Versatile Ordering Network: An Attention-based Neural Network for Ordering Across Scales and Quality Metrics

Zehua Yu, Weihan Zhang, Sihan Pan, and Jun Tao*, *Member, IEEE*

Abstract—Ordering has been extensively studied in many visualization applications, such as axis and matrix reordering, for the simple reason that the order will greatly impact the perceived pattern of data. Many quality metrics concerning data pattern, perception, and aesthetics are proposed, and respective optimization algorithms are developed. However, the optimization problems related to ordering are often difficult to solve (e.g., TSP is NP-complete), and developing specialized optimization algorithms is costly. In this paper, we propose Versatile Ordering Network (VON), which automatically learns the strategy to order given a quality metric. VON uses the quality metric to evaluate its solutions, and leverages reinforcement learning with a greedy rollout baseline to improve itself. This keeps the metric transparent and allows VON to optimize over different metrics. Additionally, VON uses the attention mechanism to collect information across scales and reposition the data points with respect to the current context. This allows VONs to deal with data points following different distributions. We examine the effectiveness of VON under different usage scenarios and metrics. The results demonstrate that VON can produce comparable results to specialized solvers. The code is available at https://github.com/sysuvis/VON.

Index Terms—visual ordering, 1D layout, metric-agnostic ordering, attention mechanism, reinforcement learning

1 Introduction

Ordering is a key ingredient in visualization approaches, as the order of entities in visualization may greatly impact the perceived patterns, leading to different understandings of data. Therefore, ordering has been studied in many visualization scenarios. For example, axis reordering has gained considerable attention in visualization with multiple axes, such as parallel coordinates [4], [75], star coordinates [41], and Radviz [23]; branch ordering is studied in river-like visualization techniques [14], [21], [22]; row and column reordering is studied in matrix visualization [10], [48], [76]; and, images ordering is studied in large image collection visualization.

The ordering techniques seek to address two central questions: What is an ideal order (quality metric), and how to achieve the ideal order (solver)? Many quality metrics are proposed to evaluate the order of visual elements in various applications, and respective solvers are developed to optimize the order for best quality. For example, GUIRO [10] incorporates 16 quality metrics for matrix representation and 70 reordering algorithms. Although the reordering algorithm often shares similar components (e.g., swapping elements), designing algorithms for individual metric still consumes a significant amount of research effort. This leads to our question: **Given an arbitrary quality metric, can machines learn to solve the ordering problem by themselves?** By answering this question, we may free the visualization researchers from designing solvers, and allow them to focus on perception and analytic tasks, which may not be tackled solely by machines.

Reinforcement learning can leverage the quality metric to evaluate solutions and improve the solver without understanding the problem structure, which is suitable in our scenario. In theoretical computer science, reinforcement learning has been applied to similar problems [47], [61], but the existing approaches still leave the following challenges open in visualization:

C1. Generalizability across quality metrics. The problem structures and factors to consider may vary across metrics. As shown in Fig. 1, the linear route structure is enforced by the traveling salesman problem (TSP), which only evaluates the distance between neighboring data points. In contrast, stress considers all pairwise relationships between points, leading to a significantly different problem structure of a complete graph. We expect that the problem structure can be more different when it comes to perceptual or aesthetic metrics. This requires the network to be able to incorporate different kinds of information.

C2. Transferability across data distributions and scales. In visualization, the data items being ordered may be dynamically generated during interaction. For example, when visualizing a collection of dog images, as shown in Fig. 1, users may view images from the entire space (A), they may view the images of a certain breed of dogs (B), or they may explore images similar to a specified one (C). The images in (A), (B), and (C) correspond to different distributions, exhibiting different point cloud structures and requiring different ordering strategies.

C3. Efficiency. The solver should produce the order in real time, especially if it is used in an interactive system. It should enhance development efficiency by minimizing the effort required to develop specialized algorithms for

- * denotes the corresponding author.
- Z. Yu, W. Zhang, and S. Pan are with the School of Computer Science and Engineering, Sun Yat-sen University, China. E-mail: {yuzh58, zhangwh79, pansh25}@mail2.sysu.edu.cn.
- J. Tao is with the School of Computer Science and Engineering, Sun Yat-sen University and the National Supercomputer Center in Guangzhou, China. E-mail: taoj23@mail.sysu.edu.cn. He is the corresponding author.

different metrics or data distributions.

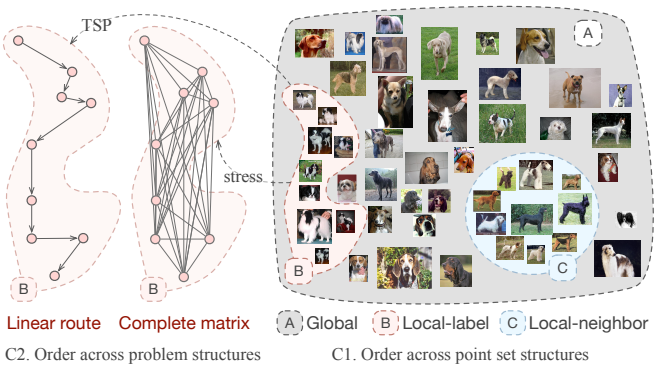


Fig. 1. Challenges in learning to order across scales and quality metrics: **C1**. The images in (A), (B), and (C) may span different regions in space and exhibit *different point set structures*. **C2**. The quality metrics may enforce *different problem structures*, as indicated by the graphs on the right. TSP enforces a linear route structure, while stress considers a complete graph between points. The different scales of points and quality metrics may require different ordering strategies to be learned.

In this paper, we propose Versatile Ordering Network (VON) to tackle the above challenges. VON is a set-to-sequence network that consumes a set of data points and produces a sequence (order). It is trained via reinforcement learning with a given quality metric as the loss function. Note that VON only uses the quality metric as a black box to evaluate the ordering results without explicitly understanding the problem structure. This allows VON to apply on arbitrary quality metric (**C1**). In addition, VON leverages the attention mechanism to collect information about global structures from all points and the local neighborhood of the ordering front. It also enables the message passing between all nodes, including the ordered and unordered nodes. Therefore, it can adapt to various kinds of underlying data patterns and quality metrics (**C1**). To order data points with different scales and spatial locations, VON is also equipped with a specially designed repositioning module. This module can be considered a learnable normalization operator that removes the location, orientation, and scale information so that the network can focus on the structure of data, especially the local neighborhood around the ordering front. The repositioning module allows the learned ordering strategy to be applied to points at unseen locations or scales (**C2**). This enables VON to order data items during interaction without further updating in most cases (**C3**).

We examine the performance of VON under two scenarios: First, ordering a subset of a large collection of data items with a given quality metric. This may apply to interactive visualization, such as filtering a large image collection. Second, ordering a given set of points according to a given quality metric. Axes reordering and matrix reordering are examples of this scenario. We comprehensively evaluate the performance of VON across metrics, data scales, and distributions with extensive experiments involving twelve datasets, twelve sampling strategies, thirteen baselines, and eleven metrics.

Our contributions are as follows. First, we propose a novel ordering network that can be used across tasks, objectives, and data distributions. By leveraging the power of reinforcement learning and a greedy rollout strategy, the network can automatically learn ordering strategies to improve itself. Second, we propose a conceptual framework for ordering and provide several realizations to implement the conceptual model. In particular, we design a repositioning module that automatically learns to place the data points in the current ordering context. This can be seen as a learnable nonlinear transformation that removes the information irrelevant to the structure, improving transferability. In addition, we conduct extensive experiments to examine the effectiveness and robustness of our method.

2 Related Work

The ordering problem has been studied in the visualization from two aspects: how to design visual quality metrics for ordering, and how to achieve the optimal ordering given a metric? Our approach is more closely related to the approaches in the second category. It aims to develop a generic solver that automatically learns how to optimize the order based on a quality metric definition. However, it does not study whether the quality metric will enhance perception or improve task performance in analytics. In this section, we discuss the techniques across various ordering applications in visualization: such as axis reordering, branch ordering, and matrix reordering. We also review the ordering techniques in theoretical computer science, with an emphasis on the learning-to-rank approaches. For the visual quality metric design, We refer the readers to the surveys [8], [9], [52], as they are less relevant.

Image ordering. Image ordering is often modeled as a 1D-grid arrangement problem. For example, SOM [46] learns the low-dimensional representations, SSM [72] arranges items based on position swapping, and LDG [30] uses local search. Kernelized sorting [24], [67] models the arrangement as a quadratic assignment problem. IsoMatch [31] uses the Hungarian algorithm for optimal matching between Isomap embedding and target grid points. DS++ [27] introduces a convex quadratic programming relaxation for the matching process. LAS/FLAS [7] designs a distance preservation metric based on neighborhood preservation and employs linear assignment ordering.

Axes reordering in visualization. Axes ordering aims to enhance patterns revealed by visualization with multiple axes, such as parallel coordinates [43] and star coordies [57], etc. These approaches often design a metric to guide the ordering optimization. For example, Peng et al. [64] reduced glyph clutter by considering monotonicity and symmetricity, and Artero et al. [4] arranged the axes based on their similarity. Tyagi et al. [75] introduced twelve local properties with customizable weights to order axes for parallel coordinates. For star plots, shape similarities of contours are often considered [3], [33]. For interaction, SMARTexplore [11] simplified the analysis of high-dimensional data, and RankAxis [53] is designed to order multiple attributes.

Branch ordering in river-like visualization. 2D River-like visualizations (e.g., streamgraphs, ThemeRivers, storylines) often use one dimension to represent time evolution and arrange branches in the other dimension to encode their relationships. Byron and Watternberg [15] formalized a series of aesthetic criteria for streamgraphs. Di Bartolomeo

et al. [21] swapped the layers in a streamgraph to reduce the wiggle value, and Bu et al. [14] introduced Sinttream to enhance readability.Zhang et al. [88] adapted 3D stream surfaces to 2D, preserving inter-branch distances. For storytelling, sequence ordering is guided by logical structures [70], user intentions [74], and minimizing edge crossings [22].

Matrix reordering. The data topology revealed by matrix representation depends on the order of nodes. However, reordering nodes often involves various factors [76]. Behrisch et al. [10] proposed GUIRO to enhance the transparency of matrix reordering algorithms. Users can explore the permutation space with 70 reordering algorithms and 16 quality metrics, viewing matrix rows or columns as 2D paths on the screen. Van Beusekom et al. [76] simultaneously ordered a collection of matrices by maximizing their Moran's I. The desired order is derived from the solution of TSP, which can be approximated by nearest neighbor heuristics. Kwon et al. [48] use an autoencoder to capture order patterns, enabling the exploration of matrix reordering results of 27 algorithms. Al-Naami et al. [2] investigated how orderable node-link layouts enhanced the visual saliency of graph clusters, by assessing the impact of factors such as graph layout and node ordering methods on the accuracy and completion time of cluster count judgments.

Other applications of ordering in visualization. Ordering is studied in other visualization scenarios as well. For example, in volumetric visualization, Frey et al. [29] ordered 2D curves based on their shape similarities using self-organization map. Weiss et al. [83] used the 1D Hilbert space-filling curve to linearize a volume and Zhou et al. [89] proposed a data-driven space-filling curve based on Hamiltonian paths. Dobler et al. [25] proposed a TSP-based algorithm to compute linear diagrams. Fuchs et al. [32] explored how to reveal topological patterns in BioFabric visualizations by proposing combinations of various quality metrics and ordering strategiessy. Nate et al. [60] proposed a Hilbert reordering scheme to reduce the memory consumption of trees in smooth particle visualization. Instead of the metric-based optimization, ranking approaches often emphasize the user involvement to weight multiple attributes for decision-making, such as ValueCharts [16], LineUp [37], WeightLifter [63], Podium [80], FairSight [1], SRVis [84], FairFuse [71], and RankAxis [53]. We refer the readers to the original papers for details as this direction is less relevant.

Ordering in theoretical computer science. In theoretical computer science, ordering is often studied in the form of traveling sales problem or Hamiltonian circle. Both problems are proved to be NP-complete without efficient deterministic solvers. Conventional approaches are often natural-inspired. For example, Dorigo et al. [26] proposed the Ant Colony System, an algorithm with multiple agents (i.e., ants). Ants cooperate by depositing information on edges to build solutions. Chen et al. [18] revealed the relationship between ranking measures and loss functions in learning-to-rank methods.

Set-to-sequence in machine learning. Set-to-sequence approaches [44] generate sequences from unordered data points, often driven by the need for permutation invariance. These approaches are often equipped with a set-input encoding module for sequence predictions, including RNN [73], reinforcement learning [6], and transformer [12], [20], [78]. Transformers, especially those like Set Transformer [49] and Pointer Network [79], are increasingly popular for ranking and selection tasks due to their attention mechanisms. For solving the TSP, transformers often frame route construction as a set-to-sequence learning task [19], [34].

Recent approaches use neural networks to learn ordering strategies [47], [66], which are similar to our work. Kool et al. [47] combined graph attention network with reinforcement learning for solving routing problems. Their approach maintains a hidden status of currently ordered nodes and outputs the next one. Other approaches may explore rendering in different scenarios, such as text ranking using CNNs [69] and attention mechanism [81], and document and image tretrieval [38], [68], item ranking [50], [65], and heterogeneous data ordering [51].

Connections and comparisons. Our approach differs from the ordering techniques in visualization, as it aims to provide a quality metric-agnostic solver to optimize the order of data items. Therefore, instead of identifying an ideal metric that enhances human perception, our approach focuses on how to optimize the order given any quality metric. This allows visualization experts to focus on the criteria that lead to an ideal order without spending additional effort to design a solver for achieving those criteria.

In comparison to the learning-based techniques for ordering, such as [47], [65], [66], our approach emphasizes the generalizability across metrics, data distributions, and data scales. This is particularly important for visualization systems: the ordering criteria may vary across applications, and the data points involved in the interaction are likely to exhibit different distributions, as shown in Fig. 1. The existing learning-to-rank approaches may be less effective in visualization scenarios, as they often assume the same distribution for point sets being ranked. In contrast, our framework is equipped with a novel repositioning module that allows the network to focus on structural information that can be transferred between distributions.

3 Versatile Ordering Network

We design versatile ordering network (VON) for learning to order data items with an arbitrary quality metric. Instead of designing a specialized algorithm to optimize for the quality metric, VON uses the quality metric to evaluate different orders of sample points, so that it can automatically learn effective ordering strategy for that specific metric.

Overview. The overarching framework is shown in Fig. 2. The data are fed into a set-to-sequence network, that converts input sample points into an output order. The output order is evaluated by the quality metric and compared against the best-observed order. The difference in order quality is computed and used to update the network. This reinforcement learning procedure forces the network to improve itself and produce better order quality. Please refer to Section 3.1 for details.

The network uses an encoder-decoder architecture. The encoder converts the data points from their original space to a latent space for ordering. The decoder takes the points in the latent space, and outputs the points one by one. At

each step, the decoder computes the context of the current ordering status and selects one unordered point for output. Please refer to Section 3.2 for details. In this section, we will start with the notation and technical background. Then, we will introduce an abstract framework and explain how the conceptual modules are realized.

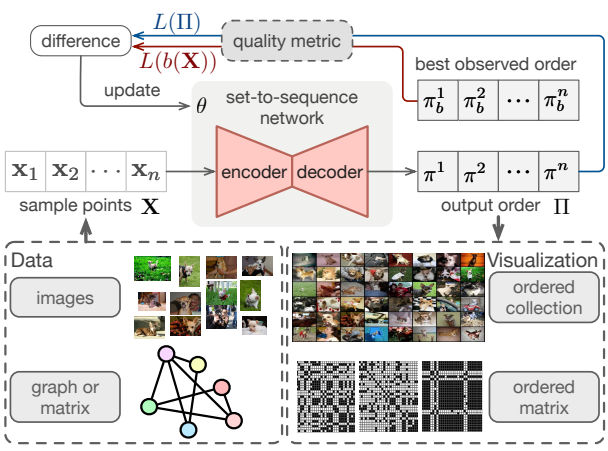


Fig. 2. An overview of VON workflow. The upper diagram illustrates the overarching framework and training process of VON, and the lower shows example data and ordering applications in visualization.

3.1 Notation and Background

This section introduces the notations and briefly explains the greedy rollout strategy, which is the core of our optimization framework. For other key techniques, such as the attention, multi-head attention (MHA), and the detailed optimization procedure, please refer to Appendix A.

Notation. As shown in Table 1, we denote the input set of data points as $\{\mathbf{x}_i\}$ and an individual point as $\mathbf{x}_i$. Similarly, we denote the set of hidden encodings at a layer as $\{\mathbf{h}_i\}$ and the encoding of an individual point as $\mathbf{h}_i$. We denote the order of a sequence as $\Pi = \{\pi^1, \pi^2, \cdots, \pi^n\}$, where each element π^t indicates the t-th data point in the ordered sequence. In general, we use the superscript to indicate the ordering step, and the subscripts to indicate specific instances. For subscripts, we use i and j to denote indices of points, o to denote all ordered points, and c to denote the ordering context (considering the ordering front and all ordered points). We use square brackets to denote the concatenation of multiple vectors. For example, $[\mathbf{h}_i, \mathbf{h}_j]$ indicates the concatenation of $\mathbf{h}_i$ and $\mathbf{h}_j$.

