# Parallel Clusters: Visual Comparison of Embeddings Based on Multi-Scale Neighborhood Analysis

Zehua Yu, Xiaoyi Li, Pengfei Liu, and Jun Tao, *Member, IEEE*

***Abstract*—Understanding embedding relationships is crucial for neural network interpretability, structural analysis, and data exploration. However, visually comparing embeddings is challenging due to the difficulty of mentally aligning structures across views. In this paper, we address this challenge by constructing multiple hierarchies of data points from different perspectives to facilitate meaningful comparisons. We introduce Parallel Clusters (ParaClus), a visual analytics system that enables multi-scale exploration of embedding structures. This is achieved through cluster formations across embeddings and attributes, along with an adaptive thresholding mechanism that defines neighborhoods. By dynamically adjusting this threshold, users can explore structures at varying scales. Our interface employs a parallel axes design, where clusters derived from the same embedding or attribute are aligned along individual axes. This layout allows users to easily compare neighboring clusters and observe relationships across embeddings. Furthermore, interactive subdivision mechanisms enable users to refine clusters based on their connections to other clusters, providing deeper insights into structural dependencies. Additionally, our system seamlessly integrates labels and scalar attributes into the clustering process, offering a unified approach to analyzing multi-attribute, time-varying, and network-derived embeddings. We evaluate the effectiveness of ParaClus through expert assessments and case studies.**

***Index Terms*—Embedding comparison, unified comparison, multi-scale neighborhood, hierarchical exploration.**

## I. Introduction

THE increasing complexity of machine learning models, particularly deep neural networks, has underscored the importance of understanding the relationships between data points in high-dimensional spaces [4], [42]. Deep learning techniques represent raw data, ranging from natural languages to images, into high-dimensional embedding spaces. While powerful in computation, embeddings are difficult to interpret, especially when one needs to compare structures across different models, features, or scales.

Received 12 September 2025; revised 17 December 2025; accepted 7 January 2026. Date of publication 15 January 2026; date of current version 13 February 2026. This work was supported by the National Natural Science Foundation of China under Grant 62172456 and Grant 62372484. Recommended for acceptance by M. Sedlmair. *(Corresponding author: Jun Tao.)*

Zehua Yu, Xiaoyi Li, and Pengfei Liu are with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510275, China (e-mail: yuzh58@mail2.sysu.edu.cn; lixy2226@mail2.sysu.edu.cn; liupf7@mail2.sysu.edu.cn).

Jun Tao is with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510275, China, and also with National Supercomputer Center in Guangzhou, Guangzhou 510330, China (e-mail: taoj23@mail.sysu.edu.cn).

Digital Object Identifier 10.1109/TVCG.2026.3654590

Comparing embeddings is crucial for several reasons. It reveals structures captured from the same data by different models or training strategies. This helps align learned representations with human-interpretable attributes, enabling validation of whether embeddings reflect domain knowledge and providing a way to assess robustness and generalization. In multimodal or heterogeneous data scenarios, comparison further clarifies whether distinct feature spaces capture consistent or complementary patterns.

However, intuitive comparison remains challenging. Traditional methods such as t-SNE [62] and PCA [38] reduce dimensionality for visualization, but often obscure meaningful relationships and overlook structural variations across embeddings [58]. As embedding spaces become more complex [21], static projections are insufficient: users require tools that support flexible, interactive, and multi-scale exploration. Three core challenges arise. First, *embeddings differ in type and dimensionality, and attributes take diverse formats, making direct comparison infeasible* [5]. Second, *structural relationships depend on scale: varying the neighborhood threshold can change cluster boundaries, reveal overlaps, or cause loss of critical information* [43], [53]. Third, *visually corresponding the same data-items across embeddings is challenging due to the overwhelming amount of data-items in datasets* [5], [54]. Addressing these challenges requires a unified approach that enables cross-type embedding comparison while supporting dynamic adjustment of structural scales.

To address these challenges, we present **Parallel Clusters** (ParaClus), a visual analytics system for exploring embeddings in a comprehensive and intuitive way. Unlike static visualizations, ParaClus provides an interactive platform where users can adjust clustering granularity, navigate across embeddings and attributes, and drill down into local structures. By combining parallel axes with dynamic clustering and rich interaction, the system helps users reveal both global patterns and fine-grained relationships.

The core of ParaClus is a parallel axis layout, where each *partition* represents *clusters* derived from either an *embedding* or an *attribute*. This design supports direct comparison across multiple *embeddings*, and *attributes*, and scales, enabling users to trace how *data-items* correspond between *clusters*. Users can interactively refine *clusters*, split *axisNodes* into sub-*clusters*, and compare them across multiple *partitions* to study intra&inter-*cluster* structures.

In addition, ParaClus integrates labels and scalar *attributes* directly into the clustering process. This allows *embeddings* to

be studied together with metadata, which is crucial for multi-attribute datasets such as time-varying signals or networks. By unifying *embeddings* and *attributes* into a common framework, the system provides a holistic view of high-dimensional data.

In summary, ParaClus contributes in the following aspects:

- *Unified comparative framework:* We use *partitions* to enable consistent comparison of *clusters* across *embeddings* and *attributes*. This extends traditional parallel coordinates beyond 1D *attributes* to cross-dimensional and cross-type relationships.
- *Interactive hierarchical exploration:* We design mechanisms for splitting and refining *clusters*, supporting progressive, multi-level exploration. These interactions help users uncover both local details and global structures within *embeddings*.
- *Dynamic neighborhood definition:* Adaptive neighborhood ranges allow users to flexibly adjust clustering granularity, define neighborhood size, and compute connection strength among clusters. This overcomes the limitations of fixed thresholds and supports exploration from coarse to fine scales.

## II. Related Work

Embeddings are often the output of the encoder in a neural network. In the exploration of networks and tasks, except for the architecture and data itself, embeddings are an important carrier of data patterns and distributions. Therefore, the representation and interpretation of embeddings has become the intersection of visualization and machine learning.

*Visualization for embeddings:* Visual analysis of embedding aims to characterize the distribution and structure in data. A common approach is to reduce the high-dimensional embedding to 3D/2D space and use a variety of visualization methods to highlight data features at various scales. Each visualization form has its own information pattern that it is good at highlighting, which inevitably loses the comprehensiveness of the features. t-SNE [62], UMAP [40], and PCA [38] are classic models that are commonly used to reduce dimensions to 2D space and visualize structures using scatter plots. In addition, now more and more complex interactions and visualization view designs are used to eliminate the loss of pattern representation. In the representation of embedding, Walchshofer et al. [64] and Yuan et al. [28] tend to understand the similarity of embedding states and provid embedding-guided insights from knowledge graphs. For the overloading of local information, Zhang et al. [76] chose to combine juxtaposition and explicit encoding to expand the differential contrast at the local scale, and Ye et al. [71] used density maps and sampling techniques. Customized visual analysis systems are widely used in exploring embeddings. AdaVis [77] uses a box embeddings based knowledge graph to model the potential one-to-many relationship between different entities. TransforLearn [22] supports users understand the data flow and model workflow of long text sequences. MV$^2$Net [52] closely combines multivariate multi-view visualization for brain network comparison. Other applications include differentiated interpretation of embeddings in machine learning [16], analysis of medical cases [35], and prediction of neuronal functions [46], etc.

*Embeddings analysis in machine learning:* In machine learning, analyzing embeddings primarily focuses on evaluating the performance of different model architectures and algorithms. In the early stage, the research was mainly oriented towards performance evaluation in applications. Ranjbar et al. [47] and Sargent [50] evaluated the performance of commonly used artificial neural network architectures in power grid forecasting and medium-to-large medical data analysis, respectively. In the analysis of the model itself, there are discussions and comparisons of Monte Carlo simulation [45], network settings [29], graph classification algorithms [39], deep neural network pruning [11], etc. Then, some models dedicated to embedding alignment have been derived to represent networks with networks. REGAL [27] uses unsupervised learning to generate comparable node embeddings in multiple graphs. Song et al. [56] proposed a graph alignment neural network (GANN) model, combined with an attention-based LSTM node sequence generation model to maintain node proximity and improve node matching efficiency. Yan et al. [69] proposed DINGAL-B for aligning dynamic knowledge graphs. Introducing visualization is also a popular choice in machine learning. In network analysis, methods like Representation Topology Divergence help compare multi-scale topologies of neural network representations [1], [33]. In data exploration, visual analytics approaches allow users to interactively select representative parameter distributions in multi-parameter ensembles [17], [51], [55]. Similarly, in social and medical studies, visualization methods have been used to analyze brain networks and group work activities across multiple sessions [44], [70].

*Comparison of embeddings:* Comparison is a standard way to intuitively show differences in visualization [23], but it is difficult to balance customization and generality in the design that supports contrast, and each specific visualization form will lose information representation from other scales. Some studies choose to explore and compare two sets of embeddings in a 2D dimensionality-reduced space. Boggust et al. [5] visualize the global data in a star-trail plot and then compare the numerical attributes of selected data points in tabular form across the two embeddings. Sivaraman et al. [54] focus on interactive comparisons of global and local relationships, as well as neighborhood structures, based on scatterplots. Bearfield et al. [2] found that people tend to compare two groups before comparing two separate bar charts, and spatial proximity is a stronger grouping cue than other visual cues. Therefore, the design of a visual system based on parallel coordinates is a good choice [3]. Bendix et al. [3] proposed Parallel Sets to illustrate categorical data. Previous works [14], [31], [49], [67] support the subset selection of embeddings based on attributes and analyze model performance by simultaneously comparing differences between the subsets. However, parallel sets have inherent limitations [19], [20], [37], such as difficulty in directly visualizing statistical distributions across multiple sets and visual clutter from overlapping information in cross-axis comparisons. To address these challenges, Cantu et al. [7] leveraged the contextual structure

of parallel sets to explore multivariate heterogeneous sequence data, focusing on local comparisons. Bok et al. [6] enhanced parallel coordinate plots (PCP) by incorporating stacked bar histograms with discrete color schemes to increase information density in non-overlapping regions.

*Connections and comparisons:* Our approach is inspired by the rich existing literature. Similar to the scatterplot-based tools, ParaClus emphasizes differences between global and local neighborhoods and supports interactive exploration. Multi-view approaches leverage coordinated perspectives to compare different embedding spaces, providing a reference for our exploration of multiple cluster relationships. Parallel-coordinate designs demonstrate the effectiveness of axis-based encodings for comparing data attributes, echoing our use of a parallel framework. Similar to cross-scale and human-in-the-loop methods, our approach also supports cross-space comparisons and flexible, user-driven exploration.

