Structure-aware Visualization Retrieval

Haotian Li
The Hong Kong University of Science and Technology
Hong Kong SAR, China
Singapore Management University
Singapore
haotian.li@connect.ust.hk

Yong Wang
Singapore Management University
Singapore
yongwang@smu.edu.sg

Aoyu Wu
The Hong Kong University of Science and Technology
Hong Kong SAR, China
awuac@connect.ust.hk

Huan Wei
The Hong Kong University of Science and Technology
Hong Kong SAR, China
hweiad@connect.ust.hk

Huamin Qu
The Hong Kong University of Science and Technology
Hong Kong SAR, China
huamin@cse.ust.hk

ABSTRACT

With the wide usage of data visualizations, a huge number of Scalable Vector Graphic (SVG)-based visualizations have been created and shared online. Accordingly, there has been an increasing interest in exploring how to retrieve perceptually similar visualizations from a large corpus, since it can benefit various downstream applications such as visualization recommendation. Existing methods mainly focus on the visual appearance of visualizations by regarding them as bitmap images. However, the structural information intrinsically existing in SVG-based visualizations is ignored. Such structural information can delineate the spatial and hierarchical relationship among visual elements, and characterize visualizations thoroughly from a new perspective. This paper presents a structure-aware method to advance the performance of visualization retrieval by collectively considering both the visual and structural information. We extensively evaluated our approach through quantitative comparisons, a user study and case studies. The results demonstrate the effectiveness of our approach and its advantages over existing methods.

CCS CONCEPTS

• **Human-centered computing** → **Visualization**; • **Information systems** → *Information retrieval*; • **Computing methodologies** → *Machine learning*.

KEYWORDS

Data Visualization, Visualization Retrieval, Visualization Similarity, Representation Learning, Visualization Embedding

CHI '22, April 29-May 5, 2022, New Orleans, LA, USA

ACM ISBN 978-1-4503-9157-3/22/04…$15.00
https://doi.org/10.1145/3491102.3502048

ACM Reference Format:
Haotian Li, Yong Wang, Aoyu Wu, Huan Wei, and Huamin Qu. 2022. Structure-aware Visualization Retrieval. In *CHI Conference on Human Factors in Computing Systems (CHI '22), April 29-May 5, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3491102.3502048

1 INTRODUCTION

Data visualization provides users with a powerful approach to analyze enormous data, communicate insights and achieve efficient decision-making. Along with the popularity of visualizations, a huge number of visualizations based on Scalable Vector Graphics (SVGs) have been created and shared online. Compared with bitmap-based visualizations, SVG-based visualizations have many advantages such as the support of interactions [1] and quality-preserving resizing. Thus, SVGs have been adopted by various online platforms to store and present visualizations, for example, Plotly[1] and Observable[2]. With such a large volume of visualizations online, how to retrieve similar visualizations has attracted growing research interest from both academia and industry [31, 32, 36] due to its significant importance for many downstream tasks. Specifically, the retrieval of similar visualizations is fundamental to downstream tasks such as creating visualization collections [32] and recommending visualizations [31].

To achieve effective retrieval of similar visualizations, the core problem is to characterize the similarity between two visualizations. Existing studies mainly focus on estimating the similarity between visualizations according to the data or perceptual similarity. The existing methods based on data similarity [31, 42, 43, 48] focus on the characteristics of data such as data distribution or metadata, ignoring the visual appearance of visualizations. Since the original data is not always available with the visualizations, the application of visualization retrieval methods based on data similarity is quite limited. Perceptual similarity mainly refers to the similarity of visualizations perceived by users, which can also reflect the data similarity. Compared to the direct computation of data similarity, the computation of perceptual similarity does not rely on the original data. To compute the perceptual similarity, existing approaches [29, 36, 61] first extract the visual feature vectors from

[1] https://plotly.com/
[2] https://observablehq.com/

visualizations and further calculate the distance between feature vectors to measure their similarity. These methods mainly extract the visual features of visualizations at the level of pixels. For example, Saleh et al. [36] measured the visualization similarity by using the color distribution of different pixels (i.e., color histograms). Recently, deep learning-based methods [29, 61] have been proposed to extract visual features automatically by treating visualizations as bitmap images (e.g., the images in ImageNet [10]). However, few prior studies have considered the *structural information of visualizations* that exists in SVGs by nature, when characterizing the perceptual similarity of visualizations.

Structural information of visualizations mainly describes the spatial and hierarchical relationship between elements, such as the position, grouping and hierarchy of the basic visual elements (e.g., *<rect>* and *<path>*). Compared with the commonly-used *visual information* (i.e., the visual features to describe the appearance of visualizations) of visualizations, structural information enables a unique perspective to characterize the appearance of visualizations at the level of visual elements instead of pixels. It provides an accurate description of how different visual elements are organized in visualizations. For example, as shown in Figure 1, a grouped bar chart with two groups of bars (Figure 1(a)) and a bar chart with only one group of bars (Figure 1(b)) seem to show the same trend and are regarded as similar charts, if only the visual information is considered by using a computer-vision-based method (e.g., convolutional neural network (CNN) models). However, the grouped bar chart actually shows how two sets of data are compared and it should not be treated as a similar visualization as the bar chart with a single group of bars. Instead, another grouped bar chart (Figure 1(c)) with both similar *structure* and appearance should be regarded similar to the query bar chart (Figure 1(a)). From the example above, it is obvious that structural information plays an important role in characterizing the perceptual similarity between visualizations. However, it still remains unclear what kind of structure-based features can be extracted and how these structure-based features can be leveraged to facilitate similar visualization retrieval.

In this paper, we aim to fill the research gap by leveraging both structural and visual information to accurately evaluate the *perceptual similarity* between visualizations. We first conducted a preliminary study to better understand users' criteria on assessing the perceptual similarity of visualizations and identified the three most important criteria, i.e., *the type of a visualization*, *the number of visual elements* and *the overall trend of visualized data*. Building upon these results, we propose to transform SVG-based visualizations to *graphs* and *bitmap images* that reflect the structure and the appearance of visualizations, respectively. Then we utilize contrastive representation learning to comprehensively delineate structural and visual information in a visualization with embedding vectors. Contrastive representation learning is a type of self-supervised learning method and can minimize the distance between similar samples and maximize the distances between diverse samples [21]. With contrastive learning, we avoid manually labeling the similarity between different visualizations, enabling us to easily generalize our approach to various visualizations. Finally, we gain an embedding vector for each visualization that characterizes its structural and visual information and is used for retrieving similar visualizations. Using the VizML corpus [19], we extensively evaluate our approach through a crowdsourced user study, multiple case studies and quantitative comparisons. The results demonstrate the effectiveness of our approach.

The major contributions of this paper are summarized as follows:

- We present a novel structure-aware approach to characterize the perceptual similarity between visualizations through embedding vectors, which enables effective similar visualization retrieval.
- We conduct extensive evaluations including a crowdsourced user study with 50 participants, multiple case studies and quantitative comparisons with existing visualization retrieval methods. The results verify the effectiveness of our structure-aware visualization retrieval approach.
- We summarize the lessons we learned during exploring the usage of structural information in visualization retrieval.

2 RELATED WORK

The related work of this study can be categorized into three parts: retrieval of visualizations, visualization similarity estimation and visualization storage formats.

2.1 Visualization Retrieval

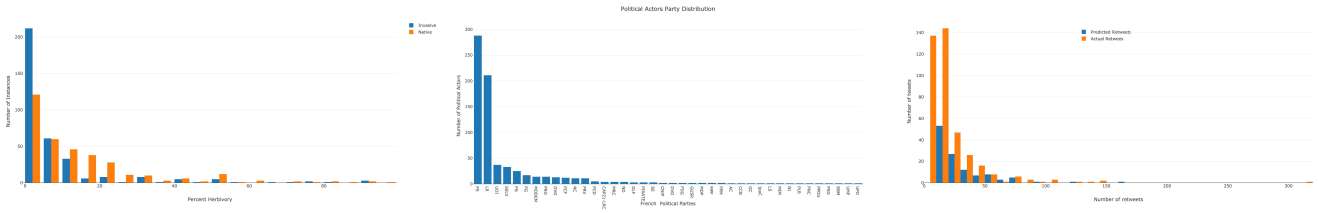
Visualization retrieval has attracted researchers' interests in recent years along with the increasing number of visualizations. According to the type of queries, there are two major classes of methods for retrieving visualizations [45], retrieval by definition and retrieval by example.

Retrieval by definition means that users can explicitly specify the criteria of retrieving visualizations using either programming language or natural language. For example, Hoque and Agrawala [18] enable users to create a JSON-like specification to indicate their target characteristics of visualizations such as encoding types. Some other prior studies [7, 27, 44, 45] also provide users with tools to search for visualization using explicit queries. Compared to retrieval by definition, retrieval by example provides an intuitive way for users to define the criteria of retrieving visualizations. Users can use existing visualizations or sketches to search for other visualizations. Several recent studies [29, 34, 36] take example visualizations as inputs and return similar ones for data exploration or visualization re-use. Zenvisage [42] and ShapeSearch [43] allow users to sketch their desired data pattern in visualizations. Then they retrieve the data which matches the pattern from the database and visualize them to users. In this line of research, one of the core problems is how to define the similarity between visualizations, which will be further discussed in Section 2.2.

Our structure-aware approach falls in the category of retrieval by example. Our approach takes SVG-based visualizations as the input and then represents the visual and structural information of them as embedding vectors for similar visualization retrieval.

2.2 Visualization Similarity

Computing the similarity of visualizations benefits various downstream tasks such as assisting in exploratory data analysis [62], querying visualizations [29] and generating visualization collections [32]. Inspired by a previous study [29], prior methods on



(a) Query (b) Visual Information (CNN) (c) Structural and Visual Information

Figure 1: This figure shows (a) a sample query and the top-1 visualizations retrieved by (b) only considering visual information using CNN and (c) considering both structural and visual information. This example shows that structural information is essential in performing similar visualization retrieval.

computing the similarity of visualizations can be roughly categorized into two classes: data similarity and perceptual similarity.

