GraphDecoder: Recovering Diverse Network Graphs from Visualization Images via Attention-Aware Learning

Sicheng Song, Chenhui Li, Dong Li, Juntong Chen, and Changbo Wang

Abstract—DNGs are diverse network graphs with texts and different styles of nodes and edges, including mind maps, modeling graphs, and flowcharts. They are high-level visualizations that are easy for humans to understand but difficult for machines. Inspired by the process of human perception of graphs, we propose a method called GraphDecoder to extract data from raster images. Given a raster image, we extract the content based on a neural network. We built a semantic segmentation network based on U-Net. We increase the attention mechanism module, simplify the network model, and design a specific loss function to improve the model's ability to extract graph data. After this semantic segmentation network, we can extract the data of all nodes and edges. We then combine these data to obtain the topological relationship of the entire DNG. We also provide an interactive interface for users to redesign the DNGs. We verify the effectiveness of our method by evaluations and user studies on datasets collected on the internet and generated datasets.

Index Terms—Information visualization, Chart mining, Semantic segmentation, Network graph, Attention mechanism

1 Introduction

DNGs (diverse network graphs) are common visualizations in various applications. They have different styles of nodes and edges than the general definition of node-link graphs [1]. Their nodes can be hollow, may contain text, and exhibit various shapes. Their edges can be curves and polylines instead of straight lines only. DNGs contain node-link graphs, flowcharts [2], E-R diagrams [3], and mind maps [4]. Fig. 1 shows examples of DNGs. People often try to modify these DNGs because of poor design or outdated data. However, most visualizations are stored in the form of raster images and released on various media [5]. Moreover, in the visualization pipeline, designers often draw sketches on paper before the visualization is formally drawn [1]. If users can convert sketches into visualizations, productivity can be increased. In these scenarios, the core problem is to extract the original data of graph visualizations. Obtaining original data is a complicated task that has become an important study of visualization [6].

There are already some existing methods to solve the problem of obtaining original data. Some previous studies [7, 8] used steganography to write original data into images, but this method hides the data in the image: the data cannot reproduce the original data from the existing chart image without a steganography model. The most common method is chart data extraction, including image processing [9] and machine learning [10]. Chart extraction focuses on extracting the original data from the raster image of the chart. At present, data extraction approaches for basic charts such as bar [11, 12] and pie charts [13, 14] are mature. There are also some methods [15, 16] to support line charts, but they are not robust enough against multiple lines or dotted lines. It is difficult to detect lines in the chart. OGR [17] first proposed using morphological methods for the extraction of network graphs. However, morphological methods need to constantly adjust the threshold of binarization. In addition, such methods can only extract solid circular nodes and cannot solve graphs with text. OGER [18] improved the edge recognition module of OGR. VividGraph [1] explored extracting data from network graphs with deep neural networks, but this method has many limitations. The graph extracted in the report was a special kind of DNG: a solid circular node-link diagram. The method only worked for graphs with straight edges and circular nodes but no text. DNGs (diverse network graphs) are more complicated than the network graphs defined in OGER and VividGraph. There are many types of DNGs [5, 19], including E-R diagrams of database design, flowcharts, mind maps, etc.

We present a novel method called **GraphDecoder** to extract data from DNGs. First, we build a deep neural network based on U-Net [20] and the attention mechanism. We design the backbone and loss function according to the characteristics of the chart and add a module with an attention mechanism to the decoder, which improves the robustness of the network. Second, we learn to imitate the process of human perception of a DNG and analyze the extracted data to obtain

• *The authors are with the School of Computer Science and Technology, East China Normal University, Shanghai 200062, China. E-mail: {scsong, dongli, jtchen}@stu.ecnu.edu.cn, {chli, cbwang}@cs.ecnu.edu.cn.*
This work was supported by the NSFC under Grant 62072183.
(Corresponding authors: Chenhui Li and Changbo Wang.)

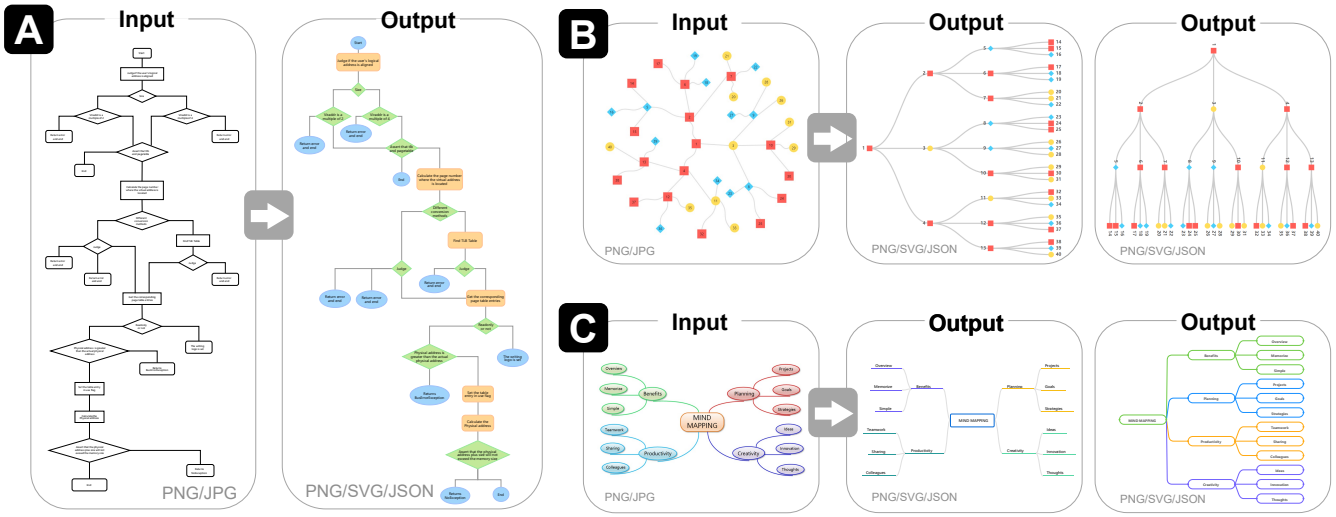


Fig. 1. The graph decoder can extract DNG data from raster images and automatically retarget them. Our method can be applied to many DNGs, including flowcharts (A), hierarchical diagrams (B), model graphs, hand-drawn sketches, and mind maps (C).

network relational data. Our method is robust to the edges of polylines and curves and supports three types of nodes: rectangles, diamonds, and ellipses. We also provide users with an interactive system. After the user uploads a raster image, the system extracts its underlying data. Users can redesign and modify data on the system interface. The system can be applied in many scenarios, such as mind maps, flowcharts, E-R diagrams, and hierarchical structure diagrams.

We perform ablation experiments on our semantic segmentation model; the results show that our neural network greatly improves the ability to extract edges from graphs. We also have applied structural and perceptual similarity evaluations on the real corpus. After redrawing the extracted results with AntV [21], we compare them with the input images. We use NetSimile [22] to evaluate the structural similarity and the visual saliency map [23] to evaluate the perceptual similarity. We established an additional corpus with different resolutions and scales of DNGs to test the robustness of our method for different scales of network structure and pixels. Our experimental results and user study show that our method offers great application potential in the redesign and modification of DNGs. We share differences between deep learning and heuristics. We also illustrate the experience of visualization image segmentation compared with natural images. Finally, we discuss the limitations of this paper and future work. Some of the limitations introduced by data-driven models can be explored in correcting the chart norm. Our contributions include three aspects:

(1) We defined the problem of data extraction from DNGs. We verified the practical importance of this study in various application domains.
(2) We designed a state-of-the-art deep learning model and a pipeline to extract DNGs.
(3) We put forward multiple evaluations to demonstrate that our method is effective and robust.

2 Related Work

Our work is mainly related to three aspects: chart data extraction, chart redesign, and attention mechanism.

2.1 Chart Data Extraction

Most visualization charts are represented as static images. The purpose of chart data extraction is to extract the original data from the bitmap. Different from some figure extraction works [24] which focus on locating the position of the image in the paper and figure classification, chart data extraction focuses on extracting original data from figures. Chart data extraction helps designers redesign and modify charts. There are many studies on chart data extraction for simple charts such as bar charts, line charts, and pie charts. For bar graphs, most methods [9, 25, 26, 27] use the analysis of connected components to extract bars in the charts. These methods are only effective for solid single-color bars. For pie charts, the existing methods [9, 13, 25, 28, 29] are based on the assumption of a solid pie chart without a three-dimensional effect. These methods mainly detect each slice by analyzing connected components [13] or deep learning methods [28, 29]. Other methods [9, 25] use curve fitting to locate each pie slice. Few methods support line charts because of the difficulty of extracting lines. Some methods [30, 31] can only extract line charts with a single line. Some recent studies [6, 28, 29] used deep neural networks for object detection to detect bars, showing better performance than traditional image processing. The latest methods use deep neural

networks to learn marked patches in the chart [15] or detect key points in the chart [16], but they are still not robust enough for multiple lines and lines of different thicknesses. Mao et al. [32] proposed a method of extracting data from meteorological facsimile charts. Poco et al. [33] focused on heatmaps with legends. Wu et al. [34] introduced a method to handle contour maps.

There are also some chart extraction frameworks and tools that are semiautomatic, such as ChartSense [14], DataThief [35], iVoLVER [36], Plot Digitizer [37], Dagra [38] and Engauge Digitizer [39]. These tools use human-supplied data to improve extraction accuracy, including chart types, key point positions, and line positions. There are also some extraction works for specific charts, such as ChemGrapher [40], a framework for extracting compounds. Some works [41, 42] are focused on science textbooks. The difference between science textbooks and DNGs is that their objects are the targets of natural images, not visualization charts. Chen et al. proposed a model [43] to extract the timeline. They proposed the GrabCut [44] method to improve the segmentation results. For the chart types above, DNG has more attributes. In addition to the structural attribute (topological relationship), DNGs also have other visual attributes (e.g., color, node shape, node size, and location of the nodes).