In comparison to the multi-view based approaches [12], [60], [66], ParaClus does not rely on additional views to mitigate overlap but instead enables multi-granularity exploration within a single framework through a dynamic neighborhood mechanism. Compared alternative designs based on parallel coordinates [13], [75], ParaClus presents both cluster-level connections and data point-level relationships, enables flexible refinement to explore hierarchies, and supports high-dimensional data (e.g., embeddings). Although Table Lens [48] and LineUp [24] share a similar visual form (i.e., horizontal bar charts) with our approach, their design facilitates the identification of data items, instead of understanding the structures of attributes. Additionally, these approaches target scalar attributes without support of embedding comparison.

Regarding the challenge of cross-space alignment, ModalChorus [74] conducts cross-scale image–text semantic comparison via scatterplots, Chen et al. [9] enhance data patterns through human-in-the-loop interaction, and SPEC [30] aligns embedding cluster structures using kernel matrices. In contrast, ParaClus emphasizes systematically organizing and comparing cluster structures within a unified parallel coordinate framework, directly integrating labels and attributes during clustering, and revealing complex cross-scale and cross-space relationships through a threshold-adjustable clustering exploration mechanism.

## III. Requirements Analysis

### A. Interview Procedure

Following the requirement analysis that combine prototype system design with interviews [8], [73], we conducted two rounds of interviews to derive requirements for embedding analysis. In the first round, we worked with two experts each with more than 10 years experience in machine learning. Their feedback shaped the initial requirements, system design, and case studies. In the second round, we interviewed seven junior researchers (graduate students with machine learning experience) to complement these requirements. This round was conducted based on the initial requirements and system, focusing on four aspects: (1) task-specific analysis of embeddings and attributes, (2) interaction modes and responsiveness, (3) multi-scale exploration, and (4) cross-space comparison. Further details are provided in the Appendix.

### B. Design Requirement

In the first round, interviews with two experts revealed three main requirements for embedding comparison: interaction, multi-scale analysis, and cross-space exploration. Then, a structured follow-up interview with seven graduate students refined these directions and uncovered one additional requirements. The three main requirements reflect a progression of understanding across scales: from clusters, to relationships among clusters, to individual data items, as follows:

*R1: Multi-resolution clustering for inter-variable comparison:* Users need to analyze embeddings at different scales, from global distributions to fine-grained subgroups. To capture structures of varying resolution and nature, the system should allow dynamic adjustment of clustering parameters (e.g., neighborhood radius, cluster number, projection method) and provide a unified visual representation for comparing multiple embedding spaces and attributes, such as labels, temporal evolution, or physical properties.

*R2: Multi-level analysis of cluster relationships:* Users require tools to examine how clusters relate at different levels, where adjustable neighborhood sizes define both intra-cluster structures and inter-cluster connections. The system should reveal these relationships to support systematic comparison across embeddings.

*R3: Interactive exploration of data items across clusters:* Users expect rich interaction capabilities (e.g., filtering, brushing, and linking) to refine clusters and identify corresponding data items across them. Such exploration should support both local inspection and global navigation by uncovering meaningful data-item-level connections between embeddings.

These requirements guided the development of the initial system, and they were further refined in the second round, which revealed an additional insight:

*R4: Dynamic sampling and exploration range:* Users require support for varying sampling strategies and the ability to dynamically adjust the exploration range in real-time. This enables examination of structural stability and reduce bias from single-sample artifacts.

Please also refer to the appendix for detailed observations and participant-specific insights.

## IV. Parallel Clusters Model and Design

Based on these requirements, we first derive the basic components to construct a conceptual model, and then design the system of ParaClus. The conceptual model unifies the elements and operations for cluster-based embedding comparison. It provides a global description of a dataset at cluster-level. The system design translates this description into an analytics interface and adds explorative operations to refine the analysis results.

*Overview:* Fig. 1 presents the architectural overview of our ParaClus, consists of three main components: data processing, interface, and exploration. The data processing supports

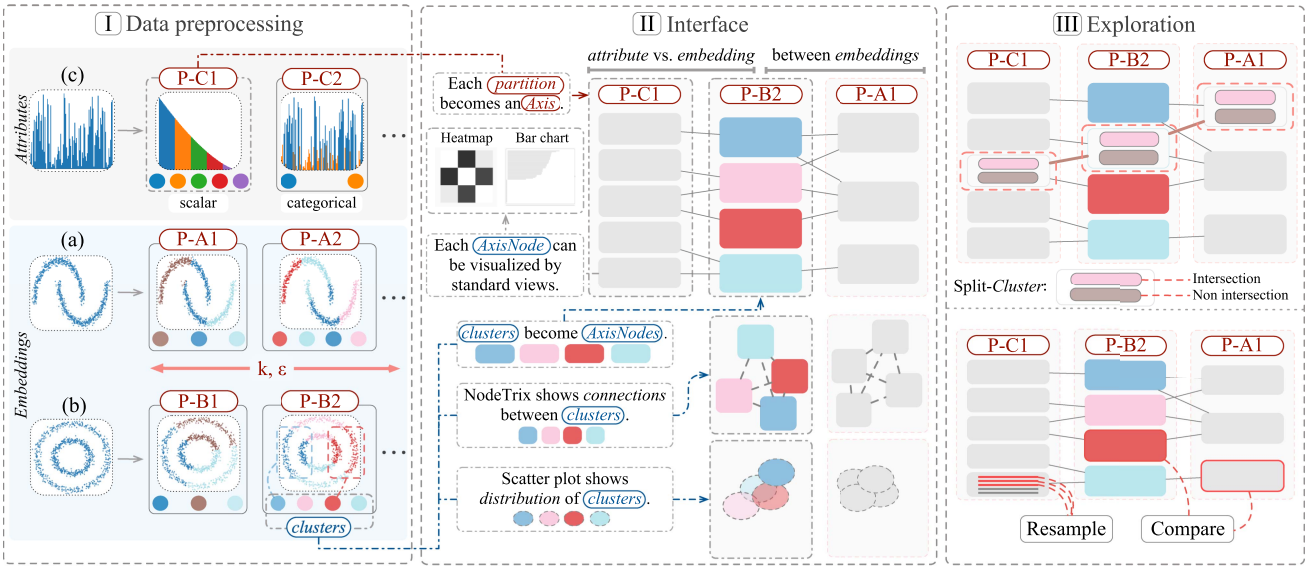


Fig. 1. Overview of ParaClus design. (I) *Variables* (including *embeddings* and *attributes*) are first processed to form *partitions*. Note that each *variable* may correspond to multiple *partitions* for different analytical purposes. (II) In ParaClus interface, each *partition* is visualized as an *axis*, with edges indicating connections between neighboring *axes*. Intra-*partition* connections and distributions of *clusters* corresponding to *embeddings* are further displayed as NodeTrix and scatter plots. (III) Explorative interactions (e.g., cluster splitting and comparison) further enables identification of intersected *data-items*. The *partitions* and *axes* are highlighted in red, and the *clusters* and *axisNodes* are highlighted in blue.

multi-resolution *clusters* (R1) and flexible connection definition (R2), generating the *partitions* to describe *variables*. Fig. 1(I) illustrates the processing of a dataset with two *embeddings* and one *attribute*, with their *partitions* labeled in red. The two *embeddings* A and B can be represented by different *partitions* (i.e., A1, A2, and B1, B2, respectively), by clustering the *data-items* using different parameters $k$ and $\epsilon$. For the *attribute* C, *data-items* can be grouped by their values as well.

The interface illustrates each *partition* as an *axis*, and visualizes the connections both between the *partitions* and with individual *partitions*, as illustrated in Fig. 1(II). Each *axis* corresponds to a *partition* and its *axisNodes* corresponds to the *clusters* of the *partition*. The connections between *partitions* are visualized between *axes*, and each individual *partition* is further visualized by a NodeTrix and a Scatter Plot directly below the corresponding *axis*.

The exploration provides interactions to split or compare the *clusters* and identify *data-items* of interest. Fig. 1(III) shows three exploration operations: split-*clusters*, compare-*clusters*, and resampling. Split-*clusters* refines *clusters* and reveals interactions beyond neighboring *axes*. Compare-*clusters* details *data-item*-level similarities between selected *clusters*, and resampling allows users to specify local regions to generate additional samples. Please refer to Section VI for details.

### A. Conceptual Model

The conceptual model forms the foundation of the ParaClus system. First, each *embedding* or *attribute* can be represented by multiple series of *clusters* (R1), with each series reflecting the analysis intent through an appropriate clustering approach and parameters. Second, connections between *clusters* are captured by their overlaps (R2), which can be flexibly defined for different analytical purposes by adjusting neighborhood size and distance metrics.

*Concepts:* The framework unifies all data by treating both intrinsic *attribute* and encoder-generated *embedding* as *variable*. Each *variable* can be described at a coarse level by a *partition*, which consists of multiple *clusters* covering all *data-items*. By specifying clustering algorithms and parameters, users may create different *partitions* to reflect their analysis purposes. Our design represents each *partition* as an *axis*, and the corresponding *clusters* as *axisNodes*. This design facilitates between-*axes* analysis at *cluster*-level. Detailed definitions are provided below:

- *Data-item* refers to the primitive elements in the raw data used for analysis, e.g., a molecular formula, an image, etc.
- *Embedding* is a latent representation generated from raw data through neural networks/encoders.
- *Attribute* is an intrinsic property of raw data, including categories, scales, numerical values, and text descriptors.
- *Variable* includes both *embedding* and *attribute* to provide a unified view for exploring the data.
- *Partition* assigns all *data-items* to *clusters*, reflecting the structure of a *variable* [59].
- *Cluster* is a group of *data-items* sharing similar properties, generated by a clustering algorithm.
- *Axis* visually represents a *partition* in ParaClus, where multiple *axisNodes* are aligned.
- *AxisNode* corresponds to a *cluster* in a *partition*. It is visualized as a node in an *axis* in ParaClus.

Note that we use *partition* instead of *clusters* because a *partition* represents a disjoint and complete division of the data based on a single *variable*. In contrast, the term *clusters* can

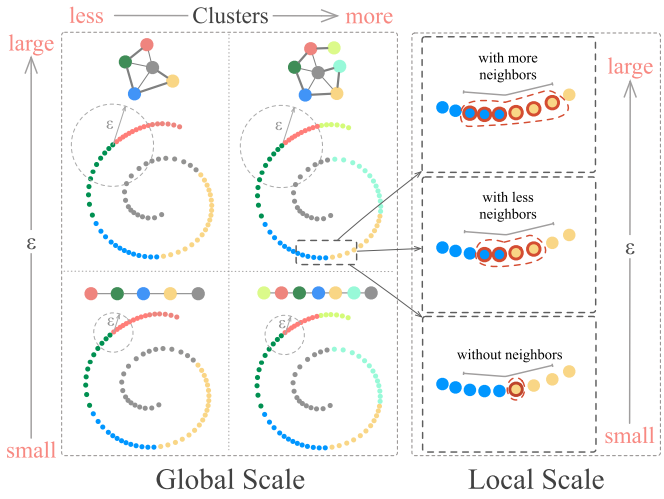


Fig. 2. An illustration of our multi-scale analysis using a toy 2D Swiss roll example. By varying the resolution of clusters and neighborhood size $\epsilon$, our analysis can create different global pictures of the same dataset, and varying local neighborhoods.

be ambiguous, sometimes referring only to a subset of *clusters* or overlapping groups. By using *partitions*, we ensure a whole and clearly defined grouping, which is essential for consistent analysis and comparison within a single *variable*.

