Squared Euclidean Distance from the point to all points in the cluster, offering a balance between the sensitivity of single-linkage and the compactness of complete-linkage.

Based on three different anomaly index computations, we derive three sub-methods of OM4AnI: **MD-AnI**, based on the *Mahalanobis Distance*; **LOF-AnI**, based on the *Local Outlier Factor (LOF)*; and **ALMD-AnI**, based on *Average-Linkage*. We evaluate the performance of these three sub-methods along with six additional baseline methods (see Section 4).

3.2 Step 2 - Discretization

We discretize the scatterplot into a matrix representation by mapping the markers onto a spatially binned grid to obtain a pixel-based representation of the scatterplot, and then transform this representation into a matrix form, as shown in Figure 2 (Step 2). This process first determines the number of cells in the grid and the number of bins along the x-axis and y-axis. After creating the grid, OM4AnI projects markers (each data point for a marker) onto the cell-based (pixel-based) grid, taking into account both data features and visual features. Data features include data coordinates, anomaly index and class labels, while visual features include marker size, marker shape, and rendering order. Through this process, OM4AnI obtains the pixel-based representation of the scatterplot. It then transforms this representation into a matrix format that stores coverage information for each pixel, including the number of data points in each pixel, their class labels, their anomaly index and their rendering order.

We take an example to explain the discretization process, as shown in Figure 3, the scatterplot space is divided into a grids with 20×20 (which is mentioned as p_x and p_y in Section 3.1) cells in (a). Five markers representing five data points (from $M1$ to $M5$) are plotted, resulting in overplotting among the markers, with rendering order $M1 \rightarrow M2 \rightarrow M3 \rightarrow M4 \rightarrow M5$. Each cell records its own coverage situation. For example, cell k_1 is covered by all five markers, and the coverage order is shown in (b), where markers from two different classes are involved. In contrast, cell k_2 is covered by only two markers, with the coverage order $M1 \rightarrow M4$ in (c). To determine how many cells each marker can cover, we compute it based on the marker parameters, as illustrated in (d), where (x_0, y_0) are the coordinates of a data item and L is the side length of the rectangle. Using the same method, we can also map markers into a grid for other shapes, such as triangles or circles (Please refer to Section 2 of the supplementary material for the detailed computation strategy).

3.3 Step 3 - Overlap Computation

We introduce the third step (see Step 3 in Figure 2) of OM4AnI, which provides a quantitative value to inform users how effective their scatterplot is at identifying anomalous data points. The output of OM4AnI ranges over $[0,1]$, where 0 indicates the worst quality and 1 represents no overlap in the designed scatterplot. The output value is computed based on the anomaly index derived in Section 3.1 and the pixel-based coverage information (i.e., how each cell is covered by markers) from Section 3.2.

3.3.1 Notations

Before introducing how to compute the output value of OM4AnI, we first establish a set of notations to describe the elements that constitute OM4AnI. The key notations include:
K: Set of all cells (pixels) in the scatterplot.
$k \in K$: A specific cell within set K.
$N_K = |K|$: Total number of cells.
M: Set of all markers (data points) in the scatterplot.
$N_M = |M|$: Total number of markers in set M.
$N_M(k) = t$: Number of markers covering cell k.
$m_i(k)$: The i-th marker covering cell k (i reflects the order of plotting in cell k), where $i \in \{1, 2, \ldots, t\}$.
$m_t(k)$: The topmost marker covering any other markers in cell k.
$w_i(k)$: Anomaly index of the i-th marker on cell k, which is calculated from Section 3.1 (each data point has only one anomaly index value).
C: Set of classes represented across all markers.
$c_i(k)$: Class of the i-th marker covering cell k.
$c_t(k)$: Class of the topmost marker covering cell k.

3.3.2 Computation

We introduce the computation of OM4AnI for anomaly identification in multi-class scatterplots. The core idea is to first summarize the anomaly index values of all occluded markers in each cell, and then compute the ratio between the aggregated anomaly index values of all occluded markers across cells and the total anomaly index values of all markers (both occluded and topmost) within those cells. The initial step involves identifying the anomaly index of the topmost marker, $q_t(k)$, in a given cell k, which occludes all other markers at that location (e.g., M5 at cell k_2 and M4 at k_1 in Figure 3). This is expressed as:

$$q_t(k) = w_t(k). \tag{1}$$

Next, we compute the total anomaly index values of all markers occluded by the topmost marker at cell k that belong to the same class as the topmost marker (e.g., M4 in (b) of Figure 3). This quantity is given by Equation (2):

$$q_s(k) = \lambda \sum_{i=1}^{t-1} w_i(k) \mathbb{1}\{c_i(k) = c_t(k)\}, \tag{2}$$

where $\mathbb{1}\{\cdot\}$ denotes the indicator function, which returns 1 if the condition is satisfied and 0 otherwise. This allows for the identification of data points that both cover a cell and belong to the same class as the topmost marker. By aggregating the total anomaly index values of such same-class occluded data points at cell k, we compute a weighted sum. This sum is then scaled by a task-specific coefficient λ (ranging from 0 to $+\infty$), which reflects the relative importance of data points occluded by others from the same class (the larger the value, the more important they are considered). For example, in the scenario shown in Figure 1, data points occluded by same-class data points do not hinder anomaly identification; therefore, a smaller λ value is used to indicate their lower importance.

Similarly, we compute the total anomaly index values of all markers occluded by the topmost marker at cell k that belong to a different class than the topmost marker (e.g., M1, M2, and M3 in (b), or M1 in (c) of Figure 3). This is expressed in Equation (3):

$$q_d(k) = \beta \sum_{i=1}^{t-1} w_i(k) \mathbb{1}\{c_i(k) \neq c_t(k)\}, \tag{3}$$

where β (ranging from 0 to $+\infty$) denotes the importance for data points (markers) occluded that belong to a different class than the topmost marker in the scatterplot. For example, in the scenario shown in Figure 1, data points covered by different-class data points affect a lot on anomaly identification so a bigger β value is used to indicate their higher importance. At this stage, we have computed the total anomaly index values of hidden markers for each cell k, including those that belong to the same class as the topmost marker and those that belong to different classes as the topmost marker. The hidden information $q_h(k)$ for each cell k is then defined as:

$$q_h(k) = q_s(k) + q_d(k). \tag{4}$$

We calculate the hidden information degree $o(k)$ at each cell k, which helps identify where anomalies are located in a scatterplot. This metric informs users whether cell k contains occluded data points with high anomaly index values, and is defined as:

$$o(k) = \frac{q_h(k) - \min(q_h)}{\max(q_h) - \min(q_h)}, \tag{5}$$

where it ranges from 0 to 1, with 0 indicating minimal hidden information and 1 indicating severe hidden information. Finally, we sum these values across all cells and assess the final overlap degree value for a multi-class scatterplot, as shown in Equation (6):

$$Q = \frac{Q_t}{Q_d + Q_s + Q_t}, \tag{6}$$

where

$$Q_t = \sum_{k \in K} q_t(k), \quad Q_d = \sum_{k \in K} q_d(k), \quad Q_s = \sum_{k \in K} q_s(k). \tag{7}$$

The final quality value, Q, effectively integrates the contributions from both the hidden same-class data points and those from different classes, relative to the total anomaly index values of all markers across all cells, thereby providing a comprehensive measure of the underlying data point structure across the cells. It ranges over $(0,1]$, where 0 corresponds to poor quality with significant overlap, while 1 indicates a perfect scatterplot with no overlap for the anomaly identification task.

3.4 Computational Complexity

All our methods involve three main computational steps: anomaly index computation, discretization, and overlap computation. Thus, the overall computational complexity can be expressed as:

$$O = O(\text{Anomaly}) + O(\text{Discretization}) + O(\text{Overlap}),$$

where $O(\text{Discretization}) = ns$ and $\mathscr{O}(\text{Overlap}) = hw$ (see Table 2 for notation details). However, $O(\text{Anomaly})$ depends on the specific anomaly index computation method used. The computational complexity of the anomaly index computation in **LOF-AnI** is $O(n \log n)$. For **MD-AnI** and **ALMD-AnI**, the anomaly index computation has a complexity of $O(n^2/k)$. Therefore, the overall computational complexity of our methods is either $O(n \log n + sn + hw)$ or $O(n^2/k + sn + hw)$ (see Table 2 for notation details). Based on this computational complexity analysis, **LOF-AnI** is less time-consuming than **MD-AnI** and **ALMD-AnI**.