The type of visualization most relevant to DNGs is the network graph. OGR [17] used morphological methods to extract network graphs. However, its network graph definition has many limitations, such as solid circle nodes and charts without text. These methods require manual adjustment of the binarization threshold and weak anti-noise ability. Users need to adjust the thresholds of binarization and morphological operations, so the robustness ability of the method is weak. OGER [18] improved the edge detection ability of OGR and can recognize dashed lines. However, it still has the same limitations as OGR. Some studies [45, 46] focused on graphs that are close to our definition of DNGs, but their input was a set of stroke vectors. They included more prior knowledge than static bitmaps. VividGraph [1] used U-Net to segment network graph images. However, due to the limitation of their training data and untuned CNN (convolutional neural network) architectures, VividGraph only works for graphs with circular nodes and straight edges but no text. Compared with their work, we design our model and semantic parsing module for wider application and higher accuracy, as shown in Sec. 7.

Optimizing CNNs for feature extraction is an important step in using deep learning methods. Haehn et al. [47] proposed the possibility of using CNNs to extract data from charts. They evaluated the performance of four models, including MLP, LeNet [48], VGG [49], and Xception [50]. Giovannangeli et al. [51] further corroborated this view. However, their methods are not suitable for high-dimensional data such as DNGs. The output value of these methods is one-dimensional (e.g., the length of one line, the area of one circle, the angle between two sides [47], or the number of nodes and edges [51]). However, our task needs to output the topological relationship of N^2, where N represents the number of nodes. There are also many visual attributes (e.g., color map, node shape, node position). After preliminary experiments, we find that the semantic segmentation method is more promising. Neural networks for semantic segmentation mainly include FCN [52], U-Net [20], and SegNet [53]. These models have achieved good results in the task of segmentation of natural scenes. However, for chart images, the segmentation task requires higher accuracy because tiny pixel errors may result in large differences. Wang et al. [54] proposed the application of the attention mechanism in computer vision to make the model learn the areas that are important to reduce errors. SENet [55] was the first to propose the channel attention mechanism of the SE module from the channelwise level, which can adaptively adjust the characteristic response value of each channel. Oktay et al. [56] designed a module with attention, which prevented target discontinuity in medical images, such as those of the pancreas, and achieved better results. Zhou [11] used an encoder-decoder network with an attention mechanism to complete the task of extracting bar graphs. In this paper, we experimentally find that the attention mechanism applied to medical image segmentation can successfully improve the segmentation of DNG images. We also discuss the difference between natural images and visualization images in Sec. 8.

2.2 Chart Redesign

There are many charts in the real world that do not conform to the principles of visualization design, and chart extraction enhances these visualizations. [33] We mainly discuss three principles related to the design of DNGs. First, color plays an important role in chart design. Wang et al. [57] applied a knowledge-based model for learning color rules to visualize images. The W3C standard [58] also specifies color rules for images on web pages. The main purpose of some chart mining work [33] is to recolor the chart. Second, different representations of DNGs will also bring different perceptions to users. The entity relationship diagram [3] has been widely approved in the field of database design, using three types of nodes and solid lines to represent data relationships. E-R diagrams help database designers build databases faster. Buzan et al. [4] first proposed the mind map, which helps users efficiently express their divergent thinking. Flow charts [2] can clearly demonstrate algorithms in computer programming. Our system provides different retargeting schemes for different kinds of DNGs. Third, obtaining original data allows further visualization past the raster image. Interactive DNGs can ensure better user experience. There are some other types of chart mining work [59, 60, 61] that

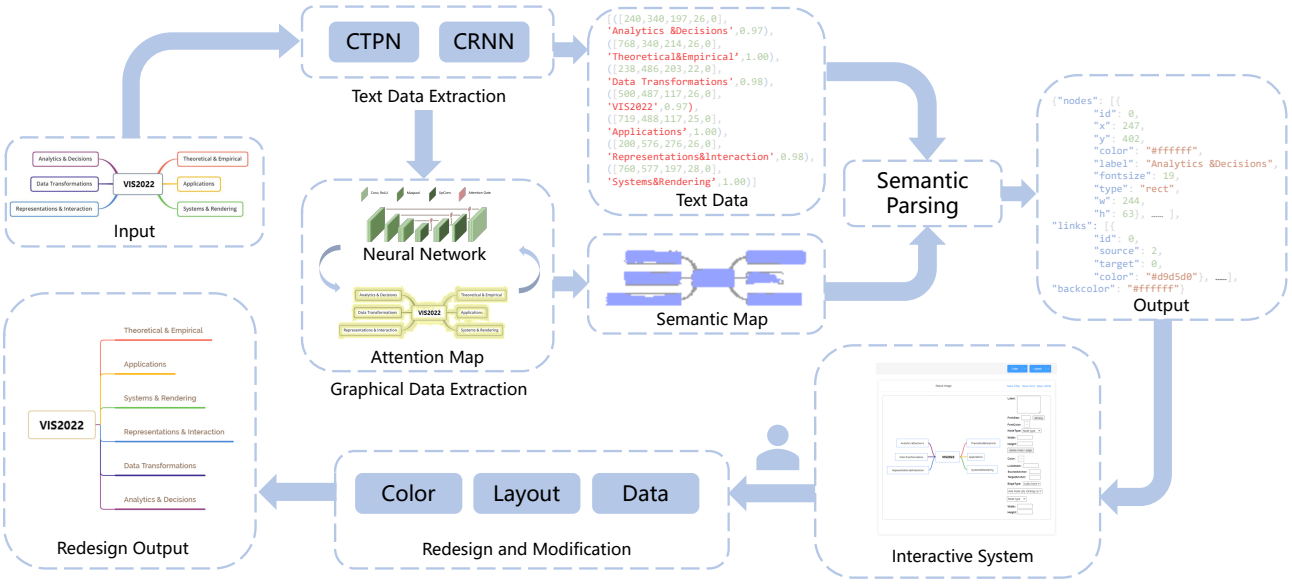


Fig. 2. The main modules and processes of GraphDecoder.

converted raster charts into vector diagrams to make the charts interactive, such as intelligent question and answer, adding table notes, and adding auxiliary lines.

In this work, we provide methods and pipelines for extracting data from DNGs and corresponding interfaces and applications for automatic redesign. However, we do not contribute the design principles of DNGs, and we assume that the target design is prior knowledge.

3 Overview

Our goal is to extract the original data from DNGs. The traditional methods [1, 18] are only suitable for certain simple graphs. DNGs have more data attributes, complex data types, difficult edge recognition, and matching problems between text and graphics. To solve these problems, we use the OCR system to preprocess raster images, use a semantic segmentation network with an attention module to locate and classify each pixel, and finally use the semantic parsing module to recover the DNG.

We propose a framework named GraphDecoder, which automatically extracts original data from DNG images. Fig. 2 shows the pipeline of our framework. The framework includes three components:

- *Text Detection Module*. To improve the performance of semantic segmentation, we first extract the text data in the chart. Through the OCR system, we obtain the context and position of the text. We remove the text area in the image and fill it with color blocks. Then, we can obtain the DNG image without text.
- *Segmentation Neural Network*. The segmentation neural network is the core module of our framework. We constructed a semantic segmentation network with an attention mechanism. This network can accurately locate pixels at which the nodes and edges are located and classify various types of nodes. By adding attention modules and improving the objective function, the network is robust to continuous curves and polylines.
- *Semantic Parsing Module*. By analyzing the connected components of the data obtained in the previous two modules, we obtain the approximate original data.

4 Methods

4.1 Dataset

Graph data extraction requires a large number of training datasets, but most works usually use small-scale datasets and keep the data private [6]. This is because most datasets require manual labeling [62], which requires high-quality control. The more advanced the visual coding, the more difficult the annotation. To the best of our knowledge, there is no existing DNG dataset for data extraction. Therefore, we built our dataset using the most popular visualization tools: D3 Library [63] in JavaScript and Matplotlib [64] and Skimage [65] in Python. The height and width of each image are defined as H and W, where $H, W \in [320, 800]$. To improve the robustness of the model, we also enhanced our dataset, including cropping, stretching, adding noise, etc. Finally, we selected $12,000$ images as the training set, $4,000$ images as the validation set, and $4,000$ images as the test set. Python and JS each draw one-third of the images, and the remaining are data-augmented images.

We set the number of nodes at $[0, 30]$, the thickness of edges at $[1, 10]$, and the number of edges at $[0, 50]$. The colors of the elements in the graphs are all random between $(0, 0, 0)$ and $(255, 255, 255)$. The height and weight of nodes are random between 5 and 100. We assume that the types of nodes are rectangles, ellipses and diamonds, and the types of edges are straight lines, polylines and curves. All random attributes are evenly distributed, ensuring that the training set covers most styles of DNGs. Some samples of our training dataset are shown in the appendix.

4.2 Text Detection Module

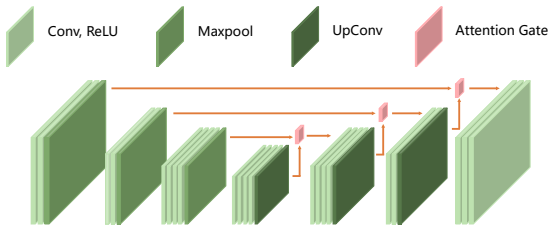
In the text detection module, we use CTPN [66] to locate our textual area. CTPN combines a CNN with an LSTM network to effectively detect horizontally distributed text in complex scenes. CTPN uses a vertical anchor regression mechanism to improve the performance of detecting small-scale text proposals. Different from the text detection in natural images, most of the texts in the chart originate in digital form. CTPN considers the characteristics of this type of text compared with Faster-RCNN [67]. The width of the vertical anchor is fixed at 16 pixels. The height varies from 11 pixels to 273 pixels (divided by 0.7 each time) for a total of 10 anchors.