The first class of methods focuses on the visualized data solely to delineate the similarity between visualizations. Some representative studies in this class include SeeDB [42], ShapeSearch [43] and VizCommender [31]. SeeDB [42] and ShapeSearch [43] define the similarity between visualizations as the similarity of data distribution or trend. These methods require that the raw data is available, which limits their application scenarios. A possible way to mitigate the issue is to extract the raw data from visualizations and then calculate the data similarity. However, since the performance of existing data extraction methods (e.g., [39]) is not satisfactory [22], the inaccurately extracted data may further affect the results of retrieval.

Another class of methods focuses on the perceptual similarity of visualizations. They extract visual features from visualizations and further utilize the distance between hand-crafted or learned feature vectors to characterize the similarity of visualizations. Hand-crafted features mainly refer to those features which are selected by the authors and can reflect certain characteristics of visualizations. For example, prior studies proposed to use color histograms [36] or histograms of gradients [32] to measure the perceptual similarity of infographics or visualization workbooks. Due to the inefficiency and complexity of selecting hand-crafted features, representations automatically learned by machine learning models have been applied recently. For example, ChartSeer [62] proposed to use an autoencoder to extract the representations of visualizations from their specifications. ScatterNet [29] and ChartNavigator [61] introduced convolutional neural networks (CNNs) on visualizations to learn their representations.

This paper aims to propose a structure-aware approach for retrieving perceptually similar visualizations. Compared with the retrieval methods based on data similarity (e.g., [42, 48]), our approach does not require the existence of original data and thus extends the scope of inputs. Different from the existing approaches which compute the perceptual similarity[3] of visualizations (e.g., [29, 31]), our approach considers both the pixel-level visual appearance and the structure of visual elements. Such a design allows our approach to better match the crowdsourced criteria of perceptual similarity, which will be introduced in Section 7.1.

[3]In the remaining part of this paper, "similarity" refers to perceptual similarity.

2.3 Visualization Format

Depending on the ultimate purposes of different visualizations, they can be stored in various formats including graphics, programs and hybrid approaches [55].

The graphics-based visualizations include two common formats: raster graphics and vector graphics. Raster graphics (i.e., bitmaps) are the most common approaches to store and share visualizations [37] due to their high compatibility. However, they are hardly editable and can lose visualization-specific information such as the chart type and visual encodings [55]. As an alternative, vector graphics like SVGs provide general users with the flexibility of modification and annotation and can preserve partial visualization-related information [54, 55] such as the relationship between visual elements. Previously, some studies have explored utilizing vector graphics to achieve visualization query by specification [18] and visualization type classification [1].

Programs are also a common way to store visualizations such as D3 [2], Vega-Lite [38] or Plotly. They preserve the visualization-related information and raw data. However, when rendering them as visualizations, extra compilers are always required, which limits their compatibility and wide usage. Recently, to combine the advantages of graphics and programs, several recent studies have also explored how to embed programs into graphics, such as [12, 35, 60].

Bitmaps and programs are commonly used in previous studies to compute the similarity of visualizations [29, 50, 62]. However, bitmaps suffer from the lack of visualization-specific information while the usage of programs limits the generalizability of related methods. Thus, in our paper, we propose to utilize the structural information (see Section 3.1) stored in scalable vector graphics (SVGs), which is widely used in spreading visualizations on the Internet due to its interactivity [1]. We combine the structural information extracted from SVGs and the visual information extracted from bitmaps to achieve effective characterization of visualization similarity.

3 BACKGROUND

In this section, we introduce the background of our research, including the structural information in SVGs (Section 3.1) and the overview of contrastive learning, which enables unsupervised representation learning for both SVGs and bitmaps (Section 3.2).

3.1 Structural Information in SVGs

Scalable vector graphics (SVGs) are files used to describe vector-based graphics using Extensible Markup Language (XML)[4]. They have been widely used in visualizations on the Internet [1]. Compared with bitmaps, SVGs can preserve more visualization-specific information such as the style of the visual elements [55]. The structural information in SVGs mainly includes the hierarchical and spatial relationship among elements and the properties of each element. The hierarchical information of visual elements can reflect how they are inherited and grouped, for example, in Figure 5, the *<path>*s of green bars are grouped under the same *<g>* while the *<path>*s of blue bars are grouped under another *<g>*. Such information of visual elements can further illustrate the usage of visual channels and reveal some information of the raw data like the number of data instances and the number of attributes. The spatial relationship, which is extracted based on the positions of visual elements, can describe how the elements are placed and a rough trend of data. The properties of each element in SVGs also encode rich information, for example, the type, style, and shape of the element. The types of elements reflect the functionality of elements, for example, *<g>* is used to group other elements and *<text>* can be rendered as graphics containing text. The styles of an element indicate how the elements are rendered. Some common styles include the color and stroke of an element. Furthermore, the shape of some visual elements can also be obtained from the properties, for example, the attribute *"d"* in *<path>* defines the shape of a path. Besides, SVGs also have some properties related to interactions (e.g., *onclick* and *onmouseover*).

In this paper, we propose to utilize the structural information extracted from SVGs to enhance the retrieval of similar static visualizations. To be more specific, we mainly consider structural information to reflect the hierarchical and spatial relationship among elements. Some side information such as the types and styles of elements is also utilized to distinguish different elements. More details are further illustrated in Section 5.1.

3.2 Contrastive Learning

Supervised deep learning approaches always require a large number of samples with labels to train a model with satisfactory performance [21]. However, corpora with high-quality labels are always hard to be obtained due to the high cost of human annotation. Thus, to reduce the effort of manually labeling, self-supervised representation learning approaches, which are sometimes considered as a subset of unsupervised learning methods, have attracted researchers in various fields, for example, computer vision [4, 5], user interface (UI) design [26] and visualization [47]. Contrastive learning is a representative approach of self-supervised learning. The basic idea behind it is to train a model which can discriminate similar and dissimilar samples [21]. As illustrated in Figure 2, a common pipeline in contrastive learning approaches contains 4 steps: *data augmentation, representation extraction, representation projection and contrastive loss computation* [4]. In the first step, a data sample will be randomly transformed (e.g., distortion for images) and the transformed samples will be considered as similar (positive) samples. Then the transformed samples are encoded to embedding vectors by an encoder in the second stage. The embedding vectors are further projected to a space where the loss is computed. After training using the pipeline above, the encoder is used solely to extract representations of data samples and different projectors can be trained for various downstream tasks such as classification.

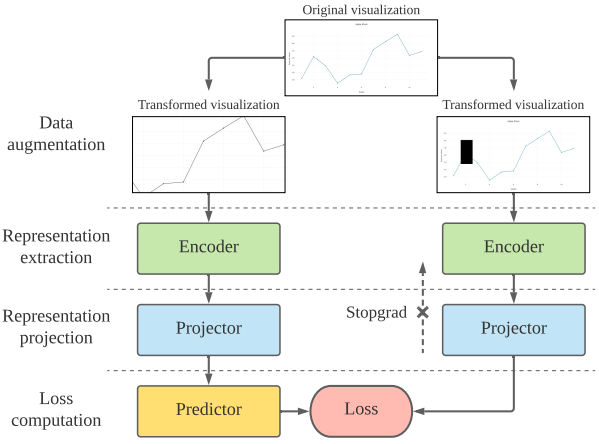


Figure 2: The basic structure of contrastive learning contains four major modules [4]: data augmentation, representation extraction, representation projection and loss computation. This figure shows SimSiam [5] as an example.

In our paper, we propose to use two CNN- and GNN-based contrastive learning models [5, 46] to generate the embedding vectors of visualizations' visual and structural information, respectively. Then the embedding vectors are concatenated and used for visualization retrieval.

3.3 Graph Neural Networks

Inspired by successful convolutional neural networks (CNNs), graph neural networks (GNNs) have been proposed to model the relationship among nodes in graphs. The basic idea behind GNNs is to propagate the features of nodes through edges and then aggregate the information on nodes to capture node features and graph structures [63]. The feature propagation and aggregation can be considered as a generalized convolutional filter on graphs. GNNs have shown outstanding performance on graph-related tasks (e.g., node classification [40] and graph classification [58]) in various application domains (e.g., UI design [33], online education [25] and visualization [52]). As introduced in Section 3.1, SVG elements are organized as trees that can also be regarded as graphs. Thus, it is intuitive to apply GNNs to learn and represent the structural information in SVGs as embedding vectors. Considering the advantages of contrastive learning (see Section 3.2), GNN-based contrastive learning approaches are suitable for extracting the embedding vectors of graphs of SVG elements in our structure-aware approach. Existing GNN-based contrastive learning approaches are mainly applied to three types of tasks [57], including node-level tasks such as node classification (e.g., DGI [49]), edge-level tasks such as link prediction (e.g., BiGi [3]) and graph-level tasks such as graph

[4] https://developer.mozilla.org/en-US/docs/Web/SVG

classification (e.g., InfoGraph [46]). Since we aim to represent the structural information in the graph of SVG elements with embedding vectors, InfoGraph [46], one of the state-of-the-art methods for graph embedding, is applied in our approach. Section 5.1 will introduce more details of InfoGraph.

4 PRELIMINARY STUDY

Before designing our structure-aware visualization retrieval approach, we conducted a preliminary study in which we collected the opinion of 54 visualization users on the criteria of perceptual similarity between visualizations. In this section, we introduce the procedure[5] of the study and summarize the important criteria.

4.1 Procedure

Our preliminary study was conducted on Prolific[6], a widely used platform for recruiting research participants. There are totally three parts in the study. In the first part, we first introduced the overall process and got the participants' consent to join the study. Then, to verify that the participant has basic knowledge of visualizations, each participant was required to answer three simple visualization-related questions, for example, *"what is the chart type of the given visualization?"*. Only participants who correctly answered the three verification questions were allowed to join the study. No other criteria were used in the participant recruitment. In the second part of the study, to encourage the participants to reflect on how they judge the similarity of visualizations, each participant was presented with five query visualizations and their retrieved top-5 similar visualizations by using visual information only. The participants were asked to give each retrieved visualization a score ranging from 1 (the least similar) to 5 (the most similar). After finishing the scoring, in the last part of the study, we asked participants to write down their criteria of scoring the retrieved visualizations in a text box.