*Multi-scale analysis based on clusters and neighborhood:* Our analysis should support the exploration of *clusters* at different levels of granularity (R2) and the definition of connections among *clusters* to capture structures across scales (R3). To achieve the desired cluster resolution, users can specify clustering algorithms and parameters. To flexibly define connections, we introduce the key concept of a variable distance threshold in neighborhood definitions. This enables users to interactively determine which *data-items* are considered neighbors, thereby establishing the strength of *cluster* connections based on the overlapping *data-items* in their neighborhoods.

The impact of the cluster resolution and the neighborhood size $\epsilon$ can be illustrated with a simple example in Fig. 2. At the global scale, we can see that more *clusters* can better approximate the overall structure of the entire dataset, but lead to more complex visual representation. The neighborhood size $\epsilon$ determines which *clusters* are overlapped and captures structures at different scales. At local scale, using appropriate $\epsilon$ allows *data-items* in different *clusters* to appear in the same neighborhood. This prevents abrupt changes at the cluster boundaries and preserves the proximity of spatially close *data-items* in different *clusters*.

*Definitions and notations:* The neighborhood of *data-items* and *clusters* avoids hard boundaries that separate close points. The *overlap* of cluster neighborhoods determines the connections of *clusters*. By distinguishing the overlapping regions among different *clusters*, we can further subdivide the *clusters* for refined exploration.

Formally, let $e_i$ be a *data-item*, its neighborhood $\mathcal{N}(e_i)$ is the set of *data-items* within distance $\epsilon$ to it. $L(\cdot)$ is the *cluster* label function that maps a *data-item* to its *cluster* index, and, similarly, $L(\mathcal{N}(e_i))$ is the set of cluster labels present in the neighborhood of *data-item* $e_i$. This enriches the label information in the label, by including the neighboring *clusters*. Note that $L(\cdot)$ can be either a single label or a vector.

Given the above definitions, the overlap of two clusters $a$ and $b$ can be derived as the *data-items* whose label vectors contain both $a$ and $b$:

$$O_{a,b} = \left\{ L(\mathcal{N}(e_i)) \;\middle|\; e_i \in E,\ \{a,b\} \subseteq L(\mathcal{N}(e_i)) \right\}. \quad (1)$$

Additionally, the labels of all possible cluster overlaps can be identified as:

$$O_A = \left\{ x | \exists\, i \in \{1,\ldots,n\},\ x \in L(\mathcal{N}(e_i)) \right\}, \quad (2)$$

where $n$ is the number of *data-items* and $i$ is an index to iterate over all *data-items*. By grouping the *data-items* with the same label into a new *cluster*, we can derive a refined *partition*, where *data-items* residing in the intersection of *clusters* can be easily identified and selected for further exploration.

### B. Realization and Visual Design

The conceptual model is realized through the following three components: *partition* generation, connection analysis, and hierarchical exploration.

*Partition generation: Partition* serves as a unified comparative unit, offering a complete and disjoint division of the data. This structure supports more reliable alignment and comparison of group structures across multiple views. For *embeddings*, clustering is performed in high-dimensional space using user-specified methods and parameters, in order to achieve desired cluster resolutions or distributions. For *attributes*, categorization follows their inherent types, e.g., categorical or scalar. Each *partition* contains multiple *clusters*, and overlap analysis between *clusters* is supported to reveal shared *data-items* and boundary structures. Each *partition* is visualized as an *axis*, and its *clusters* are showed as *axisNodes* aligned on the *axis*.

*Connection analysis:* Connections in ParaClus enable users to examine relationships both across and within *partitions*. Two types of links, supporting R1 and R2 by interactive layering and overlap representation.

*Inter-partition connection:* It unifies the comparison between different *embeddings*, different clustering results of the same *embedding*, or two *attributes*. It also facilitates the analysis of the relationship between an *embedding* and an *attribute*. The inter-*partition* connections between two *partitions* are visualized as edges between the respective *axes*.

*Internal connection:* These connections emphasize intra-*partition* relationships, which supports the understanding of a single *embedding*, similar to the global structures revealed in Fig. 2. NodeTrix is used to encode both the *data-item*-wise connections within *clusters* as matrices, and the connections between *clusters* within the same *partition* as edges between the matrices.

*Hierarchical exploration:* ParaClus provides two hierarchical mechanisms: *split-cluster* that subdivides *axisNodes* into sub-*axisNodes* within a *partition*, and *refine-cluster* that updates

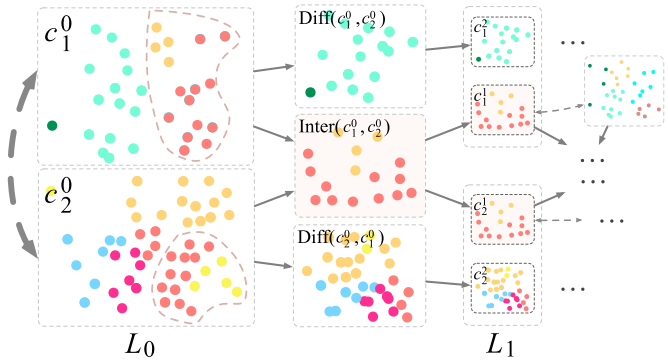


Fig. 3. The hierarchical diagram of partition *axisNodes* in *cluster-data-item*, where $L_0$ represents the initial *axisNode* and $L_1$ represents the sub-*axisNode* layer after one level of hierarchy.

a *partition*. These improve the multi-scale analysis (R2) and facilitate refined exploration at *data-item*-level (R3).

*Split-cluster:* Each *axisNode* can be subdivided into up to two layers of sub-*axisNodes* based on intersections with connected *axisNodes*. Fig. 3 illustrates one level of this process: $c_1^0$ and $c_2^0$ are two *clusters* from adjacent *partitions*. The color of each *data-item* point indicates its *attribute* group, and color semantics are shared between the two *clusters*. The index of each *data-item* point is also preserved. In $L_0$, dashed lines highlight *data-item* points with matching indices across the two *clusters*. The two *clusters* are then divided into three subsets: $c_1^1 = c_2^1 = \text{Inter}(c_1^0, c_2^0)$ represents the intersection; $c_1^2 = \text{Diff}(c_1^0, c_2^0)$ contains points in $c_1^0$ but not in $c_2^0$; and $c_2^2 = \text{Diff}(c_2^0, c_1^0)$ contains points in $c_2^0$ but not in $c_1^0$.

Subsequently, $c_1^0$ and $c_2^0$ are subdivided into two sub-*clusters*, forming the $L_1$ layer. This division can be repeated to generate $L_2$, allowing connections between adjacent axes to summarize and propagate cluster structures. Starting from $c_1^0$, $L_1$ and $L_2$ enable progressive hierarchical division along connected axes, supporting cross-dimensional and multi-scale comparisons.

*Refine-cluster:* Users can dynamically adjust clustering methods and their parameters to update an entire *partition*, or using the overlapping labels $O_A$ to refine a *partition*. Users may adjust $\epsilon$ to select an appropriate neighborhood size, ranging from 0 to 100, where 0 indicates that each node has no neighbors, and 100 indicates that every node is considered a neighbor of all nodes globally. This enables flexible re-partitioning and fine-tuning of overlap relationships, supporting both local and global structural analysis. Refinement operations update cluster assignments, connections, and *partition* visualizations in real time, allowing progressive exploration from coarse to fine-grained structures.

*Sampling:* While clustering reduces the amount of *axisNodes* to a reasonable level, the number of *data-items* in individual nodes may still introduce scalability issues for very large dataset. Therefore, we adopt an additional sampling in our data management, which includes two components: the initial sampling and dynamic sampling. The initial sample is performed before exploration. Users can specify one sampling strategy from our system or provide their own sampling results. The dynamic sampling blocks are created during exploration. Users can sample neighboring points around a chosen *data-item* to focus on local patterns. In addition, multi-round combinations enable the observation of local structures at different positions and across scales.

## V. Parallel Clusters Interface

The visual analysis interface of ParaClus is shown in Fig. 4. This interface contains of: (a) Parallel Clusters view, (b) Individual Partition view, and (I-V) five auxiliary views. The auxiliary views include (I) Auxiliary Tools, (II) Axis Ordering, (III) Gallery, (IV) Combination view, and (V) Selection List. The Parallel Clusters view shows the overview of *partitions* and their connections, the Individual Partition view shows the connections and distributions within individual *partitions*, aligned with the axes in the Parallel Cluster view. The auxiliary views provides additional information regarding the dataset.

### A. Parallel Clusters View

The Parallel Clusters view (Fig. 4(a)) is the main view for exploration. It integrates multiple *partitions* as parallel *axes* and *clusters* as *axisNodes* on the corresponding *axes*. This view extends the conventional parallel coordinates with an interactive and hierarchical design. Users can add any number of *axes* and adjust them dynamically for interactive exploration (R3). This section introduces the Parallel Cluster view from three levels: *clusters*, *partitions*, and connections.

*1)* Cluster-*Level Visualization: Cluster*-level visualization shows the *data-items* within individual *clusters* in the Parallel Clusters View. ParaClus currently provides bar chart and heatmap for individual *clusters* visualization, and we perform hierarchical enhancement based on them. The bar chart is the default visualization because it is easier to align horizontally, and the heatmap is to show the connections between *data-items*. Each mode highlights different aspects of *cluster* distribution and similarity (R2).

*Bar chart:* This mode shows the distribution of *attributes* inside a *cluster*, as shown in Fig. 5(a). Each *cluster* is wrapped in a *axisNode*, with the left edge as the grounding axis, and the *attributes* are encoded as normalized horizontal bars. This visual design is similar to Table Lens [48] and LineUp [24]. All raw data will be normalized to (0, 1) (or optional [0.5,1]). Zero values are represented as short gray bars on the left of the grounding axis.

*Heatmap:* As a complement to the cross-cluster comparison (R2), the heatmap is used to shows the similarity within and between *clusters* Fig. 5(d). Each *cluster* builds a correlation matrix based on a user-specified *attribute*. Users can also switch to numerical differences in the back end. Clusters on the same axis share one color map, making it easy to compare similarity patterns across *clusters*.