After obtaining the text area, we use CRNN [68] to recover the text content. Inspired by Bosechen [69], we rotate each image by three angles, including $90°$, $180°$ and $270°$. Then, we perform text extraction to obtain the most accurate results. In the end, we obtain a list of texts: $\{t_x, t_y, t_w, t_h, t_a, Text, Confidence\}$, where t_x, t_y, t_w, t_h, t_a and $Text$ are the center coordinates, width, height, angle and content of the text. $Confidence$ is the confidence of the text. We consider the text to be effective only when the confidence is greater than 0.95. By this method, we can suppress the background noise in the chart.

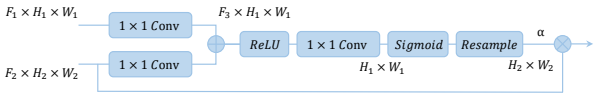
We also preprocess the input image based on the text data, including removing the pixels in the text area and filling it with pixel values around the text. This method reduces the unnecessary noise generated by the text in the graphics data in the semantic segmentation network. We perform the expansion processing of $kernel = (2, 2)$ on the image, which solves the problem of edge pixel breakage when the edge width is 1. We input the preprocessed image into the segmentation neural network for the next step of extracting graphics data.

4.3 Segmentation Neural Network

Compared with the network graphs defined in past work [1], DNG introduces hollow nodes with more shapes and nonstraight edges. This imposes greater challenges to the segmentation model. We show some examples of segmentation results from past models trained on our dataset in Fig. 4(a). First, when multiple polylines enclose a rectangular area, the model

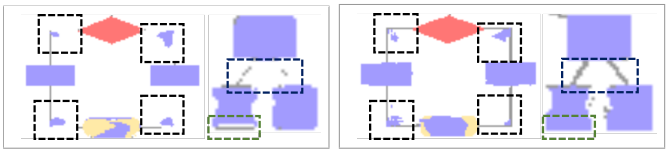


(a) Segmentation Neural Network



(b) Attention Gate

Fig. 3. Our segmentation neural network structure is based on the U-Net network, with the addition of the attention mechanism module and the modification of the loss function and the backbone network of feature extraction.



(a) VividGraph [1] (b) Ours

Fig. 4. Examples of segmentation results for different models. The challenges of model segmentation brought by DNG are shown in the box.

incorrectly identifies the polyline area as a rectangular hollow node, as shown in the black box. Second, the borders of hollow nodes are misidentified as edges, as shown in the green box. Third, the edges are difficult to recognize. The broken pixel results are shown as a blue box. Inspired by pancreas segmentation [56] in medical images, we improve the model so that it can more accurately segment the above situation, as shown in Fig. 4(b). The pancreas has the characteristics of changeable shape and area, and the phenomena of oversegmentation and undersegmentation often occur [56]. The pancreas is an organ that changes in shape and size with body health. Oversegmentation refers to the segmentation of organs that do not belong to the pancreas into the pancreatic area. Undersegmentation represents insufficient segmentation of the pancreas area. This is similar to the situation in which DNG nodes and edges have variable shapes and sizes, and the segmentation results are often discontinuous.

First, we found that noisy pixels causing targets to be discontinuous were misidentified as the background class. Most of the visualization images include more background pixels than object pixels, and we expect the model to focus more on visual codes rather than the background. We add an attention gate [56] to the

network, a module that learns the target structures. As shown in Fig. 3, $F_1 \times H_1 \times W_1$ is the feature map output by the deeper convolutional layer. $F_2 \times H_2 \times W_2$ is the lower-level feature map. F, H, and W represent features, height, and width, respectively. After the operation shown in Fig. 3(b), we obtain the attention coefficient $\alpha, \alpha \in [0, 1]$. We multiply it with the lower-level feature map and output it to the upper decoder. α is defined as follows:

$$\alpha_i^l = \sigma_2\left(\psi^T\left(\sigma_1\left(W_x^T x_i^l + W_g^T g_i + b_g\right)\right) + b_\psi\right) \quad (1)$$

where σ_1 is the ReLU function, σ_2 is the sigmoid function, g_i is the deeper-level feature map, x_i is the lower-level feature map, W_x^T, W_g^T, ψ^T are the convolution operations, and b_g, b_ψ is the bias of the convolution. Through the attention map shown in Fig. 5(a), we can determine that it depicts the border of the DNG, which makes the model focus more on the visual codes in visualizations. The color opacity of the attention map represents the α value, which means that attention increases the weight of the target area and suppresses background noise.

During training, we found that the segmentation of edge pixels is still difficult, whereas the results of other categories are sufficiently accurate. The reason for this is that edge pixels in the training set are fewer than other pixels (node type, background type). We use a hybrid training loss to solve class imbalance:

$$\mathcal{L}_{hybrid} = \mathcal{L}_{wce} + \mathcal{L}_{DL} \quad (2)$$

where $\mathcal{L}_{wce}$ is the weighted cross entropy with logits [70], and $\mathcal{L}_{DL}$ is the Dice loss [71].

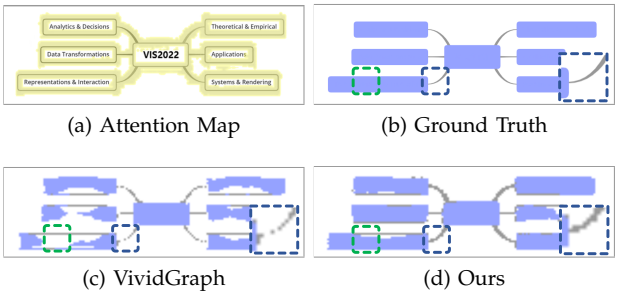
Given the ground truth of each pixel i category y_i and the prediction of network y_i', $\mathcal{L}_{wce}$ is defined as follows:

$$\mathcal{L}_{wce} = -\sum_i \left(\omega y_i \log\left(y_i'\right) + \left(1 - y_i\right) \log\left(1 - y_i'\right)\right) \quad (3)$$

where ω is the weight of each category. $y_i \in \{0, 1, 2, 3, 4\}$ represents the background, rectangle, ellipse, diamond and edge. Because most pixels in the graph are background pixels, the edges occupy the smallest proportion of pixels. We set the weight of the background class to 0.8 and the weight of the edge class to 1.25; the weights of other node classes remained 1. These hyperparameters are derived based on trial and error. This method improves the detection performance of edge pixels while maintaining node detection.

The Dice loss [71] is based on the Dice coefficient, a metric that evaluates the similarity of two samples. Dice loss is a region-dependent loss, which is different from cross entropy loss. It helps the model segment the foreground from the image to prevent situations such as the incomplete segmentation of the edge objects in Fig. 5(c). $\mathcal{L}_{DL}$ is defined as follows:

$$\mathcal{L}_{DL} = \sum_i \left(1 - \frac{2 y_i y_i' + \gamma}{y_i^2 + y_i'^2 + \gamma}\right) \quad (4)$$



(a) Attention Map (b) Ground Truth

(c) VividGraph (d) Ours

Fig. 5. Due to the attention map and hybrid loss function, our segmentation results are closer to the ground truth than are those of VividGraph. In the green box, the nodes in our semantic map are continuous, while the nodes in (c) are discontinuous. In the blue box, the pixels on the edges of our semantic map are also more continuous, which is conducive to semantic parsing.

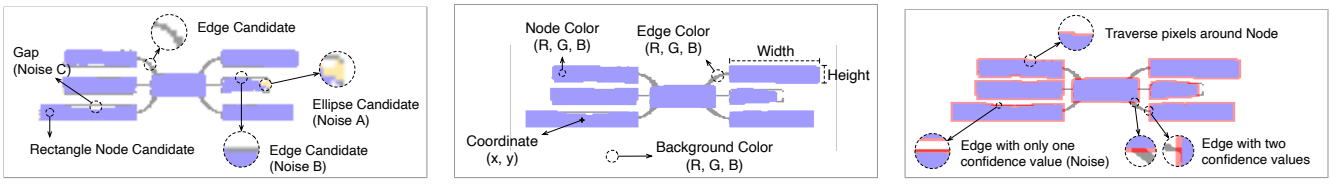
where γ is the Laplace smoothing factor, which prevents the denominator from being zero when both y_i and y_i' are zero. At the same time, the risk of overfitting can be reduced. We set the value to 1. This loss function can also help us solve the problem of sample imbalance, thereby improving the performance of our network.

Chen et al. [43] proposed that most models of natural images are designed to use low-resolution, strong semantics to improve detection, while the accuracy of segmentation suffers. We found that one way to improve segmentation accuracy (especially crispy edges) under the same resolution model is to reduce the loss of information during the convolution process, since visualization images contain fewer features than natural images. They do not require as many convolutional layers for feature extraction as natural images. We simplified the backbone from a four-layer encoder-decoder structure to three layers. To eliminate the impact of reducing the number of encoder-decoder layers on large targets such as nodes, we learn the structure of VGG19 [49] to thicken the convolutional layers of the last two backbone layers. Our model achieves better performance than VividGraph [1]. More detailed comparison data are included for the ablation experiments.

4.4 Semantic Parsing Module

After obtaining the semantic map, we need to parse the semantics to obtain complete information of the graph. To find node and edge candidates, we first extract connected components (CC) [72] from the semantic map, as shown in Fig. 6(a).