Greedy rollout strategy. At the core of our learning procedure is the greedy rollout strategy in reinforcement learning [86]. This strategy simulates the greedy selection of most likely data points through a sequence of steps, so that the model can efficiently learn the optimal solution from multiple greedy actions. Our optimization minimizes the expected loss by comparing the current model with the best-performing model, which is regularly updated to ensure consistent performance. To stabilize training, we freeze the baseline model's parameters for a certain number of epochs, similar to DQN [58]. Please refer to Appendix A for details.

TABLE 1
Notation used in this paper. The superscripts correspond to the ordering step, and the subscripts correspond to specific instances.

notation	meaning
i, j	indices of data points
$\mathbf{x}_i, \mathbf{x}_j$	input data points
$\mathbf{h}_i, \mathbf{h}_j$	hidden encodings of data points
$\bar{\mathbf{h}}$	the hidden encoding for all data points being ordered
t	the ordering step where t points are already ordered
$\mathbf{h}_o^t$	the hidden encoding of the already ordered points at step t
$\mathbf{h}_c^t$	the hidden encoding of the current ordering context at step t
π^t	the output data point at step t
$[\mathbf{h}_i, \mathbf{h}_j]$	the concatenation of two hidden encodings $\mathbf{h}_i$ and $\mathbf{h}_j$
$\mathrm{att}(\mathbf{h}_i, \mathbf{h}_j)$	the attention between hidden encodings $\mathbf{h}_i$ and $\mathbf{h}_j$

3.2 Conceptual framework

VON produces the order by sequentially outputting the data points, one at a step. As shown on the left of Fig. 3, VON considers both the overall structure of all input data points (gray area) and the structure of the previously ordered points (light red area). In particular, VON emphasizes the most recent order point (the red circle) in the ordered nodes. By stitching these three kinds of information, VON forms the context of the current ordering status (the dashed rectangle centering at the red circle). Finally, given the current context, VON determines the next data point to output.

The conceptual framework is illustrated on the right of Fig. 3. Note that the detailed network structure is omitted in this illustration, as we would like to focus on the data flow and the goal of each module. VON consists of an encoder and a decoder. The **encoder** transforms the input data points $\{\mathbf{x}_i\}$ into high-dimensional latent vectors $\{\mathbf{h}_i\}$ for ordering. The overall structure of points $\bar{\mathbf{h}}$ is obtained through a common practice by averaging the latent vectors of individual points [55].

The **decoder** takes in the latent vectors of individual points $\{\mathbf{h}_i\}$ and the overall description $\bar{\mathbf{h}}$ and output one point at a step. At each step t, the decoder executes the following modules to identify the next point to output:

- The *context computation* module updates the ordering context based on the information of the already ordered points $\mathbf{h}_o^t$, together with the individual points $\{\mathbf{h}_i\}$ and the overall structure $\bar{\mathbf{h}}$. The context computation merges these pieces of information into a single vector $\mathbf{h}_c^t$, describing the entire context to pick the next point.
- The *repositioning* module repositions the latent vector of each point $\mathbf{h}_i$ with respect to the current context $\mathbf{h}_c^t$. This can be considered as a learnable operator that normalizes the point coordinates based on the current context, as indicated by the red dashed rectangle in Fig. 3. It allows similar ordering strategies to be applied to points at different locations and scales.
- The *matching* module evaluates the reposition points $\mathbf{h}_i^t$ given the context $\mathbf{h}_c^t$. Each point is assigned a

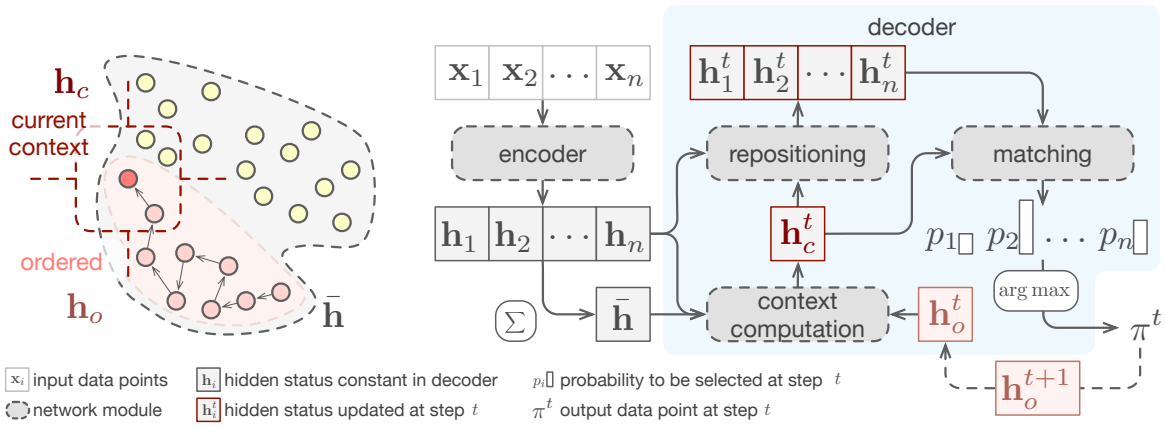


Fig. 3. The conceptual framework of VON. The left part shows the information considered in determining the next data point. The right part shows the data flow and the conceptual network modules. Note that the conceptual modules can be realized with various implementations.

probability p_i of being the next point in the sequence. The point with the largest probability will be the output π^t at step t.

At the end of step t, the hidden encoding of ordered points $\mathbf{h}_o^{t+1}$ will be updated accordingly. By repeating this procedure, the decoder will produce the entire sequence. After the entire sequence is generated, the quality metric will be applied to evaluate the sequence and guide the optimization of VON. This forces the VON to look forward and avoid greedy solutions, as the reward is evaluated on the complete sequences instead of at each step. In the following sections, we will discuss the potential realizations of each module.

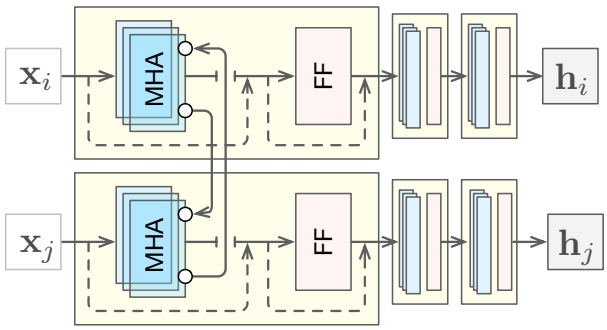


Fig. 4. The encoder network structure. Each attention layer (□) consists of a multi-head attention layer (MHA □) and a feed-forward layer (FF □). Data points interact with each other through the MHA layer. The dashed arrows indicate skip connections.

3.3 Encoder

The encoder should transform the input space into a latent space that is appropriate for ordering. As ordering multiple data points should fully consider their relationships, we use the attention mechanism from the transformer [78] to capture the interactions among points. As shown in Fig. 4, the encoder is a stack of multiple attention layers (yellow boxes), where each attention layer consists of a multi-head attention (MHA, blue box) layer and a feed-forward (FF, red box) layer.

Note that, unlike the original transformer [78], we do not incorporate the positional encoding, as the input point set is expected to be unordered. We should also note that the conceptual encoder can be realized by any other approaches that consider the joint structure formed by the data points as well. The alternative choices include the graph convolutional network and various kinds of message-passing networks. In addition, while VON expects the coordinates of points as the input, it may still handle the data points without intrinsic coordinates. Please refer to Section 3.5 for details.

3.4 Decoder

At each step, the decoder computes the context of the current ordering status, repositions all points to this context, and matches up the points with the context to identify the next point in the ordered sequence. In this section, we will explain the realizations of the conceptual modules, i.e., context computation, repositioning, and matching modules.

Context computation. The context computation aggregates information of the overall structure $\bar{\mathbf{h}}$, individual points $\{\mathbf{h}_i\}$, and the already order points $\mathbf{h}_o^t$ at step t. Similar to the encoder, we use the attention mechanism for incorporating the interactions among multiple entities:

$$\mathbf{h}_c^t = \sum_i \mathrm{att}_c([\bar{\mathbf{h}}, \mathbf{h}_o^t], \mathbf{h}_i) \cdot \mathrm{val}_c(\mathbf{h}_i), \tag{1}$$

where att_c and val_c form the attention module in context computation.

This module updates the context using both the ordered and unordered data points, for fully considering the structure formed by all points and coping with various quality metrics (**C1**). This is necessary for quality metrics involving pairwise computation (e.g., the stress in 1D stress majorization). However, for the quality metrics where only consecutive points in the resulting sequence are evaluated (e.g., path length in TSP), the ordered points may be ignored when updating the context. In addition, by using all points to update the context, we find that it is safe to concatenate the first point and the most recent point for representing the ordered nodes, i.e., $\mathbf{h}_o^t = [\mathbf{h}_{\pi_1}, \mathbf{h}_{\pi_{t-1}}]$.

Repositioning. The repositioning module aligns the individual points to the current ordering context $\mathbf{h}_o^t$. Conceptually, this can be expressed as a function that given the

latent vector of a point $\mathbf{h}_i$ and the context $\mathbf{h}_o^t$, produces the corresponding repositioned vector $\mathbf{h}_i'$. While we rely on the attention mechanism to aggregate information in most of the modules, we examine different realizations for this module, as the performance of different network structures is not fully known for repositioning. Specifically, we experiment with the following structures:

First, an *attention-based* realization. This realization attaches the latent representation to the ordered points using two attention layers:

$$\mathbf{h}_i' = \mathrm{att}_1(\mathbf{h}_o^t, \mathbf{h}_c^t) \cdot \mathrm{att}_2(\mathbf{h}_o^t, \mathbf{h}_i) \cdot \mathbf{h}_o^t + \mathbf{h}_i. \tag{2}$$

In this process, we consider two major factors in weighting the ordered points $\mathbf{h}_o^t$: what are the connections between the current ordering context and the already points (i.e., $\mathrm{att}_1(\mathbf{h}_o^t, \mathbf{h}_c^t)$), and what are the relationships between each data point and the ordered ones (i.e., $\mathrm{att}_2(\mathbf{h}_o^t, \mathbf{h}_i)$).

Second, an *multi-layer perceptron (MLP)-based* realization. This realization can be expressed formally as:

$$\mathbf{h}_i' = \mathrm{MLP}([\mathbf{h}_c^t, \mathbf{h}_i]). \tag{3}$$

This means that we will transform the latent $\mathbf{h}_i$ by jointly considering the context $\mathbf{h}_c^t$. Note that MLP may not be used solely to realize other modules, as it does not scale well with varying numbers of input data points. For repositioning, we mainly consider the interactions between individual points and the context. Therefore, MLP can be sufficient as it blends the different dimensions in the two vectors and allows them to fully interact with each other for capturing both local and global information. In our implementation, we use a two-layer MLP with ReLU activation.

Third, a *convolution-based* realization. The convolutional module transforms the latent representations with the following formula:

$$\mathbf{h}_i' = \mathrm{Conv}([\mathbf{h}_c^t, \mathbf{h}_i]) = \sum_k \mathbf{W} * [\mathbf{h}_c^t, \mathbf{h}_i]_k + b, \tag{4}$$

where k represents a channel in convolution and "$*$" is the valid cross-correlation operator. The convolution-based realization is similar to the MLP-based one, as they both concatenate the two vectors so that different dimensions can fully interact. Instead of stacking multiple layers of networks to approximate complex transformations, the convolutional module aggregates information collected from multiple channels. The effectiveness of these three variations will be examined in our experiments. Please refer to Section 4 for details.

Matching. The matching module produces a probability for each point, indicating how likely will the point be selected to be the next output. To incorporate the relationships among data points, we use two MHA layers for this module. We explicitly removed the previously ordered points from being selected by setting their probability to negative infinite. Note that the matching module does not evaluate the points using the quality metric explicitly. Instead, it learns the matching criteria guided by the metric as the loss function.

3.5 Data preparation

The input of VON is a set of data points of the same number of dimensions. For points in Euclidean space (i.e., 2D or 3D point clouds), it is straightforward to use their coordinates as input. For points without intrinsic coordinates, the input may be derived with the following strategies:

Distance-based initialization. If a distance metric is available, we may compute the distance matrix to form the input. For example, for graph data, the graph distance between nodes may be used. In this way, each row/column corresponding to a data point can be used as its coordinates. An alternative choice is to use embedding methods (e.g., t-SNE) to embed the points into an Euclidean space based on the distance metric. This strategy may be applied to graph data and axis reordering, where similarities between coordinates are often defined. Note that VON can tolerate less ideal input as its encoder will reshape the input space. Therefore, the input does not need to be perfect, as long as the data points are distinguishable.

Deep representation-based initialization. We may use deep representation approaches to embed the data points into latent space. For example, we may use an autoencoder for images, a graph representation network for graphs, and multi-layer perceptrons (MLPs) for data points with attributes. As pretraining becomes a common technique, latent embeddings are often available for various kinds of data. In our experiment, we use the deep representations of three image datasets (i.e., CIFAR-10, FashionMINST, and ImageNet) and one scholarly dataset (i.e., CORA).

4 Evaluation

In this section, we evaluate the performance of VON under different experimental settings. We first use an ablation study to determine the best instantiation of VON and examine the effectiveness of each module. The best configuration of VON will be used for the remaining experiments. For these experiments, we consider two scenarios: ordering randomly selected points and ordering fixed data points. The first scenario is suitable for ordering data points that may be repeatedly sampled from the same distribution multiple times, e.g., ordering images interactively selected from a collection. The second scenario mimics the applications such as matrix reordering and axis ordering. For the first scenario, we consider eight baselines, six metrics, six datasets, four sampling strategies, and three sampling sizes, leading to **1059 sets** of experiments for a comprehensive evaluation. For the second scenario, we compare VON with baselines in axes reordering and matrix reordering. For axes reordering, we consider four baselines and four datasets; and, for matrix reordering, we consider eight baselines, two dynamic datasets, and five embedding approaches. Please refer to Section 4.1 and the Appendix for detailed settings.

4.1 Experimental settings

This section lists the datasets, methods being compared, and quality metrics used in our experiment. Note that for datasets without coordinates, embedding approaches are used. The embedding approaches being studied are also listed. More details can be found in the Appendix. In this

section, we explain our sampling strategies, as they are closely related to our experiment.

Datasets. We experiment with twelve datasets, including six datasets with data points in respective latent embedding spaces (i.e., Fashion-MNIST, MNIST, CIFAR-10, ImageNet, CORA, and DBLP), four high dimensional datasets (i.e., Coal Disaster, Cars, AAUP, and Census Income) and two dynamic graph datasets (i.e., FLT and SCH).

Methods compared. We compare VON against three generic approaches (i.e., attention model (AM) [47], simulated annealing (SA) [36] and nearest neighbor (NN) [35]), four specialized methods for images reordering (i.e., IsoMatch [31], Kernelized Sorting (KS) [67], LAS and FLAS [7]), one specialized method for axes reordering (i.e., random swapping), and four specialized algorithms for matrix reordering (i.e., C-LO-δ_I, U-LO-δ_I, C-BC and U-BC [76]).

We choose the baseline algorithms considering their targeted problem structures, usage scenarios, and popularity. For the generic solvers, SA [85] is widely applicable, as it does not assume any specific problem structure; NN [35] simulates the most common strategy in ordering by selecting neighboring/similar nodes; and AM [47] allows us to examine performance against the SOTA learning-based approach. For the specialized solvers, SM [28] optimizes the stress, a general function modeling the pairwise relationships between points; and other baselines are selected from typical visualization applications, including image ordering, axis reordering, and matrix reordering. Please see Appendix B for detailed explanations of the baselines and their setup. We believe these approaches cover representative problem structures and usage scenarios.

Quality Metrics. Various kinds of quality metrics have been proposed to evaluate the quality of ordering results in terms of preserving structures, perception, and aesthetics. In this experiment, we study two quality metrics for the 1D layout problem (i.e., the traveling salesman problem (TSP) distance and the stress in stress majorization), three metrics (i.e., distance preservation quality (DPQ), energy in IsoMatch, and kernelized sorting objective) for images reordering, two metrics (i.e., symmetry and correlation) for axes reordering, and four metrics (i.e., Moran's I, Linear Arrangement (LA), profile (PR), and bandwidth (BW)) for matrix reordering.

Embedding methods. When the coordinates of data points are not available in a dataset (e.g., the SCH dataset), an embedding approach is needed to produce the coordinates. We examine the impact of five embedding approaches with distinct characteristics, namely, principal component analysis (PCA), truncated singular value decomposition (T-SVD), autoencoder (AE), Isomap, and locally linear embedding (LLE).

Sampling strategies. Four sample strategies are used to resemble dynamic behaviors during interaction and examine the performance across scales. The four strategies are: 1) the **global sampling ('g')** that randomly samples from all points; 2) the **nearest neighbor sampling (local-'n')** that samples the nearest points of the specified number; 3) the **radius-based sampling (local-'r')** that randomly samples the points within a given radius of a specific point; and, 4) the **label-based sampling (local-'l')** that randomly samples points with a specific label. The strategy 'g' mimics the behavior of browsing the entire collection, 'n' and 'r' emulates the behavior of scanning items similar to a selected one, and 'l' corresponds to reviewing items of the same category. Based on the findings and experimental settings in [2], [39], [43], [56], [57], [62], three sampling sizes: 50, 100, and 150 are considered for each sampling strategy. Please refer to Appendix B for details.

We further denote a model that is first trained with global samples and then with the local samples as "gl", and a model trained with local samples and then global samples as "lg". For example, "50gl" indicates that the model is trained with 50 global samples and then 50 local samples. We denote a model trained with global and local samples in random orders as "mix". These settings will be useful in discovering the detailed behavior of models in transferring knowledge across scales and distributions. Please note that for each setting, we have the same number of global and local samples.