*Hierarchical enhancement:* Both bar charts and heatmaps support hierarchical refinement (Fig. 5). For bar charts, each *cluster* can be split into two blocks. Intersecting elements keep their original colors, while non-intersecting *data-items* turn gray. Additional splits can be applied when more *axisNodes* are introduced. For heatmaps, selecting an edge moves intersecting

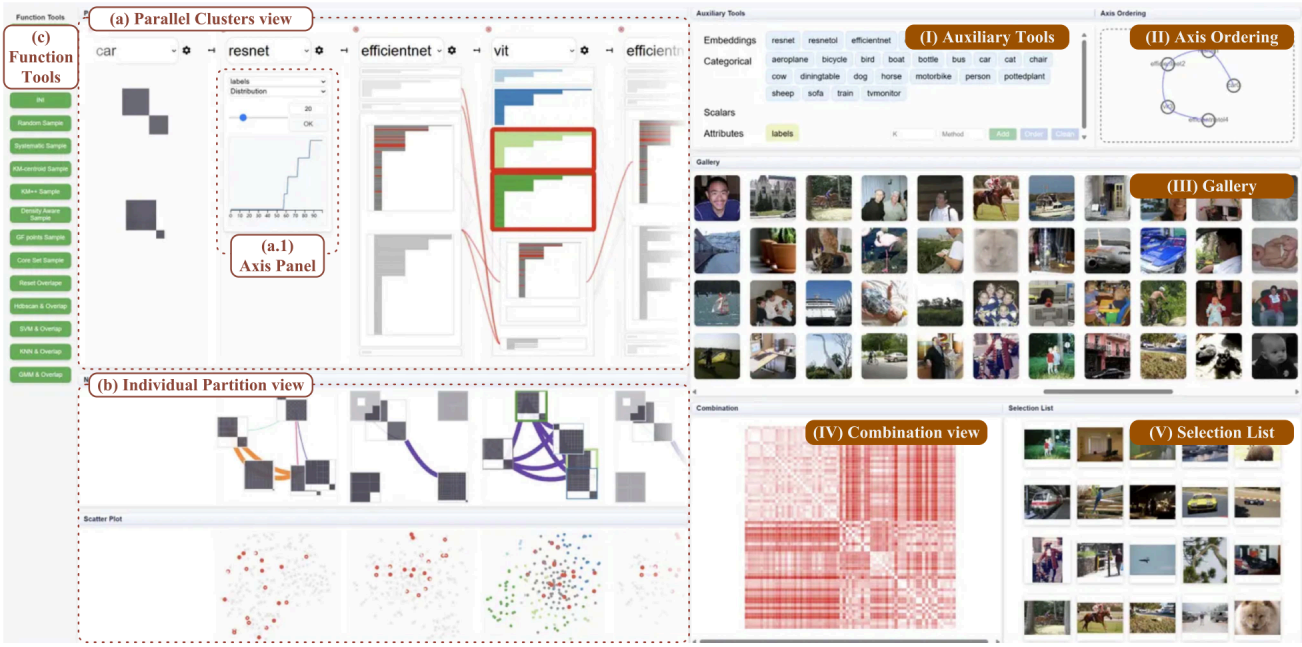


Fig. 4. The interface of ParaClus. The left side of the interface serves as the main view: (a) Parallel Clusters view, each axis represents the *partition* of a *variable*. *clusters* are displayed as color-coded nodes along the axes, with internal visual indicating *data-item* distribution. (b) Individual Partition view, the connections and the 2D spatial distribution in a *partition*. (c) Function tools, for resampling, reoverlaping, and starting a new exploration. The right side provides auxiliary views: (I) Auxiliary Tools, all comparable labels for drag-and-drop axis configuration. (II) Axis Ordering, current axis map for reordering. (III) Gallery, original data list of all *data-items*. (IV) Combination view, the combination similarity heatmap for selected *axisNodes*. (V) Selection List, the enlarged view of selected *data-items*.

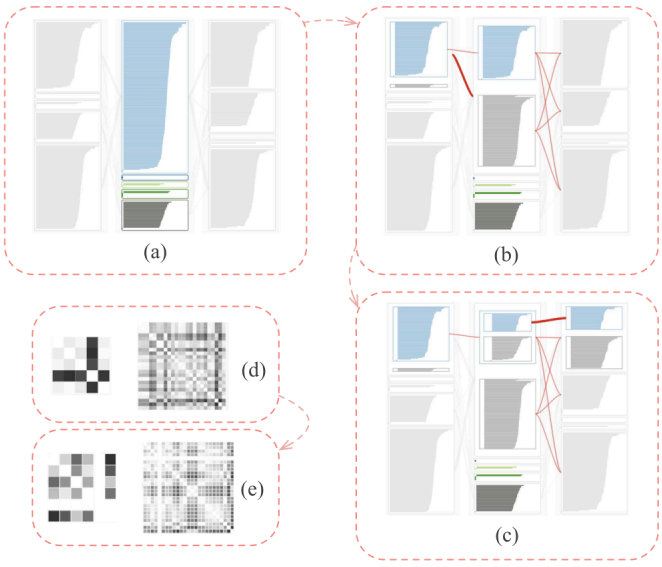


Fig. 5. Visualization of *clusters*. (a) default visualization. (b) visualization of hierarchical *axisNodes*. (c) visualization of sub-*axisNodes* and *axisNodes*. (d) heatmap of *clusters*. (e) visualization of hierarchical heatmap.

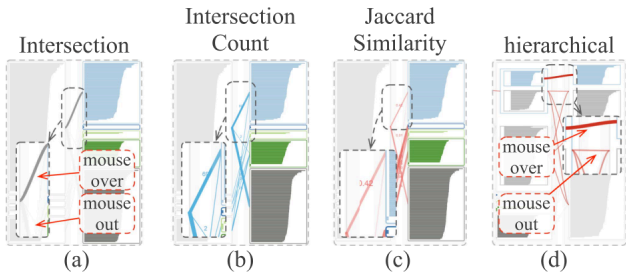


Fig. 6. Visualization of four types of connection:(a)default connections (intersection), (b) intersection counts, (c) Jaccard similarity, and (d) hierarchical intersections.

*data-items* to the upper-left corner and non-intersecting *data-items* to the lower-right corner, producing a clear separation.

*2)* Partition-*Level Visualization:* Users can add multiple *partitions* to compare different *variables* (R3). Each *partition* corresponds to an axis in the Parallel Cluster view, which shows an overview of *clusters* and *data-items*. Users may specify one main *partition*. The main *partition* uses colors to distinguish *clusters*, and all other *partitions* show *clusters* in gray. This minimizes color ambiguity and supports smooth interaction (R1). In the main *partition*, each *axisNode* encodes information through color, vertical order, and bar length, highlighting differences between *partitions* for multi-level comparison (R2).

When the *clusters* are visualized as bar charts, each *data-item*'s value comes from a user-specified numerical *attribute*. This value can be original data or the result of dimensionality reduction. Inside an *axisNode*, *data-items* are arranged from top to bottom by the bar value.

*3) Connections Between* Partitions*:* Due to the large amount of *data-items*, the connections between *partitions* are shown at the *cluster*-level (Fig. 6). Users may use *axisNode*-split operations to identify the exact *data-items* involved in the connections. They may also adjust the order of *axisNodes* between adjacent *partitions* through connection interaction to reduce the edge crossing for clarity.

Our system supports four types of connections: (a) default intersections, (b) intersection counts, (c) Jaccard similarity, and (d) hierarchical intersections. By default, adjacent axes are connected using curves, with thickness encoding intersection counts. The default intersections connection is used for interaction, so intersection numbers are not displayed. When no interaction is required, users can switch to intersection count or Jaccard similarity to see the specific values. Hierarchical connections appear when a *axisNode* is split, increasing the number of connections. To keep the view readable, these connections are semi-transparent by default and share the same hover behavior as standard intersections, highlighting on hover and fading afterward. In Fig. 6(d), two red arrows mark the highlighted and semi-transparent states.

*4) Axis Panel:* Each *axis* has a panel (Fig. 4(a.1)) for configuration. Users can use this panel to switch the *partition*, specify *attribute* values, and select *cluster* visualization types. Sliders and input boxes adjust neighborhood thresholds and generate new overlaps. A line chart at the bottom shows how average intra-class similarity changes with the threshold. This chart helps users tune parameters. The connection button sets the type of edges between axes. Options include intersection, Jaccard similarity, and ordering. Further data operations, such as resampling and reclustering, are available in the Function tools (Fig. 4(d)), which allows users to control the data range for global exploration.

### B. Individual Partition View

Individual Partition view supports analysis of *clusters* within individual *partitions*. It shows the connections between *clusters* in the same *partitions* as NodeTrix and the distributions of *data-items* as Scatter Plots. Since the *attributes* are labeled by scale or category without any overlap, *attribute*-based *partitions* do not have Individual Partition view.

In Fig. 4(b), the NodeTrix shows the autocorrelation matrix of each *axisNode* and the overlap lines between them. The color and thickness of overlap lines encode type and quantity. The matrix can be computed either by difference or autocorrelation, using the selected numerical *attribute* of the *partition*. A force-directed layout adjusts the relative positions of matrices to prevent overlap. The Scatter Plot shows the distributions of *data-items* in 2D. Colors of the Scatter Plot and NodeTrix are inherited from the corresponding *partition*, allowing users to assess density and separation across *clusters*.

### C. Auxiliary Views

The auxiliary views support adding, adjusting, and exploring details in the Parallel Clusters view. These views are shown in Fig. 4(I-V). Auxiliary Tools (I) shows all *variables*, including *embeddings* and *attributes*. Users may select any *variable* from this panel to create *partitions* with specified parameters, and map scalar *attributes* to the height of bar charts. Axis ordering (II) allows users reorder added *partitions*. Gallery (III) shows all original *data-items* in the exploration range as images. Combination view (IV) compares any two *clusters* by stitching their autocorrelation matrices and computing inter-*cluster* correlation. Selection list (V) presents the original data of selected *data-items* for detailed inspection.

## VI. Exploration Workflow

ParaClus supports an interactive workflow that guides users from global overview to local inspection. The workflow has two "Focus Modes" to center the exploration at *partitions* or their connections. The *partition* focus allows users to refine clusters based on their own properties (R1) and the connection focus enables subdivisions based on linkages across *partitions* (R2). For each mode, explorative operations (R3) are performed in three stages: refinement, selection, and iterative adjustment. Each stage provides operations that let users refine, compare, and construct *partitions*. Efficient computation and optimized visualization support real-time interaction.

### A. Focus Mode

When the user adds more than two *partitions* in the interface, the displayed information can become complex. Since the *axisNodes* of different *partitions* vary, distinguishing all of them simultaneously may lead to visual clutter. To address this, we implemented two **Focus Modes** to help users concentrate on a specific *partition* and explore *embeddings* flexible (R1).