Second, we use mathematical morphology [73] on the CCs. We use the opening operation on node CC to eliminate small islands (e.g., Noise A and B in Fig. 6). We then use the closing operation on node CC to eliminate small holes and fill gaps of the contours (e.g., Noise C in Fig. 6). We also use the closing operation on edge CC to fuse narrow breaks and long thin gulfs (e.g., blue box



(a) CC Analysis (b) Morphological Denoising (c) Relationship Calculation

Fig. 6. Semantic parsing process. In (a), we determine candidate CCs for edges and nodes by connected component analysis and obtain (b) by morphological denoising. Thus, the visual attributes of the DNG are obtained through (b). In (c), we traverse the pixels around the node to calculate the connection confidence and recover the connections.

in Fig. 4). Specifically, the opening operation is image erosion followed by dilation with a disk structuring element. The closed operation is the opposite. A disk structuring element is similar to the convolution kernel, and it is a small matrix used for image processing. We define its size as $\frac{1}{3}\sqrt{Avg(Area_j)}$, where $Avg(Area_j)$ indicates the average area of all CCs. We obtain the attributes of edges and nodes from these candidates, as shown in Fig. 6(b). The shape types of nodes (rectangle, ellipse, diamond) are classified by our segmentation model. The color, coordinate and size are calculated by CC. When the node color is different from the background color (e.g., Fig. 8), we consider the node to be solid. Conversely, (e.g., Fig. 2), we judge the node to be a hollow node.

Third, we improved the node reconnection algorithm in VividGraph by imitating the way that humans perceive DNGs so that our pipelines can cope with non-straight edges and nodes with various shapes. When people observe whether two nodes are connected, they always look for a connecting edge around a node, so our heuristic algorithm mimics the process of human perception of connection. We consider the edges of three different shapes: straight lines, polylines, and curves. They exhibit three common characteristics. First, they are all connected to the sides of nodes. Second, they are all continuous pixels rather than dashed lines. Third, each edge connects two nodes. According to these common characteristics, we design a general algorithm to solve the topological analysis of DNGs. We traverse the pixels around each node CC, as shown in Fig. 6. If a pixel i belongs to a certain node, the confidence γ_p^q that the edge belongs to this node increases, where p, q represent Node p and Edge q. Since an edge always belongs to two nodes, the two nodes with the highest confidence are connected. For Edge q, if $max(\gamma_p^q)$ or $submax(\gamma_p^q)$ is 0, we consider the edge candidate to be noise, and we delete it. We consider nodes p_1 and p_2 to be connected, where p_1 and p_2 are the indices of $max(\gamma_{p_1}^q)$ and $submax(\gamma_{p_2}^q)$. We then assign the data obtained by the text detection module to nodes or edges. For each detection result t in $TextArr^t$ with $confidence > 0.95$, we find the nearest node p or edge q and assign t to it.

The detailed process and pseudocode of the parsing module are attached in the appendix.

4.5 Large-Scale Graph Extraction

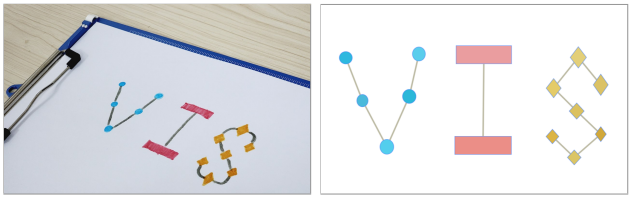
The GraphDecoder supports general scale (node number ≤ 30) DNGs. However, sometimes there are graphs with more than 30 nodes in the study, and these images often have a resolution greater than 1600×1600. This is a challenging task for chart mining because the dimensions of the input in the semantic segmentation network are stable, and high-resolution images will lose information during the normalization process.

Algorithm 1 Large-Scale Graph Mining Algorithm

Input: $C_{H\times W}$: the large-scale graph image with high resolution
Output: y': the semantic map of the large-scale graph

```
Put C_{H×W} into segmentation neural network
Obtain the semantic map y'_{entire}
Cut C_{H×W} into M pieces C^m, m = 1, 2, ..., M
Put C^m into segmentation neural network
Obtain the semantic map y'_m
Aggregate y'_m into y'_{piece}
Set α ∝ H × W
for all Pixel i do
    y'_i = { αy'_{piece} + (1 − α)y'_{entire}, y'_{piece}(i) = 4
           { αy'_{entire} + (1 − α)y'_{piece}, y'_{entire}(i) = 0, 1, 2, 3
end for
return y';
```

To overcome this problem, we analyzed the characteristics of the DNG. In the normalization process, edge information is easily lost because the edge occupies a few pixels, and the node information is still retained. Therefore, we can cut the image into pieces, extract them one by one, and then combine the semantic maps of these pieces for semantic parsing. However, when cutting, the nodes are easily cut into different shapes, which causes errors in extraction. Therefore, we combine the semantic map obtained after the normalization of the whole image and the semantic map obtained after cutting it into pieces for semantic parsing. We set a confidence $\alpha \in [0.5, 1]$, which is proportional to the resolution of the image. The confidence determines the bias of the final semantic segmentation result. The details are shown in Algorithm 1, where H, W are the height and weight of the image and y' represents the prediction of the pixel category.



(a) Input (b) Output

Fig. 7. The result of extracting data from the sketch.

5 Applications

The graph decoder can be applied in many scenarios. In this section, we mainly introduce three applications: sketch transformation, raster image extraction and chart redesign.

5.1 Sketch Transformation

The aim of visual engineers is to further reduce labor costs in visual design; this is also the driving force behind the development of visual analysis of production links. In addition to the engineering cost, the design of data visualization greatly affects the efficiency of data mining.

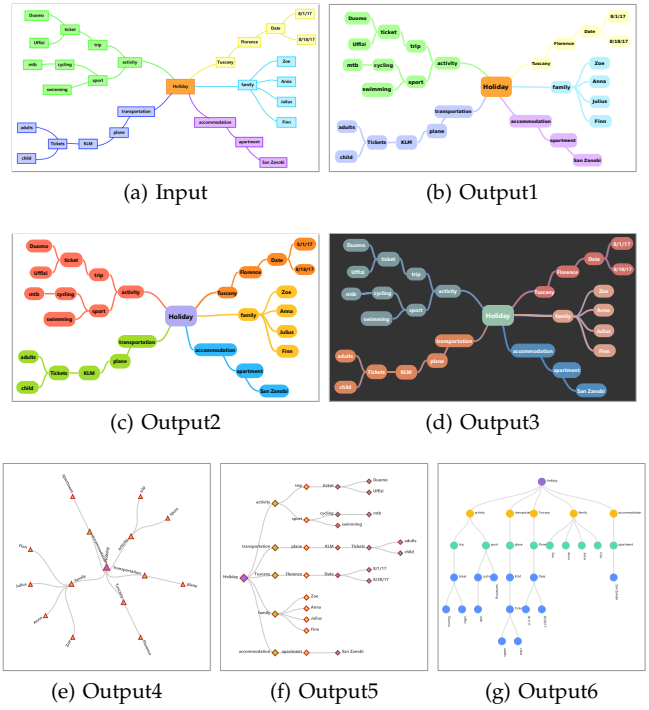
We used GraphDecoder to make some contributions in this regard. Users can transfer hand-drawn visualization works to our system, after which our system will combine an intelligent layout and color map based on the algorithm and rules to quickly generate electronic visualization works. This application can considerably reduce the requirement for visual programming, which allows designers to free their hands and quickly realize ideas.

As shown in Fig. 7(a), the users can quickly pick up the paper and pen to draw the graph when an idea occurs to them. The users then only need to take a picture of the graph they have drawn, upload it to our system, and obtain an editable and interactive DNG, as shown in Fig. 7(b). This application is not only convenient for professional visual designers, but also enables users who do not have programming skills to create as much as they want.

The current version of GraphDecoder only supports graphical data transformation of sketches compared with electronic images because the text detection module cannot effectively recognize handwritten text. This limitation will also be discussed in Sec. 8.

5.2 Raster Image Extraction

Visualization images spread on the internet generally take the form of raster images, which can vividly express data characteristics. However, we always seek to obtain their original data. For example, E-R diagrams [3] are very common among DNGs. After drawing the E-R diagram, the database programmer needs to build the database. At this time, if the original data of the E-R diagram can be obtained quickly, the time and labor costs can be reduced.



(a) Input (b) Output1 (c) Output2 (d) Output3 (e) Output4 (f) Output5 (g) Output6

Fig. 8. The results of extracting the data from mind map and redesign. (b) is the extracted result. (c-d) are the results of recoloring. (e-g) are the results of relayout.

Using graphs to represent algorithms is a way to quickly understand them. The flowchart [2] is a commonly used DNG to express algorithmic thinking. Information on these algorithmic processes can be quickly obtained through our system.

Fig. 1 shows some cases of extracting DNGs with GraphDecoder, and more cases of E-R diagrams and algorithm flowcharts are shown in the appendix.

5.3 Chart Redesign

An excellent layout and color scheme enable readers to quickly grasp information and features in the diagram [33, 74, 75]. For example, mind map [4] is an effective graphical tool for expressing divergent thinking. The mind map uses the characteristics of graphics and texts, expresses the relationships of various topics with hierarchical diagrams, and establishes memory links between topic keywords, graphics, colors, etc.

Fig. 8(a) is an image that depicts a family holiday arrangement. We input it into GraphDecoder to obtain its original data, which are generally output in the form of a JSON file. We can directly redraw the mind map by JSON to obtain the result shown in Fig. 8(b). In addition to manually modifying this graph, we provide one-click retargeting layouts and color schemes. The users only need to input a static image, and then they can easily recolor the DNG with a default recommended

color scheme as shown in Fig. 8(c) or a dark theme as shown in Fig. 8(d). They can also obtain a variety of visualization forms, as shown in Fig. 8(e-g).

The programmer of the database can quickly check the attributes of each entity and the connection relationship between the entities.

6 User Study

To expand the application of GraphDecoder, we conducted a user study. We used system trials, user interviews and questionnaires to obtain user feedback, which can improve our system.