4 Evaluation

We conducted an evaluation to demonstrate the efficiency, effectiveness, and sensitivity to marker size and rendering order of our method, compared to baseline methods (see Table 2). The evaluation consists of four steps. First, we describe the data preprocessing procedures, including how the datasets were collected, the scatterplots were generated by varying visual features (marker sizes and rendering orders), and the ground truth was constructed for evaluating the effectiveness of different methods. Second, we introduce the evaluation tasks and the corresponding metrics used to assess the performance of each method. Third, we present the baseline methods used for comparison. Finally, we analyze the results of the comparison between our method and the baseline methods to validate the performance of OM4AnI.

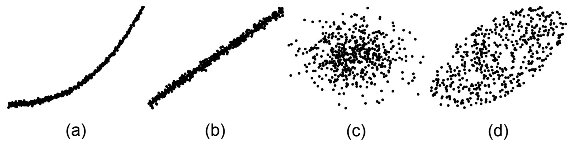

Fig. 4: The shapes of clusters in the datasets generally fall into four categories: (a) curve clusters, (b) linear clusters, (c) circular clusters, and (d) elliptical clusters.

4.1 Data Preprocessing

The dataset preprocessing consists of three steps: 1) We collected datasets from generated synthetic datasets to cover a range of data distribution patterns and Sedlmair's datasets from the work [37]; 2) We varied two visual factors—marker size and rendering order—to generate scatterplots from the collected datasets. This variation allows the scatterplots to represent diverse visual patterns. 3) We constructed the ground truth data for evaluating the effectiveness of different methods.

4.1.1 Data Collection and Generation

To evaluate OM4AnI under diverse conditions, we created synthetic datasets featuring multiple clusters with varying structural and data characteristics. The dataset generation process began by constructing functions to generate clusters of data points with specific distributions (see Figure 4), and then embedding the clusters produced by each function into a single dataset to simulate multi-class datasets (see Figure 6).

The first two functions in Figure 4 generated the curve-shaped and linear-shaped clusters shown in Figure 4a and Figure 4b. The curve-shaped cluster is based on the function $y = 0.1x^3 + 0.6x^2 + 0.4x$. Using this function, we randomly generated x values uniformly distributed in the range [0, 10]. The corresponding y values were then calculated using the function, with added noise uniformly distributed between -2 and 2. After generating all x and y values, the y values were scaled to fall within the range [0, 10]. This approach allows the function to produce curve-shaped cluster data with both x and y values constrained to the range [0, 10]. Similarly, the Figure 4b function follows the same procedure to generate a cluster of data points, except that it was based on the linear function $y = 4x$. This function also produces a variety of data points, with both x and y values falling within the range [0, 10]. Each dataset includes four attributes: x and y coordinates, shape, and class.

The other functions generate the circular-shaped and elliptical-shaped clusters. In Figure 4c, we defined a function to generate data following a normal distribution in a 2D array. The $center_x$ and $center_y$ values of the data center are randomly selected from the range 0 to 30, and the scale (standard deviation) is set to [1, 2]. The function in Figure 4d generates data points with elliptical shapes using a uniform distribution over the area of the ellipse. It defines the cluster's center (the same as for circular-shaped clusters) and axes ($axis_x$ and $axis_y$, randomly selected from the range 1 to 2), samples angles (θ) uniformly from $[0, 2\pi)$, and radial distances from $\sqrt{U}$, where U is uniformly distributed over $[0,1)$. The polar coordinates are then transformed into Cartesian coordinates using $x = r \cdot \text{axis}_x \cdot \cos(\theta)$ and $y = r \cdot \text{axis}_y \cdot \sin(\theta)$. The resulting x and y values are output as coordinates for generating scatterplots. Similar to (a) and (b), each dataset includes four attributes: x and y coordinates, shape, and class.

Each multi-class dataset was generated as follows: We first defined the number of classes (n) in a dataset, and then randomly select one of the functions to generate a cluster with a certain number of data points (m) and assign them class labels ranging from 1 to n. and then repeat the process n times to generate n clusters. Thus, the final dataset contained a total of $M = m_1 + m_2 + \ldots + m_n$ data points, where m_n denotes the number of data points in cluster n. To increase variability, clusters were randomly translated (between -5 and 5) and rotated (from 0° to 360°) within the plotting area. Additionally, a portion of the data points within each cluster was randomly reassigned to different class labels to simulate anomalous data points. This reassignment was performed at three levels: small (0 or 1–3 points), medium (0 or 4–6 points), and large (0 or 7–9 points). In each reassignment process, the value 0 was

Table 1: Summary of datasets, including our synthetic datasets and those collected from Sedlmair's datasets.

	Our data	Sedlmair's data
No. of Datasets	81	243
No. of Well-separated Datasets	41	38
No. of Data Points	100–2500	77–7776
No. of Classes	2, 3, 5	2–53
Cluster Shapes	All shapes	All shapes

assigned with a 40% probability; thus, at each level, there was a 60% probability of generating a non-zero number of anomalous points and a 40% probability not assigning any anomalous points.

This generation process allowed for control over the number of clusters, data size, and the amount of anomalies. Multiple datasets were generated for each configuration to capture variability and support robust evaluation. The features used during dataset generation were the number of data points ($m = 100$, 200, or 500), the number of classes ($n = 2$, 3, or 5), and the level of anomalies (small, medium, or large). Each combination of these features was repeated three times. In total, we generated 81 synthetic datasets (3 data point groups $\times$ 3 class groups $\times$ 3 anomaly levels $\times$ 3 repetitions = 81).

The second part of the datasets (Sedlmair's datasets) is drawn from [37], including 81 real and synthetic datasets with more than two dimensions. These real datasets were obtained by contacting colleagues or from online repositories. Sedlmair's datasets include a categorical feature (class information), while the remaining features are numerical. We processed these datasets using three commonly used dimensionality reduction (DR) methods—t-SNE [41], PCA [17], and UMAP [29]—to reduce the datasets from more than two dimensions to two, using only the numerical features (excluding the class feature). All final datasets contain three features: `1`, `2`, and `class`. The features `1` and `2` represent the reduced dimensions, and `class` contains the categorical labels. Through DR methods we totally obtained 243 datasets (81 raw datasets $\times$ 3 DR methods) in Sedlmair's datasets.

Combining our synthetic datasets with Sedlmair's datasets, we collected a total of 324 datasets ($243+81=324$; see Table 1). Based on the preprocessing steps described above, we summarized the characteristics of the collected datasets as follows: (1) the number of data points ranges from 77 to 7776; and (2) the number of classes ranges from 2 to 53.

4.1.2 Scatterplot Generation

For each dataset, we generated scatterplots by varying two visual features: marker size and rendering order. Scatterplots were created using the Matplotlib library in Python, with a figure size of $(10, 8)$ inches and a resolution of 100 DPI. Margins were set as follows: left = 0.15, right = 0.75, top = 0.9, and bottom = 0.1. Marker size was set to four values: 10, 60, 110, and 160. Additionally, data points were rendered in different sorted sequences to simulate various overplotting conditions. We used three rendering order methods: random order, in which we shuffled the data and rendered with the random sequence; category-based rendering, in which data points are rendered one class at a time; and OM4AnI order, in which data points with higher anomaly index values are rendered later.

In total, this process yielded 3,888 scatterplots (324×3 rendering order methods $\times 4$ marker sizes) for analysis. However, some of the generated scatterplots did not exhibit clear clusters or showed no separation between clusters. One of the co-authors manually annotated the scatterplots based on the criteria illustrated in Figure 5 and selected datasets whose clusters were well separated (i.e., containing only 'Separated' correlations between each pair of clusters). As a result, we obtained 492 scatterplots (41 datasets) from our synthetic datasets and 456 scatterplots (38 datasets) from [37], resulting in a total of 948 scatterplots.