*Partition Focus:* During exploration, users can switch the main *partition* by clicking it. The main *partition* uses distinct colors to differentiate its *axisNodes*, and the same color coding is applied in the Individual Partition view to represent different *clusters*. All other *partitions* use a uniform color for their *axisNodes* and in the Individual Partition view. By default, the most recently added *partition* is set as the main *partition*. Cross-*partition data-item* interactions remain effective, supporting synchronized highlighting and restoration. In this mode, connections are displayed with semi-transparency, allowing users to concentrate on exploring the *partitions* themselves.

*Connection Focus:* Selecting a connection for hierarchical exploration, the interface automatically enters connection focus mode, highlighting the connections between *partitions*. The main *partition* is styled the same as other *partitions*. Light red lines are used to connect the subsets of *axisNodes* at both ends of the selected connection with their related *axisNodes*. Sub-*axisNodes* within the intersection are shown in the same color, determined by the main *partition*'s color at the first-level division, while the non-intersecting related *axisNodes* are displayed in dark gray. All unrelated *axisNodes* are shown in a uniform light gray.

To focus exploration attention and reduce visual clutter, the two modes automatically switch and cannot coexist.

### B. Refinement

ParaClus creates a new *partition* using creates default *variables*, clustering parameters, and visualization. Users can then refine these settings. Refinement focuses on *clusters* within each *partition*. It includes clustering, overlap threshold, hierarchical refinement, and sampling.

*Cluster refinement:* Users can freely choose clustering methods and the number of *clusters* (R1). New clustering configurations generate new *variable* tags (Fig. 4(I)), which share the same interaction and visualization as the default feature. Five clustering strategies are provided (details in Appendix D), with k-Means as the default. They can be applied interchangeably, allowing users to explore overlaps under different assumptions about cluster structures. The system computes the corresponding overlap automatically and updates the visualization accordingly. This flexibility ensures that ParaClus can accommodate diverse datasets and support multi-scale exploration of embedding structures.

*Connection refinement:* Users can adjust the overlap threshold $\epsilon$ to refine the connection computation (R2), as shown on the second axis in Fig. 4(a). The range is [0,100], from no overlap to full overlap. For each coefficient, overlapping category combinations are generated. The Jaccard similarity between all categories is calculated and averaged as an overall similarity. Users can refine the overlap threshold with the similarity curve to achieve multi-scale comparison.

*Hierarchical refinement:* Clicking an intersection connection stratifies the related *clusters*. Each *axisNode* supports up to two refinement layers (R2). With two layers, three adjacent *partitions* can be aggregated to filter shared or unique intersections. For adjacent *partitions*, one layer is enough. When multiple *partitions* are linked, information flows continuously across them. The filtered *data-items* can be recorded, and the final axis of one round can serve as the starting point of the next.

*Sampling:* Users can reset to the initial sample at any time, clearing dynamic blocks. Or they can interactively select an element in *partition* as the seed to resample and recluster *variable* from the dataset to start a new exploration. Dynamic blocks are then updated in the interface. This supports progressive analysis from global to local structures (R4). To accommodate diverse analytical needs, ParaClus implements seven representative sampling strategies. Please refer to Appendix D for details.

### C. Selection

After refinement, users explore the *partitions* through selection (R3). Selection works on three levels: *partitions*, *axisNodes*, and *data-items*.

*Partition Selection:* Users compare multiple *partitions* in the Parallel Clusters view. Axis complexity indicates which *partition* should be further explored. Axes can be dragged to reorder or adjusted through the axes map (Fig. 4(II)). Users can freely add, delete, or combine *partitions*. New *partitions* generated in refinement can also be added to the queue.

*AxisNode Selection:* After selecting a *partition*, users can select a *axisNode*. Clicking highlights all *data-items* of the same category in the Individual Partition view. NodeTrix responds by highlighting the heatmap with a bold border. In the Scatter Plot, the corresponding cluster is outlined by a convex hull. Holding *Shift* allows users to select two *axisNodes* from different categories, merging their autocorrelation heatmaps into one (Fig. 4(IV)). This supports multi-scale and cross-category comparisons.

*Data-item Selection:* At any time, users can select individual *data-items*. Clicking adds them to the selection list (Fig. 4(V)). The selected *data-item* is highlighted on the *axis* and in the Scatter Plot. Hovering reveals its raw data, enabling detailed comparison with neighbors. Users can see both global distribution and local detail.

### D. Iterative Adjustment and System Support

Exploration is iterative (R4). Users move between refinement and selection, adjusting *partitions* and visualizations as analysis evolves. The system ensures responsiveness during this loop.

Data processing supports high-dimensional *embedding*, numerical *attributes*, categorical *attributes*, and scale division. Computation uses PyTorch and Python for ordinary processing, with C++ acceleration called from Python through Boost. A Flask backend manages the server. Visualization is divided into a main view for multi-scale comparison and a side view for settings and detail magnification. All front-end applications are built with D3.

## VII. Evaluation

Three case studies are performed in this section to demonstrate the usefulness of the ParaClus, including synthetic distribution comparison, molecular data analysis, and multi-classification models comparison.

### A. Case 1: Proof of Concept

*Background:* The first case compares the structures of two synthesized spaces: a sphere and a torus. In our experiment, the torus is built by folding and connecting the polar areas of the sphere. Through the comparison, we aim to identify the structural change from the sphere to the torus. Specifically, the change means that points in the two poles of the sphere become neighbors in the torus, which is difficult to be identified in scatter plots.

*Dataset:* According to Zeeman [41], spheres and tori represent fundamentally distinct classes of surfaces in topology, each with unique structural properties. Comparing these surfaces highlights differences in connectivity, genus, and local vs. global geometric arrangements. The dataset is constructed as follows: we uniformly sample 200 points on a sphere with missing top and bottom caps, and then transform it into a toroidal "doughnut" shape. The transformation is performed by taking the top and bottom ends of the sphere as starting points and converging them toward the middle of the sphere.

*Analysis process:* We created four *partitions* using PCA and t-SNE embedding spaces of sphere and torus, and used the geodesic distance as the bars for *data-items*.

We first compare the sphere and the torus using PCA, as shown by the first two *axes* in Fig. 7(a). The sphere exhibited three short-tailed and two long-tailed *clusters* with distinct means, while the torus showed the opposite pattern with more comparable means. The two long-tailed *clusters* of the sphere correspond to points around the poles, reflecting topological differences between the sphere and torus in terms of geodesic length and local curvature.

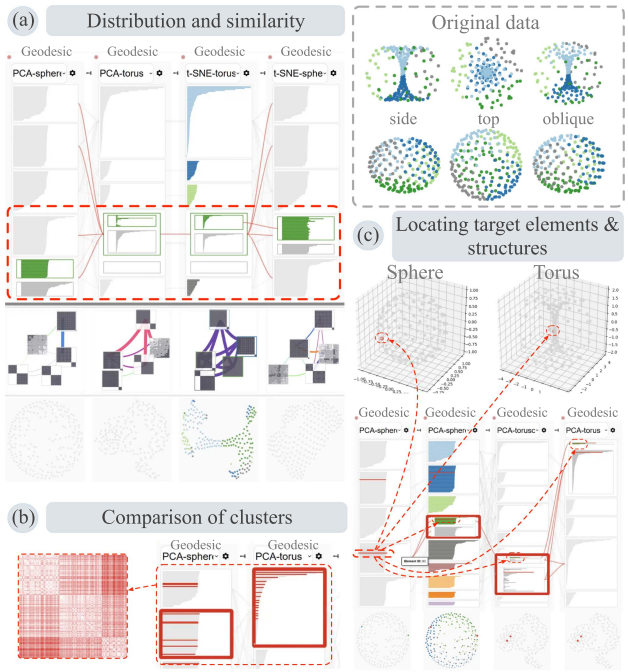


Fig. 7. Comparing the structures of sphere and torus using ParaClus.

However, torus showed greater distortion in 2D projection, making direct alignment of points challenging. Through hierarchical exploration, we identified correspondences between pole-near *clusters* on the sphere and local torus structures, with significant geodesic differences indicating sharp curvature changes and mappings to the torus' inner side, as indicated by the red box in Fig. 7(a).

Using correlation heatmaps and intra-*cluster* distributions, we observed that the torus' geometry produces dense interior regions and sparse exterior regions due to its curvature-induced folding, as shown in Fig. 7(b). This allowed us to systematically determine which regions of the sphere correspond to specific torus *clusters* in 3D.

By exploring the cluster labels of *data-items* under different neighborhoods, we identified *data-items* that are far apart in the sphere but become close in the torus, even if they still do not fall into the same *cluster*. We establish correspondences between *clusters* on the sphere and torus via connections across two axes, narrowing the focus to sphere *clusters* with low geodesic variance (non-polar regions). By adjusting the overlap threshold, we then locate torus points that are close in distance but far apart on the sphere. Finally, hierarchical decomposition with sub-*axisNodes* intersections identifies the target elements.

### B. Case 2: Molecular Data Interdependencies

*Background:* AI has increasingly demonstrated its potential in scientific research, driving advances in data-driven drug discovery and materials design. Molecular data, with its inherent multidimensional properties (structural *variables*, electronic characteristics, and reactivity), serves as an ideal candidate for data analysis and *embedding* visualization. By leveraging *embedding* techniques to map these high-dimensional molecular representations into lower-dimensional spaces, AI models can better capture subtle structural nuances. The second case was conducted by E1, exploring the GNNs encoding of Molecular data. E1 aims to evaluate various graph neural network models to assess their effectiveness in capturing intrinsic molecular structures.

*Dataset:* In this study, E1 selected the ChEBI-20-MM [36] dataset due to its diverse range of molecular types and modalities, which is conducive to systematic comparisons. This study used a total of 32,998 molecules. Our sampling procedure initially selects 200 molecules from the full dataset for exploration, and the expert can use the resampling feature to progressively add samples from regions of interest during interaction. E1 converted molecular formats to graph structures and standardized the data to ensure uniformity for subsequent embedding extraction. For the *embedding* acquisition, E1 employed four models (GraphSAGE [25], GCN [32], GIN [68], and GAT [63]) to extract meaningful structural features from the molecular graphs. These pipelines and cases provide a comprehensive perspective on how each technique captures different aspects of molecular structure, highlights the differences between methods, and facilitates detailed analysis of molecular *clusters*.

*Analysis process:* Solubility is a crucial property in the field of molecular formula prediction and analysis. To explore this aspect, xLogP, a numerical indicator of water solubility, was initially chosen as the numeric metric for *cluster* visualization. Subsequently, the Solubility *attribute* was added as a baseline *variable* and compared in parallel with four commonly used GNN models to assess the distribution of *partitions* in Parallel Clusters view, as shown in Fig. 8(a).