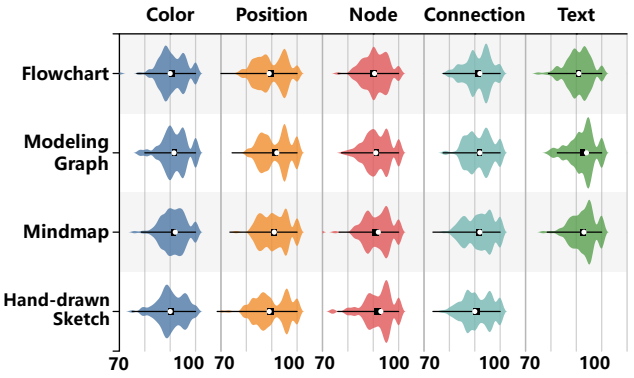
Procedure: Our user study includes five parts: (1) informed consent of users, (2) introduction of GraphDecoder, (3) system trials, (4) questionnaire, and (5) interview. We first informed the participants that their answers would be used for our research and then classified them by age and occupation. Then, we verbally introduced our system and methods to ensure that every participant could understand the functions and usage of our system. We invited users to implement our system for data extraction and transformation, then used questionnaires and interviews to obtain user feedback and evaluate the actual effectiveness of our methods.

Recruitment: We recruited 60 participants. Six of the participants were unable to understand our method due to capacity limitations, and we ultimately recovered 54 valid user data (μ_{age}=25.9 years; 32 computer professionals; 22 noncomputer professionals). Some of our participants were proficient with computers and design, such as development engineers, visualization designers, graphic designers and scientific researchers. Other participants were noncomputer professionals, such as teachers, middle school students, accountants, and homemakers.

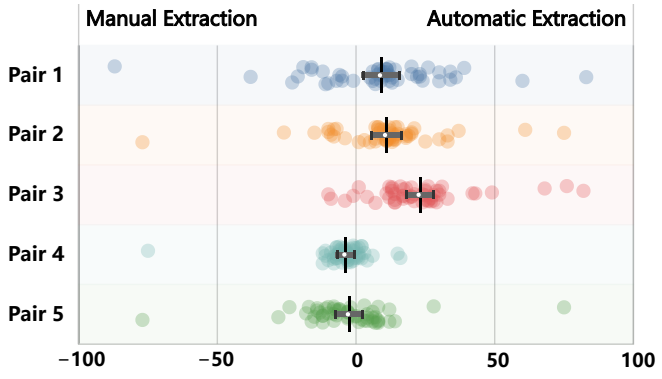
We recruited 5 additional designers (μ_{age}=25 years) working on visualization to collect the pairs in the questionnaire for pairwise comparison.

Questionnaire: We set up 13 questions in the questionnaire, including 8 scoring questions for the extraction results and 5 pairwise comparisons.

Scoring Questions: The hand-drawn sketches, modeling graphs, mind maps, and flowcharts each account for a quarter of the scoring questions of the extraction results. Participants scored the extraction results based on five aspects: color, position, node (size, shape), connection relationship and text content, with a full score of 100. Since there are no handwritten data in our text extraction model dataset, the hand-drawn sketch is not scored for text. The final result is shown in Fig. 9(a). From the results, we can determine that our extraction results can meet most of the needs of users. Most of the average scores are above 90. The reason why the position score is slightly lower than 90 is because we used a small number of morphological methods in the semantic parsing module, which caused the node position to shift.



(a) Scoring of extraction results



(b) Pairwise comparisons

Fig. 9. Questionnaire scoring results. (a) is the result of 8 scoring questions for the extraction results. (b) is the result of the 5 pairwise comparisons. Pairs 1-5 correspond to Questions 12-16 in the questionnaire. Circles depict group averages (± 95% confidence intervals).

Therefore, it is necessary to avoid traditional methods in the pipeline as much as possible in the future.

Pair Collection: Our pairwise comparisons necessitated two groups. (1) Manual-extracted group: The designers used their skill for tools such as Photoshop, PowerPoint, Processon, etc. to redraw and redesign DNGs by observing the image. (2) Autoextracted group: The designers used our system to extract and redesign DNGs.

We provided five designers with a DNG in bitmap form and then asked them to redraw one DNG using their tools and another one using our system. We supplied a total of 5 DNGs and conducted a total of 25 trials. We calculated the time T, the number of left mouse clicks N_m, and the number of keyboard strokes N_k, which were used for the two groups. The autoextracted group achieved T of 85 seconds, N_m of 32.4, and N_k of 9. The manual-extracted group achieved T of 1165.8 seconds, N_m of 465.2, and N_k of 424.8. This shows that our system reduces designer time for redesigning DNGs.

For each DNG given, we selected pairs with similar and correct results and added them to the pair comparison. There are five pairs in total.

Pairwise Comparison: For the fairness of the comparison, we randomly marked the images of the two

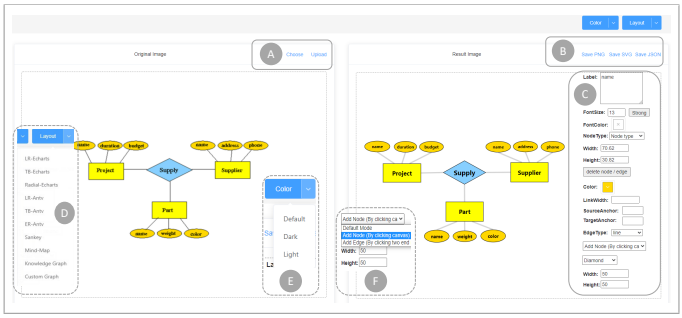


Fig. 10. The interface of our system. The left side of the interface is the input, and the right side is the output. Elements include (A) choose and upload, (B) save, (C) interactive panel, (D) automatic layout option, (E) automatic color option, (F) mode switch.

groups as Picture 1 or Picture 2. The participants did not know which was the autoextracted result. We designed a rating slider with 100 on the far right and -100 on the far left. The larger the positive value was, the greater the user preferred Picture 2 on the right. The larger the absolute value of the negative number, the greater the user preferred Picture 1 on the left. Scoring sliders are often employed for preference tasks [76] and have high score reliability [77]. The slider was initially set to 0, which made the questions unbiased [78]. When quantifying the results, we set the score of the picture from the autoextracted group to a positive value. The complete result is shown in Fig. 9(b). The scores are all close to the neutral value in the five comparisons, indicating that participants showed no preference between the two groups. This also shows that designers can use our systems to redesign DNGs similar to manual results in less time. The complete survey data related to the questionnaire have been attached in the appendix for further research.

Interview: Each participant also experienced our beta system. We then conducted interviews with users to improve our system. In the interview, we found that users of different professional groups have different requirements for the system. For noncomputer professionals, the types of DNGs they are most concerned about are hand-drawn sketches and mind maps. Many of them do not master visual programming, especially older users. A woman who has been engaged in middle school education for more than ten years said that the data transformation of hand-drawn sketches saves her a lot of time compared with the drawing module in Microsoft Office. Another lady in the company's financial management mentioned in the interview that mind maps are often used in daily life. However, those who are not design professionals will generally search for existing mind maps on the internet to imitate. Our system can help to quickly modify the material in the form of raster images on the internet. Based on the feedback we received, we have also improved the usability of our tool, including by enlarging the text edit box and merging the same attributes of nodes and edges. These suggestions make our system friendlier and more durable.

For computer professionals, in addition to the first two DNGs in their work, there are also many algorithm flowcharts and modeling graphs. Even though most of them have mastered programming and have a certain design foundation, they primarily indicated that the interactive DNG extraction system can help them complete their work faster. In the interview, three web designers all expressed that the function of extracting DNGs and saving JSON files is very convenient. A data engineer praised our redesign of the E-R diagram, which helped him understand the database structure more quickly. It was said that our system should add more automatic layout options and automatic color schemes for different types of DNGs. We adopted the provided opinions and increased the automatic layout options to a total amount of ten. Two color matching strategies are provided. One traditional color matching strategy is used to map the same color to the other color, and the other color matching strategy for modeling graphs and flowcharts is used to map nodes of the same shape to the same color.

Our revised system test version interface is shown in Fig. 10. Area A is the button for selecting and uploading the input image. Users can download the output in area B. We provide three output formats, namely, PNG, SVG, and JSON. Area C is our DNG interactive panel, and users can customize almost all DNG attributes in this area. There are three modes: the default mode, the node addition mode, and the edge addition mode, as shown in Area F. Area D and Area E are our options for automatic color matching and automatic layout. More demonstrations of our system have been attached in the additional materials.

7 Evaluation

Our evaluation experiment includes two parts. First, we compared our segmentation module with a state-of-the-art method [43] used in extracting timelines. We also conducted ablation experiments on our semantic segmentation network to prove the effectiveness of our modifications to U-Net in VividGraph [1]. Second, we conducted evaluations of structural similarity and perceptual similarity on the real corpus and the virtual corpus to prove the effectiveness of our method.

We deploy our pipeline on a PC with Intel Core i7-9700F, NVIDIA GeForce 1660 Ti, and 16 GB of memory. The deep learning framework is implemented based on Keras [79].

7.1 Segmentation Module Evaluation

The accuracy of segmentation is a key component of the accuracy of extracting DNG data. We designed a special segmentation model for DNGs. Some work has

TABLE 1
Evaluation results of semantic segmentation model

Methods		FIoU↑	MIoU↑	Back_IoU↑	Rect_IoU↑	Elli_IoU↑	Dia_IoU↑	Edge_IoU↑
Chen et al. [43]		0.9529	0.5242	0.9773	0.4114	0.5612	0.5435	0.1276
VividGraph [1]		0.9895	0.9407	0.9944	0.9762	0.9859	0.9537	0.7831
Ours	Attention	0.9908	0.9479	0.9951	0.9801	0.9876	0.9659	0.8107
	$\mathcal{L}_{hybrid}$	0.9902	0.9441	0.9948	0.9791	0.9862	0.9585	0.8019
	Backbone	0.9901	0.9439	0.9947	0.9821	0.9863	0.9659	0.7906
	Attention+$\mathcal{L}_{hybrid}$	0.9920	0.9536	0.9957	**0.9866**	**0.9910**	0.9712	0.8236
	Attention+Backbone	0.9904	0.9454	0.9948	0.9820	0.9855	0.9689	0.7959
	Backbone+$\mathcal{L}_{hybrid}$	0.9908	0.9465	0.9952	0.9793	0.9881	0.9627	0.8070
	Attention+Backbone+$\mathcal{L}_{hybrid}$	**0.9924**	**0.9562**	**0.9959**	0.9857	0.9908	**0.9717**	**0.8367**

also used segmentation to extract other types of graphs. Chen et al. proposed a model [43] to extract the timeline. Since the model does not include the algorithm of the node connection, we only compare the segmentation modules. For a fair comparison, we train this module on our training dataset.