Figure 6 shows four scatterplots generated from two datasets: (a) and (b) are from one dataset, while (c) and (d) are from another. The difference between (a) and (b) lies in the marker size: (a) uses a size of 60, whereas (b) uses a size of 110. The difference between (c) and (d) is the rendering order: (c) uses a category-based rendering order, while (d) uses OM4AnI order. The *CCOP* serves as the ground truth, we introduce it in Section 4.1.3.

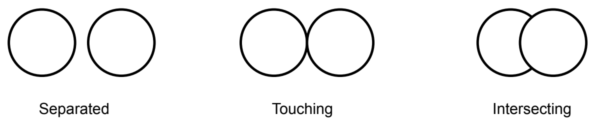

Fig. 5: Illustration of the relationship between each pair of clusters. Only datasets with clearly separated correlations between each cluster pair were retained in our evaluation.

4.1.3 Constructing the Ground Truth

Human judgment-based evaluation is not suitable as ground truth, as our methods are designed to quantify hidden anomalies. As noted by Sedlmair and Aupetit [36], "There might be patterns in the data that a human simply cannot directly 'see', or what she sees might just be an artifact of how the data has been abstracted and encoded." Moreover, constructing a ground truth by simply counting the number of data points occluded by others from different classes fails to accurately capture how much anomaly information is hidden, since many data points are only partially covered.

Therefore, in this paper, we constructed the ground truth by calculating the number of pixels in each marker that are occluded by markers from different classes; we refer to this as Cross-Class Occluded Pixels (*CCOP*). First, we implemented Step Two in Figure 2 to discretize the scatterplot into a matrix representation, where each cell in the matrix represents a pixel in the scatterplot. We then calculated how many pixels each marker covers and identified which specific pixels it covers. Finally, we computed the *CCOP* for each marker and summed the total *CCOP* across all markers.

4.2 Tasks and Measures

We designed our evaluation tasks according to three aspects: efficiency under different data sizes, overall effectiveness, and sensitivity to marker size and rendering order. For the efficiency of OM4AnI, we first grouped the datasets based on the number of data points and then analyze the time cost trends of each method as the number of data points increases.

For the effectiveness of VQMs, the task is grounded in the ground truth and focuses on identifying the correlation between the total number of *CCOP* and the values produced by each VQM method. Since the methods operate on different value scales (e.g., some range from 0 to 1, while others extend from 0 to $+\infty$) and exhibit different directions of correlation with anomaly information (e.g., some show a positive correlation while others show a negative correlation), We use the presence or absence of monotonic trends in the relationship between quality values and *CCOP* to compare the effectiveness of the methods.

For sensitivity to visual features, we assess whether the quality values change under different visual conditions. Specifically, we examine how the values produced by different methods vary across four marker sizes and three rendering orders. To do this, we grouped all scatterplots into different sets based on marker size or rendering order and then analyzed how the values produced by different methods vary across the different levels of each of the two visual features.

4.3 Baseline Methods

After reviewing related works on VQMs for measuring the overlap of scatterplots (see Section 2.2.2), we selected six baseline methods. Among the six baseline methods, only **DSC** is a task-based approach, while the other five methods are task-agnostic. Regarding the granularity level, **GDOD**, **KDOD**, and **DSC** operate at the data level, **VI** and **PBBOD** at the marker level, and **MP** at the pixel level. Thus, the three methods operating at the data level are not sensitive to marker size, whereas the other three baseline methods are sensitive to marker size. However, all these baseline methods are not sensitive to rendering order.

Table 2: Summary of our methods and baseline methods.

Method Groups	Abbreviation	Direction of Correlation	Computational Complexity	Value Range	Task Dependency	Granularity Level
Our methods	**MD-AnI**	Higher-is-Better	$O(n^2/k+sn+hw)$	[0,1]	Task-based	Pixel level
	LOF-AnI	Higher-is-Better	$O(n\log n+sn+hw)$	[0,1]	Task-based	Pixel level
	ALMD-AnI	Higher-is-Better	$O(n^2/k+sn+hw)$	[0,1]	Task-based	Pixel level
Baseline methods	**VI**	Higher-is-Better	$O(hwn)$	[0,1]	Task-agnostic	Marker level
	GDOD	Lower-is-Better	$O(n+hw)$	[0,1]	Task-agnostic	Data level
	PBBOD	Lower-is-Better	$O(n^2+hwn)$	[0,1]	Task-agnostic	Marker level
	MP	Lower-is-Better	$O(n+hw)$	$[0,\infty)$	Task-agnostic	Pixel level
	KDOD	Lower-is-Better	$O(hwn)$	$[0,\infty)$	Task-agnostic	Data level
	DSC	Lower-is-Better	$O(nk)$	[0,1]	Task-based	Data level

Note: "Higher-is-Better" indicates that higher metric values correspond to better quality with less overlap; "Lower-is-Better" indicates that lower measure scores correspond to better quality with more overlap. "n" denotes the data size; "k" denotes the number of classes; "s" denotes the number of cells occupied by a single marker; "h" denotes the number of cells in vertical dimension; "w" denotes the number of cells in horizontal dimension.
Abbreviations: **MD-AnI** = Mahalanobis Distance-AnI; **LOF-AnI** = LOF-AnI; **ALMD-AnI** = Average Linkage Method-AnI; **VI** = Visibility Index [40]; **GDOD** = Grid Density Overlap Degree [24]; **PBBOD** = Pairwise Bounding Box Overlap Degree [18]; **MP** = M_{pix} [42]; **KDOD** = Kernel Density Overlap Degree [32]; **DSC** = Distance Consistency [39].

Only **VI** produces higher values when the scatterplot has less overlap, while the other five methods exhibit the opposite correlation. Regarding computational complexity, **PBBOD** is the most time-consuming method compared to the other five baseline methods. We introduce the six methods for comparison (see Table 2):

(1) **VI** [40] is a task-agnostic approach to measure the percentage of glyphs that are not completely overlapped by other glyphs, thereby assessing the visibility of markers, with a computational complexity of $O(hwn)$ (see notation in Table 2). As it is computed at the marker level, the measure can be affected by marker size. The value ranges from 0 to 1, where 0 indicates low quality and 1 indicates perfect quality.
(2) **GDOD** [24] is a task-agnostic approach that calculates the number of grids in the scatterplot and maps data points onto these grids; if more than one data point falls within a single grid, overlap is identified. The computational complexity is $O(n+hw)$. As it is a data-level computation, the method is not sensitive to either marker size or rendering order. The measure ranges from 0 to 1, where 0 indicates poor quality and 1 indicates good quality.
(3) **PBBOD** [18, 33] is a task-agnostic approach to measure the ratio of the intersection area between each pair of markers to the smaller of the two marker areas. The final value is computed as the average of these ratios. The computational complexity is $O(n^2+hwn)$, as it requires computing interactions between each pair of data points. It is sensitive to marker size, since it is computed at the marker level, but it is not sensitive to rendering order. The measure ranges from 0 to 1, where 0 indicates low quality with high overlap, and 1 indicates high quality with minimal overlap.
(4) ***MP*** [42] is a task-agnostic approach that counts the number of pixels occupied by more than two markers as well as those occupied by at least one marker. The computational complexity is $O(n+hw)$. As a pixel-level computation, it is sensitive to marker size but not sensitive to rendering order. Its range extends from 0 to $+\infty$, where 0 indicates high quality with minimal overlap, and quality decreases as the measure value increases.
(5) **KDOD** [32] is a task-agnostic approach that estimates the density distribution of data points using kernel density estimation. The computational complexity is $O(hwn)$. It is not sensitive to marker size or rendering order, as it is computed at the data level. The measure ranges from 0 to $+\infty$, where 0 indicates better quality with less overlap.
(6) **DSC** [39] is a task-based approach that measures cluster separation by computing the proportion of data points whose nearest class centroid (center of mass) does not match their true class. Each point is assigned to the class whose centroid is closest in the scatterplot space. The computational complexity is $O(nk)$. It is not sensitive to marker size, as it is computed at the data level. The measure ranges from 0 to $+\infty$, where a lower value indicates better class separability.