4.2 Ablation study

In the Ablation study, We first examine the performance of the three variants of VON and the impact of sampling strategies that produce the training data. Then, we study the individual impact of each key module in the best variant. The three variants are VONs based on attention mechanism (VON-a), MLP (VON-m), and convolution (VON-c), corresponding to Eq. (2) to Eq. (4), respectively.

TABLE 2
Ordering performance using Fashion-MNIST dataset and TSP quality metric. The content in the parentheses following a model name denotes the sampling strategy used to train that model. **I**. Comparing models trained with point sets of 50 points. **II**. Comparing VON-m models trained with different point sets of different sizes and scales. **III**. Comparing models with mixed-scale sampling strategies.

		50g	50l	100g	100l	150g	150l
I	VON-a(50g)	608.1	349.7	920.8	639.4	742.7	826.3
	VON-a(50l)	1024.3	246.0	1312.3	448.1	609.3	693.6
	VON-m(50g)	**588.3**	183.3	**879.8**	357.1	504.5	540.5
	VON-m(50l)	635.9	**177.6**	940.1	**343.8**	**488.6**	**526.1**
	VON-c(50g)	674.4	229.3	965.7	433.9	616.5	658.9
	VON-c(50l)	663.8	212.4	1030.4	408.5	566.7	605.3
	AM(50g)	616.6	427.3	1018.7	1087.9	1759.5	2007.4
	AM(50l)	677.8	229.1	1083.8	445.2	605.4	680.3
II	VON-m(100g)	**599.5**	185.7	**954.9**	357.6	515.3	563.5
	VON-m(100l)	635.5	**182.3**	971.5	**320.9**	531.0	563.9
	VON-m(150g)	751.0	208.3	1113.9	400.7	**487.9**	541.6
	VON-m(150l)	698.8	193.5	1220.7	386.2	581.0	**538.2**
III	VON-m50mix	614.0	217.8	**919.8**	402.6	545.9	**590.9**
	VON-m(50gl)	**609.5**	199.6	941.6	379.2	**540.4**	**590.9**
	VON-m(50lg)	694.0	**180.7**	1077.3	**357.2**	558.4	592.9
	AM(50mix)	693.4	462.0	1003.1	1003.6	1010.2	1474.4
	AM(50gl)	689.2	402.3	1128.3	1057.2	1403.7	1628.2
	AM(50lg)	2150.1	260.3	2373.9	620.1	951.7	1025.0

Identifying the best VON variant. Table 2.I shows the results of VON-a, VON-m, and VON-c, together with the results of AM for reference. Each model is trained with two sets of training data (i.e., 50g and 50l). We do not create training data with multiple sampling strategies to reduce the undesired impact of sampling strategies.

We can see that VON-m outperforms the other models in every case, especially in terms of its transferability across scales. We find that VON-m(50l) (trained on local data)

outperforms the other models on the global data 100g, and VON-m(50g) (trained on global data) delivers smaller distances than the other models on the local data 100l.

In contrast, VON-a and AM show a clear performance decline on testing data at different scales. For example, VON-a(50l) results in large distances in 50g and 100g tests. This indicates that the attention mechanism is less effective in repositioning the points with linear combination. AM's performance drop is even more notable, as it does not incorporate any mechanism to transfer knowledge across scales. For example, AM(50g) decreases from 616.6 in 50g to 1759.5 in 150g. Therefore, *in the remaining experiments, we will use VON-m as our model and compare it with other models*.

Additionally, we discuss the sampling strategy in detail and examine the impact of each module. Please refer to the Appendix C.I for details.

4.3 Performance for Ordering in Dynamic Scenario

We conduct extensive experiments to examine the performance of the best variant of VON (VON-m) in a dynamic scenario, where the point sets are sampled from a large collection for visualization. Specifically, for general ordering task using both the image and scholarly data, we compare VON with eight baselines with different characteristics: AM (learning-based), SA (naturally inspired), NN (heuristic), SM (specialized for the stress metric), using three quality metrics with different problem structures: TSP (linear route), stress (complete graph), and Moran's I (distance matrix). Additionally, for image ordering, we further consider four baselines (LAS, FLAS, IsoMatch, and Kernelized Sorting) with their respective objectives as optimization goals. We perform 1059 sets of experiments on six datasets. Each set of experiments is performed on 100 groups of testing points to deliver stable results.

4.3.1 Quantitative performance

In this section, we first discuss the general performance of VON-m using all six datasets (as shown in Fig. 5), and then the image ordering performance using the image datasets against specialized algorithms (as shown in Fig. 6). Note that we use performance improvement percentages for comparison instead of the original metric values, because the original values may vary substantially in magnitude.

Overall performance using all datasets. For the general ordering scenario, VON-m outperforms the baselines in all the 120 pairs of comparisons, with all the improvement percentages being positive in Fig. 5. Even for individual datasets, VON-m improves the baselines in 629/630 sets of experiments. The only exception is found in the Fashion-MNIST dataset, where the TSP distance of NN in 150r sampling is 0.29% smaller than VON-m, as shown in Fig. 4 in Appendix C. This shows that VON-m delivers stable performance across datasets, metrics, and sampling strategies in general ordering tasks.

Improvement for different metrics. We find that the pattern of performance improvement for different metrics varies across baselines. AM and SA share a similar pattern, where the improvement in TSP is large and the improvement in stress is small. Since stress considers all pairwise distances, we expect it to be less sensitive to the order of points, leading to smaller improvement. However, we observe an opposite trend for NN. This indicates that the nearest neighbor heuristic excels in planning linear routes, but falls short in providing a global picture needed to match the complete graph structure of stress. Note that stress majorization produces the largest stress as it does not explicitly take the order into consideration.

Improvement for different sampling strategies. For the learning-based approach (AM), the performance gaps in global samples are larger than the gaps in local samples for all three metrics. This indicates the lack of generalizability across scales in AM. For the conventional approaches (SA and NN) without the learning procedure, their performance across scales relies on the metric. For Moran's I and TSP, the improvement over these two approaches in global samples is larger than that in local samples; for stress, the trend is the opposite. Since SA and NN do not depend on data distribution, this may indicate that it is easier to solve Moran's I and TSP locally, while it is easier to solve stress globally.

Improvement for different sample sizes. Because VON-m is trained on 50mix, we expect a decrease in its performance when the number of samples increases from 50 to 100 and 150. This may further lead to a decrease in performance improvement. For the learning-based approaches (AM and VON-m), their performance becomes similar when less knowledge can be leveraged from the training data. For the conventional approaches (SA and NN), as their performance does not rely on the training data, the performance gap between these approaches and VON-m may shrink as well. In Fig. 5, while observing this decrease in most cases, two exceptions (AM for TSP and SA for TSP) are found. While the behavior of AM may still be attributed to the transferability, we suspect that the pairwise swapping strategy of SA may be less effective in solving a linear route problem when the number of points increases.

Performance in image ordering. For image ordering, VON-m outperforms 38/48 pairs of comparison against specialized algorithms on their respective metrics, as shown in Fig. 6. It performs better than FLAS and IsoMatch in every setting, and LAS in most settings except for 50r and 150r. Using the KS objective function, VON-m outperforms KS in the four cases with a sampling size of 50, but loses in the other cases. The performance gap becomes larger when the size increases from 100 to 150. As VON-m is trained on the sample size of 50, the performance loss may be due to the difference between the testing data and the training data. We suspect that the ordering strategy should be adjusted when dealing with different distributions for certain metrics. In this case, we recommend collecting training data from users to better align the training and testing data.

Findings. The extensive experiments demonstrate the strong impact of the complicated connections between the algorithm design, data distributions, and problem structures. This highlights the importance of transferability across scales and metrics, which is unavailable in existing learning-based approaches. For the general ordering tasks, we should note that the performance gap between AM and other approaches mainly originates from the diverse data distributions. For the main distribution (50g) in the training data, AM outperforms the other three baselines in

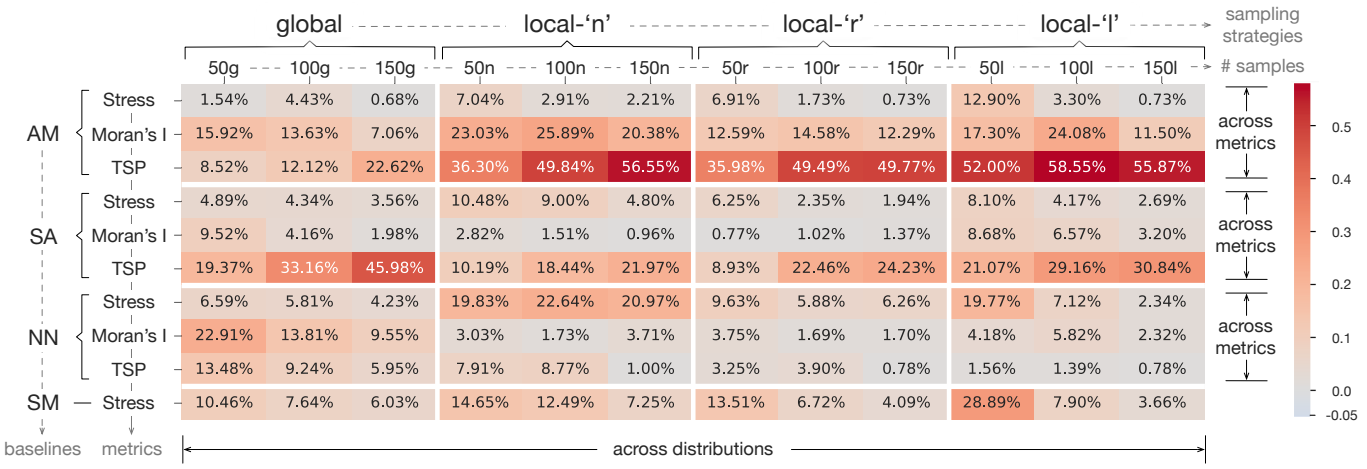


Fig. 5. Average performance improvement of VON over AM, SA, NN, and SM for three metrics, using the MNIST, Fashion-MNIST, CIFAR-10, DBLP, Cora, and ImageNet datasets. All dataset is experimented with three sampling strategies: global, nearest neighbor (local-'n'), and radius-based (local-'r'). The first three datasets with labels are further examined with the label-based (local-'l') sampling. The experiment of each dataset is conducted over 100 sets of samples.

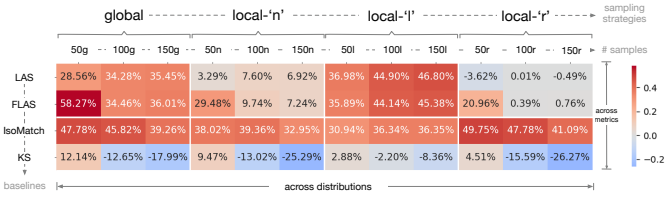


Fig. 6. Average performance improvement of VON over LAS, FLAS, IsoMatch, and KS using the MNIST, Fashion-MNIST, CIFAR-10, and ImageNet datasets. Each dataset is experimented with four sampling strategies: global, nearest neighbor (local-'n'), radius-based (local-'r'), and label-based (local-'l') sampling. The experiment of each dataset is conducted on 100 sets of samples. Each specialized method uses its own applicable metric: LAS/FLAS uses DPQ, IsoMatch uses Energy Value, and KS uses its own objective function.

6 out of 7 cases, except SA for Moran's I. However, when the testing data deviates from the main pattern in training data, AM's performance quickly decreases. In contrast, *VON achieves better generalizability, as evidenced by the performance improvement over generic approaches in every case, and over specialized algorithms in a considerable portion of cases (38/48) for image ordering.*

More importantly, the performance variation across scenarios infers the difficulty in selecting an appropriate solver for a given scenario. This either involves the prohibitive trial-and-error effort to examine multiple ordering techniques or requires the chosen technique to generate better results in all cases. In light of this observation, we emphasize the importance of worst case performance. In our experiments, every baseline may deliver the worst performance among all approaches in certain scenarios, e.g., AM for TSP with local samples, SA for TSP with global samples, and NN in stress with local-'nn' samples. Additionally, when considering the worst cases for individual datasets, every baseline has at least a 50% performance gap compared to VON, as shown in Fig. 4. In contrast, VON is at least comparable to the baselines in the worst cases for general ordering tasks, and it outperforms the image ordering algorithms in a large portion of cases (38/48). *This renders it to be a reasonable choice in most scenarios without exhaustive examination.* However, the performance gap between VON-m and KS, when the sampling size deviates from training data, also emphasizes that we should keep the training data as close to the real scenario as possible.

4.3.2 *Qualitative case study*

In Fig. 7, we qualitatively compare the ordering results of VON-m and three other approaches using 50 randomly selected images from the CIFAR-10 and Fashion-MNIST datasets, respectively. The quality of the order can be visually assessed from the ordered images, where effective approaches should place similar items close to each other.

For Fashion-MNIST, the difference in ordering quality is easier to perceive. We find that VON-m, AM, and NN produce better results, clearly dividing clothes, bags, and shoes into three groups; SA is less pronounced, as the items of different categories often appear in a mixed order.

For CIFAR-10, the quality is somewhat difficult to visually evaluate in general methods, as the similarity may be defined in multiple dimensions (e.g., color and content). We can still observe a clear pattern from the results of VON-m and NN. For example, VON-m seems to place images with green background in the first row, images with sky background in the second and third row, and images with animals in the two bottom rows; similarly, NN tends to place images with sky in the second and third row, vehicles in the third row, and animals in the last row. The visual assessment is in line with the quantitative results, where VON-m and NN excel in solving TSP.

However, we should note that the visual assessment incorporates both the similarity measure and ordering quality. For this paper focusing on generic solver, we still mainly rely on the quantitative results. For more results regarding other metrics, please refer to Appendix C.II.

4.4 Performance for Ordering in Static Scenario

For the static scenario, we study two typical applications: matrix reordering and axes reordering. In these applications,

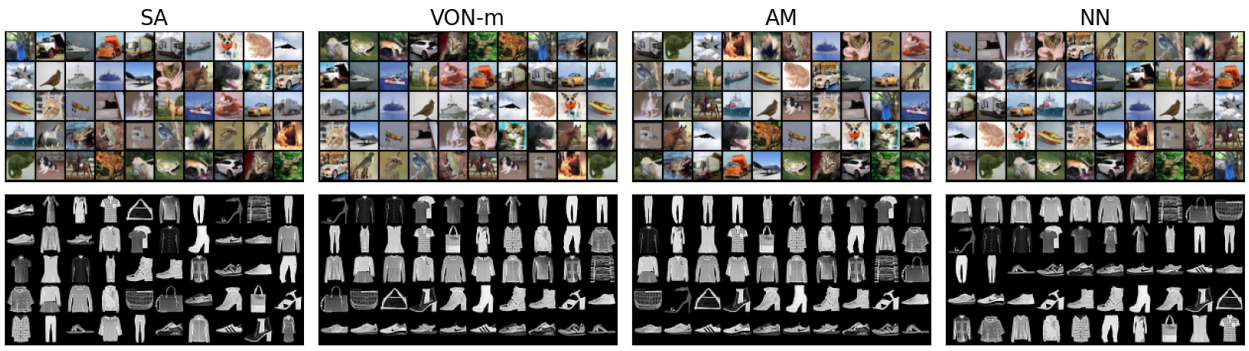


Fig. 7. Comparison of image ordering results using TSP as the quality metric. The first row shows the results using CIFAR-10 dataset and the second row shows those using Fashion-MNIST. The four columns show the results of SA, VON-m, AM, and NN from left to right, respectively.

TABLE 3
Axes reordering performance of VON-m, AM, SA, RS using Coal Disaster, Cars, AAUP, and Census Income datasets. The star plot is assessed with the symmetry quality metric [64], and the parallel coordinate is assessed with correlation [75].

		Coal Disaster (5)	Cars (7)	AAUP (14)	Census Income (42)
star plot	VON-m	**0.0743**	**0.2166**	**0.1321**	**0.2484**
	AM	**0.0743**	0.2285	0.1492	0.2602
	SA	**0.0743**	**0.2166**	0.1539	0.2641
	RS	**0.0743**	**0.2166**	0.1482	0.2603
parallel coordinate	VON-m	**0.0244**	**0.3593**	0.4516	**0.0789**
	AM	0.0264	0.3714	0.4664	0.0843
	SA	**0.0244**	0.3749	**0.4500**	0.0808
	RS	**0.0244**	0.3754	0.4569	0.0839

the data points (i.e., axes or matrix rows/columns) do not change dynamically.

4.4.1 Performance on axes reordering

We compare VON-m against AM, SA, and random swapping [64] in reordering the axes of star plots and parallel coordinates. We experiment with four datasets that are commonly used in axes reordering [64]: the Coal Disaster dataset with 5 attributes, Cars with 7 attributes, AAUP with 14 attributes, and Census Income with 42 attributes.

Quantitative result. We used symmetry [64] as the quality metric for star plot axes reordering, and correlation [75] for parallel coordinate reordering. Note that we transform the metric for consistency, so that smaller values indicate better performance. Table 3 shows that, for the datasets with smaller numbers of axes (Coal Disaster and Cars), most approaches can identify the optimal solutions. This leads to identical metric values, i.e., 0.0743 in Coal Disaster, 0.2166 in Cars for star plots, and 0.0244 in Coal Disaster for parallel coordinates. However, when the number of axes increases, the solution space grows factorially and it becomes prohibitive to search the entire space for the optimal solution. For Census Income with 42 axes, we can observe larger gaps between VON-m and other approaches. Overall, VON-m performs the best in 7/8 cases. The only exception is the parallel coordinates reordering using AAUP, where VON-m ranks second (0.4516) right after SA (0.4500).