We conducted ablation experiments on our dataset to prove the effectiveness of our contribution to the semantic segmentation model. We made three improvements to the model, including adding a module with an attention mechanism, using a hybrid loss function, and simplifying the backbone of the model. We evaluate our semantic segmentation model using intersection over union (IoU), mean IoU (MIoU), and frequency-weighted IoU (FIoU), which are defined as follows:

$$\begin{cases} IoU = \frac{y_i \bigcap y_i'}{y_i \bigcup y_i'} \\ MIoU = \frac{1}{k+1} \sum_{i=0}^{k} \frac{p_{ii}}{\sum_{j=0}^{k} p_{ij} + \sum_{j=0}^{k} p_{ji} - p_{ii}} \\ FIoU = \frac{1}{\sum_{i=0}^{k} \sum_{j=0}^{k} p_{ij}} \sum_{i=0}^{k} \frac{p_{ii}}{\sum_{j=0}^{k} p_{ij} + \sum_{j=0}^{k} p_{ji} - p_{ii}}, \end{cases} \tag{5}$$

where p_{ij} represents the situation. The ground truth is i, and the prediction is j. k is the number of pixel classifications. We set $k = 5$ in our task.

IoU represents the overlap area ratio of the ground-truth bounding box and predicted bounding box, which is usually employed for evaluation [80, 81]. Due to the characteristics of the graphs, the background pixels account for a large proportion, so MIoU is more suitable for evaluating our model than is FIoU. The experimental results are shown in Table 1. From the results, we can determine that the IoU of each category has improved, especially the IoU of the edge. Therefore, our modifications to the model have effectively improved the semantic segmentation capabilities of graphs (especially edges). This is because the attention mechanism we introduced can solve the oversegmentation and undersegmentation problems caused by DNG features, as shown in Fig. 4.

7.2 Structural and Perceptual Evaluation

We propose two methods to evaluate the effectiveness of our extracted results. The first method is NetSimile [22], which is a method widely used to compare the similarity of two network structures. NetSimile is defined as follows:

$$NetSim(Pre, Gt) = \sum_{i=1}^{n} \frac{\left\| Sig_{pre}^{i} - Sig_{gt}^{i} \right\|}{\left\| Sig_{pre}^{i} \right\| + \left\| Sig_{gt}^{i} \right\|} \tag{6}$$

where Sig_{pre}^{i} is the signature vector of the extracted network structure and Sig_{gt}^{i} is the signature vector of the ground truth. The signature vector contains 35-dimensional features, including node in and out degree, clustering coefficient, and ego. This method measures the similarity of the network structure by comparing the Canberra distance of two signature vectors. The lower NetSimile is, the higher the structural similarity. The advantage of using this metric to evaluate structural similarity is that there is no need for one-to-one correspondence of node numbers. If a node is not detected, this method can also correctly measure the similarity of the two networks. To demonstrate NetSimile more intuitively, we compare the network structure of the ground truth with the network structure of the same number of nodes but with no edges to obtain the largest NetSimile $NetSim_{max}$. Then, the structural similarity $StruSim$ is defined as follows:

$$StruSim(Pre, Gt) = (1 - \frac{NetSim(Pre, Gt)}{NetSim_{max}}) \times 100\% \tag{7}$$

where $StruSim$ ranges from 0 to 100%. $StruSim = 100\%$ means that the two network structures are the same.

The other method is to compare the visual saliency map of the input images and the redrawn images. The information of nodes and links will affect the metrics, including size, location, color, etc. We used AntV to redraw the extracted data into a DNG image. We then used the visual saliency model proposed by Bylinskii et al. [23] to output the visual saliency map of the two images that need to be compared. This visual saliency model can output the important areas of each graph and assign all pixels a score from 0 to 255. We evaluate the perceptual similarity by comparing the difference in the average scores of the two images. Similarly,

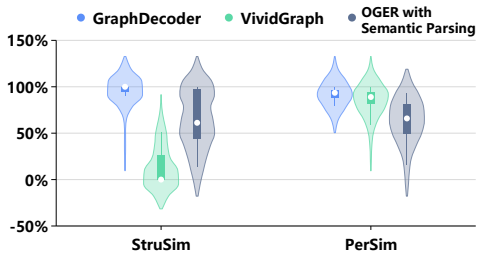


Fig. 11. The results of evaluation experiments on a real data corpus. The circles show the median values. Error bars depict the interquartile range.

we also divide the difference between this score and the ground truth score into a perceptual similarity $PerSim \in [0, 100\%]$. $PerSim$ is defined as follows:

$$PerSim(Pre, Gt) = 100\% - \frac{|Sali_{pre} - Sali_{gt}|}{Sali_{gt}} \quad (8)$$

$Sali_{gt}$ is the visual saliency map score of the ground truth, and $Sali_{pre}$ is the visual saliency map score of our extraction results.

We collected 100 images from E-Charts, D3, AntV, Google, and Xmind, and user hand-drawn sketches were used as our evaluation dataset. We compared our method with VividGraph [1] and OGER [18] (an improved version of OGR [17]). VividGraph does not have a text extraction module, so we use our text detection module to extract texts for fair comparison. VividGraph uses an untuned segmentation model, and its reconnection algorithm is not suitable for the connection of nonstraight edges. Our real dataset contains many DNGs with hollow nodes and nonstraight edges (as shown in Fig. 1), which results in the poor $StruSim$ achieved by VividGraph. OGER has many limitations. OGER cannot extract texts and classify nodes. It cannot analyze the shape of nodes and edges, nor can it match textual data and graphical data. We added our semantic parsing module to enable a fair comparison. This also enables OGER to handle connections of nonstraight edges in the dataset. However, compared with end-to-end methods, OGER still has many limitations in comparison. Since OGER is entirely a heuristic method, we need to adjust the segmentation threshold for different resolutions and different scales of network graphs and binarization parameters for different colormaps. We also resize the images to a suitable resolution and set the proper threshold. These operations are not counted in the time of the OGER.

The evaluation results are shown in Fig. 11. Compared with VividGraph and OGER, our method achieves the best results for both the $StruSim$ and $PerSim$ metrics. The results show that our method can extract data from DNGs in the real world with better quality than other state-of-the-art methods. The error of $PerSim$ is primarily due to the rich styles of many DNGs in the real world, such as gradient colors. When we redraw, our system does not have a gradient color option, and the graph is filled with a single color. However, we can find that our methods still correctly identify the connection relationships of these DNGs from $StruSim$. The graph decoder can handle most DNGs, especially when DNGs include noise, nonsolid nodes, and different edge sizes, which are all difficult for morphological methods.

In addition, we found that image resolution and structural complexity are important factors affecting extraction accuracy. Therefore, we constructed an additional set of evaluation data corpora with different resolutions and numbers of nodes. To ensure that the evaluation corpora and the training dataset do not overlap, we chose AntV as the visualization tool for the additional corpus. Each image in the corpus has a different layout, different colors, and different node sizes and edge types. We chose three resolutions of 640×640, 960×960, and 1280×1280 and three types of node numbers around 10, 20, and 30. We conducted experiments for comparison with OGER and VividGraph on nine sets of corpora. The experimental data are shown in Fig. 12. Our method achieves state-of-the-art results in terms of all three metrics. OGER with semantic parsing is a complete heuristic method, so its results change with the resolution and the number of nodes. Although we set optimal thresholds for each set of OGER to enable smooth binarization and skeletonization, our results are also superior to it. This is because OGER's segmentation method does not work for hollow nodes. The segmentation model of VividGraph causes the phenomena of oversegmentation and undersegmentation, and the node reconnection algorithm can only recover the connections of straight edges, so $StruSim$ is lower. These results confirm that our methods can extract DNGs of different scales.

7.3 Time Performance

Our method also consumes less time than OGER and VividGraph. We first focus on real-world datasets. GraphDecoder spent an average of 3.115 (95% CI: $3.108 - 3.122$, $p<0.01$) seconds on this corpus, while OGER spent an average of 6.585 (95% CI: $6.507 - 6.663$, $p<0.01$) seconds; VividGraph spent an average of 3.184 (95% CI: $3.157-3.210$, $p<0.01$) seconds. Our methods are faster than OGER because our segmentation uses deep learning methods. This time gap increases with the image resolution. Our time performance is also improved over VividGraph because our semantic parsing module traverses each node: the corresponding time complexity is $O(N)$. VividGraph traverses every two nodes, with complexity $O(N^2)$.