4.4 Parameter Settings and Algorithm Implementation

We describe the parameter selection process for both our methods and the baseline methods. For all of our methods, we set β to 10 and λ to 0, as in this scenario, anomalies are data points located in a cluster to which they do not belong. This means that data points from different classes within the same cluster are the ones users want to observe. The number of pixels is determined by h and w (see Section 3.4), where we set h to 800 and w to 1000. These values correspond to the pixel dimensions used when generating scatterplots in Section 4.1.2, where $h = 100$ DPI $\times$ 8 inches and $w = 100$ DPI $\times$ 10 inches. For the baseline methods whose calculations depend on pixel or grid dimensions, we used the same number of pixels as in our methods ($w = 1000$ and $h = 800$) to ensure comparability.

OM4AnI was implemented in Python, utilizing the `scikit-learn` and `scipy` libraries to implement various machine learning algorithms, including Mahalanobis Distance t-SNE, UMAP, and PCA. For evaluating different VQMs, all computational tasks were executed in parallel on the Aire high-performance computing (HPC) platform at the University of Leeds, using standard CPU nodes equipped with AMD Dual 84-core 2.2 GHz (9634 Genoa-X) processors and 768 GB of DDR5 memory.

4.5 Results

We compared the performance of our three method with six baseline methods to evaluate their efficiency, effectiveness, and sensitivity under various conditions. In this section, we first evaluate the time cost of each method under varying numbers of data points. We then assess the effectiveness of OM4AnI by analyzing the correlation between its results and the number of *CCOP*. Finally, we examine the sensitivity of OM4AnI to marker size and rendering order.

4.5.1 Time Cost Analysis

We measured the computational time of all methods across varying data sizes (see Figure 7). For each method, we imposed a maximum time limit of 15 seconds; if computation reached this limit, we terminated the process and proceeded to the next instance. This threshold reflects the assumption that users are unlikely to wait more than 15 seconds for the results of a VQM.

Figure 7 plots computational time against data size, with each line representing a different method. Notably, **PBBOD** exhibits substantial variation as data size increases, reaching 4.5 seconds even at a small data size of approximately 300. The **PBBOD** curve terminates at this point, as it fails to produce results within 4.5 seconds under our experimental conditions. In contrast, the remaining methods maintain runtimes below 2 seconds until the data size reaches 8,000. Beyond 10,000, **ALMD-AnI** displays a steeper increase in computational time relative to the other methods, approaching approximately 4 seconds at a data size of 14,000. Excluding **PBBOD**, our three proposed methods incur higher computational times than the baseline methods.

4.5.2 Effectiveness Analysis

To evaluate the effectiveness of OM4AnI, we plotted a violin chart in Figure 8 comprising nine subplots. The first three subplots correspond

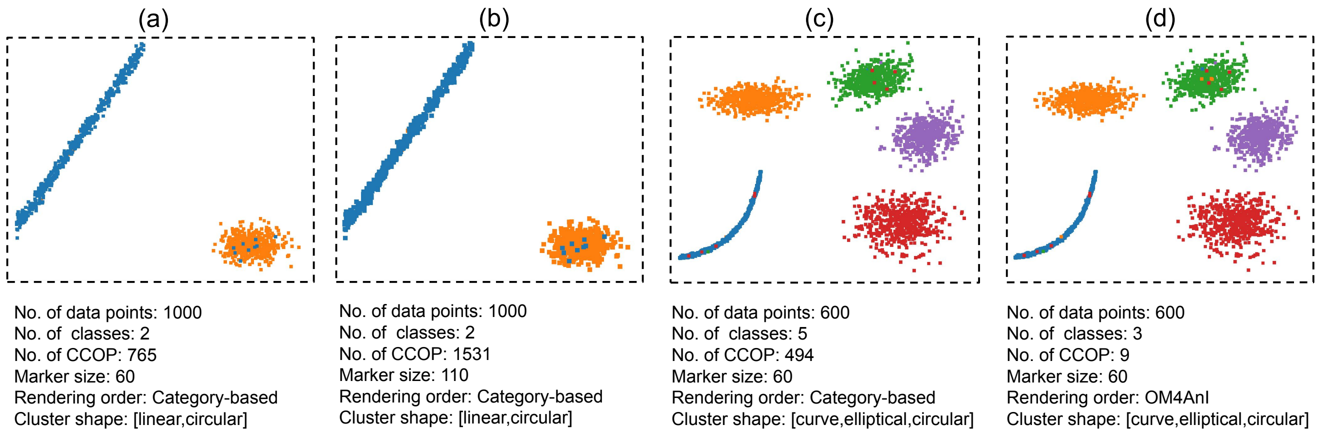

Fig. 6: Examples of generated scatterplots. Each scatterplot is characterized by six features: number of data points, number of classes, number of *CCOP*, marker size, rendering order, and cluster shape. The difference between (a) and (b) is the marker size; (b) has a larger marker size, which leads to more *CCOP*. The difference between (c) and (d) lies in the rendering order method: the OM4AnI order reduces the number of *CCOP*.

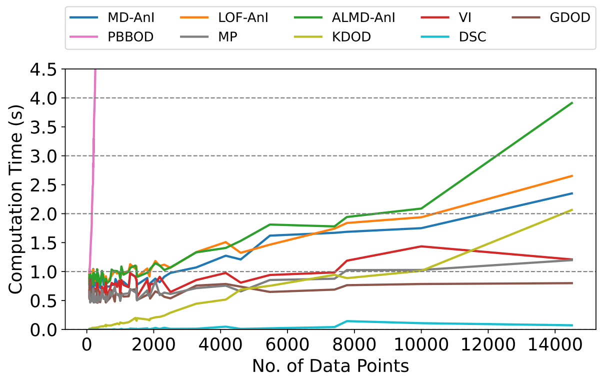

Fig. 7: Computational time versus dataset size. The x-axis denotes the number of data points, and the y-axis denotes computational time (in seconds). Each curve represents a different method; **PBBOD** is the most time-consuming.

to our method—each employing a different anomaly-index computation strategy—while the remaining six depict baseline methods. Because VQM values differ in both scale and correlation direction with overlap extent, our primary objective in analyzing these violin plots is to identify monotonic trends between No. of *CCOP* and quality scores generated from different methods. Additionally, each red dot and its associated label reference the matching scatterplots in Figure 6, allowing us to inspect *CCOP* values and the corresponding scatterplot results for four selected examples.

Overall, the three methods exhibit more consistent monotonic trends than the six baseline methods. When the *CCOP* count is zero, all three methods cluster around 1, although their minimum values drop below 0.25. As the *CCOP* count increases, the distributions for our methods display a pronounced decreasing trend, with only a slight deviation at the final x-axis value (10 433). The four scatterplot examples—indicated by red dots—can be ordered by decreasing quality (VQM) values as follows: d (average *CCOP* = 9), c (average *CCOP* = 494), a (average *CCOP* = 765), and b (average *CCOP* = 1 531). This ordering highlights a strong negative correlation between quality (VQM) values and *CCOP*, thereby validating the monotonic consistency of our methods.

Compared to our methods, the baseline methods are less effective. **VI** does not exhibit a monotonic trend, and many of the results generated by **GDOD** are close to 0. Although **PBBOD** shows a relatively better monotonic trend, it lacks results for two groups corresponding to *CCOP* values of 4943 and 10433. **MP** also fails to demonstrate a monotonic trend. Similarly, **KDOD** does not exhibit a monotonic trend, with most values clustering around 0.03. **DSC** likewise does not display a clear monotonic trend, with most of its values concentrated near 0. The red dots further illustrate the weakness of the baseline methods: in **VI**, **GDOD**, **MP**, **KDOD**, and **DSC**, the quality values do not show meaningful changes with respect to the number of *CCOP*. In contrast, **PBBOD** does not include red dots because it did not compute the results for the four scatterplots, which is consistent with the time cost results in Section 4.5.1, indicating that **PBBOD** failed to produce results in many cases within 15-second computation under our experiment environment.