Qualitative study. Fig. 8 shows the axes reordering results for parallel coordinates, using AAUP and Cars datasets as examples. The visual quality seems to be more closely related to the nature of data rather than the approach used. We can see stronger correlations between neighboring axes in AAUP than those in Cars. Unlike SA and RS, which rely on randomness to explore the solution space, VON-m and AM seem to learn different ordering strategies. AM tends to place the axes with discrete values in the middle, leading to distracting structures. In contrast, *VON-m tends to place the discrete axes at the two ends, showing a clearer pattern*. Note that SA achieves better results in AAUP, but its visualization may not demonstrate clear advantages.

4.4.2 Performance on matrix reordering

In this section, we examine the performance of VON in matrix reordering. We use the FLT dataset and Moran's I for comparison in the dynamic graph setting as [76]. In this setting, VON orders the rows and columns in multiple matrices simultaneously, guided by the average Moran's I overall matrices. Note that in graph data, nodes may not associate with coordinates and embedding approaches may be used to generate the coordinates. We will study the impact of embedding approaches in Section 5. In this experiment, we average the adjacency matrices of graphs at individual steps to produce the initial coordinates.

Quantitative results. Fig. 10 shows the distribution of Moran's I at individual steps for each approach. The cyan dashed lines indicate the average Moran's I, which is used to evaluate the overall quality for dynamic graphs [76]. In terms of the average, VON-m (0.399) outperforms the other generic approaches, namely, AM (0.229), NN (0.105), and SA (0.090), by a large margin. Compared with the specialized algorithms, VON-m clearly outperforms U-BC (0.143) and C-BC (0.236) and delivers similar performance as the SOTA approaches, i.e., C-LO-δ_I (0.433) and U-LO-δ_I (0.419). This shows the overall effectiveness of VON-m to solve the simultaneous reordering problem. In terms of the minimum of Moran's I (i.e., the quality of the worst matrix), VON-m (0.191), C-LO-δ_I (0.193), and U-LO-δ_I (0.119) are the only three approaches that are able to maintain positive minimums. This shows their ability to produce stable results for individual graphs.

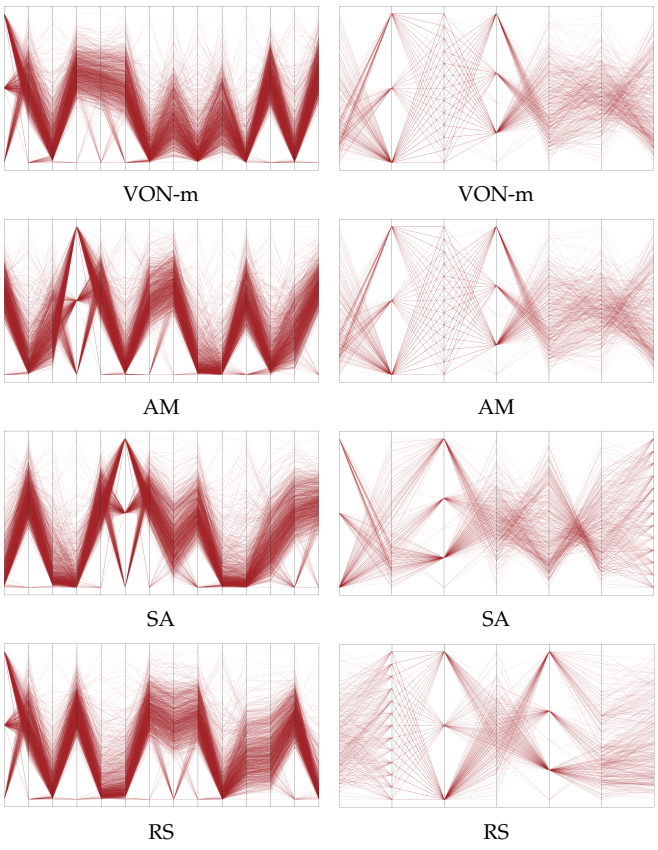


Fig. 8. Comparison of parallel coordinates reordering results of our VON-m, AM, and random swapping (RS), using (a) 'AAUP' and (b) 'Cars' datasets and the correlation metric.

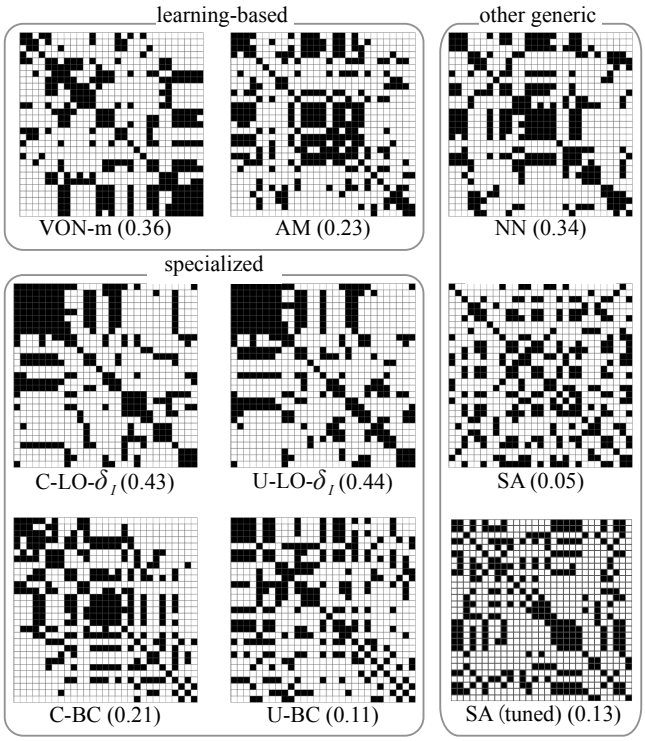


Fig. 9. Matrix reordering results using FLT dataset and Moran's I. The results of different approaches are shown using a single graph G_{78}, which best reflects the overall performance (in terms of correlation). The number below each matrix shows the corresponding Moran's I for G_{78}. Note that the rows and columns are ordered simultaneously for all 96 graphs in FLT, and the performance of approaches may not be assessed in a single step. Please refer to Fig. 7 in the Appendix for more results sampled at an interval of ten.

Note that although the input of VON-m (the average graph) is known to be less ideal [76], VON does not solely rely on the input to guide the ordering. Instead, it uses the quality metric to transform the input space for ordering. As Moran's I is computed at each graph, VON can still collect sufficient information to learn effective strategy.

In Fig. 9, we use the graph G_{78} as a representative graph, which is selected based on its correlation to the average scores. This aims to reflect the overall performance, although some methods may deliver worse performance than average at this specific graph, e.g., 0.05 at G_{78} vs 0.09 on average for SA. We should further note that, while we fix the parameters for all methods to examine their usability across different tasks and scales, SA may benefit from parameter training on specific tasks. In Fig. 9, we can see that Moran's I of SA increases from 0.05 to 0.13 at G_{78} with a more exhaustive searching setting. Its average performance also increases from 0.09 to 0.258. Please refer to Appendix C.IV.2 for more discussion.

Qualitative results. Fig. 9 shows the matrix reordering results with a typical graph G_{78}. The high-quality matrices with larger Moran's I exhibit larger black blocks, meaning that they preserve the communities in the corresponding graphs. We can see that VON-m, C-LO-δ_I, U-LO-δ_I, and NN produce high-quality matrix reordering results for this graph, with Moran's I larger than 0.3. However, we should also note that the order is determined simultaneously over all graphs, and the single graph does not reflect the overall quality. For example, while AM outperforms NN on average, its performance (0.23) falls behind NN (0.34).

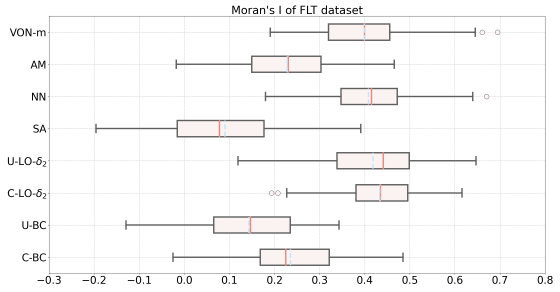


Fig. 10. Matrix reordering performance using FLT dataset and Moran's I metric. The boxplot shows the distribution of Moran's I values over 96 graphs at individual time steps in FLT, the blue dashed line shows the mean value, and the red line shows the median. Larger Moran's I is better.

4.5 Discussion

Scope of application. VON is designed to be generic and work across applications in visualization. Our experiment demonstrates that it can be applied to different scenarios, where the points are fixed (e.g., matrix reordering and axes

ordering) or dynamically generated in interaction. It can optimize for various quality metrics. In terms of data, our experiment shows that VON can be applied to points with or even without coordinates.

We expect that VON may be used for more complicated data, as pre-training on many modalities of data becomes more readily available in the era of large models. For example, well-trained encodings for text and images can be accessed easily nowadays. Additionally, a rich suite of representation techniques can be used to encode one's own data as well. For example, for heterogeneous graphs, meta-path-based embedding learning techniques [42], [82] can provide the encodings that preserve the semantic meaning in the graph. However, we should also note that VON is still somewhat sensitive to the input embedding, as shown in Section 5. Therefore, improving the encoder for robustness against input should be further investigated.

VON's potential applications are diverse. It may be used by developers as a *component in visual analytics interface* to order items interactively, by visualization researchers as a reliable baseline for examining future ordering algorithms, and by perception researchers as a solver for customized metrics without existing optimization algorithms. For example, Appendix C.II shows the gap between metric values and human perception, confirming the need for perceptual metrics. In this scenario, VON can be a reasonable choice when a specialized solver is not immediately available, allowing perception researchers to focus on metric design without worrying about how to optimize the metric. Additionally, VON's adaptability to user-specific requirements positions it as a potential solution in user-centric applications and human-in-the-loop analyses.

Efficiency. In terms of timing efficiency, VON may not outperform the most advanced specialized algorithms, but it still delivers a reasonable response time for interactive systems. More importantly, VON can be used in any system where ordering is required without additional effort to develop specialized algorithms.

In terms of memory efficiency, the amount of network parameters and the data in one batch should be considered. By using the attention mechanism for modeling pairwise relationships between points, the size of network parameters only relies on the number of dimensions but not the number of nodes, which reduces the required space. In the training stage, the space for data relies on the number of nodes, the number of dimensions, and the batch size. In our experiment, for 2D points with a batch size of 100, ordering 20 points requires 1.5GB of graphics memory, ordering 50 points requires 4.6GB, and ordering 100 points requires 8.1GB, which is often available on modern graphics cards. After training, in the inference stage, the memory costs for ordering 20, 50, and 100 points become 1.2GB, 1.4GB, and 1.9GB, respectively. When the memory size is a concern, one may reduce the batch size to support larger point sets. One may also train VON on smaller sets and apply it to order larger ones. Additionally, we should note that the memory required in the inference stage is much smaller as only one point set is processed at a time. This means we may train VON on more powerful machines and use it on less powerful ones.

5 Conclusions and Future Work

We propose VON that can facilitates various kinds of ordering tasks across scales and quality metrics. We propose a conceptual framework for embedding the input points into a latent space that is optimized for ordering task. The framework uses attention to collect global information regarding all data points being ordered and local information regarding the current ordering status as well. We introduce a novel repositioning module that places the data points into the ordering context, so that it may learn common structures regardless of the location and scale of point sets. We demonstrate with experiments that our approach can outperform most baselines, especially in generalizability and transferability. We also examine the performance against specialized SOTA approaches and show that VON can achieve comparable ordering results. We evaluate VON on twelve data sets with eleven metrics, covering various kinds of applications.

Future work. Our method still suffers from several limitations. In the future, firstly, we would like to improve the efficiency of VON in both training and inference stages. We may further leverage knowledge distillation to reduce the size of network for interactive performance in inference, and collect knowledge from similar datasets of problem structures to speedup the training procedure. This may help VON to better adapt to new tasks. Secondly, to capture global and local structures more effectively, we may further design structures that are specifically designed for repositioning purpose. Thirdly, we want to explore the encoding embedding process to enhance the information feed into the model. Finally, we can also design or discuss special methods for costs that cannot be minimized, making them applicable in VON. Finally, we would like to explore the possibility of learning ordering strategies directly from users. This may further reduce the cost to design quality metrics for different applications.

Acknowledgements

This research was supported in part by the National Natural Science Foundation of China through grants 62172456 and 62372484. The authors would like to thank the anonymous reviewers for their insightful comments.

References

[1] Y. Ahn and Y.-R. Lin. FairSight: Visual analytics for fairness in decision making. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1086–1095, 2020.

[2] N. Al-Naami, N. Médoc, M. Magnani, and M. Ghoniem. Improved visual saliency of graph clusters with orderable node-link layouts. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):1028–1038, 2025.

[3] M. Ankerst, S. Berchtold, and D. A. Keim. Similarity clustering of dimensions for an enhanced visualization of multidimensional data. In *Proceedings IEEE Symposium on Information Visualization*, pages 52–60. IEEE, 1998.

[4] A. Artero, M. de Oliveira, and H. Levkowitz. Enhanced high dimensional data visualization through dimension reduction and attribute arrangement. In *International Conference on Information Visualisation*, pages 707–712, London, UK, July 2006. IEEE.

[5] Aviz. Multipiles. https://aviz.fr/~bbach/multipiles/. Accessed: 2023-03.

[6] D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*, 2016.

[7] K. U. Barthel, N. Hezel, K. Jung, and K. Schall. Improved evaluation and generation of grid layouts using distance preservation quality and linear assignment sorting. In *Computer Graphics Forum*, volume 42, pages 261–276, 2023.

[8] M. Behrisch, B. Bach, N. Henry Riche, T. Schreck, and J.-D. Fekete. Matrix reordering methods for table and network visualization. *Computer Graphics Forum*, 35(3):693–716, 2016.

[9] M. Behrisch, M. Blumenschein, N. W. Kim, L. Shao, M. El-Assady, J. Fuchs, D. Seebacher, A. Diehl, U. Brandes, H. Pfister, et al. Quality metrics for information visualization. *Computer Graphics Forum*, 37(3):625–662, 2018.

[10] M. Behrisch, T. Schreck, and H. Pfister. GUIRO: User-guided matrix reordering. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):184–194, 2020.

[11] M. Blumenschein, M. Behrisch, S. Schmid, S. Butscher, D. R. Wahl, K. Villinger, B. Renner, H. Reiterer, and D. A. Keim. SMARTexplore: Simplifying high-dimensional data analysis through a table-based visual analytics approach. In *IEEE Conference on Visual Analytics Science and Technology*, pages 36–47, Berlin, Germany, Oct 2018. IEEE.

[12] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[13] M. J. Brusco, H.-F. Köhn, and S. Stahl. Heuristic implementation of dynamic programming for matrix permutation problems in combinatorial data analysis. *Psychometrika*, 73:503–522, 2008.

[14] C. Bu, Q. Zhang, Q. Wang, J. Zhang, M. Sedlmair, O. Deussen, and Y. Wang. SineStream: Improving the readability of streamgraphs by minimizing sine illusion effects. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1634–1643, 2021.

[15] L. Byron and M. Wattenberg. Stacked graphs – geometry & aesthetics. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1245–1252, 2008.

[16] G. Carenini and J. Loyd. Valuecharts: analyzing linear models expressing preferences and evaluations. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '04, page 150–157. Association for Computing Machinery, May 2004.

[17] CCF. Tiers a and b from china computer federation. https://www.ccf.org.cn/. Accessed: 2019-03.

[18] W. Chen, T.-Y. Liu, Y. Lan, Z.-M. Ma, and H. Li. Ranking measures and loss functions in learning to rank. *Advances in Neural Information Processing Systems*, 22, 2009.

[19] X. Chen, J. Shen, W. Xia, J. Jin, Y. Song, W. Zhang, W. Liu, M. Zhu, R. Tang, K. Dong, et al. Set-to-sequence ranking-based concept-aware learning path recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 5027–5035, Washionton, DC, USA, Feb 2023. AAAI Press.

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, Jun 2019. Association for Computational Linguistics.

[21] M. Di Bartolomeo and Y. Hu. There is more to streamgraphs than movies: Better aesthetics via ordering and lassoing. *Computer Graphics Forum*, 35(3):341–350, 2016.

[22] S. Di Bartolomeo, Y. Zhang, F. Sheng, and C. Dunne. SEQUENCE BRAIDING: Visual overviews of temporal event sequences and attributes. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1353–1363, 2021.

[23] L. Di Caro, V. Frias-Martinez, and E. Frias-Martinez. Analyzing the role of dimension arrangement for data visualization in Radviz. In *Advances in Knowledge Discovery and Data Mining*, pages 125–132, 2010.

[24] N. Djuric, M. Grbovic, and S. Vucetic. Convex kernelized sorting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 893–899, 2012.

[25] A. Dobler and M. Nöllenburg. On computing optimal linear diagrams. In *Proceedings of International Conference on Diagrammatic Representation and Inference*, pages 20–36, Rome, Italy, Sep 2022. Springer-Verlag.