We also evaluated the time performance based on additional datasets. As shown in Fig. 12(c), we found that at the same resolution, when the number of nodes increases, the average time spent by all three methods increases slightly. However, when the resolution increases, the average time consumption of the OGER

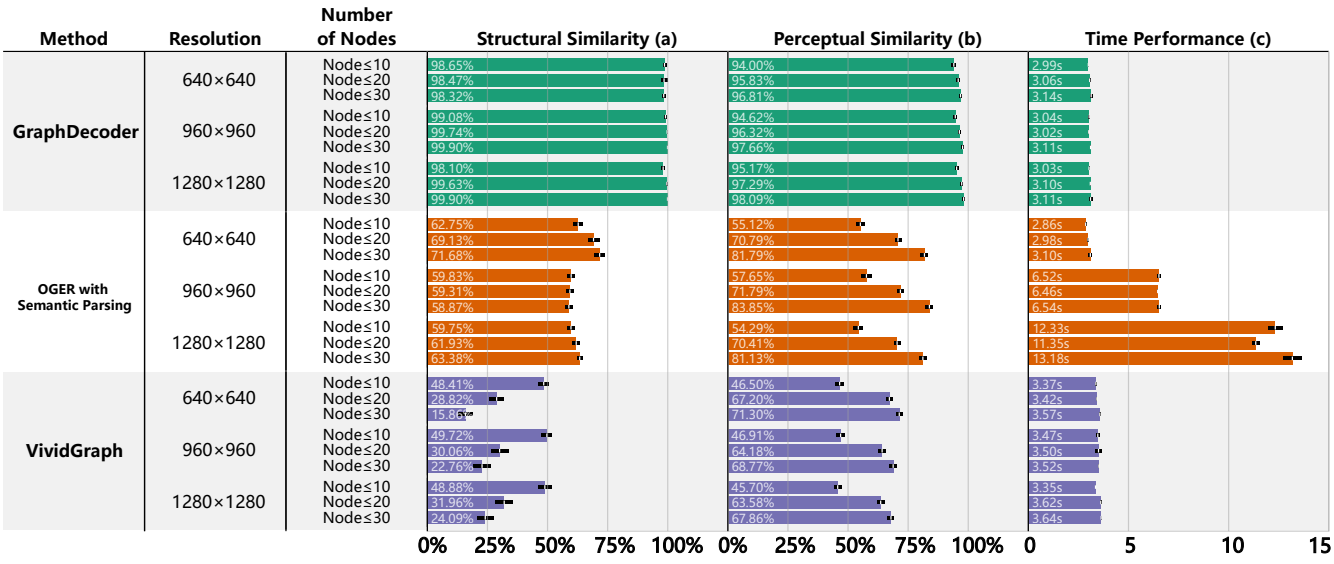


Fig. 12. The results of evaluation experiments on additional sets of evaluation data corpora. In (a), we used structural similarity to evaluate the extraction accuracy of relationships. In (b), we used perceptual similarity to evaluate the extraction accuracy of various visual attributes. In (c), we show the time spent by each method. The white values represent the mean values. Error bars depict ± 95% CIs for mean values.

increases significantly, while the average time consumption of the others remains stable. This illustrates the time performance advantage of the deep learning module. In addition, our time performance is better than that of VividGraph under the same conditions. This again proves that our method is more efficient than VividGraph in computing node relations.

8 Discussion

8.1 Deep Learning or Heuristic Methods

The popularity of deep learning brings tremendous changes to visualization research [82]. We believe that deep learning should be used in scenarios for which heuristic methods have limitations. If heuristic methods work well, then deep learning is of little significance.

We designed a deep learning model to segment nodes and edges. Compared with heuristic methods such as OGR [17] and OGER [18], deep learning models have better robustness and time performance. First, heuristic methods need a suitable threshold for binarization, but there is no perfect threshold for the various colormaps of real-world DNGs to distinguish all targets and backgrounds. Second, deep learning solves the limitation of the heuristic method for hollow nodes because heuristic methods encounter difficulty distinguishing the edges and outlines of hollow nodes. Third, existing heuristic methods for segmenting network graphs do not include node classification modules. They are only suitable for graphs with circular solid nodes. If we attempt to complete the node classification, heuristic segmentation methods need to add more complex thresholds and rules. However, our deep learning model has integrated these.

In our semantic parsing module, we use heuristic methods to analyze semantic maps. We did not use deep learning [11, 41, 42] to extract relationships for two reasons. First, the heuristic method can already effectively analyze the relationship, as shown in the structural evaluation in section 7. Second, we extract many attribute values from the semantic maps, including relationships, positions, colors, sizes, and shapes. A single deep learning model cannot accurately regress too many attribute values. If we design a model for each attribute, the pipeline will be redundant. For this part, the heuristic method is simple and effective.

8.2 Visualizations or Natural Images

There are works [6, 43] which propose chart extraction in relation to the difference between graphic and natural elements. Most of the existing CNN models and attention mechanisms are designed based on natural images. On the one hand, they are concerned with large objects such as people, trees, and vehicles. On the other hand, they segment low-level semantics. For example, a person is recognized, but the result is slightly wider and shorter than the ground truth. This has no effect on the desired outcome. However, the situation is different for charts, especially DNGs.

Visualization images should be more focused on the ambiguity of basic geometric shapes. DNGs have many edges and nodes with various shapes. Since most of the targets of DNG are basic geometric shapes, they are easily confused with hollow nodes due to the ambiguity caused by edges during segmentation. When designing a segmentation model for visualization images, we suggest paying attention to the undersegmentation and

oversegmentation categories of the attention region and loss function. In DNGs, both of these categories are edges.

Additionally, due to the difficulty of labeling and the lack of public corpora, the training sets for most data-driven visualization work [1, 11, 83] are synthetic datasets. The scale of such datasets is smaller than that of common datasets of natural images (e.g., COCO [84]). Since medical image segmentation also requires high-level semantics and is implemented with the characteristics of small datasets, we found that some methods can be borrowed for visualization image segmentation.

The recovery of various visual attributes from visualizations is actually a multiple perception task [47, 85]. Bar graphs require the perception of area or length, pie charts require perception of angle, and line graphs require the perception of the direction and position of lines. DNGs necessitate more tasks, including the determination of area, shape, color, text, relationship and position. Every perception task requires high-level semantics.

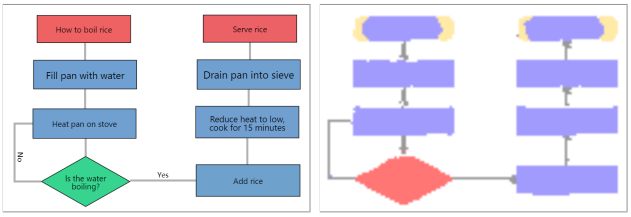
To solve complex perceptual tasks, we believe that imitating the process of human understanding visualization can be considered. The models of Chen et al. [43] focus on global information and local information of timelines, respectively. Our pipeline also divides the complex perception task of DNG into two steps: first obtaining global information through a segmentation model and then recovering local information through semantic parsing.

8.3 Limitations and Future Work

The current version of GraphDecoder also has certain limitations.

First, the data-driven segmentation module also makes the model constrained by the training set. The semantic manual annotation of DNGs is a complicated task, so we designed a synthetic dataset as a training set. We have strived to cover various elements of different DNGs, including visualization tools, node shape, node size, node distribution, edge type, edge thickness, colormap, and text. The DNGs that the current version of GraphDecoder can handle represent 85.19% of participants in our user study. However, the styles of DNG in the real world are endless. We introduce noise into the training set via data augmentation, but with no elements (e.g., chart title, chart annotation) other than the DNG content. For the removal of these elements, known methods [24, 33] can be used in combination. We plan to make the code of DNG generation and our datasets available for the visualization community and construct a larger dataset of DNGs in the future.

Second, a surprising finding is that rectangles or rounded rectangles may be misidentified as ellipses (Fig. 13) when the input DNG is a flowchart because the flowcharts in the training set are regular, which means that the first node of a standard flowchart is



(a) Input (b) Segmentation Result

Fig. 13. A case of segmentation fault. The boundaries of the red nodes are incorrectly identified as ellipses.

an ellipse [2]. This phenomenon indicates that our segmentation model not only learns local features but also learns global features of various DNGs. We can exploit this limitation to discover irregular DNG images in the future.

Third, we retained the heuristic method in the semantic parsing module. We attempted to transfer the end-to-end method [11] of other chart types to DNGs but found that it did not function successfully. This is because the DNGs contain excessive data attributes. The heuristic semantic parsing module ensures the robustness and correctness of the original data obtained from the semantic map but also increases the time consumption. At the same time, the heuristic module cannot effectively solve the ambiguity of cross-edges. We can use the direction vector judgment [18] to solve a part of the cross-edge problem, but the overall effect is not good enough. We plan to improve the semantic parsing module.

Finally, although the current OCR system is mature, there are still some limitations. For example, it is difficult to extract hand-drawn text, which means that we can only handle hand-drawn DNGs without text. However, this is a common problem in the OCR field. We will replace the text detection module with a better OCR module in the future.

9 Conclusion

We proposed a method to extract the original data of DNG images. Our system, GraphDecoder, can be applied for sketches, raster image extraction and chart design. Our method is suitable for mind maps, E-R diagrams, flowcharts, and tree diagrams. We designed an attention-aware semantic segmentation neural network to improve the ability to extract DNGs. Our evaluation of a real data corpus and additional data corpora demonstrates that GraphDecoder can efficiently and correctly extract DNGs of different scales, resolutions, and types. We also shared our experience in data extraction from advanced and complex graphs such as DNGs and discussed how to improve and optimize our work in the future.

REFERENCES

[1] S. Song, C. Li, Y. Sun, and C. Wang, "Vividgraph: Learning to extract and redesign network graphs from visualization images," *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[2] I. Nassi and B. Shneiderman, "Flowchart techniques for structured programming," *ACM Sigplan Notices*, vol. 8, no. 8, pp. 12–26, 1973.

[3] P. P.-S. Chen, "The entity-relationship model—toward a unified view of data," *ACM transactions on database systems (TODS)*, vol. 1, no. 1, pp. 9–36, 1976.

[4] T. Buzan, *Use both sides of your brain*. EP Dutton New York, 1983.

[5] Y. Liu, X. Lu, Y. Qin, Z. Tang, and J. Xu, "Review of chart recognition in document images," in *Visualization and Data Analysis 2013*, vol. 8654. International Society for Optics and Photonics, 2013, pp. 384–391.