4.5.3 Sensitivity Analysis

We analyzed the sensitivity of the methods to two visual features to assess how they respond to changes in visual feature values. As mentioned earlier, for each dataset, we generated scatterplots using four marker sizes and three rendering orders. In this analysis, we divided all generated scatterplots based on the levels of marker size or rendering order, depending on which visual feature's sensitivity was being examined. We then plotted violin charts to show how the VQM values are distributed at each level in one of the two visual features and how they change across different levels of each feature (see Figure 9 and Figure 10), in order to analyze the sensitivity of the methods to these visual features.

Figure 9 illustrates the sensitivity of different methods to marker size. Similar to Figure 8, the first row presents our three methods, while the subsequent violin plots display the baseline methods. Each violin plot depicts the full distribution of VQM values across all scatterplot configurations for the corresponding method. Additionally, each violin plot is annotated with labels referencing Figure 6a and Figure 6b, which are generated from the same dataset and share identical features except for marker size—60 for (a) and 110 for (b).

To analyze the sensitivity of the methods to marker size, we examined the distribution of VQM values across different marker sizes. All of our methods show a continuous decrease in values from size 10 to 160. **VI** also demonstrates a continuous decrease, indicating its sensitivity to marker size. Two other baseline methods—**PBBOD** and **MP**—exhibit sensitivity to marker size, with their values increasing as marker size increases. In contrast, the remaining three baseline methods—**GDOD**, **KDOD**, and **DSC**—do not show sensitivity to marker size, as their distributions remain largely unchanged. By examining the red dots that correspond to the two scatterplots, from marker size 60 to marker size 110, all three of our methods and **VI** show decreasing trends from scatterplot Figure 6a to Figure 6b, while **MP** shows an increasing trend. **GDOD**, **KDOD**, and **DSC** show no change when the marker size changes. In contrast, **PBBOD** did not compute the VQM results for the two scatterplots within in 15 seconds due to its high time cost.

Figure 10 shows the sensitivity of different methods to rendering order. The first difference from Figure 9 is the x-axis of each subplot;

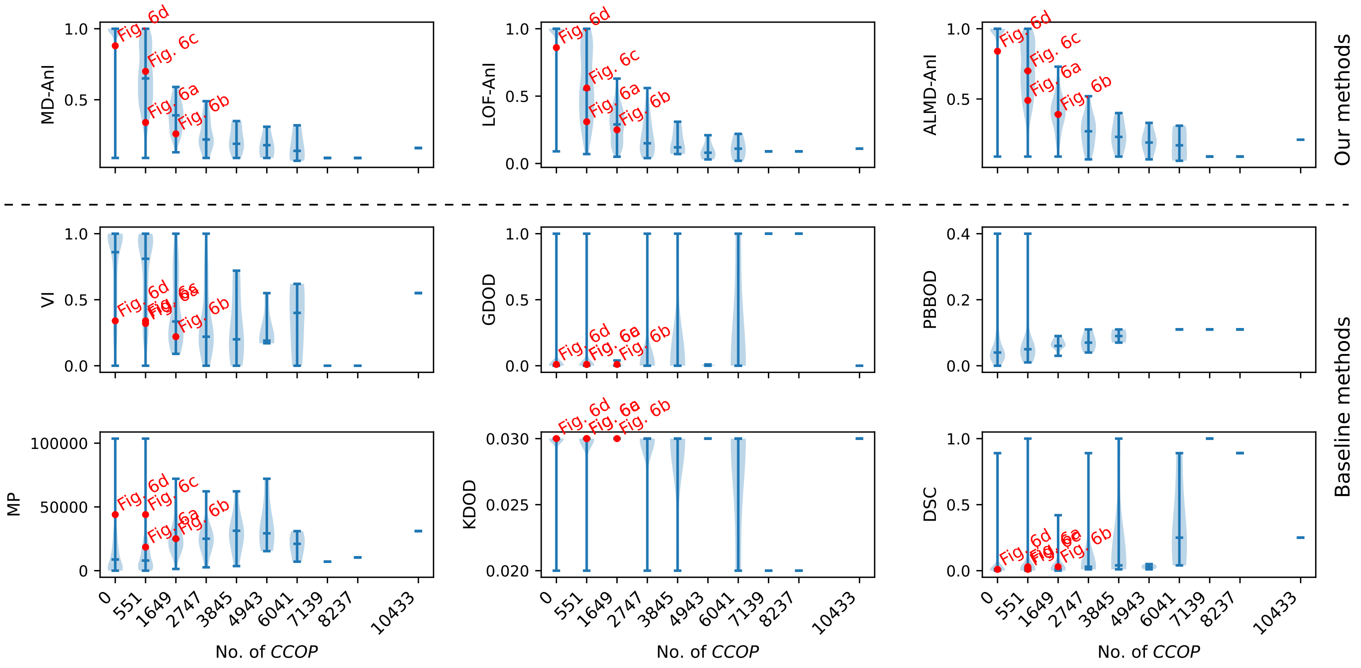

Fig. 8: Distribution of VQM values vs the average number of *CCOP*. The x-axis denotes the average *CCOP* count within each range, and the y-axis denotes the VQM values for each method. The three proposed methods (first row) exhibit more consistent and monotonic trends compared to the six baseline methods; only at the largest *CCOP* value (10,433) they show a slight deviation. Red markers correspond to the scatterplots in Figure 6, illustrating that our methods yield distinctly separable quality values, whereas the baseline methods do not.

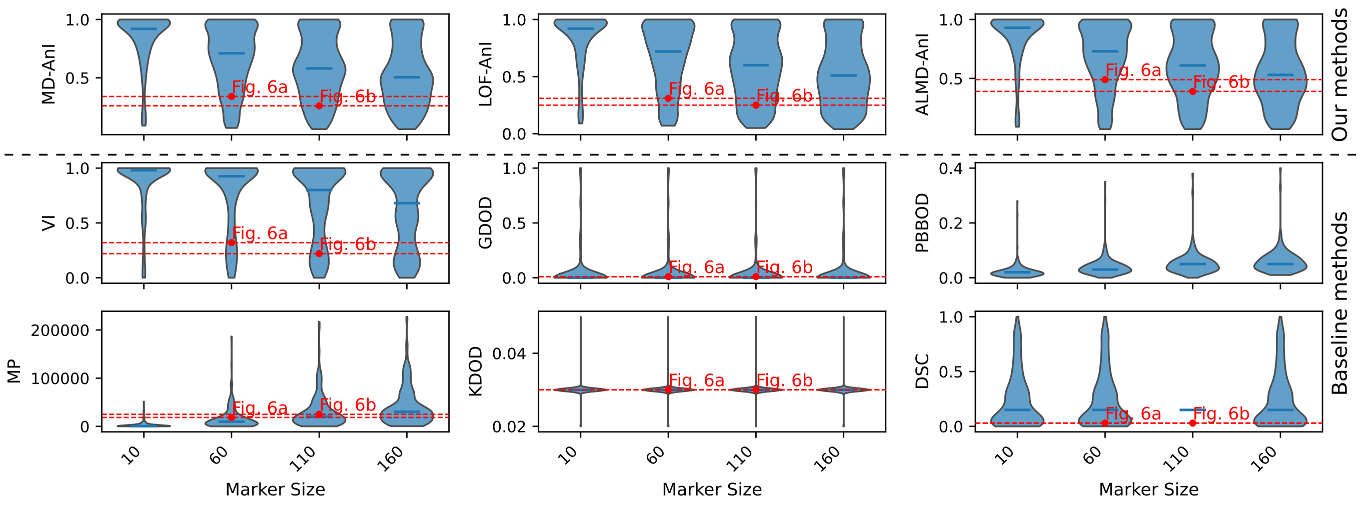

Fig. 9: Distribution of VQM values across different marker sizes. The x-axis denotes marker size, and the y-axis denotes VQM values for each method. All three proposed methods, as well as **VI**, **PBBOD**, and **MP**, exhibit sensitivity to marker size, whereas **GDOD**, **KDOD**, and **DSC** remain largely invariant across the four tested sizes. Red markers correspond to the scatterplots in Figure 6, illustrating the evolution of VQM values as marker size increases from 60 (Fig. 6a) to 110 (Fig. 6b).

in this case, the x-axis represents the rendering order, which includes three categories: OM4AnI, category-based, and random. The second difference is the inclusion of red dots indicating the corresponding scatterplots. In this case, we added scatterplots Figure 6c and Figure 6d, as they are based on the same datasets and features, except for the rendering order.