[26] M. Dorigo and L. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[27] N. Dym, H. Maron, and Y. Lipman. DS++: a flexible, scalable and provably tight relaxation for matching problems. *ACM Trans. Graph.*, 36(6), nov 2017.

[28] Y. K. Emden R. Gansner and S. North. Graph drawing by stress majorization. In *Graph Drawing*, pages 239–250, Berlin, Heidelberg, 2005.

[29] S. Frey. Temporally dense exploration of moving and deforming shapes. *Computer Graphics Forum*, 40(1):7–21, 2021.

[30] S. Frey. Optimizing grid layouts for level-of-detail exploration of large data collections. In *Computer Graphics Forum*, volume 41, pages 247–258, 2022.

[31] O. Fried, S. DiVerdi, M. Halber, E. Sizikova, and A. Finkelstein. Isomatch: Creating informative grid layouts. In *Computer Graphics Forum*, volume 34, pages 155–166, 2015.

[32] J. Fuchs, A. Frings, M.-V. Heinle, D. A. Keim, and S. D. Bartolomeo. Quality metrics and reordering strategies for revealing patterns in biofabric visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 31(1):1039–1049, 2025.

[33] J. Fuchs, P. Isenberg, A. Bezerianos, F. Fischer, and E. Bertini. The influence of contour on similarity perception of star glyphs. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2251–2260, 2014.

[34] Y. L. Goh, Z. Cao, Y. Ma, Y. Dong, M. H. Dupty, and W. S. Lee. Hierarchical neural constructive solver for real-world TSP scenarios. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 884–895, Barcelona, Spain, Aug 2024. Association for Computing Machinery.

[35] J. Goldberger, G. E. Hinton, S. Roweis, and R. R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems*, Vancouver, Canada, Dec 2004. MIT Press.

[36] V. Granville, M. Krivanek, and J.-P. Rasson. Simulated annealing: a proof of convergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):652–656, 1994.

[37] S. Gratzl, A. Lex, N. Gehlenborg, H. Pfister, and M. Streit. LineUp: Visual analysis of multi-attribute rankings. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2277–2286, 2013.

[38] Q. Guo, G. Haotong, X. Wei, Y. Fu, Y. Yu, W. Zhang, and W. Ge. RankDNN: Learning to rank for few-shot learning. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 728–736, Washionton, DC, USA, Feb 2023. AAAI Press.

[39] X. Han, C. Zhang, W. Lin, M. Xu, B. Sheng, and T. Mei. Tree-based visualization and optimization for image collection. *IEEE Transactions on Cybernetics*, 46(6):1286–1300, 2016.

[40] N. Henry and J.-D. Fekete. MatrixExplorer: a dual-representation system to explore social networks. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):677–684, 2006.

[41] R. Hu, B. Chen, J. Xu, O. van Kaick, O. Deussen, and H. Huang. Shape-driven coordinate ordering for star glyph sets via reinforcement learning. *IEEE Transactions on Visualization and Computer Graphics*, 27(6):3034–3047, 2021.

[42] Z. Hu, Y. Dong, K. Wang, and Y. Sun. Heterogeneous graph transformer. In *Proceedings of the Web Conference*, pages 2704–2710, 2020.

[43] J. Johansson and C. Forsell. Evaluation of parallel coordinates: Overview, categorization and guidelines for future research. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):579–588, 2015.

[44] M. Jurewicz and L. Derczynski. Set-to-sequence methods in machine learning: a review. *Journal of Artificial Intelligence Research*, 71:885–924, 2021.

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

[46] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[47] W. Kool, H. van Hoof, and M. Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, New Orleans, USA, May 2019.

[48] O.-H. Kwon, C.-H. Kao, C.-H. Chen, and K.-L. Ma. A deep generative model for reordering adjacency matrices. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[49] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh. Set Transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753, California, USA, Jun 2019. PMLR.

[50] J. Li, H. Zeng, L. Peng, J. Zhu, and Z. Liu. Learning to rank method combining multi-head self-attention with conditional generative adversarial nets. *Array*, 15:100205, 2022.
[51] Y. Li, H. Xiong, L. Kong, Q. Wang, S. Wang, G. Chen, and D. Yin. S2phere: Semi-supervised pre-training for web search over heterogeneous learning to rank data. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4437–4448, California, USA, Mar 2023.
[52] I. Liiv. Seriation and matrix reordering methods: An historical overview. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 3(2):70–91, 2010.
[53] Q. Liu, Y. Ren, Z. Zhu, D. Li, X. Ma, and Q. Li. RankAxis: Towards a systematic combination of projection and ranking in multi-attribute data exploration. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):701–711, 2023.
[54] Z. Meng, S. Liang, H. Bao, and X. Zhang. Co-embedding attributed networks. In *Proceedings of ACM International Conference on Web Search and Data Mining*, pages 393–401, Melbourne, Australia, Feb 2019. ACM.
[55] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, Nevada, USA, Dec 2013. Curran Associates Inc.
[56] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.
[57] M. Miller, X. Zhang, J. Fuchs, and M. Blumenschein. Evaluating ordering strategies of star glyph axes. In *IEEE Visualization Conference*, pages 91–95. IEEE, 2019.
[58] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.
[59] P. A. Moran. Notes on continuous stochastic phenomena. *Biometrika*, 37(1/2):17–23, 1950.
[60] N. Morrical, S. Zellmann, A. Sahistan, P. Shriwise, and V. Pascucci. Attribute-aware RBFs: Interactive visualization of time series particle volumes using RT core range queries. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):1150–1160, 2024.
[61] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác. Reinforcement learning for solving the vehicle routing problem. In *Proceedings of Advances in Neural Information Processing Systems*, pages 9860–9870, Montréal, Canada, Dec 2018. Curran Associates, Inc.
[62] G. Nguyen and M. Worring. Interactive access to large image collections using similarity-based visualization. *Journal of Visual Languages & Computing*, 19(2):203–224, 2008.
[63] S. Pajer, M. Streit, T. Torsney-Weir, F. Spechtenhauser, T. Möller, and H. Piringer. WeightLifter: Visual weight space exploration for multi-criteria decision making. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):611–620, 2017.
[64] W. Peng, M. O. Ward, and E. A. Rundensteiner. Clutter reduction in multi-dimensional data visualization using dimension reordering. In *IEEE Symposium on Information Visualization*, pages 89–96. IEEE, 2004.
[65] P. Pobrotyn, T. Bartczak, M. Synowiec, R. Białobrzeski, and J. Bojar. Context-aware learning to rank with self-attention. *arXiv preprint arXiv:2005.10084*, 2020.
[66] M. Prates, P. H. Avelar, H. Lemos, L. C. Lamb, and M. Y. Vardi. Learning to solve NP-complete problems: A graph neural network for decision TSP. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4731–4738, Hawaii, USA, Jan 2019. AAAI Press.
[67] N. Quadrianto, L. Song, and A. Smola. Kernelized sorting. *Advances in Neural Information Processing Systems*, 21, 2008.
[68] A. Rahangdale and S. Raut. Deep neural network regularization for feature selection in learning-to-rank. *IEEE Access*, 7:53988–54006, 2019.
[69] A. Severyn and A. Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382, Santiago, Chile, Aug 2015. Association for Computing Machinery.
[70] D. Shi, X. Xu, F. Sun, Y. Shi, and N. Cao. Calliope: Automatic visual data story generation from a spreadsheet. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):453–463, 2021.
[71] H. Shrestha, K. Cachel, M. Alkhathlan, E. Rundensteiner, and L. Harrison. FairFuse: Interactive visual support for fair consensus ranking. In *2022 IEEE Visualization and Visual Analytics (VIS)*, pages 65–69, Oklahoma, USA, Oct 2022. IEEE.
[72] G. Strong, R. Jensen, M. Gong, and A. C. Elster. Organizing Visual Data in Structured Layout by Maximizing Similarity-Proximity Correlation. In *International Symposium on Visual Computing*, pages 703–713, 2013.
[73] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, pages 3104–3112, Montreal, Canada, Dec 2014. MIT Press.
[74] T. Tang, R. Li, X. Wu, S. Liu, J. Knittel, S. Koch, T. Ertl, L. Yu, P. Ren, and Y. Wu. PlotThread: Creating expressive storyline visualizations using reinforcement learning. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):294–303, 2021.
[75] A. Tyagi, T. Estro, G. Kuenning, E. Zadok, and K. Mueller. PC-Expo: A metrics-based interactive axes reordering method for parallel coordinate displays. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):712–722, 2023.
[76] N. van Beusekom, W. Meulemans, and B. Speckmann. Simultaneous matrix orderings for graph collections. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):1–10, 2022.
[77] L. van der Maaten. t-Distributed stochastic neighbor embedding. https://lvdmaaten.github.io/tsne/. Accessed: 2023-03.
[78] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, California, USA, Dec 2017. Curran Associates Inc.
[79] O. Vinyals, M. Fortunato, and N. Jaitly. Pointer networks. *Advances in Neural Information Processing Systems*, 28, 2015.
[80] E. Wall, S. Das, R. Chawla, B. Kalidindi, E. T. Brown, and A. Endert. Podium: Ranking data using mixed-initiative visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):288–297, 2018.
[81] B. Wang and D. Klabjan. An attention-based deep net for learning to rank. *arXiv preprint arXiv:1702.06106*, 2017.
[82] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu. Heterogeneous graph attention network. In *The World Wide Web Conference*, pages 2022–2032, 2019.
[83] J. Weissenböck, B. Fröhler, E. Gröller, J. Kastner, and C. Heinzl. Dynamic Volume Lines: Visual comparison of 3d volumes through space-filling curves. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):1040–1049, 2019.
[84] D. Weng, R. Chen, Z. Deng, F. Wu, J. Chen, and Y. Wu. SRVis: Towards better spatial integration in ranking visualization. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):459–469, 2019.
[85] D. Weyland. Simulated annealing, its parameter settings and the longest common subsequence problem. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, GECCO '08, page 803–810, New York, USA, 2008. Association for Computing Machinery.
[86] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinforcement Learning*, pages 5–32, 1992.
[87] XmdvTool. Xmdvtool home page. http://davis.wpi.edu/~xmdv/. Accessed: 2023-03.
[88] J. Zhang, J. Tao, J.-X. Wang, and C. Wang. SurfRiver: Flattening stream surfaces for comparative visualization. *IEEE Transactions on Visualization and Computer Graphics*, 27(6):2783–2795, 2021.
[89] L. Zhou, C. R. Johnson, and D. Weiskopf. Data-driven space-filling curves. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1591–1600, 2021.

Zehua Yu is a PH.D. student of computer science at Sun Yat-sen University. He received a MS degree in electric and communication engineering from Shantou University in 2022. His major research interest is scientific visualization, visual analytics, deep learning and time-series analysis.

Weihan Zhang is a PH.D. student of computer science at Sun Yat-sen University. He received a Bachelor's degree in software engineering from Chongqing University in 2022. His research interests include scientific visualization and visual analytics.

Sihan Pan is a master student at Sun Yat-sen University and National Supercomputer Center in Guangzhou. She obtained a Bachelor's degree in computer science and technology from Sun Yat-sen University in 2023. Her major research interest is scientific visualization and visual analytics.

Jun Tao is an associate professor of computer science at Sun Yat-sen University and National Supercomputer Center in Guangzhou. He received a Ph.D. degree in computer science from Michigan Technological University in 2015. His major research interest is scientific visualization, especially in applying information theory, deep learning, and optimization techniques to interactive flow visualization and multivariate data exploration.

APPENDIX

A. BACKGROUND OF ATTENTION AND OPTIMIZATION

Attention. As attention is heavily used in VON, we also provide a simplified notation in this paper. An attention module is parameterized by three learnable matrices: $\mathbf{W}_q$, $\mathbf{W}_k$, and $\mathbf{W}_v$. These three matrices are used to linearly transform a vector $\mathbf{h}_i$ into three vectors: namely, the query $\mathbf{q}_i = \mathbf{W}_q \cdot \mathbf{h}_i$, the key $\mathbf{k}_i = \mathbf{W}_k \cdot \mathbf{h}_i$, and the value $\mathbf{v}_i = \mathbf{W}_v \cdot \mathbf{h}_i$. The attention between two data points $\mathbf{h}_i$ and $\mathbf{h}_j$ is defined as:

$$\begin{aligned}\mathrm{att}(\mathbf{h}_i, \mathbf{h}_j) &= \mathbf{q}_i \cdot \mathbf{k}_j \\ &= (\mathbf{W}_q \mathbf{h}_i)^T \cdot (\mathbf{W}_k \mathbf{h}_j) \\ &= \mathbf{h}_i \cdot (\mathbf{W}_q^T \cdot \mathbf{W}_k) \cdot \mathbf{h}_j. \end{aligned} \tag{5}$$

This indicates that the attention between two nodes can be described by a single matrix $(\mathbf{W}_q^T \cdot \mathbf{W}_k)$ for transforming the product of $\mathbf{h}_i$ and $\mathbf{h}_j$. In this paper, we use $\mathrm{att}(\mathbf{h}_i, \mathbf{h}_j)$ to denote the attention module without introducing the matrices. Similarly, we denote the value vectors as functions of respective inputs, i.e., $\mathrm{val}(\mathbf{h}_i) = \mathbf{W}_i \cdot \mathbf{h}_i$. To emphasize that the attention modules are trainable and they may vary across layers, we use subscripts to denote different attention modules, e.g., att_k and val_k for the k-th attention head.

Multi-head attention (MHA). MHA uses multiple attention heads to capture information from different facets. Each attention head updates a vector $\mathbf{h}_i$ by collecting information from all other data points (i.e., $\mathbf{h}_j$), and the attentions from multiple heads are aggregated through summation:

$$\mathbf{h}_i^{l+1} = \sum_k \sum_j \mathrm{att}_k(\mathbf{h}_i^l, \mathbf{h}_j^l) \cdot \mathrm{val}_k(\mathbf{h}_j^l), \tag{6}$$

where l is the index of the attention layer and k is the index of the attention head. In this way, the resulting vector $\mathbf{h}_i'$ is able to sense the neighbor of the data point. By computing the pairwise attention for all points, this mechanism incorporates the interactions between points.

Quality metric and loss function. Formally, the quality metric for ordering can be any function that takes an order Π of a point set X as input, evaluates the quality of Π, and outputs a scalar value indicating the quality. VON uses the quality metric as the loss function $L(\Pi)$, so that the network learns to optimize the metric.

Greedy rollout strategy and optimization. Inspired by [86], we use the greedy rollout strategy in reinforcement learning to improve the ordering neural network over itself, leveraging information in the previous ordering samples. The term "greedy rollout" refers to the selection of the most promising actions. In our scenario, the greedy rollout strategy means that the network outputs the most likely data point at each step to form a sequence. The use of the greedy strategy allows VON to focus on learning the difference between the optimal solution and the greedy solution, which is more effective than learning from scratch.

Formally, the optimization minimizes the expectation of loss function $\mathbf{L_E}(\theta|X) = \mathbf{E}_{p_\theta(\Pi|X)}[L(\Pi)]$, where X is the point set to be ordered, Π is the order, and $L(\Pi)$ is the loss evaluated on the order Π. We used the best-observed strategy b as the baseline in the gradient estimator:

$$\nabla \mathbf{L_E}(\theta|X) = \mathbf{E}_{p_\theta(\Pi|X)}[(L(\Pi) - L(b(X)))\nabla log p_\theta(\Pi|X)], \tag{7}$$

where $L(b(X))$ is the loss evaluated on the order produced by the baseline b. Note that baseline b is the model with the best performance, and it will be regularly updated during the training process. Similar to DQN [58], we stabilize the baseline by freezing its parameters p_θ for a fixed number of every epoch. By using the best-observed model as the baseline, we consistently challenge the model against the top-performing model in each epoch. We use Adam [45] as the optimizer for parameter updates.

B. DETAIL OF EXPERIMENTAL SETTING

Data sets. We use six data sets including four data sets with data points in respective latent embedding spaces (i.e., FashionMNIST, CIFAR-10, ImageNet, CORA, and DBLP), two dynamic graph datasets (i.e., FLT and SCH), and four high dimensional datasets for axes ordering (i.e., Cars, AAUP, Coal Disaster and Census Income):

- *FashionMNIST* is a black and white clothing dataset containing images in ten categories. Each image is 28 $\times$ 28 pixels in size.
- *CIFAR-10* is a universal object dataset with ten categories, where the image size is 32 $\times$ 32 pixels.
- *ImageNet* is a large high-definition image dataset. We use dog images with 120 breeds. While the original images are different in size, we reduce their sizes to 64 $\times$ 64 pixels.
- *CORA* is a scholarly dataset with machine learning papers. It covers 2708 papers in seven categories.
- *DBLP* is a dataset from the DBLP public bibliography. We use a coauthor network extracted by Meng et al. [54] from the publications in 172 computer science conferences [17].
- *FLT* is the "flashtap" data that was used for MultiPiles [5]. It represents a functional brain connectivity network in a Parkinson's disease study.
- *SCH* is a dynamic graph dataset, where the graphs encode social interaction between children and teachers at a primary school.
- *MNIST* is a large collection of handwritten digits.
- *Cars* is a dataset of car models, comprising seven attributes (e.g., years, makes, and types) for each of the total 398 car models.
- *AAUP* is the faculty salary data collected from the March-April 1994 issue of Academe [64], [87].
- *Coal Disaster* records the coal-mining disasters between March 1851 and March 1962 [64], [87].
- *Census Income* is part of the U.S. Census Bureau database with 42 demographic and employment-related attributes such as age, education, occupation, and marital status [64], [87].