After the study, we summarized the responses from participants. Since there may be ambiguity in understanding the criteria mentioned by participants, we first classified the major criteria into six major categories and two co-authors of this paper labeled all responses individually. If the annotations were inconsistent on any response, we examined and discussed together to reach an agreement on these cases.

4.2 Results

The six major criteria and their frequency are shown in decreasing order in Figure 3. In the results, we can notice that there are three important criteria (i.e., visualization type, the trend of data and the number of visual elements) with much higher frequency than other criteria. The results of our preliminary study also align with a previous study [23] well. Specifically, the number of visual elements and the trend of data are also considered when measuring the difference between two visualizations in the previous research [23]. Thus, the type of visualization, the trend of data and the number of visual elements are necessary to be considered explicitly in our approach when characterizing similarity of visualizations.

[5]The protocol of the preliminary study and the user study has been approved by the Institutional Review Board of our institution.
[6]https://www.prolific.co/

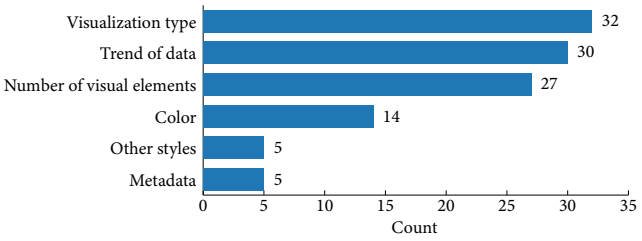


Figure 3: We categorized different criteria mentioned by participants into six classes and identified the three most important criteria based on their frequency. *Other styles* refer to the styles of visual elements other than colors, for example, the space between bars and the width of bars. *Metadata* refers to the meta-information of data in the visualizations, for example, the range and the type of data.

5 METHOD

In this section, the method of our structure-aware visualization retrieval is introduced. An overview is shown in Figure 4. To extract and represent the structural information in a visualization, we first construct a graph of visual elements with features and then apply a GNN encoder to generate the embedding vector of it (Section 5.1). Then we also render the visualization to a bitmap and use a CNN model to encode the visual information as an embedding vector as well (Section 5.2). Here we applied contrastive representation learning to train both CNN and GNN encoders since it can eliminate human efforts on data annotation. Finally, we normalize and concatenate the embedding vectors of structural and visual information for similar visualization retrieval (Section 5.3).

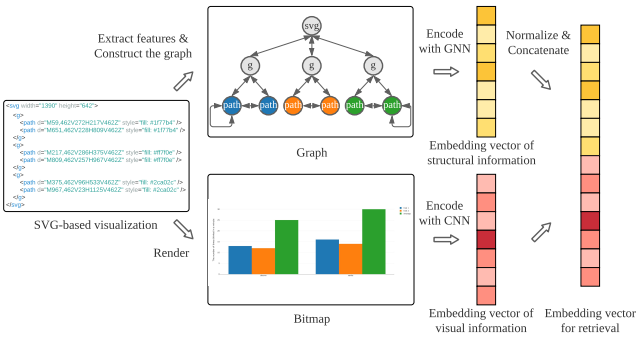
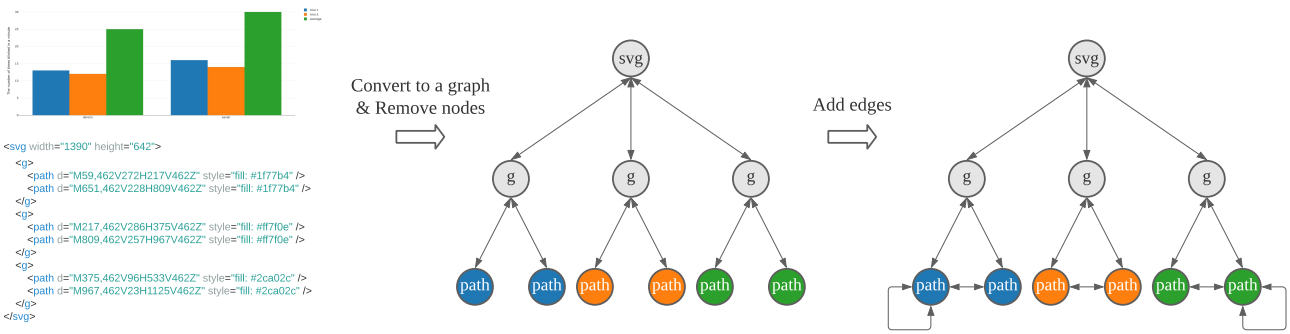


Figure 4: Our approach extracts both structural and visual information from visualizations first. Then the two types of information are encoded to embedding vectors separately. Finally, two embedding vectors are normalized and concatenated as the final representation of the visualization.



(a) Rendered visualization and SVG (b) Initial graph of visual elements (c) Final graph of visual elements

Figure 5: This figure illustrates how we transform an SVG to a graph of visual elements. In the SVG, each bar can be either represented by a *<path>* or *<rect>*. Due to the limited space, we only show partial SVG and the corresponding subgraph. Each node represents an element in the SVG. The colors of nodes in the graph indicate the corresponding bars in the visualization. In (c), all leaf nodes are with self-loop edges, which are not all shown in this figure.

5.1 Representation Learning of Structural Information

As introduced before, structural information in SVGs can reflect the hierarchical and spatial relationship between visual elements explicitly. To utilize the structural information, we first extract visual element-level features and construct a graph of visual elements. Then, we apply a GNN-based graph contrastive learning method to generate the embedding vector of the structural information.

Feature Extraction. In the first step, we aim to extract features to describe the characteristics of elements in SVGs. These features are designed to reflect the types, styles and shapes and positions of elements. To make our approach simple and generalizable, we only extract basic features inspired by Beagle [1]. The *types* of elements can reflect their functionality in an SVG. We represent the types of elements with one-hot encoding. The *style* features of elements mainly include the color, the stroke width and the opacity of the element, which are the most common styles of elements. According to the results of our preliminary study, we also consider the styles of SVG elements. Though the styles of visualizations such as colors are not the most frequent criteria when deciding the similarity between visualizations, it is still considered by some participants. Thus, we take the commonly used styles into consideration and ignore those infrequent ones such as the stroke style.

How to define *shape* features for different types of visual elements is challenging due to their characteristics. For some visible visual elements like bars in bar charts and scatters in scatter plots, we are able to describe each visual element with the area, center, height and width of its bounding box. However, it is not enough to describe lines in line charts with these simple features. Two lines with the same bounding box can represent totally different trends of data. To deal with this issue, we further introduce two features to describe the shape of lines in line charts: the number of vertices in the line [1] and the trend of the line. Inspired by a prior study [23], LOESS regression [8] is applied to model the rough trend of a line based on vertices. Since LOESS is a non-parametric regression approach, we are not able to extract a fixed number of features to describe the trend. Thus, we use the predicted values of LOESS on five evenly sampled vertices as features of the trend. Since *<text>* can hardly be described by the features above, we further add the length of the text as a feature. Furthermore, the relationship between positions of elements within the same group is also necessary to reflect the overall trend in the visualizations. Thus, we sort the visual elements according to their positions on the horizontal and vertical axes and introduce the differences in positions as features of each element as well. Finally, for elements without certain features (e.g., *<g>* does not have specified width and height), we fill zeros as the placeholders. All features are also scaled according to their ranges. For example, positions are scaled using the height and width of the entire SVG.

Graph Construction. After extracting the features for individual elements in visualizations, we construct graphs of visual elements based on the tree structure of elements in SVGs. Such graphs of visual elements are used to represent the hierarchical and spatial relationships between different visual elements in visualizations. Since the graph is the input to a graph neural network (GNN) in the latter stage, so we further process them to facilitate the propagation of features between nodes. An overview of graph construction is shown in Figure 5.

First, the tree of elements in an SVG is transformed to a bidirectional graph where elements and hierarchical relationships are treated as nodes and edges respectively. Then, we remove the visual elements which serve as references such as the legend and grid lines. This step is conducted to reduce the noise in the graph of visual elements. In the next step, we remove the *<g>* elements that connect with one or two other elements. The rationale behind this operation is that *<g>* is used to group other elements and thus not meaningful if it has zero or one child. Removing these *<g>* elements can reduce unnecessary nodes and edges in the graph. Finally, we add two types of edges to augment the graph: the self-loop edges and the edges between neighbor elements (see Figure 5(c)). The self-loop edges are added to preserve each element's own features during the feature propagation. The edges between neighbor elements are designed to reflect the spatial relationship between elements

by propagating their spatial features. Here neighbor elements are defined as the elements which are next to each other after sorting according to their positions as described previously.

Contrastive Learning Structure. To represent the structural information as an embedding vector, we utilize InfoGraph [46], a state-of-the-art graph contrastive representation learning model. The core idea of InfoGraph is to maximize the mutual information between the embedding vector of the whole graph and the embedding vectors of substructures (e.g., individual or a group of nodes) in the same graph. To be more specific, it takes pairs of graphs as the input of a GNN encoder. Then it optimizes the encoder by maximizing the mutual information between the embedding vector of one graph and the embedding vector of a subgraph inside, and minimizing the mutual information between the embedding vectors of a graph and the subgraph in another one. Finally, it uses the encoder to generate the embedding vectors of all nodes and aggregates them as the representation of the graph.

In our method, we train InfoGraph to generate the embedding vectors of graphs of visual elements, which encode the structural information in SVGs.

5.2 Representation Learning of Visual Information

In the previous section, we introduce how we extract the representation of structural information hidden in SVGs with a state-of-the-art graph contrastive learning method. Though the structural information can partially reflect the appearance of a visualization, for example, the size and position of a visual element, it still lacks the ability to comprehensively describe the appearance of visualizations. Thus, inspired by previous studies [29, 61] which utilize convolutional neural networks (CNNs) to extract the visual features of visualizations, our approach utilizes contrastive learning to generate the representation of visualizations using bitmaps as input. In the rest of this section, we briefly introduce the structure of the contrastive learning structure we used and then illustrate how we adapt it to extract visual information from visualizations.