Based on the violin-plot distributions in Figure 10, only our three proposed methods exhibit sensitivity to rendering order: their VQM values decrease slightly as the rendering strategy shifts from OM4AnI order to category-based to random. In contrast, all six baseline methods remain essentially unchanged across the three orders. The red markers for scatterplots (d) and (c) corroborate this finding: only our methods' quality values differ between OM4AnI and category-based rendering, whereas the baseline methods show no variation. Additionally, **PBBOD** fails to compute results for these two scatterplots within the 15-second time limit due to its high computational cost.

5 Discussion and Limitations

Effectiveness and Efficiency. The evaluation showed that our methods outperformed the baseline methods. Specifically, **ALMD-AnI** and **MD-AnI** consistently achieved better continuously monotonic trends with respect to the number of *CCOP*, outperforming all the baseline methods (see Figure 8). This is primarily because **ALMD-AnI** and **MD-AnI** accounts for all the data, allowing it to better handle complex

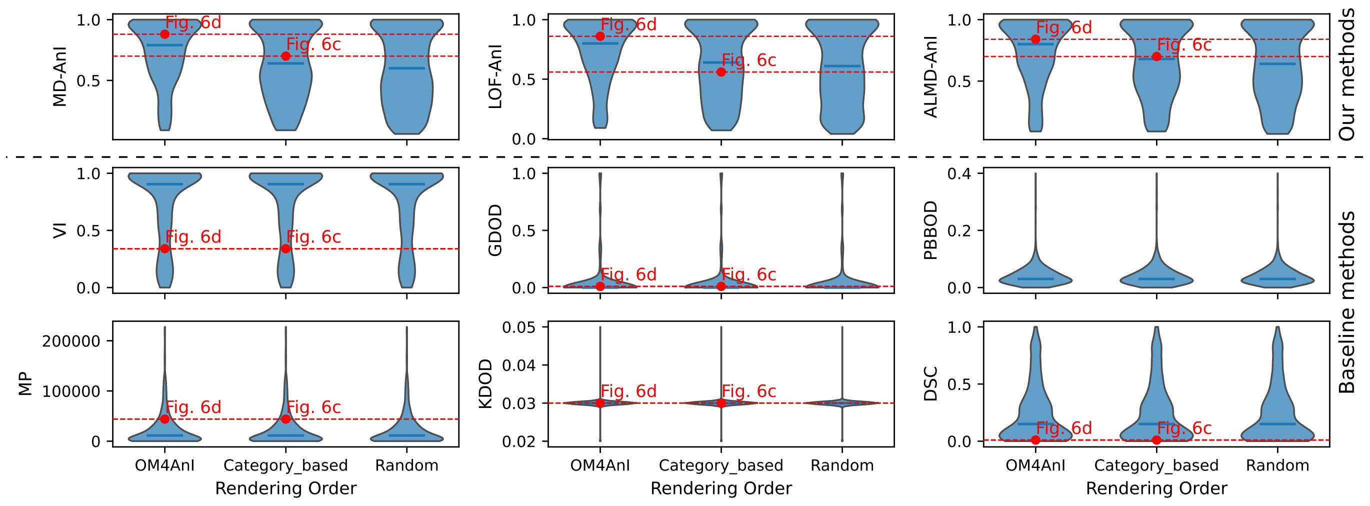

Fig. 10: Distribution of VQM values across different rendering orders. The x-axis denotes rendering order (OM4AnI order, category-based order, random order), and the y-axis denotes VQM values for each method. Only the three proposed methods exhibit sensitivity to rendering order, with their values decreasing from OM4AnI to category-based to random order; all baseline methods remain largely unchanged. Red markers correspond to the scatterplots in Figure 6, illustrating the change in VQM values as rendering order shifts from OM4AnI order (Figure 6d) to category-based (Figure 6c).

distribution shapes.

In terms of efficiency, compared to the remaining baseline methods (except for **PBBOD**), our methods required more computation time. Nevertheless, based on the experiments, when the dataset size reached 14,000, our methods took only 1–2 seconds longer than the other baseline methods, which is considered acceptable. In general, Figure 7 shows results that are consistent with the theoretical computational complexity: **PBBOD** is the most time-consuming method with $O(n^2)$ complexity, followed by our three methods with $O(n^2/k)$ and $O(n \log n)$ complexities. The remaining baseline methods have lower complexity, at $O(n)$.

Our methods consider three candidates while computing the anomaly index: Mahalanobis distance, LOF, and the Average Linkage method based on Euclidean distance. Mahalanobis distance quantifies how far a point is from the center of a data distribution while accounting for the spread, scale, and correlation between the two variables (x and y coordinates). Therefore, using Mahalanobis distance to calculate a point's anomaly index relative to its cluster partially reflects the underlying distribution. This allows it to accurately assess anomalies in both spherical and elliptical clusters, including rotated elliptical clusters. LOF is a density-based anomaly detection method that compares the local density of each point to that of its neighbors. It does not assume any particular distribution or trend, and instead estimates the anomaly index solely based on local density. The Average Linkage method, based on Euclidean distance, estimates the anomaly index by computing each point's average Euclidean distance to all other points. This method may misclassify points in elongated clusters as outliers, making it less effective for data with elongated distributions. There is no method that perfectly fits all distribution scenarios. In future work, we plan to enhance our quality metrics by calculating the anomaly index in a way that better accounts for each cluster's shape and its corresponding anomaly patterns. This may involve either automatically selecting an appropriate anomaly index computation strategy for each class in a scatterplot or allowing users to choose a method based on their specific task.

Sensitivity to marker size and rendering order. Our methods show good sensitivity to both marker size and rendering order, as the computation considers the number of pixels each marker occupies and their rendering sequence. All the other baseline methods are not sensitive to rendering order, as they do not consider which marker should be rendered earlier or later. Regarding sensitivity to marker size, among all our methods and the baseline methods, only three methods are not sensitive to marker size (**GDOD**, **KDOD**, and **DSC**), which aligns with the theoretical explanation introduced in Section 4.3.

Usability of OM4AnI. As introduced earlier, our method provides a quantitative value for a scatterplot, enabling users to determine whether anomalous data points are occluded. Therefore, users can use OM4AnI as a metric to select the best-performing scatterplot from multiple candidates generated with different parameters and visual factors. Beyond selection, our method can also be employed as a metric for evaluating scatterplots for specific task, as demonstrated in the work of Sedlmair and Aupetit [36]. Another application of our method is to improve scatterplot design by adjusting parameters and visual features (Please refer to Section 3 of the supplementary material for an example application). Since our method can compute the quality of scatterplot without rendering the scatterplot, it can be easily incorporated as an objective function in optimization algorithms for automatically generating improved scatterplots.

Scalability. Multi-class scatterplots typically use different colors as the third visual channel (i.e., using color to distinguish between classes). However, there are alternative ways to represent this channel, such as using different shapes. Since OM4AnI computes quality scores based on pixels, it is sensitive to marker shapes. Therefore, OM4AnI is scalable to other multi-class scatterplots that use shapes as the third visual channel. Besides, our current method assumes that the labels for false positive and false negative anomalies are unknown. We compute anomaly indexes based on the relative positions of individual data points within their respective clusters. However, if the anomaly labels and their types are known in advance, our method can be adapted accordingly. The main idea is to assign anomaly indexes of either 0 or 1 to each data point. We then calculate the total number of pixels corresponding to markers labeled as 1. The greater the number of pixels labeled as 1 that are occluded, the lower the quality value of the scatterplot. We plan to implement this approach in future work.