The results indicate that the embeddings generated by the GAT model closely correlate with the solubility distribution. Among the four models, GAT demonstrates the highest degree of *partition* structure and *axisNode* distribution matching with the solubility *attribute*. This makes GAT particularly suitable for tasks requiring accurate alignment with solubility-related properties. After removing the Solubility *attribute* to focus solely on model-specific characteristics, it was observed that GraphSAGE and GAT exhibit the closest resemblance in terms of both *axisNodes* and 2D distribution, as in Fig. 8(b). This suggests that GraphSAGE, like GAT, is capable of preserving structural information while capturing *variable* interactions. On the other hand, GCN and GIN exhibit distinct behavior, particularly in detecting anomalous *clusters*. GCN and GIN show higher sensitivity to identifying and isolating erroneous or outlier data, making them more reliable for anomaly detection tasks or datasets prone to noisy *data-items*. This heightened sensitivity leads to distinct global *partitions* compared to GAT and GraphSAGE.

Due to the relatively high sensitivity of GCNs to anomalies, E1 first compared *axisNodes* within the GCN and identified two pairs of *axisNodes*, as shown in Fig. 8(b). Each pair consists of an abnormally small *cluster* contrasted with a *cluster* exhibiting the expected distribution. Next, multiple sampling rounds were conducted to examine local patterns under different conditions,

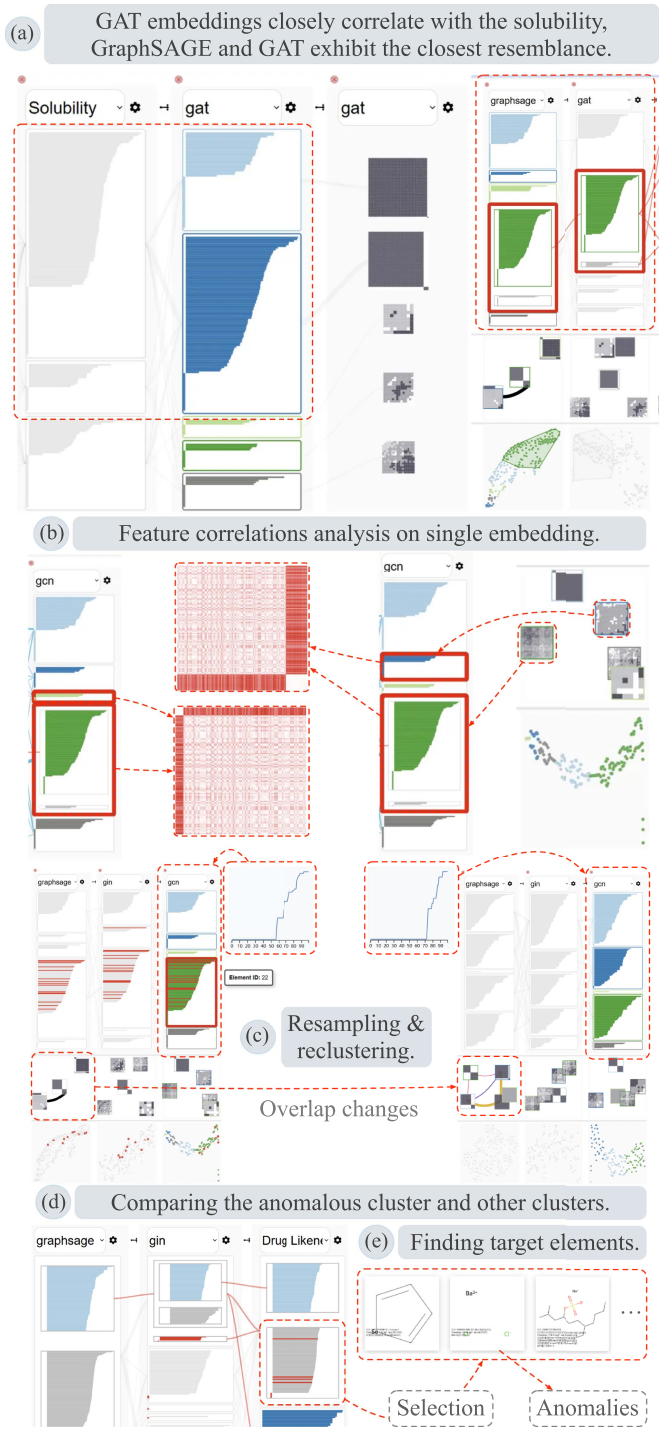


Fig. 8. Exploration and analysis of the Molecular Data Interdependencies using ParaClus.

while adjusting the overlap threshold to track changes in the *clusters* of the target *data-items*, as shown in Fig. 8(c). Finally, hierarchical exploration was used to precisely locate the specific anomalous *data-items*, as shown in Fig. 8(d) and (e). This showed that GCN and GIN are similar in anomaly detection, while GAT and GraphSAGE are more suitable for maintaining overall distribution consistency.

The findings suggest that each model has advantages. GAT is better for applications requiring precise property alignment. GraphSAGE is useful for structural similarity analysis. GCN is well-suited for detecting outliers and handling noisy data. GIN is also good at anomaly detection while maintaining robust intra-*cluster* consistency.

### C. Case 3: Image Classification Models

*Background:* In the field of computer vision, multi-class image classification is a crucial and widely applied research direction. The goal is to assign an input image to one of several predefined categories. This task is essential in various applications such as medical image diagnosis, object detection, autonomous driving, and image retrieval. E2 operated this case, he wanted to explore the reasons for the performance differences between the three SOTA methods: ResNet, EfficientNet, and ViT.

*Dataset:* The PASCAL Visual Object Classes (VOC) dataset [18] is used here. It is a widely used benchmark in computer vision for tasks such as object detection, image classification, and semantic segmentation. This study used a total of 17,125 images. The user sampled 200 images of VOC to explore and compare the networks. The models is the official version from PyTorch.

*Analysis process:* The purpose of this case is to explore the strengths and weaknesses of three models: ResNet [26], EfficientNet [57], and ViT [15], and understand the reasons behind their performance differences. Among the three models, ResNet significantly outperforms the other two, with EfficientNet and ViT showing progressively weaker performance. This discrepancy prompts the need to investigate the underlying factors that contribute to these differences. From the outset, the three models exhibit distinctly different initial states. For ResNet, the *data-item* distribution among *clusters* is highly uniform, indicating a well-balanced classification layout. The distribution of EfficientNet is highly imbalanced, with nearly 80% of the images grouped into a single *cluster*. Moreover, the range of multi-label classifications varies significantly. At first glance, the *cluster* distribution of ViT appears similar to ResNet, but upon reordering, the differences between ViT and ResNet become even more pronounced, surpassing those between EfficientNet and ResNet. This might be the reason for ViT's comparatively poor performance Fig. 9(a).

Before delving into the internal structure of *partitions*, it is essential to investigate the reasons behind EfficientNet's anomalous clustering. To address this, E2 experimented with different scales of overlap to refine the classification. When the overlap scale is set to 50, it becomes possible to separate most categories, although the intra-*cluster* distribution remains uneven Fig. 9(b). In the 2D visualization, many points still intersect, and compared to results without overlap, the effect appears to be merely scattered classification from a mixed multi-*cluster* region. In contrast, ResNet's classification layout is more distinct, with most *axisNodes* of the same type grouped. Next, the user examines the relationship between *embeddings* and *attributes* to identify which *clusters* are difficult to separate. Upon inspection, the

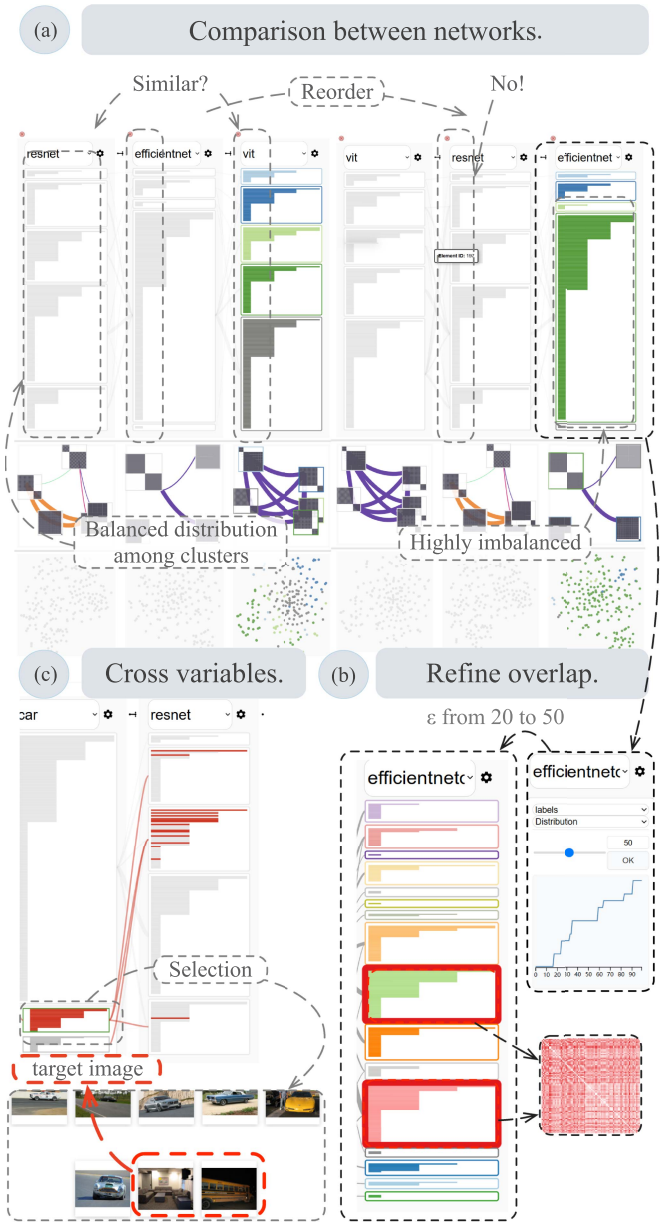


Fig. 9. Exploration and analysis of the Image Multi-classification Models using ParaClus.

user found that almost every *cluster* contains images assigned to multiple *clusters*. Taking "car" as an example, the user expanded a small intersection cluster from EfficientNet and examined each overlapping image. E2 discovered a room and a bus classified as "car" Fig. 9(c). Visually, the background of these images does resemble that of cars, suggesting that EfficientNet struggles to distinguish images with complex or ambiguous visual contexts. Upon further analysis, the EfficientNet's classification errors often arise when:

- There are too many indistinct objects in the image.
- The background is simple, uniform color and structure, which makes the model more likely to make correct classifications.
- Complex visual structures or multi-object scenes tend to be grouped erroneously.