Contrastive Learning Structure. In our method, SimSiam [5] is applied to generate the representation of visual information. Compared to other state-of-the-art contrastive learning approaches such as SimCLR [4] and BYOL [15], SimSiam has a similar performance with less demand for computational resources. It utilizes a structure based on Siamese networks (see Figure 2). Two randomly transformed samples (denoted as s_1 and s_2) of the original sample are fed into the same encoder which is followed by a projection head. After projection, both samples are transformed to embedding vectors (denoted as $\mathbf{v_1}$ and $\mathbf{v_2}$ respectively) , which are further fed into a prediction head with a bottleneck structure. The outputs of the prediction head are denoted as $\mathbf{p_1}$ and $\mathbf{p_2}$ correspondingly. Then the loss is calculated based on the negative cosine similarity with a stop-gradient operation as $Loss = -1/2 * (cos(\mathbf{v_1}, stopgrad(\mathbf{p_2})) + cos(\mathbf{v_2}, stopgrad(\mathbf{p_1})))$. Here the stop-gradient operation controls how the model is optimized with gradients and proved to be beneficial [5].

Pre-processing and Data Augmentation. The original SimSiam is designed for general images. To adapt it for visualizations, we modify the pre-processing of input bitmaps and data augmentation. First, since the input of the encoder has to be a square bitmap, we need to resize or pad visualizations to be square. Resizing input images is used in SimSiam [5]. However, resizing is not appropriate for visualizations since a resized visualization can alter the trend of the original one [23], which may affect the judgment on the similarity of visualizations. Thus, we pad the original bitmap to be a squared one, which can better preserve the original trend of data. The next step is to apply data augmentation techniques to generate transformed data instances as mentioned in Section 3.2. Chen *et al.* [4] have summarized eight common approaches used for image augmentation, including cropping and resizing, cutting out, flipping, rotating, blurring, applying noise or filter and distorting the colors. However, not all of them are suitable for visualizations. For example, flipping the visualization can generate a visualization with a different trend of data as shown in Figure 6. Thus, in the training of our CNN model for visualizations, we select some of the data augmentation techniques based on those used in SimSiam [5]. First, cropping and resizing, and partially cutting out are applied inspired by the closure principle in Gestalt principles of perception, which states that people have the ability to fill in the blanks and make the object in the image complete. Also, the color distortion is randomly applied to some visualizations by adding color jitter or transforming the bitmap to a greyscale one. The rationale behind this is that color is not always used to encode information in visualizations and is considered not the most important criteria according to our preliminary study (see Figure 3). In Figure 2, we present two transformed visualizations in which the left one is cropped and converted to a greyscale one while the right one is cut out and added color jitter (padding is not shown in the figure).

After generating the transformed samples, we use the original settings of SimSiam to train the encoder. At the end of the training, other parts of the model are disposed and the encoder is kept to generate the embedding vector of visual information.

(a) Original **(b) Flipped**

Figure 6: The flipped visualization is dissimilar to the original one due to the change of overall trend. Thus, flipping is invalid for generating similar samples.

5.3 Visualization Retrieval

In Sections 5.1 and 5.2, we introduce how to represent both structural and visual information using low-dimensional embedding vectors. Then, with the two embedding vectors of a visualization, we perform the similar visualization retrieval.

In our approach, we normalize and concatenate both the embedding vectors into one vector, which is the final representation

of a visualization. Initially, we tried to apply multi-modal autoencoders [11, 30] to learn the joint embedding vector of both structural and visual information. However, the performance of the learned embedding vector using quantitative metrics in Section 6.3 is not better than simply concatenation. We suspect the reason is that encoding two vectors into a single vector is a lossy compression, which introduces extra noise. Thus, to avoid extra training and additional noise, we use concatenation as our approach to generate the final embedding vector of a visualization. After obtaining the embedding vectors of visualizations, the cosine similarity between the embedding vectors of two visualizations is used to measure the distance between them. When a visualization is selected for query, we calculate its similarity to other visualizations and rank the similarity scores to retrieve the most similar ones.

6 EVALUATION

By using the large-scale SVG-bitmap visualization corpus (Section 6.1), we conducted extensive evaluations to assess our structure-aware visualization retrieval approach including quantitative comparisons (Section 6.3), a user study (Section 6.4) and case studies (Section 6.5). We also introduce the model settings used in evaluations (Section 6.2). The results verify the effectiveness of our structure-aware visualization retrieval approach.

6.1 Corpus

Since our approach is a deep learning-based method (Sections 5.1 and 5.2), we need a relatively large-scale corpus to train and test it. Thus, by using the links to visualizations provided in VizML [19], we built a crawler to collect the SVG-bitmap visualization pairs from Plotly Chart Studio[7]. Following previous practices using the VizML corpus [19, 24], we also kept one visualization per user. Also, we removed those pairs with invalid bitmaps or SVGs, for example, empty visualizations or incomplete SVGs. Finally, 51,037 SVG-bitmap visualization pairs were collected. The corpus contains five types of visualizations: bar charts, box plots, histograms, line charts and scatter plots. Since the difference of histograms and bar charts mainly lies in the data transformation, which is beyond our scope, they are categorized into the same class in the following evaluations. A detailed distribution of visualization types is shown in Table 1. In our evaluations, we follow the practice in Screen2Vec [26] and randomly sampled 90% of pairs of each visualization type as the training set and the rest was used for testing.

Table 1: Statistics of the SVG-bitmap visualization corpus.

Number of SVG-bitmap pairs		51,037
Number of each visualization type	Bar/Histogram	12,608
	Box	3,269
	Line	15,488
	Scatter	19,672

6.2 Model Settings

We compared the retrieved visualizations using four methods to demonstrate the effectiveness of our structure-aware approach:

[7] https://chart-studio.plotly.com/

- **Visual Information by Histogram of Oriented Gradients (HOG).** HOG [9] is one of the most widely used feature descriptors of images and has been applied to compute the similarity between visualizations in a recent study [32].
- **Visual Information by CNN.** CNNs have been widely applied to extracting the visual information of visualizations and computing the similarity [29, 32, 61]. Following the method described in Section 5.2, we extract the visual information using SimSiam [5] with ResNet-50 [17] as the encoder. Empirically, we set the training epochs as 200, the batch size as 128 and the learning rate as 0.025. The embedding dimension is set as 512. We did not use any pre-trained model as suggested by Haehn *et al.* [16].
- **Structural Information by GNN.** As mentioned in Section 5.1, we use InfoGraph [46] with Graph Isomorphism Network (GIN) [58] as the encoder to generate the embedding vectors of structural information. Based on the original experiment settings, we set the training epoch as 40, the batch size as 128 and the learning rate as 0.001. Since the radii of graphs in our corpus are mostly lower than or equal to 2, we set the layers of the encoder as 2. The embedding dimension is 512 as well.
- **Structural and Visual Information Fusion.** This is our structure-aware approach which jointly considers structural and visual information by normalizing and concatenating the embedding vectors generated by CNN and GNN as mentioned in Section 5.3.

After extracting the embedding vectors of visualizations using the four approaches above, we calculate the similarity score between visualizations as mentioned in Section 5.3. In the following sections, we use ***V-HOG***, ***V-CNN***, ***S-GNN*** and ***S&V-Fusion*** to simplify the names of the four approaches above.

6.3 Quantitative Evaluation

We first conducted a series of quantitative evaluations to assess the effectiveness of our approach in comparison with baselines.

Metrics. As described in Section 6.1, there are several types of visualizations in our corpus. Thus, we need to use some metrics which are unified across different types of visualizations. Two simple metrics are utilized to approximately probe the performance of different approaches based on the criteria identified in our preliminary study. The first metric is to measure the *visualization type consistency* between a query visualization and the retrieved visualizations. To be more specific, we count the number of retrieved visualizations that are of the same type as the original visualization in a top-k query. Then the number of such visualizations is divided by k to obtain the *type-consistent rate* of each query visualization. Finally, the average and the standard deviation of *Type-Consistent Rates* of all visualizations in the test set are computed and denoted as *average type-consistent rate* (TCR_{ave}) and *standard deviation of type-consistent rates* (TCR_{std}).

The second metric is to measure the *difference of the numbers of visible visual elements* between query and retrieved visualizations. Here the visible visual elements refer to the elements that are the leaf nodes in our graph. Since the non-leaf elements (i.e., <*g*> and <*svg*>) are mainly used to group or contain other elements and are

not directly visible, they cannot be observed by visual information-based models (i.e., *V-CNN* and *V-HOG*). To make the metric fair to all methods, we only count the visible visual elements. Examples of visible visual elements are shown as colored nodes in Figure 5. First, we calculate the normalized difference of the numbers of visible visual elements between query and retrieved visualizations. Then, similar to *TCR*, we also compute the *average Differences of the numbers of Visual Elements* (DVE_{ave}) and *standard deviation of differences of the numbers of visual elements* (DVE_{std}).

Among all metrics, a smaller value of TCR_{std}, DVE_{ave} or DVE_{std} is better, since a smaller value of these metrics shows that the method can achieve a more stable performance or smaller difference between the query visualization and the retrieved visualizations. On the other hand, a larger value of TCR_{ave} is appreciated, since it demonstrates that the model can retrieve more visualizations of the same type.

Results. Table 2 shows the overall results of the four methods. The results are the average values from five runs of each method. Our structure-aware approach consistently outperforms other approaches using TCR_{ave} and TCR_{std} and is better than visual information-based methods in DVE_{ave} and DVE_{std}. The results of DVE_{ave}s align with the results by Haehn *et al.* [16]. They conducted point-cloud experiments and demonstrated that CNNs do not perform well in estimating the difference of numbers of visual elements. In our experiments, *V-CNN* also performed worse when using DVE_{ave} as the metric. Compared to *V-CNN*, both *S-GNN* and *S&V-Fusion* can distinguish the number of visual elements well, which shows the necessity of considering structural information in characterizing the similarity of visualizations.