Limitations. (1) Our method has limitations when considering the opacity of scatterplots. In the current approach, we assume all data points are fully opaque, meaning that if a data point is occluded by others—regardless of how many—it will be considered completely invisible to users. In future work, we plan to extend our current approach by considering the opacity of marker sizes. We propose the following strategies to improve our method. First, we aim to utilize the relevant theory of opacity markers to analyze the correlation between human perception and the cumulative effect of transparent markers. Exploring this relationship will help us understand how opacity values and the number of transparent markers influence scatterplot readability, and consequently, affect the reliability of our method. Second, we also plan to conduct a user study to further investigate the correlation between opacity, the number of occluded markers, and marker visibility. Based on the findings, we intend to derive practical guidelines for applying opacity in various overplotting scenarios, thereby supporting the development of more effective VQM. (2) When marker sizes are reduced, our measuring score increases because overplotting is minimized. However, this leads to a limitation: if the marker size becomes so small that the markers are no longer visually distinguishable to users, the output value may produce unrealistically high values due to the near absence of overplotting. This is not practical for real-world applications, where human visual perception must be considered. (3) Our current datasets contain fewer than 20,000 data points. The computational time cost may increase as the data size becomes larger (e.g., in the millions). In such cases, it may be necessary to explore alternative efficient methods for computing the anomaly index. (4) Our method performs well on well-separated multi-class scatterplots. However, it has limitations in cases with low class separability. We did not include such unseparated cases in our evaluation because, to the best of our knowledge, alternative approaches may be more suitable for identifying anomalies when multi-class scatterplots exhibit significant overlap.

6 Conclusion

This paper introduced a visual quality measure (VQM) —OM4AnI for assessing overlap in the context of anomaly identification in multi-class scatterplots. The proposed method first computes the anomaly index values using embedding methods (e.g., Mahalanobis distance), then projects all markers onto a grid (each cell in the grid represents a pixel) while incorporating key visual features such as marker size and rendering order. It then calculates cell-level quality values, which are aggregated to produce the final quality score. Evaluation results comparing OM4AnI with six baseline methods show that OM4AnI outperforms them by exhibiting more monotonic trends with respect to the ground truth and greater sensitivity to rendering order. Overall, OM4AnI provides a meaningful assessment of how well a scatterplot supports anomaly identification. It also shows promise for applications in scatterplot evaluation and optimization through automated parameter tuning.

Acknowledgments

This research was supported by the Engineering and Physical Sciences Research Council (EP/X029689/1). We are grateful to Michael Sedlmair for providing access to his raw datasets.

The datasets and code used in this study are available at https://github.com/llqsee/O4AnI. Supplementary material can be accessed at https://liqunliu.com/research/O4AnI/Supp.pdf.

References

[1] M. M. Abbas, M. Aupetit, M. Sedlmair, and H. Bensmail. *ClustMe*: A visual quality measure for ranking monochrome scatterplots based on cluster patterns. *Computer Graphics Forum*, 38(3):225–236, 2019. doi: 10.1111/cgf.13684 2

[2] M. Aupetit and M. Sedlmair. SepMe: 2002 new visual separation measures. In *2016 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 1–8. IEEE, Taipei, Taiwan, 2016. doi: 10.1109/PACIFICVIS.2016.7465244 1, 2

[3] M. Behrisch, M. Blumenschein, N. W. Kim, L. Shao, M. El-Assady, J. Fuchs, D. Seebacher, A. Diehl, U. Brandes, H. Pfister, T. Schreck, D. Weiskopf, and D. A. Keim. Quality metrics for information visualization. *Computer Graphics Forum*, 37(3):625–662, 2018. doi: 10.1111/cgf.13446 2

[4] E. Bertini and G. Santucci. By chance is not enough: preserving relative density through non uniform sampling. In *Proceedings. Eighth International Conference on Information Visualisation, 2004. IV 2004.*, pp. 622–629. IEEE, London, UK, 2004. doi: 10.1109/IV.2004.1320207 2

[5] E. Bertini and G. Santucci. Quality metrics for 2D scatterplot graphics: Automatically reducing visual clutter. In A. Butz, A. Krüger, and P. Olivier, eds., *Smart Graphics*, vol. 3031 of *Lecture Notes in Computer Science*, pp. 77–89. Springer, Berlin, Germany, 2004. doi: 10.1007/978-3-540-24678-7_8 2

[6] E. Bertini and G. Santucci. Improving 2D scatterplots effectiveness through sampling, displacement, and user perception. In *Ninth International Conference on Information Visualisation (IV'05)*, pp. 826–834. IEEE, London, UK, 2005. doi: 10.1109/IV.2005.62 1, 2, 3

[7] E. Bertini and G. Santucci. Visual quality metrics. In *BELIV'06: Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaLuation methods for Information Visualization*, pp. 1–5. Association for Computing Machinery, Venice, Italy, 2006. doi: 10.1145/1168149.1168159 2

[8] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, pp. 93–104. Association for Computing Machinery, New York, NY, USA, 2000. doi: 10.1145/342009.335388 3

[9] D. B. Carr, R. J. Littlefield, W. L. Nicholson, and J. S. Littlefield. Scatterplot matrix techniques for large N. *Journal of the American Statistical Association*, 82(398):424–436, 1987. doi: 10.2307/2289444 2

[10] A. Chatzimparmpas, K. Kucher, and A. Kerren. Visualization for trust in machine learning revisited: The state of the field in 2023. *IEEE Computer Graphics and Applications*, 44(3):99–113, 2024. doi: 10.1109/MCG.2024.3360881 2

[11] A. Chatzimparmpas, R. M. Martins, I. Jusufi, K. Kucher, F. Rossi, and A. Kerren. The state of the art in enhancing trust in machine learning models with the use of visualizations. *Computer Graphics Forum*, 39(3):713–756, 2020. doi: 10.1111/cgf.14034 2

[12] P. J. Clark and F. C. Evans. Distance to nearest neighbor as a measure of spatial relationships in populations. *Ecology*, 35(4):445–453, 1954. doi: 10.2307/1931034 1, 3

[13] L. Deng. The MNIST database of handwritten digit images for machine learning research [Best of the Web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. doi: 10.1109/MSP.2012.2211477 2

[14] G. Ellis and A. Dix. The plot, the clutter, the sampling and its lens: occlusion measures for automatic clutter reduction. In *Proceedings of the working conference on Advanced visual interfaces - AVI '06*, p. 266. Association for Computing Machinery, Venezia, Italy, 2006. doi: 10.1145/1133265.1133318 3

[15] G. Ellis and A. Dix. A taxonomy of clutter reduction for information visualisation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1216–1223, 2007. doi: 10.1109/TVCG.2007.70535 2, 3

[16] T. Fujiwara, O. H. Kwon, and K. L. Ma. Supporting analysis of dimensionality reduction results with contrastive learning. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):45–55, 2020. doi: 10.1109/TVCG.2019.2934251 1

[17] F. L. Gewers, G. R. Ferreira, H. F. D. Arruda, F. N. Silva, C. H. Comin, D. R. Amancio, and L. D. F. Costa. Principal component analysis: A natural approach to data exploration. *ACM Computing Surveys*, 54(4):1–34, 2022. doi: 10.1145/3447755 6

[18] G. M. Hilasaca, W. E. Marcílio-Jr, D. M. Eler, R. M. Martins, and F. V. Paulovich. A grid-based method for removing overlaps of dimensionality reduction scatterplot layouts. *IEEE Transactions on Visualization and Computer Graphics*, 30(8):5733–5749, 2024. doi: 10.1109/TVCG.2023.3309941 3, 7

[19] T. Jaunet, R. Vuillemot, and C. Wolf. DRLViz: Understanding decisions and memory in deep reinforcement learning. *Computer Graphics Forum*, 39(3):49–61, 2020. doi: 10.1111/cgf.13962 1

[20] H. Jeon, Y.-H. Kuo, M. Aupetit, K.-L. Ma, and J. Seo. Classes are not clusters: Improving label-based evaluation of dimensionality reduction. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):781–791, 2024. doi: 10.1109/TVCG.2023.3327187 1, 2

[21] H. Jeon, G. J. Quadri, H. Lee, P. Rosen, D. A. Szafir, and J. Seo. *CLAMS*: A cluster ambiguity measure for estimating perceptual variability in visual clustering. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):770–780, 2024. doi: 10.1109/TVCG.2023.3327201 2