Baselines. We compare three variations of VON against four generic approaches (i.e., AM [47], SA [36], NN [35]), one layout algorithm (i.e., SM [28]), four specialized algorithms

(i.e., C-LO-δ_I, U-LO-δ_I, C-BC and U-BC) for matrix reordering [8], Random Swapping for axes reordering, and four specialized methods (i.e., LAS/FLAS [7], Isomatch [31] and Kernelized Sorting [67]) for images reordering as baselines:

- *Attention model (AM)* [47] is an attention-based model designed for solving routing problems.
- *Simulated annealing (SA)* [36] is a randomized search algorithm that may accept worse solutions during the search process in order to escape local optima and find the global optimum.
- *Nearest neighbor (NN)* [35] is a distance-based classification algorithm.
- *Stress majorization (SM)* [28] is a graph layout algorithm that preserves the target distance between data points in lower dimension. For ordering purpose, we use this algorithm to embed the data points in 1D.
- *C-LO-δ_I, U-LO-δ_I, C-BC, and U-BC* [76] are four specifically designed algorithms for matrix reordering based on the Moran's I.
- *Random Swapping* is a randomized algorithm that optimizes the order from an initial configuration by randomly swapping two data points. It can be used to solve axes reordering [64].
- *IsoMatch* [31] is a method to arrange collections of objects by minimizing an energy value, which is designed to preserve a given distance measure.
- *Kernelized Sorting* [67] is an approach that performs matching by requiring a similarity measure only within each class. It achieves this by maximizing the dependency between matched pairs of observations using the Hilbert-Schmidt Independence Criterion.
- *LAS/FLAS* [7] designed based on Linear Assignment Sorting (LAS). LAS combines ideas of self-organizing map and the swap sorting method to optimally swap all vectors simultaneously. FLAS handles larger quantities of images by replacing the global assignment with multiple local swaps.

For the **the three generic solvers**, the following paragraphs provide further details:

AM uses attention mechanisms and reinforcement learning for training. AM formulates a problem instance s as a graph with n nodes, where each node $i \in \{1, \ldots, n\}$ is represented by features x_i. For TSP, x_i is the coordinate of the node i, and the graph is fully connected (including self-connections). Generally, the model can be considered a Graph Attention Network [47], incorporating graph structure through a masking procedure. A solution (order) $\pi = (\pi_1, \ldots, \pi_n)$ is defined as a permutation of the nodes, where $\pi_t \in \{1, \ldots, n\}$ and $\pi_t \neq \pi_{t'}$ for $t \neq t'$. The attention-based encoder-decoder model defines a stochastic policy $p(\pi|s)$ for selecting a solution π given a problem instance s, factorized and parameterized by θ. The encoder generates embeddings for all input nodes, while the decoder produces the sequence π one node at a time, using the embeddings, a mask, and context. For ordering, after a partial order is fixed, the task is to find a sequence from the last node, through unvisited nodes, back to the first. The decoder context includes embeddings of the first and last node, and a mask indicates visited nodes.

SA starts with an initial solution and generates a neighboring solution at each step, calculating its energy difference. If a new solution is better (i.e., has lower energy), the solution will be accepted; otherwise, the solution will only be accepted with a certain probability determined by the current "temperature" and energy difference, following [85]: $P = \exp\left(-\frac{\Delta E}{T}\right)$, where P represents the acceptance probability, ΔE is the cost difference between the new solution and the current solution, T is the current temperature. The probability of accepting worse solutions allows the algorithm to escape local optima, while the decreasing temperature is designed for convergence. In the ordering problem, SA explores the solution space by randomly swapping elements and optimizes the ordering objective function. Behrisch et al. [8] reported that Brusco et al. [13] could find the optimal solution for a 35×35 connection in 20 to 40 minutes with the right settings.

NN categorizes new instances based on the majority class of their nearest neighbors. The nearest neighbor heuristic is a path-based approach that starts with a randomly selected single node and gradually adds the closest node to form a path, ultimately connecting the start and end nodes to form a tour. The algorithm always begins with the first node in the input to ensure deterministic results. In ordering tasks, NN builds a relatively ordered arrangement by minimizing the distance between adjacent elements.

For **1D layout**, *SM* is an optimization algorithm used for multidimensional scaling (MDS), aiming to optimize the layout of points by minimizing a stress function. In ordering, it projects points onto a one-dimensional space so that the one-dimensional distances after projection are as consistent as possible with the original high-dimensional distances.

For **matrix reordering**, we use a union approach [8] combines all graphs in G into a single graph H on the same vertex set V, where each edge weight reflects how often it appears across the graphs in G. This is equivalent to summing all 0-1 adjacency matrices. Any ordering algorithm can then be applied to H, with the resulting order used for all graphs in G. For a vertex v, $N(H, v) = \sum_{G \in G} N(G, v)$ is a vector of length n, where each entry represents the edge's weight, 0 to k. We further elaborate on the baselines for matrix reordering:

U-LO-δ_I works on weighted graphs, requiring a distance measure δ, with MultiPiles using $\delta(u, v) = L2(N(H, u), N(H, v))$, the Euclidean distance between neighborhoods. While squaring does not affect distance comparisons, it may change the final ordering due to how distances are summed.

U-BC, usually used in unweighted settings, extends to weighted graphs by multiplying the weights of intersecting edges and summing them across all pairs.

C-LO-δ_I focuses on pairwise distances, ensuring $\delta(u, v)$ is low when u and v have similar neighborhoods across many graphs. Unlike the union approach, which aggregates neighborhoods first, the collection-aware method calculates $\delta(u, v) = \sum_{G \in G} L2(N(G, u), N(G, v))$. Alternative distance measures can replace $L2$, and squaring distances alters the clustering step.

C-BC revert to counting crossings, but separately for each graph in G, and then sum the results. In a collection-aware

method, the target rank of a vertex is determined for each graph individually, and then aggregated. It computes the median rank of each vertex's neighbors in each graph, then computes the overall median of these ranks. If a vertex has no neighbors in a graph, it's excluded. Ordering is done based on these medians of medians.

For **axes reordering**, *Random swapping* is a simple optimization technique used to explore solution spaces by randomly selecting two elements from a given solution and swapping their positions. This method begins with an initial solution, which may be generated randomly or based on a heuristic. The swap operation is applied iteratively, where two elements are randomly chosen and exchanged, leading to a new solution. After each swap, the solution is evaluated to determine if it improves the objective function, such as minimizing distance or maximizing a score. The new solution may be accepted based on improvement or according to a probabilistic rule, depending on the specific algorithm. While random swapping is easy to implement and encourages exploration of the solution space, it may not always guarantee efficiency, especially in large spaces, as purely random swaps do not always lead to significant improvements.

The **four specialized algorithms for image ordering** are explained in detail as follows: *IsoMatch* aligns images into a specified spatial arrangement based on a given distance matrix. The inputs include a set of images I and a symmetric pairwise distance matrix D, where d_{ij} represents the distance between images I_i and I_j. IsoMatch first uses nonlinear dimensionality reduction to project the images into a 2D space, producing an initial layout. This layout is then transformed to roughly fit the desired output pattern. A bipartite graph is constructed, and a bipartite matching algorithm minimizes the movement of images to their final positions, resulting in a distance-preserving output arrangement.

KS aims to maximize the dependence between two sets of variables X and Y by permuting Y to align with X, using the Hilbert-Schmidt Independence Criterion (HSIC) as the dependence measure. HSIC has several advantages: it is robust to small changes, computationally efficient since it only requires kernel matrices, and allows flexibility in kernel selection, enabling the incorporation of prior knowledge. The sorting problem is formulated as an optimization, which maximizes the trace of the product of kernel matrices for X and the permuted Y: $\pi^* = \arg\max_{\pi \in \Pi_m} \text{tr}\left(K\pi^T L\pi\right)$ This optimization is a special case of the quadratic assignment problem and is NP-hard. However, it is proven that, for simple cases such as scalar random variables with linear kernels, the optimal permutation of Y is equivalent to sorting Y in the same order as X. Specifically, for $X = Y = \mathbb{R}$, with kernels $k(x, x') = xx'$ and $l(y, y') = yy'$, sorting Y in ascending order or its reverse gives the optimal solution. Kernelized sorting generalizes the idea of sorting, using kernel functions to capture relationships between X and Y indirectly through their feature spaces rather than their direct values.

LAS/FLAS are two linear assignment sorting algorithms. LAS combines the ideas of Self-Organizing Maps (SOM) and Swap Sorting (SSM) to optimally swap all vectors simultaneously for image sorting. Initially, input vectors are randomly placed on a map, which is then spatially low-pass filtered to create a smoothed version representing neighborhoods. The input vectors are subsequently assigned to their best matching positions on the map. While LAS provides high sorting quality, it becomes computationally expensive for large image sets. To address this, Fast Linear Assignment Sorting (FLAS) modifies LAS by replacing global assignments with multiple local swaps, significantly improving speed while maintaining sorting quality for large datasets.

Parameter settings. In all experiments, we fix the parameters of VON and the baselines, aiming to examine the method's usability without extensive tuning. For the specific parameter settings of VON and various baselines, please refer to our code. For the baselines, we report their settings as follows:

For the simulated annealing, the temperature setting significantly affects the final results. We follow the guidance of Dennis et al. [85], which suggests a cooling rate between 0.8 and 0.99, with no specific requirements for restarts, temperature length, or other parameters. Based on this, we set the cooling rate to 0.9, the temperature length to 100n, the number of restarts to 5, and the stopping temperature to 0.1. For the matrix reordering, where SA delivers the worst performance, we include an additional experiment to examine the impact of parameter tuning, as discussed in Appendix C.IV.2.

For the image ordering methods, both IsoMatch and KS require a target grid to which the images are mapped. Because image ordering can be considered as mapping the images to a 1D grid, we use a target grid of $1 \times m$ for these two methods. For the comparison with LAS/FLAS, the methods require the number of images to be a perfect square. Thus, for 50 or 150 input images, only the first 49 and 144 images are ordered, respectively. VON-m uses the same subset for a fair comparison. The evaluation focuses on VON-m's objective functions rather than direct performance comparisons with the baselines, so the slight differences in image numbers do not affect VON-m's assessment. Default parameter settings from each method's code are used for all experiments, and VON-m's parameters remain unchanged for consistency. For LAS/FLAS, the default parameters are LAS (radius factor=0.85) and FLAS (nc=49, radius factor=0.7). In the experimental setup, we maintain the default parameter settings for these methods as provided in their code.

Quality Metrics. We study two quality metrics (i.e., TSP and stress) for 1D layout, four metrics (i.e., Moran's I, LA, PR, and BW) for matrix reordering, two metrics (i.e., Symmetry and Correlation) for axes reordering, and three metrics (i.e., DPQ, Energy Value, and KS objective function) for images reordering:

- *TSP* distance is the total path length in the traveling salesman problem (TSP). In terms of visualization, minimizing TSP distance will place similar items closer and enhance perception. For example, TSP is used in ordering rows and columns in a matrix [8], [40].
- *Stress* is the loss in stress majorization, a widely used algorithm for graph layout [28]. Stress majorization preserves the target distance in low dimension by

minimizing the stress $\sum w_{ij}(|x_i - x_j| - d_{ij})^2$. While stress majorization is commonly used for 2D layout, it is adopted to arrange river branches in 1D [88]. Stress majorization brings similar items closer as well.

- *Moran's I* is a statistical measure used to determine spatial autocorrelation in a dataset [59]. It measures the similarity between the values of a variable and neighboring variables. This metric was used in matrix reordering [76].
- *Linear arrangement (LA)* is the sum of the distances between the vertices of the edges of a graph (i.e., $LA(\phi, G) = \sum_{(u,v)\in E} \lambda((u, v), \phi, G))$ [8].
- *Profile (PR)* is the sum, for each column i of the matrix, which can be intuitively understood as the "raggedness": the distance from the diagonal (with coordinates (i, i)) to the farthest-away non-zero cell for that column (with coordinates (i, j)) [8].
- *Bandwidth (BW)* is the maximum distance between two vertices given an order ϕ [8].
- *Symmetry* is the balanced and proportionate similarity found in two halves of an object, mirroring each other across a central axis or point. It can be used in axes reordering for star plots [57].
- *Correlation* is a statistical measure that evaluates to what extent two or more variables fluctuate together. It can be used in axes reordering [57].
- *Energy value* is a general objective function to evaluate how the solutions preserve pairwise distance in IsoMatch [31].
- *KS objective function* measures the similarity between two sets of images in high-dimensional spaces under a certain arrangement using the matrix trace, which guides the optimization process to find the optimal arrangement. It calculates the kernel matrix of the arranged images and grid data, performs a de-centering operation, and finally uses a locally optimal smoothing formula as the cost [67].
- *Distance Preservation Quality (DPQ)* is the ratio of the p-norms of the distance preservation gains of the actual arrangement to a perfect arrangement. Larger DPQ indicates better quality [7].

Embedding methods. We study five embedding approaches:

- *Principal component analysis (PCA)* is a linear dimensionality reduction technique that transforms high-dimensional data into a lower-dimensional representation while preserving the differences along axes exhibiting the largest variations.
- *Truncated singular value decomposition (T-SVD)* is a dimensionality reduction method that works well with sparse data by selecting a subset of singular values and their components.
- *Autoencoder (AE)* is a neural network-based technique for unsupervised dimensionality reduction and feature learning.
- *Isomap* is a nonlinear dimensionality reduction technique that focuses on preserving the pairwise geodesic distances (shortest path distances) between data points in a lower-dimensional space.
- *Locally linear embedding (LLE)* is another nonlinear dimensionality reduction method that seeks to preserve local relationships between data points.

Sampling sizes. We decide the sampling sizes by referencing similar scenarios. For example, in for visualizing image collections, Nguyen et al. [62] and Han et al. [39] evaluated the visualization with 25, 50, 75, 100, and 200 images, with 100 identified as an effective choice for balancing usability and clarity. Similarly, image search engines like Google Images typically display around 50 images per page on a 27-inch 4K screen at default zoom settings. In visualizing graph clusters, graphs with 50 and 100 nodes were used to demonstrate correlations between nodes in matrix visualizations [2]. For axis reordering tasks, examples commonly feature numbers of axes ranging from 5 to 14 [43], [57]. While a famous early study [56] suggested that the cognitive load of objects that humans could perceive simultaneously was about seven, with two floating up and down. A larger number of data items may be used as an overview before detailed investigation. Based on these references, three sampling sizes: 50, 100, and 150 are considered for each sampling strategy.

C. Additional Results

C.I Ablation Study

C.I.1 Impact of VON variants and sampling strategies

Further investigation of training data sizes and scales. We further examine the performance using training samples of varying point set sizes, as shown in Table 2.II. We still find that the model performs best when the size and scale of training point sets match those of testing sets. This further confirms our hypothesis that data of different scales exhibit different structures. For optimal performance, we suggest training the model on samples of equivalent scales. However, in general, we find that VON-m may better transfer knowledge across sample sizes and scales, as shown in Table 2.I.

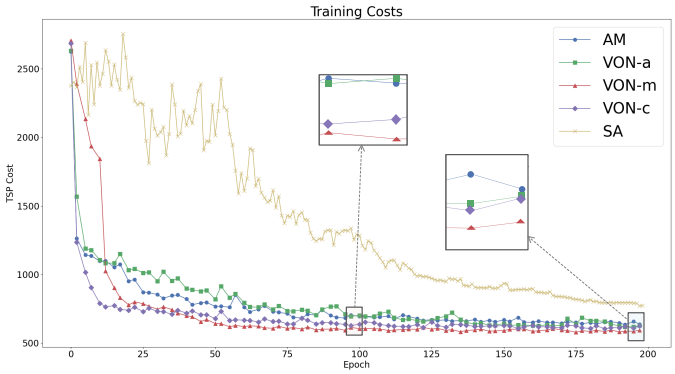


Fig. 1. The training loss over 200 epochs for ordering images in Fashion-MNIST dataset using TSP metric. All models are trained using 50mix.

Impact of mixed sampling strategies. Prior experiments reveal the impact of training data using individual sampling strategies. Here, we further study three mixed strategies

(namely, mix, gl, and lg), to examine the impact of how the global and local samples are ordered in the training data. As shown in Table 2.III, the three models of VON-m trained with different data share similar performance, demonstrating a superior ability to learn across scales. In contrast, the performance of AM-based models heavily relies on the order of training samples. The models seem to be shaped by the first half of training samples. For example, while delivering reasonable performance in local data, AM(50lg) results in large distances in global data, such as 50g (2150.1, $\times 3.10$ as VON-m(50lg)) and 100g (2373.9, $\times 2.20$ as VON-m(50lg)). Mixing the training samples across scales seems to achieve a balance: it may not lead to the best results, but it often delivers decent results and avoids the worst. Therefore, *we will examine the performance using the mix of the four sampling strategies in the remaining experiments.*

Performance over training epochs. We study the learning efficiency by investigating how the loss changes over training epochs for each model. In Fig. 1, while VON-c and AM have a rapid downward trend in the early stages of training, VON-m shows the lowest loss after 100 epochs. The loss of SA fluctuates in the first 50 epochs and gradually decreases after that. Over the entire 200 epochs, the loss of SA is much higher than the learning-based techniques. The learning-based techniques seem to discover a reasonable ordering strategy by 100 epochs.