Table 2: Results of our quantitative evaluation. *TCR* denotes type-consistent rate and *DVE* denotes differences in numbers of visual elements. *ave* and *std* denote average and standard deviation values, respectively. The best results when performing each top-k retrieval are in bold. The results show our structure-aware approach achieves the best performances among the four methods.

Top-k	Method	TCR_{ave}	TCR_{std}	DVE_{ave}	DVE_{std}
1	*V-HOG*	0.6597	0.4738	45.1266	555.2019
	V-CNN	0.7231	0.4474	17.1826	268.8787
	S-GNN	0.7372	0.4401	**0.1555**	**0.4888**
	S&V-Fusion	**0.7601**	**0.4270**	0.3674	1.6752
5	*V-HOG*	0.6074	0.3249	42.9230	253.5407
	V-CNN	0.6992	0.3150	24.4112	183.4934
	S-GNN	0.7058	0.3175	**0.2215**	**0.8603**
	S&V-Fusion	**0.7383**	**0.3096**	0.6145	3.3837
10	*V-HOG*	0.5877	0.2948	46.3089	195.6248
	V-CNN	0.6884	0.2921	26.3127	149.2130
	S-GNN	0.6893	0.2983	**0.2628**	**1.0391**
	S&V-Fusion	**0.7246**	**0.2900**	0.7909	4.2785
20	*V-HOG*	0.5614	0.2674	49.7751	163.7496
	V-CNN	0.6764	0.2785	25.1575	108.1764
	S-GNN	0.6713	0.2855	**0.3073**	**0.9769**
	S&V-Fusion	**0.7076**	**0.2809**	1.0432	5.1894

To further understand the pros and cons of different approaches, we investigated their TCR_{ave} values of each visualization type and show the average confusion matrix in five runs among all types in Figure 7. From the heatmaps, we can first notice that *V-HOG* is obviously worse than other approaches, probably due to its simplicity by nature. Thus, in the rest of our paper, we will mainly focus on the comparison among the other three approaches. *S&V-Fusion* is consistently better than other approaches, which demonstrates the advantages of our structure-aware approach. An interesting observation is that, though *S-GNN* and *V-CNN* have close performance in terms of the overall TCR_{ave}, they have clear differences in TCR_{ave} of single visualization types. *V-CNN* has an advantage on bar charts and box plots while *S-GNN* performs better on line charts and scatter plots. We checked the detailed results and speculate that the possible reasons are as below. First, visual elements in bar charts and box plots often occupy a large space in the visualization and their shapes are easy to be distinguished by *V-CNN*. However, lines and scatters are less obvious in visualizations and sometimes can be confused with each other, which may lead to the inaccurate perception by CNNs (see Figure 9(a)). Compared to *V-CNN*, *S-GNN* explicitly considers all the visual elements as nodes in the graph of visual elements. Thus, it performs better on line charts and scatter plots. The disadvantages of *S-GNN* are mainly shown by the results of box plots and bar charts. A possible reason is that *S-GNN* only describes visual elements roughly using their bounding boxes. Boxes, bars and scatters can be confused when their bounding boxes are similar.

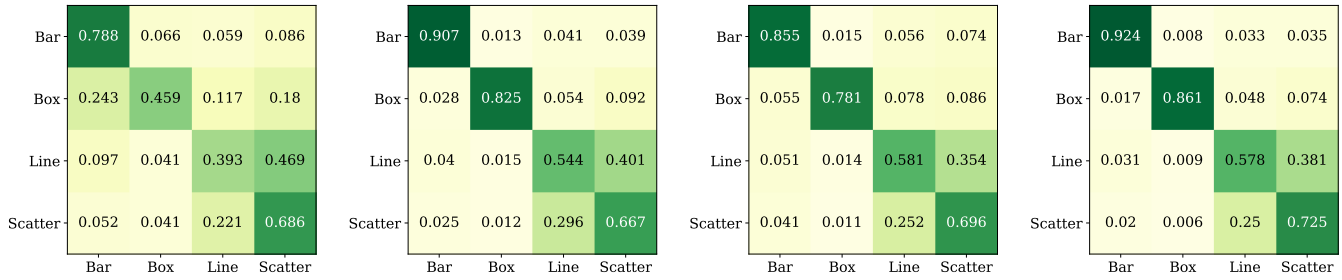
In summary, the results show that both *V-CNN* and *S-GNN* have their advantages when dealing with certain types of visualizations. Thus, the structural information in SVG-based visualizations is essential to be jointly considered with visual information to improve the performance of visualization retrieval. The results of *S&V-Fusion* in Figure 7(d) confirms its effectiveness compared to other approaches.

6.4 User Study

The effectiveness of our approach was further evaluated in a user study with 50 participants recruited from Prolific. We asked the participants to score the similarity between the query visualization and retrieved ones. In this section, we introduce the protocol and results of the user study.

Dataset. In the user study, we first randomly sampled 40 visualizations in the test set. For each sampled visualization, we retrieved the top-5 similar visualizations using the four approaches described in Section 6.2. Then each participant is presented with 5 query visualizations and the corresponding retrieved visualizations for scoring. We further added an attention check question to ask the participant to label the similarity between the same visualizations. The visualizations used in the user study are available through https://structure-vis-retrieval.github.io/.

Procedure. The user study contains two parts. Similar to the preliminary study, the first part was to introduce the procedure, collect consent from participants and test the visualization knowledge of the participants. After they passed the test, we moved to the second part. Specifically, we introduced the criteria for visualization similarity collected in our preliminary study and emphasized that *the*



(a) *V-HOG* **(b) *V-CNN*** **(c) *S-GNN*** **(d) *S&V-Fusion***

Figure 7: The heatmaps show the TCR_{ave} values of different chart types of four approaches when performing top-5 queries. The horizontal axis denotes the query visualization type and the vertical axis denotes the retrieved visualization type. A block with a darker color indicates a higher TCR_{ave}.

visualization type, *trend of data* and *number of visual elements* are of high priority based on the results of the preliminary study. We also showed several examples to illustrate these criteria for scoring the similarity between visualizations. The purpose of introducing these criteria and the corresponding examples is to calibrate the participants' judgment and eliminate the effect of extreme scores given by some participants. Then the participants were asked to apply these criteria to score the retrieved top-5 similar visualizations for each query visualization using a 5-point likert scale where 1 means the least similar and 5 means the most similar.

Results. The results of our user study are shown in Figure 8. We calculated the average similarity scores of retrieved visualizations by each method (*S&V-Fusion*: 2.8607, *V-CNN*: 2.7707, *S-GNN*: 2.4833 and *V-HOG*: 2.002). A higher score indicates that the retrieved visualizations are more similar to the query one. According to the results, *S&V-Fusion* outperforms all other models with statistical significance ($p < 0.001$) tested with Wilcoxon Signed-rank tests.

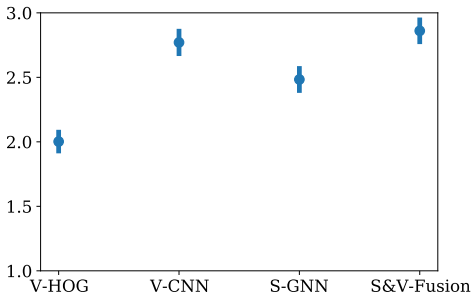


Figure 8: The average similarity scores of retrieved visualizations are shown with 95% confidence intervals. *S&V-Fusion* outperforms others with statistical significance ($p < 0.001$).

6.5 Case Study

In this section, we present some examples shown in the user study based on the three most important criteria for visualization similarity. These cases further illustrate the pros and cons of our structure-aware approach compared to others. Since *V-HOG* consistently performs worse than others, in this section, we mainly compare *S&V-Fusion*, *S-GNN*, and *V-CNN*.

Type of Visualizations. As Table 2 and Figure 7 show, *S&V-Fusion* and *S-GNN* have advantages in distinguishing the type of charts, especially between line charts and scatter plots. A case regarding this phenomenon is identified in our user study, as shown in Figure 9(a). The query visualization in this case is a line chart with an increasing trend. However, with the representation generated by *V-CNN*, a scatter plot with a similar trend is considered as a similar visualization (bounded with a red box in Figure 9(a)). Though they share some common features such as the trend of data, they should not be considered as similar visualizations according to the criteria of visualization types, which is identified in our preliminary study (see Section 3). Both *S&V-Fusion* and *S-GNN* do not have such kinds of mistakenly retrieved visualizations, which indicates that they can better maintain the visualization type consistency. Another case in Figure 9(b) also reflects a similar issue of *V-CNN*. *V-CNN* retrieved multiple stacked bar charts for a plain bar chart while the other two methods only returned similar plain bar charts.

Number of Visual Elements. The query visualization in Figure 9(a) is a line chart with three lines. *S-GNN* achieves the best results using this criterion since most of the retrieved visualizations have three lines. Most of the retrieved visualizations by *V-CNN* are with a single line, which fails to meet the criterion on the numbers of visual elements.

Trend of Data. Though *S-GNN* shows its advantages in distinguishing chart types and identifying the difference between numbers of visual elements, its capability of considering the trend of data is not as good as *V-CNN*. As both cases show, almost all the retrieved visualizations by *S-GNN* have trends that are different from the query visualization, which reflects its drawback in effectively representing the trend of data in visualizations. Thus, we considered both the visual and structural information in *S&V-Fusion* to mitigate such an issue. As both cases show, *S&V-Fusion* achieves a better performance in retrieving visualizations with similar trends while preserving the advantages in *the type of visualizations* and *the number of visual elements*.