The analyses of small, non-overlapping *clusters* show that ResNet captures semantic features more effectively. EfficientNet and ViT rely on color and structural cues, often misclustering images with complex content. This highlights ResNet's advantage in multi-class classification, where subtle semantic distinctions are critical.

## VIII. Discussion

In this section, we will discuss the data processing scale and scalability that ParaClus can accommodate, as well as the limitations in the comparative exploration process.

*Scalability:* The system can handle datasets with tens of thousands of data points, but this often affects efficiency and performance. Sampling methods are commonly applied to maintain the statistical distribution characteristics of the data while improving efficiency. According to the central limit theorem [34], when a large number of independent and identically distributed samples are drawn from a population, the distribution of the sample means tends to approximate a normal distribution. Although a sample size of 30 is often considered a rule of thumb for achieving this approximation, practical applications usually recommend a sample size of at least 100 to ensure a more accurate estimate of the distribution characteristics. The human visual system struggles with large datasets. When the number of data points exceeds a certain threshold, the visual representation becomes too dense, leading to confusion and difficulty in recognizing patterns. Some studies suggest that the limit is around 200 data points, but this varies [61]. To avoid information overload, we should consider this limitation to ensure effective interpretation. Front-end rendering also faces performance bottlenecks when handling large-scale data, consuming significant computational resources and causing slow page responsiveness or crashes. Sampling reduces the number of data points, improving rendering efficiency and ensuring a smooth user experience. To balance data accuracy and visualization efficiency, we use various sampling methods, including uniform, random, stratified, and density-preserving sampling.

*Limitations and future works. Continuous Comparison:* While continuous comparison sequences facilitate identifying edges and overlaps between different embedding sets, their limited hierarchical structure impedes complex association analyses across multimodal embeddings (e.g., text and images). This constraint disrupts information flow across multiple axes. To address this, future designs should develop hierarchical visualization techniques that support extensive information linkages while preventing information overload from excessively long connections. For instance, integrating focus+context exploration methods can enhance the analysis of hierarchical embeddings by allowing users to navigate different levels of detail effectively.

*Reordering:* The current system supports only basic, rule-based node reordering. To more effectively reveal patterns within partitions, implementing a collaborative ordering algorithm across element, node, and partition layers is advisable. Leveraging Large Language Models can facilitate real-time

semantic understanding, enabling more efficient and intelligent information layout.

*System Complexity:* During the case study with the experts, we observed that they required assistance at certain points to achieve their goals, rather than throughout the entire process. For instance, in Case 2, E1 demonstrated rapid comprehension of concepts like *partition* and *cluster* upon initial exposure to ParaClus. However, when adding axes or comparing axes in the Parallel Clusters view, E1 encountered challenges in mapping these concepts to their visual representations, such as interpreting the meaning of visual encodings within each *axisNode*, and understanding the relationship between hierarchical *axisNodes* and their primitive counterparts. To improve learnability, we provided experts with demonstration videos and manuals for reference, supplemented by real-time assistance. After several guided sessions and self-directed exploration, the experts were able to use the system proficiently for data exploration. This observation indicates that ParaClus carries a certain level of complexity for novice users.

However, the complexity observed in ParaClus is partially inherent to general visual analytics tools, which rely on higher levels of abstraction to support diverse tasks and usage contexts [10], [65], [72]. As prior work has noted, such generality inevitably increases conceptual and interaction complexity, making complete simplification impractical without sacrificing expressive power.

To mitigate this challenge, rather than attempting to further simplify the design, we would like to explore the use of large language model (LLM) to guide user exploration. The LLM will act as a cognitive mediator that assists users in interpreting visual encodings, understanding interaction logic, and aligning analytical intents with system operations, thereby smoothing the learning process while preserving the system's generality.

## IX. Conclusion

We propose Parallel Clustering (ParaClus), a visualization analysis system that enables multi-scale exploration of embedding structures based on cluster formation across *embeddings* and *attributes*, and an adaptive threshold mechanism to define neighborhoods. In addition, it supports *variable* alignment and refinement exploration through visualization and interactive segmentation designed with parallel axes. It unifies *embeddings* and *attributes*, and all *variables* can be used as a *partition* to form a comparative view. It addresses the cross-view alignment challenge of embedding comparison by building multiple data point hierarchies from different perspectives. It solves the problem of cross-scale loss of data information and inability to compare across dimensions in embedding exploration. Finally, expert evaluation and case studies are used to verify the effectiveness.

## Acknowledgments

The authors would like to thank the anonymous reviewers for their insightful comments.

## References

[1] S. Barannikov et al., "Representation topology divergence: A method for comparing neural network representation," in *Proc. 39th Int. Conf. Mach. Learn.*, 2022, pp. 1607–1626.
[2] C. X. Bearfield, C. Stokes, A. Lovett, and S. Franconeri, "What does the chart say? Grouping cues guide viewer comparisons and conclusions in bar charts," *IEEE Trans. Vis. Comput. Graph.*, vol. 30, no. 8, pp. 5097–5110, Aug. 2024.
[3] F. Bendix, R. Kosara, and H. Hauser, "Parallel sets: Visual analysis of categorical data," in *Proc. IEEE Symp. Inf. Visual.*, 2005, pp. 133–140.
[4] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.
[5] A. Boggust, B. Carter, and A. Satyanarayan, "Embedding comparator: Visualizing differences in global structure and local neighborhoods via small multiples," in *Proc. 27th Int. Conf. Intell. User Interfaces*, 2022, pp. 746–766.
[6] J. Bok, B. Kim, and J. Seo, "Augmenting parallel coordinates plots with color-coded stacked histograms," *IEEE Trans. Vis. Comput. Graph.*, vol. 28, no. 7, pp. 2563–2576, Jul. 2022.
[7] A. Cantu, M. E. Micó-Amigo, S. Del Din, and S. J. Fernstad, "Parallel assemblies plot, a visualization tool to explore categorical and quantitative data: Application to digital mobility outcomes," in *Proc. IEEE 16th Pacific Visual. Symp.*, 2023, pp. 21–30.
[8] D. Cashman et al., "A user-based visual analytics workflow for exploratory model analysis," in *Comput. Graph. Forum*. Hoboken, NJ, USA: Wiley, 2019, pp. 185–199.
[9] C. Chen et al., "Human-guided image generation for expanding small-scale training image datasets," *IEEE Trans. Vis. Comput. Graph.*, vol. 31, no. 6, pp. 3809–3821, Jun. 2025.
[10] S. Chen et al., "Supporting story synthesis: Bridging the gap between visual analytics and storytelling," *IEEE Trans. Vis. Comput. Graph.*, vol. 26, no. 7, pp. 2499–2516, Jul. 2020.
[11] H. Cheng, M. Zhang, and J. Q. Shi, "A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, pp. 10558–10578, Dec. 2024.
[12] W. Cui, G. Strazdins, and H. Wang, "Visual analysis of multidimensional Big Data: A scalable lightweight bundling method for parallel coordinates," *IEEE Trans. Big Data*, vol. 9, no. 1, pp. 106–117, Feb. 2023.
[13] Z. Deng et al., "Visual comparative analytics of multimodal transportation," *Vis. Inform.*, vol. 9, no. 1, pp. 18–30, 2025.
[14] D. Dingen et al., "RegressionExplorer: Interactive exploration of logistic regression models with subgroup analysis," *IEEE Trans. Vis. Comput. Graph.*, vol. 25, no. 1, pp. 246–255, Jan. 2019.
[15] A. Dosovitskiy et al., "An image is worth 16 x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 611–631.
[16] M. El-Assady, R. Sevastjanova, F. Sperrle, D. Keim, and C. Collins, "Progressive learning of topic modeling parameters: A visual analytics framework," *IEEE Trans. Vis. Comput. Graph.*, vol. 24, no. 1, pp. 382–391, Jan. 2018.
[17] M. El Meseery and O. Hoeber, "Geo-coordinated parallel coordinates (GCPC): Field trial studies of environmental data analysis," *Vis. Inform.*, vol. 2, no. 2, pp. 111–124, 2018.
[18] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes (VoC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, 2010.
[19] E. E. Firat, A. Denisova, M. L. Wilson, and R. S. Laramee, "P-lite: A study of parallel coordinate plot literacy," *Vis. Inform.*, vol. 6, no. 3, pp. 81–99, 2022.
[20] E. E. Firat, B. Swallow, and R. S. Laramee, "PCP-ed: Parallel coordinate plots for ensemble data," *Vis. Inform.*, vol. 7, no. 1, pp. 56–65, 2023.
[21] T. Fujiwara, O.-H. Kwon, and K.-L. Ma, "Supporting analysis of dimensionality reduction results with contrastive learning," *IEEE Trans. Vis. Comput. Graph.*, vol. 26, no. 1, pp. 45–55, Jan. 2020.
[22] L. Gao, Z. Shao, Z. Luo, H. Hu, C. Turkay, and S. Chen, "TransforLearn: Interactive visual tutorial for the transformer model," *IEEE Trans. Vis. Comput. Graph.*, vol. 30, no. 1, pp. 891–901, Jan. 2024.
[23] M. Gleicher, "Considerations for visualizing comparison," *IEEE Trans. Vis. Comput. Graph.*, vol. 24, no. 1, pp. 413–423, Jan. 2018.
[24] S. Gratzl, A. Lex, N. Gehlenborg, H. Pfister, and M. Streit, "LineUp: Visual analysis of multi-attribute rankings," *IEEE Trans. Vis. Comput. Graph.*, vol. 19, no. 12, pp. 2277–2286, Dec. 2013.

[25] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, Red Hook, NY, USA, 2017, pp. 1025–1035.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[27] M. Heimann, H. Shen, T. Safavi, and D. Koutra, "Regal: Representation learning-based graph alignment," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 117–126.

[28] C.-W. Hsuan Yuan, T.-W. Yu, J.-Y. Pan, and W.-C. Lin, "KGScope: Interactive visual exploration of knowledge graphs with embedding-based guidance," *IEEE Trans. Vis. Comput. Graph.*, vol. 30, no. 12, pp. 7702–7716, Dec. 2024.

[29] D. Hunter, H. Yu, M. S. Pukish III, J. Kolbusz, and B. M. Wilamowski, "Selection of proper neural network sizes and architectures—A comparative study," *IEEE Trans. Ind. Informat.*, vol. 8, no. 2, pp. 228–240, May 2012.

[30] M. Jalali, B. D. Nia, and F. Farnia, "Towards an explainable comparison and alignment of feature embeddings," 2025, *arXiv:2506.06231*.

[31] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. Chau, "ActiVis: Visual exploration of industry-scale deep neural network models," *IEEE Trans. Vis. Comput. Graph.*, vol. 24, no. 1, pp. 88–97, Jan. 2018.