C.I.2 Impact of individual modules

In this section, we study the individual impact of core modules using the best variant (i.e., VON-m) and the most balanced sampling strategy (i.e., mix). Specifically, we study the *encoder* to evaluate the benefit of transforming the points to latent space for ordering, and study the *reposition* module as it is the core contribution to our decoder design. We compare the original model VON-m with three variants (namely, VON-m$_{\bar{e}}$ without the encoder, VON-m$_{\bar{r}}$ without repositioning, and VON-m$_{\bar{b}}$ without both) using 1D layout and matrix reordering.

Ablation study using 1D layout. We use the Fashion-MNIST dataset and TSP distance for this study. Table 1 shows VON-m$_{\bar{b}}$ (without both encoder and repositioning) performs significantly worse in all tests compared to other models. The variants VON-m$_{\bar{e}}$ and VON-m$_{\bar{r}}$ are consistently outperformed by the original VON-m, highlighting the critical roles of these modules. However, their individual impact seems to be similar, with VON-m$_{\bar{e}}$ excelling in 50l, 100g, and 100l, while lagging in 50g, 150g, and 150l. Notably, VON-m$_{\bar{r}}$ (without repositioning) closely matches VON-m in 50g (675.9 vs 609.5) but performs much worse in 50l (456.4 vs 199.6). This underscores the loss of transferability across distributions without the repositioning module, especially given the varied local sample distributions.

Ablation study using matrix reordering. As shown in Table 2, the original VON-m models outperform the three variants for all three metrics, affirming the significance of both modules. However, the reposition module is less important in this task, as matrix reordering considers all data items and becomes less demanding in transferability. VON-m$_{\bar{r}}$, with the encoder but lacking the repositioning, surpasses the other two variants in every metric. Among the other variants, VON-m$_{\bar{e}}$ slightly outperforms VON-m$_{\bar{b}}$, though the difference is less pronounced. This suggests that while the repositioning module may partially offset the absence of the encoder, it cannot fully replace it.

TABLE 1
Ablation study on the performance in 1D layout using TSP distance and Fashion-MNIST dataset. VON-m$_{\bar{e}}$, VON-m$_{\bar{r}}$ and VON-m$_{\bar{b}}$ are the variants of VON-m without the encoder, repositioning, and both, respectively. All models are trained with data samples of "50mix".

	50g	50l	100g	100l	150g	150l
VON-m	**609.5**	**199.6**	**385.3**	**379.2**	**540.4**	**590.9**
VON-m$_{\bar{e}}$	1381.3	295.9	623.3	743.2	1046.9	1304.0
VON-m$_{\bar{r}}$	676.9	456.4	849.7	912.6	985.1	1054.3
VON-m$_{\bar{b}}$	2925.1	682.7	2797.1	1857.8	4456.7	4515.9

Findings. In general, the ablation study shows that the original module VON-m always performs the best in every case, and the encoder has a significant impact in every setting. The study also confirms the impact of the repositioning module, when the transferability is required to order data points across distribution (i.e., the 50l, 100g, 100l, 150g, and 150l settings for 1D layout). However, when the data follows the same distribution (i.e., matrix reordering and the 50g setting in 1D layout), the repositioning module only has a slight marginal effect.

TABLE 2
Ablation study on the performance in matrix reordering using SCH dataset and three metrics (i.e., LA, PR and BW). VON-m$_{\bar{e}}$, VON-m$_{\bar{r}}$ and VON-m$_{\bar{b}}$ are the variants of VON-m without encoder, repositioning, and both, respectively.

	LA	PR	BW
VON-m	**51.77k**	**11.60k**	**206.88**
VON-m$_{\bar{e}}$	67.66k	12.94k	221.94
VON-m$_{\bar{r}}$	56.72k	12.35k	214.41
VON-m$_{\bar{b}}$	68.18k	13.29k	224.65

C.II Dynamic Scenario

Demo. We develop a demo system to explore the CIFAR-10 image collection. Fig. 2 (a) shows the exploration results using Moran's I as the quality metric. We select a region containing four categories of images, indicated by the light blue, blue, green, and red dots, respectively, as shown in (a.I). Using Moran's I, the ordered images in (a.II) and the axis in (a.III) are correlated to their categories, with most dog images (light blue) on the left and the other three categories on the right. Users may further select images along the axis in (a.III) to investigate details. In Fig. 2 (b) and (c), we switch the quality metric to TSP and stress, respectively. The order of images is still similar. Although several planes appear between the dog pictures, the overall appearance of the images is still similar. We should note that Moran's I seem to better separate the images of different categories. This may be due to the computation of Moran's I considers all pairwise relationships among images, instead of relationships between neighbors as in TSP. Please refer to the supplemental video for a better viewing experience of this usage case.

Performance in dynamic scenario for individual datasets. Fig. 4 shows the complete results for individual

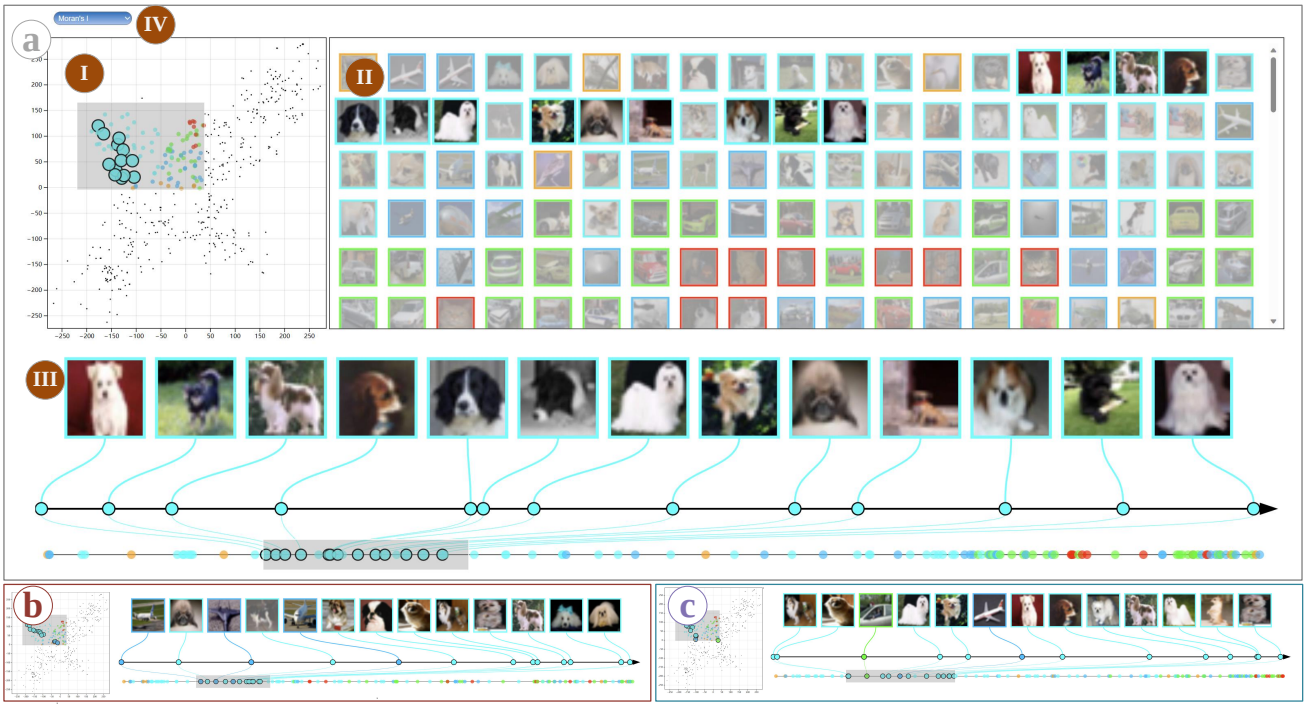


Fig. 2. Exploring CIFAR-10 dataset using VON. The demo interface consists of: (I) a scatter plot for selecting images, (II) the collection of selected images ordered by VON, (III) a detailed view for exploring selected images along an axis, and (IV) a drop-down list for selecting quality metrics. The color of a dot indicates the category of the corresponding image. (a), (b), and (c) show the ordering results using Moran's I, TSP, and stress majorization, respectively.

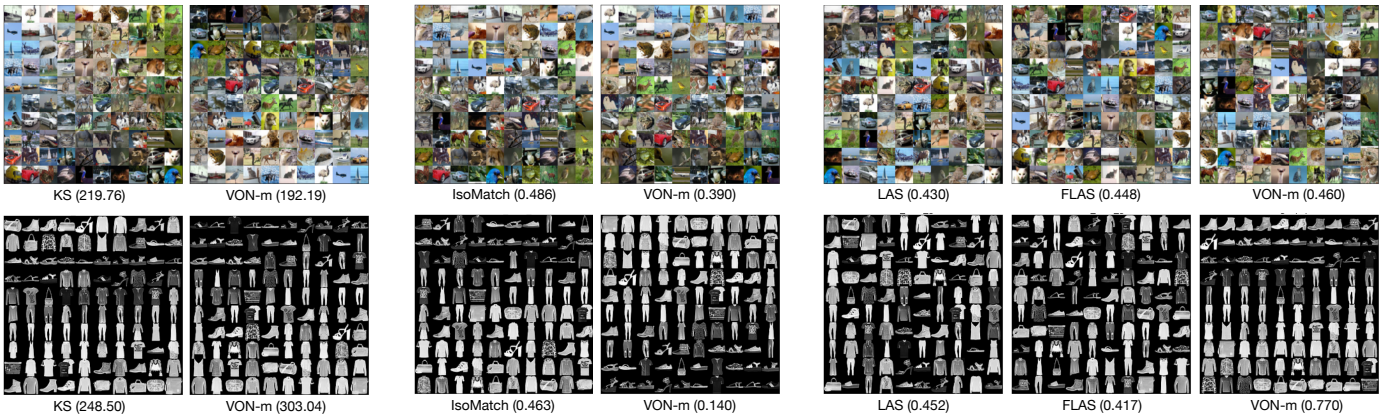
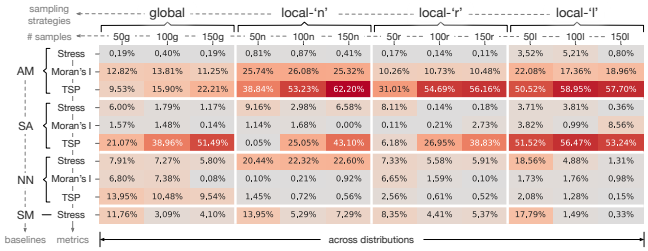


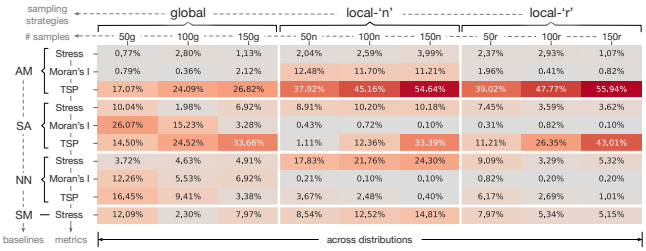
Fig. 3. Comparison of image ordering results using the respective quality metrics of specialized baselines. The first row shows the results of CIFAR-10 dataset and the second row shows the results of Fashion-MNIST. Each label shows the corresponding ordering approach and evaluated metric value. Note that the metric depends on the baseline being compared, and the images are ordered in 1D and displayed in the squared grid from left to right and top to bottom.

datasets for dynamic scenarios. Please refer to Section 4.3.1 for the analysis of the average results over these datasets. Compared to the average improvement in Fig. 5, the improvement patterns for the individual datasets are similar in general. For example, similar to the average, we find significant improvement of VON-m over AM and SA in TSP for each dataset, especially using local sampling strategies. We also find that VON-m consistently outperforms the baselines, with a single negative improvement (-0.29%).

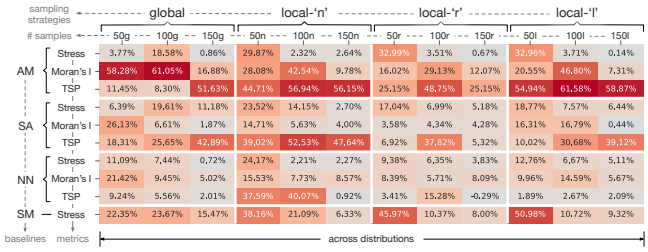
However, differences also exist. For example, in CIFAR-10 and DBLP, VON-m shows greater performance improvement over NN in Moran's I for the global samples, but less improvement for the local samples, when comparing to the average. Especially for the 50g samples, the improvement over NN reaches the maximum (51.17%) in all cases. In the Fashion-MNIST and MNIST datasets, VON-m shows more improvement over the baselines, especially for AM and SA. However, unlike the average, the performance improvement often becomes most significant using the medium number of samples (100). Interestingly, we find that the improvement over NN in TSP is much larger than average in 50n (37.5% vs. 7.91%) and 100n (40.07% vs. 8.77%). This is unexpected, because NN usually excels in TSP and even achieves better performance than VON-m with the only negative improvement (-0.29%) in TSP with 150r samples of the same dataset. In the ImageNet and CORA datasets, different from the



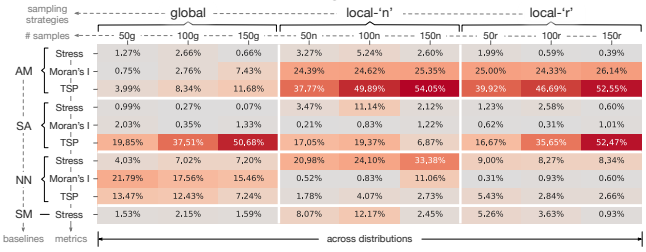
(a) MNIST



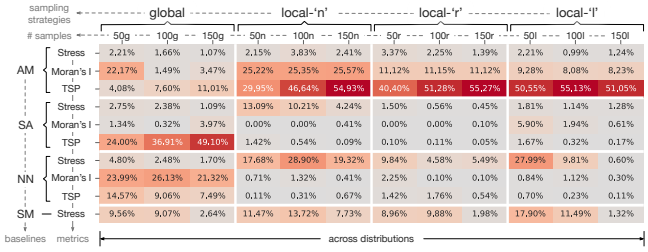
(b) ImageNet



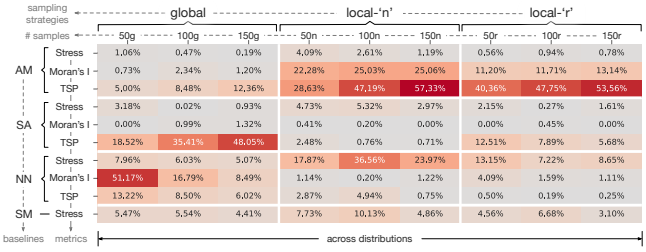
(c) Fashion-MNIST



(d) CORA

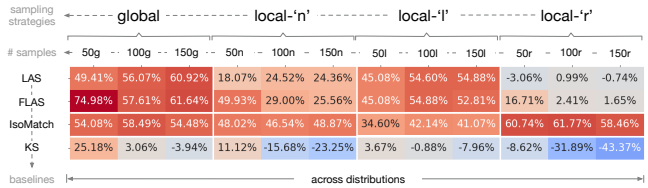


(e) CIFAR-10

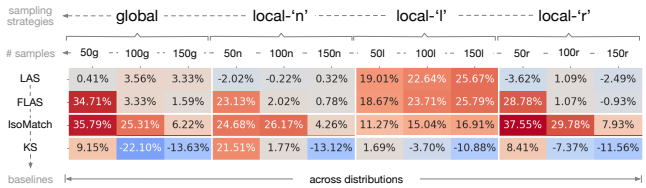


(f) DBLP

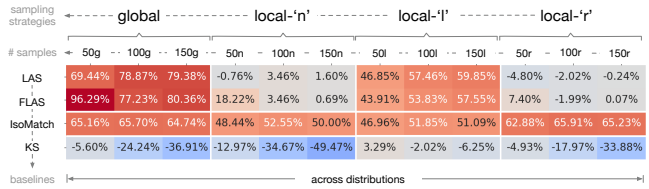
Fig. 4. Performance improvement of VON over AM, SA, NN, and SM for three metrics, using (a) MNIST, (b) ImageNet, (c) Fashion-MNIST, (d) CORA, (e) CIFAR-10, and (f) DBLP datasets.



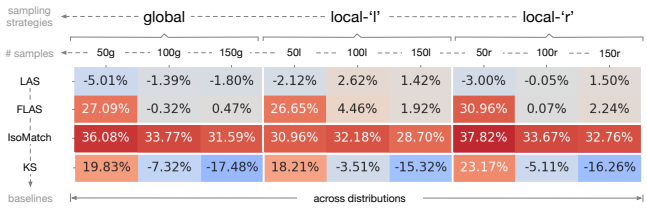
(a) MNIST



(b) CIFAR-10



(c) Fashion-MNIST



(d) ImageNet

Fig. 5. Performance improvement of VON over LAS, FLAS, IsoMatch, and KS for their respective metrics, using (a) MNIST, (b) CIFAR-10, (c) Fashion-MNIST, (d) ImageNet datasets.

average, VON-m shows a greater advantage over NN in stress, especially for the local-'n' samples.

For image reordering, we compare VON-m against IsoMatch [31] using energy value, Kernelized Sorting (KS) [67] using its objective, and LAS/FLAS [7] using DPQ. Fig. 5 presents the detailed performance improvements across different datasets. It shows that VON-m generally outperforms FLAS and IsoMatch on all image datasets, especially when the testing size matches the size (50) used in training. However, we also find gaps between VON-m and LAS or KS, particularly for KS when the size of testing sets deviates from the training data. This indicates that the ordering strategy learned by VON-m becomes less effective when applied to larger collections of data. We also examine the results qualitatively, by showing the ordering results of two randomly selected collections of 100 images in Fig. 3.