[22] J. Johansson and M. Cooper. A screen space quality method for data abstraction. *Computer Graphics Forum*, 27(3):1039–1046, 2008. doi: 10.1111/j.1467-8659.2008.01240.x 2

[23] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. Chau. ActiVis: Visual exploration of industry-scale deep neural network models. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):88–97, 2018. doi: 10.1109/TVCG.2017.2744718 1

[24] D. A. Keim, M. C. Hao, U. Dayal, H. Janetzko, and P. Bak. Generalized scatter plots. *Information Visualization*, 9(4):301–311, 2010. doi: 10.1057/ivs.2009.34 3, 7

[25] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. doi: 10.1162/neco.1989.1.4.541 2

[26] R. Li, C. Yin, S. Yang, B. Qian, and P. Zhang. Marrying medical domain knowledge with deep learning on electronic health records: A deep visual analytics approach. *Journal of Medical Internet Research*, 22(9):e20645, 2020. doi: 10.2196/20645 1

[27] L. Liu, R. Vuillemot, P. Rivière, J. Boy, and A. Tabard. Generalizing OD maps to explore multi-dimensional geospatial datasets. *The Cartographic Journal*, 61(1):49–68, 2024. doi: 10.1080/00087041.2024.2325191 2

[28] A. Mayorga and M. Gleicher. Splatterplots: Overcoming overdraw in scatter plots. *IEEE Transactions on Visualization and Computer Graphics*, 19(9):1526–1538, 2013. doi: 10.1109/TVCG.2013.65 2

[29] L. McInnes, J. Healy, and J. Melville. UMAP: Uniform Manifold Approximation and Projection for dimension reduction. Preprint, Sep 2020, arXiv:1802.03426 [stat.ML]. doi: 10.48550/arXiv.1802.03426 6

[30] G. J. McLachlan. Mahalanobis distance. *Resonance*, 4(6):20–26, 1999. doi: 10.1007/BF02834632 3

[31] C. Park, S. Yang, I. Na, S. Chung, S. Shin, B. C. Kwon, D. Park, and J. Choo. VATUN: Visual analytics for testing and understanding convolutional neural networks. In *EuroVis 2021 – Short Papers*, pp. 7–11, 2021. doi: 10.2312/EVS.20211047 1

[32] E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962. doi: 10.1214/aoms/1177704472 3, 7

[33] A. B. Ramos-Guajardo, G. González-Rodríguez, and A. Colubi. Testing the degree of overlap for the expected value of random intervals. *International Journal of Approximate Reasoning*, 119:1–19, 2020. doi: 10.1016/j.ijar.2019.12.012 7

[34] A. Sarikaya and M. Gleicher. Scatterplots: Tasks, data, and designs. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):402–412, 2018. doi: 10.1109/TVCG.2017.2744184 1, 2

[35] M. Schaefer, L. Zhang, T. Schreck, A. Tatu, J. A. Lee, M. Verleysen, and D. A. Keim. Improving projection-based data analysis by feature space transformations. In P. C. Wong, D. L. Kao, M. C. Hao, and C. Chen, eds., *Visualization and Data Analysis 2013. Proceedings of SPIE-IS&T Electronic Imaging*, vol. 8654, p. 86540H, 2013. doi: 10.1117/12.2000701 3

[36] M. Sedlmair and M. Aupetit. Data-driven evaluation of visual quality measures. *Computer Graphics Forum*, 34(3):201–210, 2015. doi: 10.1111/cgf.12632 2, 6, 10

[37] M. Sedlmair, A. Tatu, T. Munzner, and M. Tory. A taxonomy of visual cluster separation factors. *Computer Graphics Forum*, 31(3):1335–1344, 2012. doi: 10.1111/j.1467-8659.2012.03125.x 5, 6

[38] Q. Shen, Y. Wu, Y. Jiang, W. Zeng, A. K. H. Lau, A. Vianova, and H. Qu. Visual interpretation of recurrent neural network on multi-dimensional time-series forecast. In *2020 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 61–70. IEEE, Tianjin, China, 2020. doi: 10.1109/PacificVis48177.2020.2785 1

[39] M. Sips, B. Neubert, J. P. Lewis, and P. Hanrahan. Selecting good views of high-dimensional data using class consistency. *Computer Graphics Forum*, 28(3):831–838, 2009. doi: 10.1111/j.1467-8659.2009.01467.x 2, 7

[40] D. K. Urribarri and S. M. Castro. Prediction of data visibility in two-dimensional scatterplots. *Information Visualization*, 16(2):113–125, 2017. doi: 10.1177/1473871616638892 1, 3, 7

[41] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. https://jmlr.org/papers/v9/vandermaaten08a.html. 2, 6

[42] C. van Onzenoodt, A. Huckauf, and T. Ropinski. On the perceptual influence of shape overlap on data-comparison using scatterplots. *Computers & Graphics*, 90:169–181, 2020. doi: 10.1016/j.cag.2020.05.028 3, 7

[43] C. Waldeck and D. Balfanz. Mobile liquid 2D scatter space (ML2DSS). In *Proceedings. Eighth International Conference on Information Visualisation, 2004. IV 2004*, pp. 494–498. London, UK, 2004. doi: 10.1109/IV.2004.1320190 2

[44] C. Wang, X. Wang, and K.-L. Ma. Visual summary of value-level feature attribution in prediction classes with recurrent neural networks. Preprint, Aug 2020, arXiv:2001.08379 [cs.LG]. doi: 10.48550/arXiv.2001.08379 1

[45] T. Zahavy, N. Ben-Zrihem, and S. Mannor. Graying the black box: Understanding DQNs. In *Proceedings of the 33rd International Conference on Machine Learning*, vol. 48 of *Proceedings of Machine Learning Research*, pp. 1899–1908. PMLR, New York, NY, USA, 2016. https://proceedings.mlr.press/v48/zahavy16.html. 1

Liqun Liu is Research Fellow in the School of Computer Science at the University of Leeds. His research interests include Human-Computer Interaction (HCI), data visualization, and explainable AI using visualization techniques, particularly focusing on spatiotemporal data in traffic-related applications. Before joining the University of Leeds, he completed his Ph.D. in Computer Science at Université de Lyon in France.

Leonid Bogachev is Professor in the Department of Statistics, University of Leeds. He has Ph.D. in Probability & Statistics, and was a Leverhulme Research Fellow and a Turing Fellow. His broad expertise is in applied probability and statistics, including extreme value modeling and machine learning. He is a co-founder of a start-up 4-Xtra Technologies Ltd. and IP co-inventor, with EU and US patents pending.

Mahdi Rezaei is Associate Professor of Computer Vision and Machine Learning and the team leader of the Computer Vision Research Group at the University of Leeds, Institute for Transport Studies. He has published over 65 peer-reviewed journal articles and conference papers focusing on the development of Autonomous Vehicles (AVs), Driver Monitoring Systems (DMS), Explainable AI (XAI), and Intelligent Transportation Systems.

Nishant Ravikumar is Lecturer in AI at the School of Computer Science, University of Leeds. He completed his PhD in 2017 at the University of Sheffield. Following this, he spent two years as a postdoctoral researcher at the Pattern Recognition Lab, Friedrich-Alexander-University Erlangen-Nuremberg, Germany, before joining the University of Leeds in 2019.

Arjun Khara is Associate Professor of Communication, at the School of Design, University of Leeds. He completed his PhD at the Department of Typography and Graphic Communication, University of Reading. He has an MA in Computer Games Art and Design from Goldsmiths, University of London (Valedictorian Orator).

Mohsen Azarmi is a PhD student in Transport Studies at Institute for Transport Studies, University of Leeds. He holds a BSc and MSc in Computer Science and his research interest mainly focuses on applications of Computer Vision and ML in Autonomous Vehicles, traffic perception, and driver behavior monitoring.

Roy Ruddle is Professor of Computing at the University of Leeds and Director of Research Technology at the Leeds Institute for Data Analytics (LIDA). He has a multidisciplinary background, and has been a Humboldt Research Fellow and a Turing Fellow. He conducts basic and applied research in visualization, human-computer interaction, data quality and visual analytic workflows.