[32] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 2713–2726.

[33] A. Kumpf, J. Stumpfegger, P. F. Härtl, and R. Westermann, "Visual analysis of multi-parameter distributions across ensembles of 3D fields," *IEEE Trans. Vis. Comput. Graph.*, vol. 28, no. 10, pp. 3530–3545, Oct. 2022.

[34] S. G. Kwak and J. H. Kim, "Central limit theorem: The cornerstone of modern statistics," *Korean J. Anesthesiol.*, vol. 70, no. 2, 2017, Art. no. 144.

[35] B. C. Kwon et al., "RetainVis: Visual analytics with interpretable and interactive recurrent neural networks on electronic medical records," *IEEE Trans. Vis. Comput. Graph.*, vol. 25, no. 1, pp. 299–309, Jan. J 2019.

[36] P. Liu, J. Tao, and Z. Ren, "A quantitative analysis of knowledge-learning preferences in large language models in molecular science," *Nature Mach. Intell.*, vol. 7, pp. 315–327, 2025.

[37] N. F. Lobato, T. D. O. de Araujo, B. S. Meiguins, and C. G. R. dos Santos, "A review of techniques for reducing shortcomings in classical information visualization charts," in *Proc. 28th Int. Conf. Inf. Visualisation*, 2024, pp. 1–7.

[38] A. Maćkiewicz and W. Ratajczak, "Principal components analysis (PCA)," *Comput. Geosci.*, vol. 19, no. 3, pp. 303–342, 1993.

[39] S. Mascarenhas and M. Agarwal, "A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for image classification," in *Proc. Int. Conf. Disruptive Technol. Multi-Disciplinary Res. Appl.*, 2021, pp. 96–99.

[40] L. McInnes, J. Healy, N. Saul, and L. Großberger, "UMAP: Uniform manifold approximation and projection," *J. Open Source Softw.*, vol. 3, no. 29, 2018, Art. no. 861.

[41] B. Mendelson, *Introduction to Topology*. North Chelmsford, MA, USA: Courier Corporation, 1990.

[42] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[43] K. R. Moon et al., "Visualizing structure and transitions in high-dimensional biological data," *Nature Biotechnol.*, vol. 37, no. 12, pp. 1482–1492, 2019.

[44] R. Ohnishi et al., "Visualization of members' activities in a series of group works by time-series feature comparison method," in *Proc. 15th Int. Congr. Adv. Appl. Informat. Winter*, 2023, pp. 69–74.

[45] J. D. Olden, M. K. Joy, and R. G. Death, "An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data," *Ecological Modell.*, vol. 178, no. 3/4, pp. 389–397, 2004.

[46] H. Park et al., "NeuroCartography: Scalable automatic visual summarization of concepts in deep neural networks," *IEEE Trans. Vis. Comput. Graph.*, vol. 28, no. 1, pp. 813–823, Jan. 2022.

[47] M. Ranjbar, S. Soleymani, N. Sadati, and A. M. Ranjbar, "Electricity price forecasting using artificial neural network," in *Proc. Int. Conf. Power Electron. Drives Energy Syst.*, 2006, pp. 1–5.

[48] R. Rao and S. K. Card, "The table lens: Merging graphical and symbolic representations in an interactive focus context visualization for tabular information," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 1994, pp. 318–322.

[49] D. Ren, S. Amershi, B. Lee, J. Suh, and J. D. Williams, "Squares: Supporting interactive performance analysis for multiclass classifiers," *IEEE Trans. Vis. Comput. Graph.*, vol. 23, no. 1, pp. 61–70, Jan. 2017.

[50] D. J. Sargent, "Comparison of artificial neural networks with other statistical approaches: Results from medical data sets," *Cancer, Interdiscipl. Int. J. Amer. Cancer Soc.*, vol. 91, no. S8, pp. 1636–1642, 2001.

[51] J. Sharko et al., "Heat map visualizations allow comparison of multiple clustering results and evaluation of dataset quality: Application to microarray data," in *Proc. 11th Int. Conf. Inf. Visual.*, 2007, pp. 521–526.

[52] L. Shi et al., "MVNet: Multi-variate multi-view brain network comparison over uncertain data," *IEEE Trans. Vis. Comput. Graph.*, vol. 28, no. 12, pp. 4640–4657, Dec. 2022.

[53] G. Singh et al., "Topological methods for the analysis of high dimensional data sets and 3D object recognition," *PBG, Eurographics*, vol. 2, pp. 91–100, 2007.

[54] V. Sivaraman, Y. Wu, and A. Perer, "Emblaze: Illuminating machine learning representations through interactive comparison of embedding spaces," in *Proc. 27th Int. Conf. Intell. User Interfaces*, 2022, pp. 418–432.

[55] H. Song, Z. Dai, P. Xu, and L. Ren, "Interactive visual pattern search on graph data via graph representation learning," *IEEE Trans. Vis. Comput. Graph.*, vol. 28, no. 1, pp. 335–345, Jan. 2022.

[56] L. Song, W. Tu, S. Zhou, and E. Zhu, "GANN: Graph alignment neural network for semi-supervised learning," *Pattern Recognit.*, vol. 154, 2024, Art. no. 110484.

[57] M. Tan and Q. V. Le, "EfficientNet:Rethinking model scaling for convolutional neural networks," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

[58] S. Tilouche, V. P. Nia, and S. Bassetto, "Parallel coordinate order for high-dimensional data," *Statist. Anal. Data Mining, ASA Data Sci. J.*, vol. 14, no. 5, pp. 501–515, 2021.

[59] A. Topchy, A. K. Jain, and W. Punch, "Clustering ensembles: Models of consensus and weak partitions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 12, pp. 1866–1881, Dec. 2005.

[60] D. Trye, M. Apperley, and D. Bainbridge, "Extending the heatmap matrix: Pairwise analysis of multivariate categorical data," in *Proc. 27th Int. Conf. Inf. Vis.*, 2023, pp. 29–36.

[61] E. R. Tufte and P. R. Graves-Morris. *The Visual Display of Quantitative Information*. Cheshire, CT, USA: Graphics Press, 1983.

[62] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 11, pp. 2579–2605, 2008.

[63] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 2920–2931.

[64] C. Walchshofer, A. Hinterreiter, K. Xu, H. Stitz, and M. Streit, "Provectories: Embedding-based analysis of interaction provenance data," *IEEE Trans. Vis. Comput. Graph.*, vol. 29, no. 12, pp. 4816–4831, Dec. 2023.

[65] J. Wang, S. Liu, and W. Zhang, "Visual analytics for machine learning: A data perspective survey," *IEEE Trans. Vis. Comput. Graph.*, vol. 30, no. 12, pp. 7637–7656,Dec. 2024.

[66] Q. Wang, Z. Xu, C. Zhu-Tian, Y. Wang, S. Liu, and H. Qu, "Visual analysis of discrimination in machine learning," *IEEE Trans. Vis. Comput. Graph.*, vol. 27, no. 2, pp. 1470–1480, Feb. 2021.

[67] J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Viegas, and J. Wilson, "The what-if tool: Interactive probing of machine learning models," *IEEE Trans. Vis. Comput. Graph.*, vol. 26, no. 01, pp. 56–65, Jan. 2020.

[68] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 9104–9120.

[69] Y. Yan, L. Liu, Y. Ban, B. Jing, and H. Tong, "Dynamic knowledge graph alignment," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 4564–4572.

[70] X. Yang, L. Shi, M. Daianu, H. Tong, Q. Liu, and P. Thompson, "Block-wise human brain network visual comparison using nodetrix representation," *IEEE Trans. Vis. Comput. Graph.*, vol. 23, no. 1, pp. 181–190, Jan. 2017.

[71] Y. Ye, R. Huang, and W. Zeng, "VISAtlas: An image-based exploration and query system for large visualization collections via neural image embedding," *IEEE Trans. Vis. Comput. Graph.*, vol. 30, no. 7, pp. 3224–3240, Jul. 2024.

[72] Y. Ye, R. Huang, K. Zhang, and W. Zeng, "Unified visual comparison framework for human and AI paintings using neural embeddings and computational aesthetics," *IEEE Comput. Graph. Appl.*, vol. 45, no. 2, pp. 19–30, Mar./Apr. 2025.

[73] Y. Ye, F. Sauer, K.-L. Ma, K. Aditya, and J. Chen, "A user-centered design study in scientific visualization targeting domain experts," *IEEE Trans. Vis. Comput. Graph.*, vol. 26, no. 6, pp. 2192–2203, Jun. 2020.

[74] Y. Ye, S. Xiao, X. Zeng, and W. Zeng, "ModalChorus: Visual probing and alignment of multi-modal embeddings via modal fusion map," *IEEE Trans. Vis. Comput. Graph.*, vol. 31, no. 1, pp. 294–304, Jan. 2025.

[75] C. Zhang, Y. Chen, J. Yang, and Z. Yin, "An association rule based approach to reducing visual clutter in parallel sets," *Vis. Inform.*, vol. 3, no. 1, pp. 48–57, 2019.
[76] C. Zhang, T. Schultz, K. Lawonn, E. Eisemann, and A. Vilanova, "Glyph-based comparative visualization for diffusion tensor fields," *IEEE Trans. Vis. Comput. Graph.*, vol. 22, no. 1, pp. 797–806, Jan. 2016.
[77] S. Zhang, H. Li, H. Qu, and Y. Wang, "AdaVis: Adaptive and explainable visualization recommendation for tabular data," *IEEE Trans. Vis. Comput. Graph.*, vol. 30, no. 9, pp. 5923–5938, Sep. 2024.

**Zehua Yu** received the MS degree in electric and communication engineering from Shantou University in 2022. He is currently working toward the PhD degree in computer science with Sun Yat-sen University. His research interests include visual analytics, reinforcement learning, time-series analysis, and scientific visualization.

**Xiaoyi Li** received the Bachelor of Engineering degree in computer science and technology from Chongqing University in 2025. She is currently working toward the postgraduation degree in computer science with Sun Yat-sen University. Her research interests include scientific visualization, visual analytics, and deep learning.

**Pengfei Liu** is currently working toward the PhD degree with the School of Computer Science and Engineering, Sun Yat-sen University. He is with Peng Cheng Laboratory. His research focuses on the development and application of multi-modal large language models, with a particular emphasis on integrating knowledge priors and enhancing model interpretability.

**Jun Tao** (Member, IEEE) received the PhD degree in computer science from Michigan Technological University in 2015. He is currently an associate professor of computer science with Sun Yat-sen University and National Supercomputer Center in Guangzhou. His research focuses on scientific visualization, especially in applying information theory, deep learning, optimization techniques to interactive flow visualization and multivariate data exploration.