In all the cases, although the quantitative difference may be marginal to distinguish perceptually, VON-m produces comparable results to the respective baselines in general. This confirms to some degree that VON-m can be used in image ordering tasks.

To conclude, although the overall patterns are similar, the improvement percentage in a specific case may significantly fluctuate, leading to a large improvement value. We should note that although VON-m may not outperform the specialized methods in every case, it is generally usable, especially when a specialized algorithm is not available. This renders VON-m a reliable choice to avoid the worst cases.

Timing performance. We use the Fashion-MNIST dataset to compare the timing performance of different approaches for ordering 50 images, as shown in Table 3. The timing performance of the two learning-based approaches (VON-m and AM) are similar as they share similar sizes of networks. For these approaches, the training time varies across metrics. The reason is that the evaluation time across metrics is different, and these approaches need to evaluate the metrics during training. Their ordering time is stable and interactive, because once the metric is learned by the network, these approaches do not rely on the evaluation for ordering. In contrast, SA needs to evaluate the metrics during the ordering stage, leading to significantly different ordering time, which can be prohibitive. NN uses a fixed heuristic for ordering, achieving similar ordering performance as VON-m and AM.

TABLE 3
The timing results for dynamic scenario using Fashion-MNIST datasets. For the learning-based approaches, both training and ordering time are reported.

		VON-m	AM	SA	NN	SM
TSP	Train	2s/e	2s/e	-	-	-
	Order	0.82s	0.81s	2.23s	0.86s	-
Stress	Train	240s/e	240s/e	-	-	-
	Order	0.82s	0.81s	932.59s	0.86s	20.93s
Moran's I	Train	11s/e	11s/e	-	-	-
	Order	0.82s	0.81s	619.52s	0.86s	-

C.III Star plot

Visualization results for the star plot. Fig. 6 displays the axes reordering results reordering results for the star plot using the Cars and Coal Disaster datasets. The symmetry metric is used following Section 4.4.1. A smaller symmetry value indicates a better order of axes. In Coal Disaster with only five axes, all approaches identify the optimal solution. However, in the visualization results, we find that the orders produced by different approaches are visually distinct. In Cars with seven axes, a higher complexity can be observed in the metric, as VON-m and RS find the numerically optimal solution while AM does not. However, in the visualization results, we find that VON-m seems to learn a different strategy from AM and RS. VON-m tends to place attributes with high and low values alternately, resulting in 'star'-like shapes. In contrast, AM and RS place attributes with similar values contiguously, leading to 'fan'-like shapes. This example demonstrates a gap between the metric and human perception: while VON-m and RS share similar metric values, the visualization results of AM and RS seem to be visually closer. As our approach focuses on optimizing for metrics, we will not further explore this aspect. However, this example highlights the necessity of developing metrics that are better aligned with human perceptions. *Our VON can facilitate this kind of research by providing a reliable and readily available tool for optimizing the customized metrics, so that perception researchers can focus on the design of metrics and visually assess whether the optimized results are consistent with human perception.*

Timing performance. We use the Census Income dataset, which has the most number of axes, to examine the timing performance. The computation time for one epoch is around half a second using each approach: it takes 0.61s for VON-m, 0.50s for AM, 0.36s for SA, and 0.36s for RS. Note that learning-based approaches (VON-m and AM) are slower, because they require additional representation time beyond metric evaluation. This may damage the performance in static scenario, but it will be worth the time in dynamic scenario, as the network will collect useful information for ordering efficiently.

C.IV Matrix reordering

C.IV.1 Impact of embedding approaches

The embedding approach is used to generate the initial point coordinates for graphs without node coordinates. In this experiment, we study its impact by embedding the adjacency matrix of the SCH dataset for ordering.

t-SNE and 2-dimensional embedding. We only use t-SNE for 2D embedding, as t-SNE [77] recommends an additional dimension reduction approach for high dimensional data (e.g., PCA for dense data and T-SVD for sparse data) before applying t-SNE. We compare t-SNE with "PCA-8D to t-SNE" (using PCA to reduce data in 8-dimension and then applying t-SNE) and "T-SVD-8D to t-SNE" (using T-SVD before t-SNE). The results are shown in Table 5. PCA and T-SVD may help to reduce noise in the high dimensional data, which leads to better ordering results than the original t-SNE in all three metrics.

Impact of embedding approaches and numbers of dimensions. We further study other embedding approaches using various numbers of dimensions. Surprisingly, we find that the two linear dimension reduction approaches (i.e., PCA and T-SVD) perform exceptionally well. These two approaches share similar performance to each other and outperform other competitors in almost every setting. For example, for LA with 16D embedding, T-SVD (43.97k) and PCA (44.23k) perform similarly, while all the other approaches have LA values larger than 64k. In terms of the number of dimensions, we do not find a clear clue to favor a smaller or a larger number of dimensions. In general, we will follow the suggestion from the developer of t-SNE to use PCA for dense data and T-SVD for sparse data, and we will suggest using 2D embedding space for efficiency.

C.IV.2 Additional results

Timing performance. The timing results for matrix reordering are shown in Table 6. VON-a and VON-m cost similar

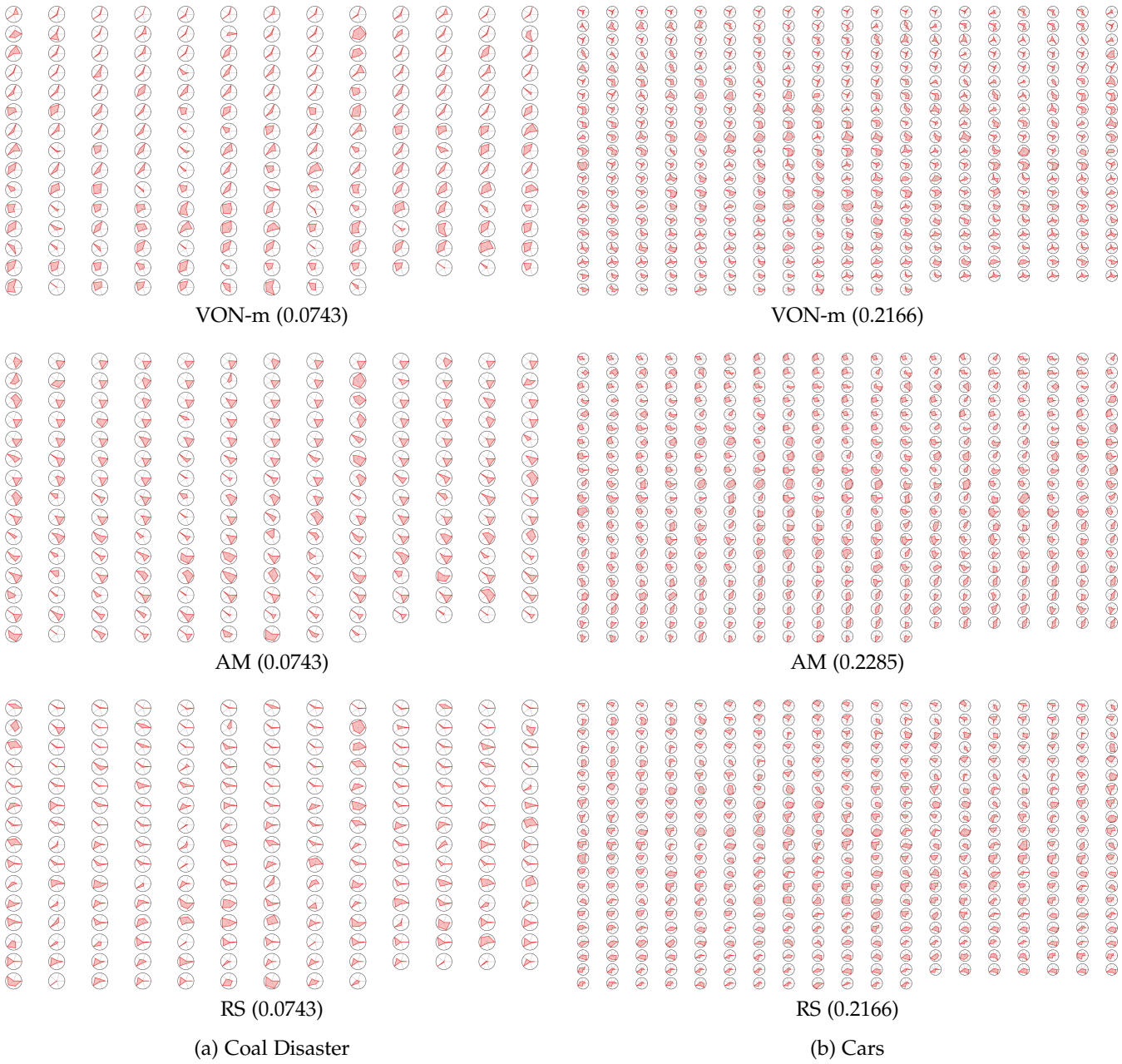


Fig. 6. Comparison of star plot axes reordering results of VON-m, AM, and random swapping (RS) using (a) Coal Disaster and (b) Cars datasets.

TABLE 4
Matrix reordering performance of VON-m using SCH dataset and different embedding approaches. Each approach embeds the data points into 2, 8, 16, and 32 dimensions to produce the input for VON-m. The ordering results are evaluated using LA, PR, and BW metrics [10].

	2-dim			8-dim			16-dim			32-dim		
	LA	PR	BW	LA	PR	BW	LA	PR	BW	LA	PR	BW
T-SVD	**40.71k**	11.21k	167.59	52.19k	9.16k	182.29	**43.97k**	**9.50k**	**188.94**	58.53k	**11.88k**	208.94
PCA	41.47k	**8.88k**	**162.18**	**51.52k**	**9.12k**	**179.35**	44.23k	10.02k	201.65	**54.78k**	12.07k	**188.12**
LLE	68.60k	13.74k	204.12	70.71k	12.72k	196.35	80.78k	14.38k	214.65	102.4k	15.92k	222.65
AE	59.52k	12.04k	207.24	67.28k	12.25k	215.35	63.59k	13.48k	195.18	70.56k	13.41k	188.76
Isomap	58.25k	12.30k	203.59	60.47k	11.95k	196.94	64.29k	11.66k	210.76	67.76k	12.60k	208.47

training and ordering time as the other learning-based approach AM. VON-c has similar training time, but requires more time for ordering. These learning-based approaches are all much faster than SA, which requires a large amount of random moves to optimize the order. The specialized approaches are several times faster in matrix ordering, while VON is still interactive with less than one second in response time. Additionally, we should note that VON

TABLE 5
Perfomance of VON-m on matrix reordering using SCH dataset and different embedding approaches to generate initial coordinates. The results are evaluated using LA, PR, and BW metrics.

	LA	PR	BW
PCA-8D to t-SNE	40.01k	8.28k	**174.59**
T-SVD-8D to t-SNE	**39.08k**	**8.23k**	178.41
t-SNE	51.77k	11.60k	206.88

TABLE 6
The timing results for matrix reordering using FTL datasets. For the learning-based approaches, both training and ordering time is reported. For the other approaches, only the ordering time is included.

	VON-a	VON-m	VON-c	AM	SA
Train	17s/e	17s/e	17s/e	17s/e	-
Order	0.81s	0.81s	1.14s	0.81s	40s

	C-LO-δ_I	U-LO-δ_I	U-BC	C-BC	NN
Order	0.21s	0.19s	0.16s	0.17s	0.70s

can be used for various metrics, which reduces the effort to develop and implement specialized algorithms.

Visualization results. Fig. 7 shows the visualization results of matrix reordering using FLT dataset. The results are evenly sampled with an interval of 10, starting from 3. Please refer to the supplemental material for the complete results. We find that VON-m, C-LO-δ_I, U-LO-δ_I, and NN deliver better performance than the other approaches, as we can observe several matrices with large block of black cells, especially at $G3$, $G33$, and $G53$. For the other four approaches, these kinds of blocks are rare, indicating the lack of ability to capture graph communities. This is inline with the quantitative results in Section 4.4.2. We do find the C-LO-δ_I and U-LO-δ_I achieve slightly larger Moran's I values, but the improvement seems to be marginal both quantitatively and qualitatively.

The impact of parameter tuning for simulated annealing (SA). We conduct additional experiments to verify the matrix ordering results using SA, as our default parameter setting leads to inferior performance for this task. SA involves three key parameters: initial temperature, restart, and cooling rate. The suggested cooling rate is between 0.8 and 0.99 [85], and we use the median 0.9 as the cooling rate in our default setting. The restart is essentially the number of repeated experiments. In our default setting, we use 5 to strike a balance between the performance gain and the additional cost of time.

Our default setting of SA leads to an average Moran's I of 0.09 for the 96 graphs using FLT dataset, which is similar to the UB-C performance reported by Vanbeusekom et al. [76]. Following Brusco et al. [13]'s settings, we use a cooling rate of 0.95 and restart of 20 for comparison. This setting leads to a decreased rate of cooling and an increased number of repeated randomized procedure. With this setting, the mean Moran's I increases from 0.09 to 0.258, but the timing cost to sort one collection also rises from 40 seconds to 261.3 seconds. Considering the timing performance, we do not experiment the more exhaustive settings. For a qualitative assessment, please refer to Fig. 7.

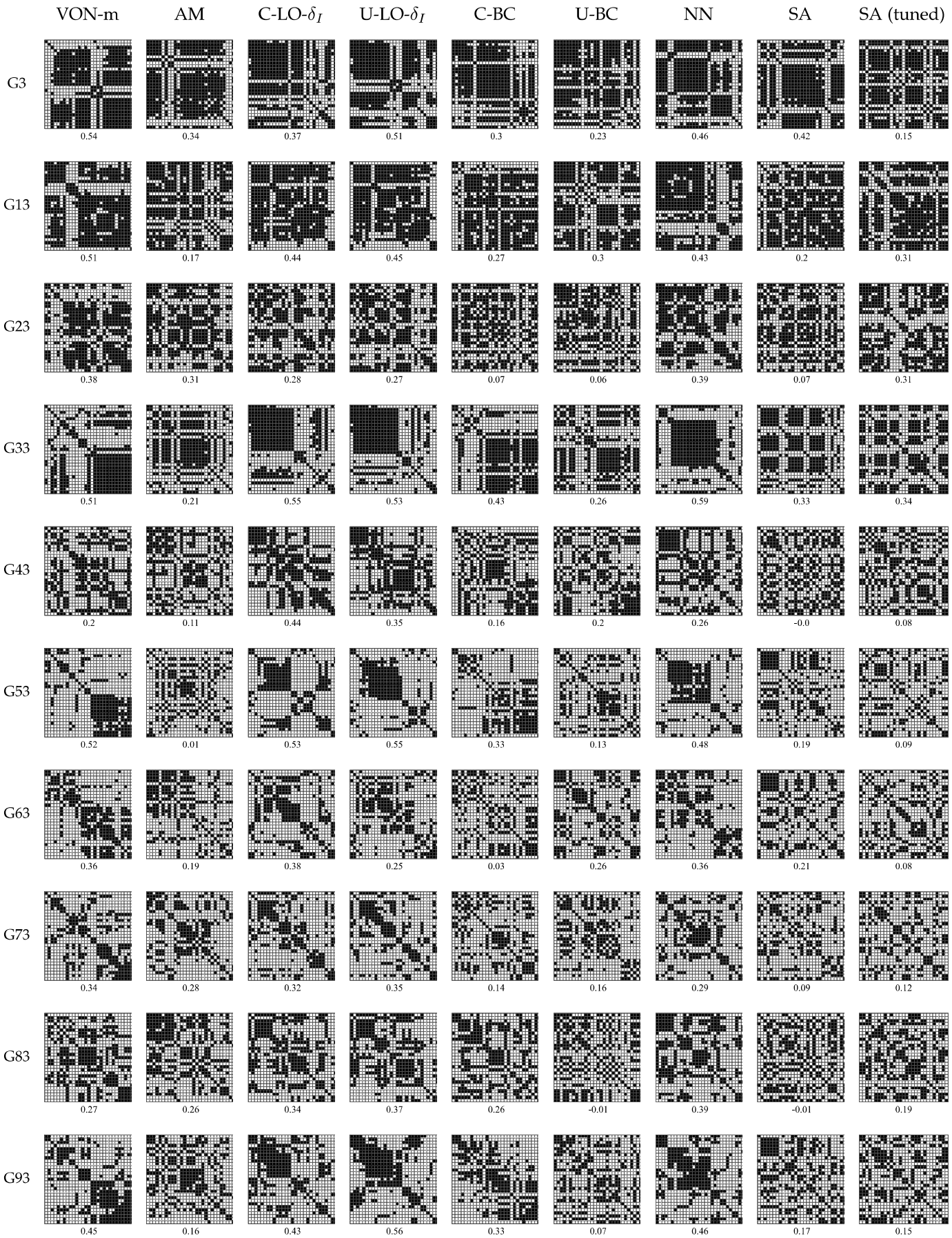


Fig. 7. Comparison of matrix reordering results of VON-m, AM, C-LO-δ_I, U-LO-δ_I, C-BC, U-BC, NN, SA, and SA with parameter tuning using the FLT dataset. Each row shows the results corresponding to one time step of the dynamic graph, evenly sampled from all steps starting from time step 3.