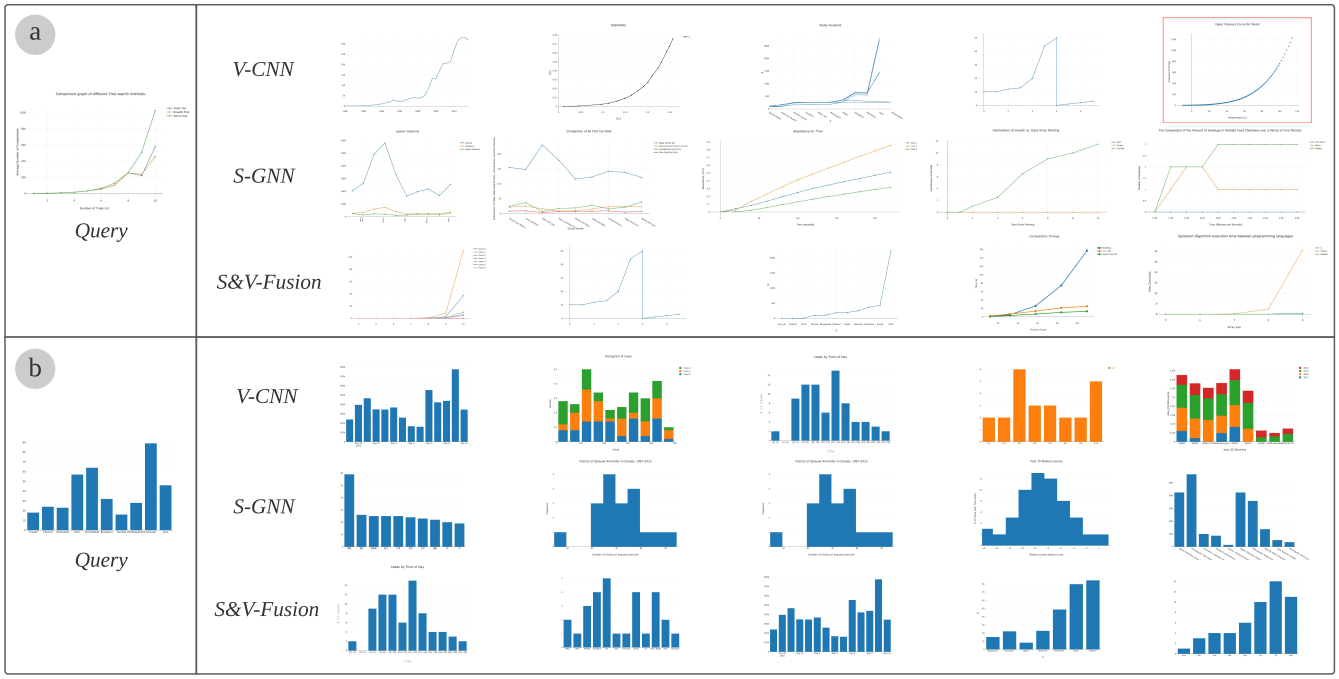


Figure 9: This figure shows examples of query and top-5 retrieved similar visualizations by *V-CNN, V-HOG* and *S&V-Fusion.*

7 DISCUSSION

In this section, we discuss the *lessons* learned in developing our structure-aware visualization retrieval methods, the *generalizability* of our approach, some potential *application scenarios* and the *limitations* of our approach.

7.1 Lessons

In our approach, we explicitly consider the structural information through deep learning techniques, which enables a new perspective of characterizing the similarity of visualizations. To facilitate future studies along this direction, we conclude two important lessons, the *necessity of structural information* in characterizing visualization similarity and *deep learning model customization* for visualizations.

Necessity of Structural Information. The similarity between visualizations can be measured through various aspects such as visualization types, colors and trends of data. Thus, to identify key criteria when determining similarity between visualizations, we conducted a preliminary study with general visualization users and summarized three most important criteria, *the visualization type*, *the trend of data* and *the number of visual elements*. These three criteria indicate the necessity of considering structural information in visualizations since they consider the similarity more at the level of visual elements instead of the level of pixels. Most of the prior studies merely treat visualizations as bitmap images and extract features based on pixels, which ignores the important structural information of visual elements. Compared with these approaches, our method shows a promising direction of explicitly considering the structural information in SVGs to characterize visualization similarity. We treat each visual element in the visualization as a node with basic features in the graph and use edges to represent their spatial and hierarchical relationships, which can effectively reflect the numbers and groups of visual elements. Since the structural information is mainly at the level of visual elements, we further leverage the pixel-level visual information to make the characterization of visualization more fine-grained. As our case studies in Section 6.5 show, *S&V-Fusion* achieves better performance in fulfilling the requirements of retrieving similar visualizations such as the numbers of visual elements and visualization types.

Deep Learning Model Customization. Deep learning models have been proved to be effective in many applications such as image processing and natural language understanding. There is also increasing interest in applying deep learning approaches in visualization-related tasks such as automatic visualization generation [19] and visualization similarity characterization [61]. In our approach, we also apply state-of-the-art contrastive learning models to generate embedding vectors of visualizations. However, we noticed that these general deep learning models may not be directly applied to visualizations. For example, as we mentioned in Section 5.2, flipping is a commonly used operation for data augmentation in general image processing. However, it is not suitable for visualizations since the trend of data is entirely altered. Thus, when applying these models, certain customizations should be carefully conducted. Furthermore, visualization-tailored deep learning models are also desired. For example, compared with shape features, the bias of CNN models towards texture features has been observed [13]. However, in visualizations, we focus more on the shapes of visual elements instead of the texture. Thus, a tailored

CNN model which emphasizes shape features might be more suitable for visualization-related tasks and can replace the ResNet-50 in our approach.

7.2 Generalizability and Application Scenarios

This section discusses the generalizability of our approach and some potential application scenarios of our method. The generalizability is from two perspectives: generalizability to visualizations created using other packages (e.g., D3) and generalizability to multi-view visualizations (e.g., dashboards or visual analytic systems).

Visualizations Created by Other Packages. In our evaluation, we used a visualization corpus crawled from Plotly. However, it is not restricted to visualizations created using Plotly and can be extended to visualizations created using other visualization packages. The requirements of leveraging our approach to retrieve visualizations include a *unified structure* of visualizations and the *consistent usage* of SVG elements in the visualizations. Here a unified structure of visualizations requires a consistent way of grouping visual elements. Our approach does not require a specific criterion of grouping visual elements as long as a unified way of grouping visual elements is applied, for example, grouping *<path>*s by data columns in Plotly. The consistent usage of SVG elements requires that the same type of SVG elements is employed to render the same type of visualizations. For example, all the bars in bar charts are plotted with *<path>*s.

Visualizations created using specification-based packages such as Vega-Lite [38] and Plotly usually fulfill the requirements of our approach. However, when dealing with the visualizations created with tools with more flexibility, such as Adobe Illustrator[8] and D3 [2], our approach may meet difficulties due to the inconsistent usage of SVG elements. For example, when creating visualizations with D3, bars can be created using *<path>*, *<rect>* or *<polygon>* in different visualizations. A potential future direction is to improve the generalizability of our structure-aware approach to handle visualizations created with various tools.

Visualizations with Multiple Views. Multi-view visualizations have been widely used to accommodate data with a huge number of attributes [6]. Along with its popularity, characterizing multi-view visualizations also attracts researchers' interest [6, 56]. Our approach also has the potential to capture the structural information of multi-view visualizations by considering more factors, such as the hierarchical and spatial relationship among views.

Application Scenarios. With the popularity of data visualizations, an emerging research direction is to treat them as a data format and to propose visualization-specific methods for storing, querying and analyzing enormous visualizations [55]. Our approach has the potential to enable various downstream applications towards this direction. First, our approach can provide a new way to perform the nearest neighbor query on stored visualizations based on their structural information. This can also boost the re-use of visualization codes since structural information is highly related to the implementation of a visualization. Second, a more effective visualization retrieval approach can enhance the large-scale analysis of visualizations. It allows users to group similar visualizations and conduct further analysis such as understanding general users' preferences on visual designs. Third, it can facilitate the construction of large-scale visualization corpora. Most of the existing visualization corpora only provide high-level labels of visualizations such as the visualization type [19] and color usage [59]. With our approach, fine-grained labels (e.g., the trend of data) of visualizations can be easily labeled based on similar visualization retrieval.

[8] https://www.adobe.com/products/illustrator.html

7.3 Limitations

The evaluations above have demonstrated the effectiveness of our approach, but it is not without limitations.

Evaluation. To extensively evaluate our approach, we have conducted a user study, quantitative comparisons and case studies. However, there are also limitations in our evaluation. Among the three most important criteria in evaluating the perceptual similarity of visualizations, We have designed two quantitative metrics to measure the performance of various visualization retrieval methods from the perspective of visualization type consistency and the difference of numbers of visual elements. However, we did not apply a quantitative metric to evaluate the consistency between trends of data in visualizations. The major reason is that the trends of data in visualizations are complex and difficult to be accurately quantified in a universal way. For example, in box plots, the trend of data can refer to either the distribution of data in each box or the difference of data between multiple boxes. Existing studies also failed to define a universal metric to quantitatively evaluate the trend of data of different types of visualizations (e.g., [23, 51, 53]). To mitigate the issue that the similarity between data trends in visualizations cannot be quantitatively evaluated, we further conducted the user study to verify our effectiveness by asking participants to explicitly consider trends of data when scoring the similarity between visualizations. Furthermore, as shown in our user study, though our methods outperform other approaches in terms of the 5-point likert scale, the score (i.e., 2.8607) is still not perfect. Also, the confusion between line charts and scatter plots can be further reduced. We suspect the reason is that some line charts are with circles on the data points (see Figure 9), which may affect the measurement of similarity. In the future, more research is expected to improve our structure-aware approach for visualization retrieval, for example, developing visualization-tailored models as mentioned in Section 7.1.

Feature Fusion. In our structure-aware approach for visualization retrieval, we represent the structural and visual information as two embedding vectors. According to our preliminary comparison with other approaches, we simply concatenate them as the final embedding vector of the visualization (see Section 5.3). However, this step is not without limitations. First, the concatenated embedding vector of a visualization will take additional space to store, which can lead to inefficiency. Second, the performance of simple concatenation may be limited due to some redundant information inside two embedding vectors, for example, colors. The redundant information can result in bias when calculating the similarity. To address the potential drawbacks, it is possible to fuse the visual and structural information based on additional information such as the interaction records used in Screen2Vec [26]. However, due to the lack of such visualization corpus, we have left it as our future work.

8 CONCLUSION

Scalable Vector Graphic (SVG)-based visualizations have been widely used and shared online. Along with their popularity, retrieving similar SVG-based visualizations has become a critical task. In this paper, we propose a structure-aware visualization retrieval approach based on the most common perceptual visualization similarity criteria surveyed in our pilot study. Despite the widely used visual information in prior studies, our approach further considers the often ignored but essential structural information to advance the performance. To consider both types of information, we convert SVGs to bitmaps for visual information extraction and graphs for structural information extraction. Then contrastive representation learning technique is employed to generate the low-dimensional embedding vectors of visual and structural information, respectively. The corresponding visualization is then represented by the concatenation of embedding vectors for visualization retrieval. We conducted extensive evaluations, including quantitative comparisons with prior approaches, a user study and multiple case studies, to demonstrate the effectiveness of our approach.

In future work, we would like to extend our structure-aware visualization retrieval approach to more visualization types, e.g., multi-view visualizations. Another interesting direction is to explore the possibility of directly extracting structural information from bitmap-based visualizations and investigate how our approach can work for the retrieval of bitmap-based visualizations.

ACKNOWLEDGMENTS

This research was supported by the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant (Grant number: 20-C220-SMU-011). Yong Wang is the corresponding author. We would like to thank Mr. Tianyuan Yao for constructive suggestions and anonymous reviewers for their feedback.

REFERENCES

[1] Leilani Battle, Peitong Duan, Zachery Miranda, Dana Mukusheva, Remco Chang, and Michael Stonebraker. 2018. Beagle: Automated Extraction and Interpretation of Visualizations from the Web. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–8.

[2] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D^3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2301–2309.

[3] Jiangxia Cao, Xixun Lin, Shu Guo, Luchen Liu, Tingwen Liu, and Bin Wang. 2021. Bipartite Graph Embedding via Mutual Information Maximization. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 635–643.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *Proceedings of the 37th International Conference on Machine Learning*. 1597–1607.

[5] Xinlei Chen and Kaiming He. 2021. Exploring Simple Siamese Representation Learning. In *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 15750–15758.

[6] Xi Chen, Wei Zeng, Yanna Lin, Hayder Mahdi AI-maneea, Jonathan Roberts, and Remco Chang. 2021. Composition and Configuration Patterns in Multiple-View Visualizations. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2021), 1514–1524.

[7] Zhe Chen, Michael Cafarella, and Eytan Adar. 2015. DiagramFlyer: A Search Engine for Data-Driven Diagrams. In *Proceedings of the 24th International Conference on World Wide Web*. 183–186.

[8] William S. Cleveland. 1979. Robust Locally Weighted Regression and Smoothing Scatterplots. *Journal of the American Statistical Association* 74, 368 (1979), 829–836.

[9] Navneet Dalal and Bill Triggs. 2005. Histograms of Oriented Gradients for Human Detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 1. 886–893.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A Large-scale Hierarchical Image Database. In *Proceedings of the 2009 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 248–255.

[11] Fangxiang Feng, Xiaojie Wang, and Ruifan Li. 2014. Cross-modal Retrieval with Correspondence Autoencoder. In *Proceedings of the 22nd ACM International Conference on Multimedia*. 7–16.

[12] Jiayun Fu, Bin Zhu, Weiwei Cui, Song Ge, Yun Wang, Haidong Zhang, He Huang, Yuanyuan Tang, Dongmei Zhang, and Xiaojing Ma. 2021. Chartem: Reviving Chart Images with Data Embedding. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2021), 337–346.

[13] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. 2018. ImageNet-trained CNNs are Biased towards Texture; Increasing Shape Bias Improves Accuracy and Robustness. *CoRR* abs/1811.12231 (2018).

[14] Theo Gevers and Arnold WM Smeulders. 2000. PicToSeek: Combining Color and Shape Invariant Features for Image Retrieval. *IEEE Transactions on Image Processing* 9, 1 (2000), 102–119.

[15] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. 2020. Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning. *CoRR* abs/2006.07733 (2020).

[16] Daniel Haehn, James Tompkin, and Hanspeter Pfister. 2019. Evaluating 'Graphical Perception' with CNNs. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2019), 641–650.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385 (2015).

[18] Enamul Hoque and Maneesh Agrawala. 2020. Searching the Visual Style and Structure of D3 Visualizations. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2020), 1236–1245.

[19] Kevin Zeng Hu, Michiel A. Bakker, Stephen Li, Tim Kraska, and César A. Hidalgo. 2019. VizML: A Machine Learning Approach to Visualization Recommendation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

[20] Anil K Jain and Aditya Vailaya. 1996. Image Retrieval Using Color and Shape. *Pattern Recognition* 29, 8 (1996), 1233–1244.

[21] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. 2020. A Survey on Contrastive Self-supervised Learning. *CoRR* abs/2011.00362 (2020).

[22] Daekyoung Jung, Wonjae Kim, Hyunjoo Song, Jeong-in Hwang, Bongshin Lee, Bohyoung Kim, and Jinwook Seo. 2017. ChartSense: Interactive Data Extraction from Chart Images. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 6706–6717.

[23] Hyeok Kim, Ryan Rossi, Abhraneel Sarma, Dominik Moritz, and Jessica Hullman. 2021. An Automated Approach to Reasoning About Task-Oriented Insights in Responsive Visualization. *CoRR* abs/2107.08141 (2021).

[24] Haotian Li, Yong Wang, Songheng Zhang, Yangqiu Song, and Huamin Qu. 2021. KG4Vis: A Knowledge Graph-Based Approach for Visualization Recommendation. *CoRR* abs/2107.12548 (2021).

[25] Haotian Li, Huan Wei, Yong Wang, Yangqiu Song, and Huamin Qu. 2020. Peer-inspired Student Performance Prediction in Interactive Online Question Pools with Graph Neural Network. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2589–2596.

[26] Toby Jia-Jun Li, Lindsay Popowski, Tom M. Mitchell, and Brad A. Myers. 2021. Screen2Vec: Semantic Embedding of GUI Screens and GUI Components. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.

[27] Zhuo Li, Sandra Carberry, Hui Fang, Kathleen F McCoy, Kelly Peterson, and Matthew Stagitis. 2015. A Novel Methodology for Retrieving Infographics Utilizing Structure and Message Content. *Data & Knowledge Engineering* 100 (2015), 191–210.

[28] Ying Liu, Dengsheng Zhang, Guojun Lu, and Wei-Ying Ma. 2007. A Survey of Content-based Image Retrieval with High-level Semantics. *Pattern Recognition* 40, 1 (2007), 262–282.

[29] Yuxin Ma, Anthony K. H. Tung, Wei Wang, Xiang Gao, Zhigeng Pan, and Wei Chen. 2020. ScatterNet: A Deep Subjective Similarity Model for Visual Analysis of Scatterplots. *IEEE Transactions on Visualization and Computer Graphics* 26, 3 (2020), 1562–1576.

[30] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. 2011. Multimodal Deep Learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*. 689–696.

[31] Michael Oppermann, Robert Kincaid, and Tamara Munzner. 2021. VizCommender: Computing Text-Based Similarity in Visualization Repositories for Content-Based Recommendations. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2021), 495–505.

[32] Michael Oppermann and Tamara Munzner. 2021. VizSnippets: Compressing Visualization Bundles Into Representative Previews for Browsing Visualization Collections. *IEEE Transactions on Visualization and Computer Graphics* (2021).

[33] Akshay Gadi Patil, Manyi Li, Matthew Fisher, Manolis Savva, and Hao Zhang. 2021. LayoutGMN: Neural Graph Matching for Structural Layout Similarity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11048–11057.

[34] Chunyao Qian, Shizhao Sun, Weiwei Cui, Jian-Guang Lou, Haidong Zhang, and Dongmei Zhang. 2020. Retrieve-Then-Adapt: Example-based Automatic Generation for Proportion-related Infographics. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2020), 443–452.

[35] Mohammad Raji, Jeremiah Duncan, Tanner Hobson, and Jian Huang. 2021. Dataless Sharing of Interactive Visualization. *IEEE Transactions on Visualization and Computer Graphics* 27, 9 (2021), 3656–3669.

[36] Babak Saleh, Mira Dontcheva, Aaron Hertzmann, and Zhicheng Liu. 2015. Learning Style Similarity for Searching Infographics. In *Proceedings of the 41st Graphics Interface Conference*. 59–64.

[37] Arvind Satyanarayan, Bongshin Lee, Donghao Ren, Jeffrey Heer, John T. Stasko, John Thompson, Matthew Brehmer, and Zhicheng Liu. 2020. Critical Reflections on Visualization Authoring Systems. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2020), 461–471.

[38] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2017. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017), 341–350.

[39] Manolis Savva, Nicholas Kong, Arti Chhajta, Li Fei-Fei, Maneesh Agrawala, and Jeffrey Heer. 2011. ReVision: Automated Classification, Analysis and Redesign of Chart Images. In *Proceedings of the 24th ACM Symposium on User Interface Software and Technology*. 393–402.

[40] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In *Proceedings of the 2018 European Semantic Web Conference*. 593–607.

[41] Cordelia Schmid and Roger Mohr. 1997. Local Grayvalue Invariants for Image Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19, 5 (1997), 530–535.

[42] Tarique Siddiqui, Albert Kim, John Lee, Karrie Karahalios, and Aditya Parameswaran. 2016. Effortless Data Exploration with zenvisage: An Expressive and Interactive Visual Analytics System. *Proceedings of the VLDB Endowment* 10, 4 (2016).

[43] Tarique Siddiqui, Paul Luh, Zesheng Wang, Karrie Karahalios, and Aditya Parameswaran. 2020. ShapeSearch: A Flexible and Efficient System for Shape-based Exploration of Trendlines. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 51–65.

[44] Noah Siegel, Zachary Horvitz, Roie Levin, Santosh Divvala, and Ali Farhadi. 2016. FigureSeer: Parsing Result-Figures in Research Papers. In *Proceedings of the 14th European Conference on Computer Vision*. 664–680.

[45] Holger Stitz, Samuel Gratzl, Harald Piringer, Thomas Zichner, and Marc Streit. 2018. KnowledgePearls: Provenance-Based Visualization Retrieval. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2018), 120–130.

[46] Fan-Yun Sun, Jordan Hoffman, Vikas Verma, and Jian Tang. 2019. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *Proceedings of the 10th International Conference on Learning Representations*.

[47] Gleb Tkachev, Steffen Frey, and Thomas Ertl. 2021. S4: Self-Supervised learning of Spatiotemporal Similarity. *IEEE Transactions on Visualization and Computer Graphics* (2021).

[48] Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, and Neoklis Polyzotis. 2015. SeeDB: Efficient Data-Driven Visualization Recommendations to Support Visual Analytics. *Proceedings of the VLDB Endowment* 8, 13 (2015), 2182–2193.

[49] Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2018. Deep Graph Infomax. *CoRR* abs/1809.10341 (2018).

[50] Qianwen Wang, Zhutian Chen, Yong Wang, and Huamin Qu. 2021. A Survey on ML4VIS: Applying Machine Learning Advances to Data Visualization. *IEEE Transactions on Visualization and Computer Graphics* (2021).

[51] Yunhai Wang, Fubo Han, Lifeng Zhu, Oliver Deussen, and Baoquan Chen. 2017. Line Graph or Scatter Plot? Automatic Selection of Methods for Visualizing Trends in Time Series. *IEEE Transactions on Visualization and Computer graphics* 24, 2 (2017), 1141–1154.

[52] Yong Wang, Zhihua Jin, Qianwen Wang, Weiwei Cui, Tengfei Ma, and Huamin Qu. 2019. DeepDrawing: A Deep Learning Approach to Graph Drawing. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2019), 676–686.

[53] Leland Wilkinson, Anushka Anand, and Robert Grossman. 2005. Graph-theoretic Scagnostics. In *Proceedings of the 2005 IEEE Symposium on Information Visualization*. 21–21.

[54] Aoyu Wu, Wai Tong, Tim Dwyer, Bongshin Lee, Petra Isenberg, and Huamin Qu. 2020. MobileVisFixer: Tailoring Web Visualizations for Mobile Phones Leveraging an Explainable Reinforcement Learning Framework. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2020), 464–474.

[55] Aoyu Wu, Yun Wang, Xinhuan Shu, Dominik Moritz, Weiwei Cui, Haidong Zhang, Dongmei Zhang, and Huamin Qu. 2021. Survey on Artificial Intelligence Approaches for Visualization Data. *CoRR* abs/2102.01330 (2021).

[56] Aoyu Wu, Yun Wang, Mengyu Zhou, Xinyi He, Haidong Zhang, Huamin Qu, and Dongmei Zhang. 2021. MultiVision: Designing Analytical Dashboards with Deep Learning Based Recommendation. *CoRR* abs/2107.07823 (2021).

[57] Lirong Wu, Haitao Lin, Cheng Tan, Zhangyang Gao, and Stan Z Li. 2021. Self-supervised Learning on Graphs: Contrastive, Generative, or Predictive. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[58] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How Powerful are Graph Neural Networks? *CoRR* abs/1810.00826 (2018).

[59] Linping Yuan, Wei Zeng, Siwei Fu, Zhiliang Zeng, Haotian Li, Chi-Wing Fu, and Huamin Qu. 2021. Deep Colormap Extraction from Visualizations. *CoRR* abs/2103.00741 (2021).

[60] Peiying Zhang, Chenhui Li, and Changbo Wang. 2021. VisCode: Embedding Information in Visualization Images using Encoder-Decoder Network. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2021), 326–336.

[61] Tianye Zhang, Haozhe Feng, Wei Chen, Zexian Chen, Wenting Zheng, Xiao-Nan Luo, Wenqi Huang, and Anthony KH Tung. 2021. ChartNavigator: An Interactive Pattern Identification and Annotation Framework for Charts. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[62] Jian Zhao, Mingming Fan, and Mi Feng. 2020. ChartSeer: Interactive Steering Exploratory Visual Analysis with Machine Intelligence. *IEEE Transactions on Visualization and Computer Graphics* (2020).

[63] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph Neural Networks: A Review of Methods and Applications. *AI Open* 1 (2020), 57–81.

A EFFICIENCY

To evaluate the efficiency of our structure-aware approach, we conducted preliminary experiments on a server with two Intel Xeon Silver 4210R processors, two Nvidia Quadro RTX 6000 GPU Cards and 128G memory. First, we tested the speed of generating the embedding. The average time is 1.4052 ms for each visualization (CNN for visual information: 1.2027 ms, GNN for structural information: 0.2025 ms). Then we ran experiments to identify the relationship between the retrieval time and the size of the corpus after generating embedding vectors. The results are shown in Table 3. We can notice that the retrieval time is almost growing linearly with respect to the size of the corpus (with Pearson correlation coefficient = 0.9998) and is not correlated with the number of returned visualizations. When there are 50000 visualizations with stored embedding vectors in the corpus, the time cost of returning the top-5 similar visualizations of a new visualization is about 6.1082 ms, which is satisfactory. In the future, more experiments can be conducted with a larger corpus to further evaluate the efficiency of our structure-aware approach. Also, how to reduce the time of generating embedding vectors and retrieving similar visualizations should be explored.

Table 3: This table shows the average time (in *ms*) of retrieving the top similar visualizations from a corpus with our structure-aware approach. The values are the average values of five runs. The horizontal axis represents the numbers of visualizations in a corpus and the vertical axis represents the top-k query.

	1000	5000	10000	50000
Top-1	0.0642	0.4084	0.8252	4.7140
Top-5	0.0653	0.4108	0.8251	4.7030
Top-10	0.0657	0.4067	0.8326	4.7123
Top-20	0.0656	0.4011	0.8282	4.7124

B FEATURES

In this section, we list and discuss the features used in our structure-aware approach.

List of features. Table 4 shows the features in four groups and the corresponding SVG element types. The SVG element types include *open* <*path*>, *closed* <*path*>, <*text*> and <*g*>/<*svg*>. A *closed* <*path*> contains at least a path that starts and ends at the same point. On the other hand, an *open* <*path*> does not have a path that starts and ends at the same point. As introduced in Section 5.1, the motivation of distinguishing different types of paths is to design various features to delineate the shapes of visual elements in visualizations accurately. Specifically, *closed* <*path*>s likely appear in box plots, bar charts and scatter plots while *open* <*path*>s are likely in line charts. Thus, considering the characteristics of these visualization types, we introduce the *trend* features for open <*path*>s since the *width* and *length* of the bounding box are not sufficient to reflect the shape of lines. On the other hand, since closed <*path*>s have the same starting and ending point, it is not suitable to describe them using *trend*.

Importance of features. We discuss the importance of features in our approach. The groups of features in Table 4 are designed to match the criteria of visualization similarity collected in our preliminary study. Thus, the most important group of features among all should be the group of *position* features and the least important group of features should be *style* features. Furthermore, according to the empirical observation during the development of our structure-aware approach, we summarize two findings. First, if we only keep *Type* features, the performance will be worse than keeping all features. Thus, the other features are helpful when delineating the similarity of visualizations. Second, the feature of trend can improve the estimation of the similarity of line charts, which matches our motivation of applying LOESS regression to generate the *trend* features. In the future, it will be helpful to conduct quantitative ablation studies for an in-depth understanding of the importance of features.

Limitation of features. The limitations of the features in our paper are reflected. The first limitation of current features is the lack of consideration for complex visual elements. Due to our scope of the standard statistical charts (i.e., box plot, bar chart, line chart and scatter plot), we only consider the potential SVG elements in these charts and propose the corresponding features. However, in practice, there can be SVG elements that represent complex visual elements, for example, curved <*path*>s and <*ellipse*>. Another limitation is that the features are limited to the SVG elements used in Plotly. Different packages may use different SVG elements to draw the same type of charts, for example, Vega-Lite and Plotly use <*path*>s to draw bar charts while D3 uses <*rect*>s. A potential method to handle this issue is to convert such elements, for example, <*rect*>s and <*polygon*>s, to <*path*>s when constructing the graph of visual elements. In future work, it is worth more exploration to improve the generalizability of features to fit complex visual elements and different representations of the same visual element.

C ROBUSTNESS

As introduced in Section 6, the results of our quantitative evaluation and the user study show that our approach outperforms other methods consistently. However, our evaluation lacks a comprehensive evaluation on the robustness of our structure-aware visualization retrieval approach towards minor changes of visualizations. The major challenge that hinders us from the evaluation of robustness is the lack of a universal definition of the robustness of visualization retrieval. We also investigated several widely-recognized studies [14, 20, 41] and a survey [28] on image retrieval to understand how robustness is examined for image retrieval methods. Among all the evaluations towards robustness in these papers, only the evaluation on the effect of image size changes is suitable for visualizations. Other evaluations can hardly be applied to visualization retrieval (e.g., the effect of occlusion, input noise and viewpoint changes). In our study, we have scaled the *size* and *position* features of visual elements according to the size of the entire SVG instead of using pixel numbers directly. The usage of relative scales makes the proposed method more robust to different sizes of visualizations. In future work, more studies should be conducted to deepen the understanding of the robustness of visualization retrieval methods.

Table 4: This table summarizes the features which characterize the SVG elements. The colored blocks represent that the feature is valid for the element type. A *closed <path>* contains one or more paths that start and end at the same point. An *open <path>* has to start and end at different points and does not contain any closed *<path>*.

Category	Feature	Closed *<path>*	Open *<path>*	*<text>*	*<g>*/ *<svg>*	Details
Type	Element type	■	■	■	■	The type of the element
Style	Stroke color	■	■	■		The stroke color of the element in RGB
	Stroke width	■	■	■		The stroke width of the element
	Stroke opacity	■	■	■		The stroke opacity of the element
	Fill color	■	■	■		The fill color of the element in RGB
	Fill opacity	■	■	■		The fill opacity of the element
Shape	Area	■	■			The area of the element
	Width	■				The width of the element's bounding box
	Length	■				The length of the element's bounding box
	Vertex Number		■			The number of the element's vertices
	Trend		■			The trend of an open *<path>*
	Length			■		The number of characters
Position	Center	■	■	■		The center position of the element's bounding box
	Delta	■	■			The difference in center positions of two neighbor elements