[6] K. Davila, S. Setlur, D. Doermann, U. K. Bhargava, and V. Govindaraju, "Chart mining: a survey of methods for automated chart analysis," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[7] P. Zhang, C. Li, and C. Wang, "Viscode: Embedding information in visualization images using encoder-decoder network," *IEEE Transactions on Visualization and Computer Graphics*, 2020.

[8] J. Fu, B. Zhu, W. Cui, S. Ge, Y. Wang, H. Zhang, H. Huang, Y. Tang, D. Zhang, and X. Ma, "Chartem: Reviving chart images with data embedding," *IEEE Transactions on Visualization and Computer Graphics*, 2020.

[9] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer, "Revision: Automated classification, analysis and redesign of chart images," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 2011, pp. 393–402.

[10] J. Yuan, C. Chen, W. Yang, M. Liu, J. Xia, and S. Liu, "A survey of visual analytics techniques for machine learning," *Computational Visual Media*, vol. 7, no. 1, pp. 3–36, 2021.

[11] F. Zhou, Y. Zhao, W. Chen, Y. Tan, Y. Xu, Y. Chen, C. Liu, and Y. Zhao, "Reverse-engineering bar charts using neural networks," *Journal of Visualization*, pp. 491–435, 2021.

[12] R. Burns, S. Carberry, and S. Elzer Schwartz, "An automated approach for the recognition of intended messages in grouped bar charts," *Computational Intelligence*, vol. 35, no. 4, pp. 955–1002, 2019.

[13] P. De, "Automatic data extraction from 2d and 3d pie chart images," in *2018 IEEE 8th International Advance Computing Conference (IACC)*. IEEE, 2018, pp. 20–25.

[14] D. Jung, W. Kim, H. Song, J.-i. Hwang, B. Lee, B. Kim, and J. Seo, "Chartsense: Interactive data extraction from chart images," in *Proceedings of the 2017 chi conference on human factors in computing systems*, 2017, pp. 6706–6717.

[15] N. Siegel, Z. Horvitz, R. Levin, S. Divvala, and A. Farhadi, "Figureseer: Parsing result-figures in research papers," in *European Conference on Computer Vision*. Springer, 2016, pp. 664–680.

[16] J. Luo, Z. Li, J. Wang, and C.-Y. Lin, "Chartocr: Data extraction from charts images via a deep hybrid framework," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 1917–1925.

[17] C. Auer, C. Bachmaier, F. J. Brandenburg, A. Gleißner, and J. Reislhuber, "Optical graph recognition," in *International Symposium on Graph Drawing*. Springer, 2012, pp. 529–540.

[18] R. Opmanis, "Optical graph edge recognition." in *VISIGRAPP (3: IVAPP)*, 2018, pp. 184–191.

[19] E. Brynjolfsson and K. McElheran, "The rapid adoption of data-driven decision-making," *American Economic Review*, vol. 106, no. 5, pp. 133–39, 2016.

[20] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[21] "Antv," https://g6.antv.vision/.

[22] M. Berlingerio, D. Koutra, T. Eliassi-Rad, and C. Faloutsos, "Netsimile: A scalable approach to size-independent network similarity," *arXiv preprint arXiv:1209.2684*, 2012.

[23] Z. Bylinskii, N. W. Kim, P. O'Donovan, S. Alsheikh, S. Madan, H. Pfister, F. Durand, B. Russell, and A. Hertzmann, "Learning visual importance for graphic designs and data visualizations," in *Proceedings of the 30th Annual ACM symposium on user interface software and technology*, 2017, pp. 57–69.

[24] J. Chen, M. Ling, R. Li, P. Isenberg, T. Isenberg, M. Sedlmair, T. Moller, R. S. Laramee, H.-W. Shen, K. Wunsche *et al.*, "Vis30k: A collection of figures and tables from ieee visualization conference publications," *IEEE Transactions on Visualization and Computer Graphics*, 2021.

[25] A. Balaji, T. Ramanathan, and V. Sonathi, "Chart-text: A fully automated chart image descriptor," *arXiv preprint arXiv:1812.10636*, 2018.

[26] W. Dai, M. Wang, Z. Niu, and J. Zhang, "Chart decoder: Generating textual and numeric information from chart images automatically," *Journal of Visual Languages & Computing*, vol. 48, pp. 101–109, 2018.

[27] R. A. Al-Zaidy and C. L. Giles, "A machine learning approach for semantic structuring of scientific charts in scholarly documents." in *AAAI*, 2017, pp. 4644–4649.

[28] J. Choi, S. Jung, D. G. Park, J. Choo, and N. Elmqvist, "Visualizing for the non-visual: Enabling the visually impaired to use visualization," in *Computer Graphics Forum*, vol. 38, no. 3. Wiley Online Library, 2019, pp. 249–260.

[29] X. Liu, D. Klabjan, and P. NBless, "Data extraction from charts via single deep neural network," *arXiv preprint arXiv:1906.11906*, 2019.

[30] M. K. I. Molla, K. H. Talukder, and M. A. Hossain, "Line chart recognition and data extraction technique," in *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, 2003, pp. 865–870.

[31] V. K. Reddy and C. Kaushik, "Image processing based data extraction from graphical representation," in *2015 IEEE International Conference on Computer Graphics, Vision and Information Security (CGVIS)*. IEEE, 2015, pp. 190–194.

[32] K. Mao, X. Chen, K. Zhu, D. Hu, and Y. Li, "A method to extract essential information from meteorological facsimile charts," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 33, no. 01, p. 1954001, 2019.

[33] J. Poco, A. Mayhua, and J. Heer, "Extracting and retargeting color mappings from bitmap images of visualizations," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 1, pp. 637–646, 2017.

[34] R.-Q. Wu, X.-R. Cheng, and C.-J. Yang, "Extracting contour lines from topographic maps based on cartography and graphics knowledge," *Journal of Computer Science and Technology*, vol. 9, no. 02, pp. 58–64, 2009.

[35] A. Flower, J. W. McKenna, and G. Upreti, "Validity and reliability of graphclick and datathief iii for data extraction," *Behavior modification*, vol. 40, no. 3, pp. 396–413, 2016.

[36] G. G. Méndez, M. A. Nacenta, and S. Vandenheste, "ivolver: Interactive visual language for visualization extraction and reconstruction," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 2016, pp. 4073–4085.

[37] A. J. Kadic, K. Vucic, S. Dosenovic, D. Sapunar, and L. Puljak, "Extracting data from figures with software was faster, with higher interrater reliability than manual extraction," *Journal of clinical epidemiology*, vol. 74, pp. 119–123, 2016.

[38] "Dagra data digitizer," https://www.datadigitization.com/.

[39] M. Mitchell, B. Muftakhidinov, T. Winchen, A. Wilms, B. van Schaik, badshah400, Mo-Gul, T. G. Badger, Z. Jedrzejewski-Szmek, kensington, and kylesower, "markummitchell/engauge-digitizer: Nonrelease," Jul. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3941227

[40] M. Oldenhof, A. Arany, Y. Moreau, and J. Simm, "Chemgrapher: optical graph recognition of chemical compounds by deep learning," *Journal of Chemical Information and Modeling*, vol. 60, no. 10, pp. 4506–4517, 2020.

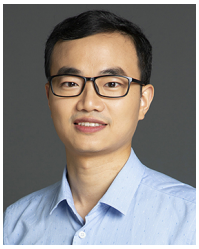
[41] A. Kembhavi, M. Salvato, E. Kolve, M. Seo, H. Hajishirzi, and A. Farhadi, "A diagram is worth a dozen images," in *European conference on computer vision*. Springer, 2016, pp. 235–251.

[42] D. Kim, Y. Yoo, J.-S. Kim, S. Lee, and N. Kwak, "Dynamic graph generation network: Generating relational knowledge from diagrams," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4167–4175.

[43] Z. Chen, Y. Wang, Q. Wang, Y. Wang, and H. Qu, "Towards automated infographic design: Deep learning-based auto-extraction of extensible timeline," *IEEE transactions on visualization and computer graphics*, vol. 26, no. 1, pp. 917–926, 2019.

[44] C. Rother, V. Kolmogorov, and A. Blake, "" grabcut" interactive foreground extraction using iterated graph cuts," *ACM transactions on graphics (TOG)*, vol. 23, no. 3, pp. 309–314, 2004.
[45] J. Wu, C. Wang, L. Zhang, and Y. Rui, "Offline sketch parsing via shapeness estimation," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.
[46] X.-L. Yun, Y.-M. Zhang, J.-Y. Ye, and C.-L. Liu, "Online handwritten diagram recognition with graph attention networks," in *International Conference on Image and Graphics*. Springer, 2019, pp. 232–244.
[47] D. Haehn, J. Tompkin, and H. Pfister, "Evaluating 'graphical perception'with cnns," *IEEE transactions on visualization and computer graphics*, vol. 25, no. 1, pp. 641–650, 2018.
[48] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[49] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[50] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.
[51] L. Giovannangeli, R. Bourqui, R. Giot, and D. Auber, "Toward automatic comparison of visualization techniques: Application to graph visualization," *Visual Informatics*, 2020.
[52] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.
[53] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.
[54] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7794–7803.
[55] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.
[56] O. Oktay, J. Schlemper, L. L. Folgoc, M. Lee, M. Heinrich, K. Misawa, K. Mori, S. McDonagh, N. Y. Hammerla, B. Kainz *et al.*, "Attention u-net: Learning where to look for the pancreas," in *Proceedings of the 1st Conference on Medical Imaging with Deep Learning (MIDL 2018)*, 2018, pp. 1–10.
[57] L. Wang, J. Giesen, K. T. McDonnell, P. Zolliker, and K. Mueller, "Color design for illustrative visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1739–1754, 2008.
[58] B. Caldwell, M. Cooper, L. G. Reid, G. Vanderheiden, W. Chisholm, J. Slatin, and J. White, "Web content accessibility guidelines (wcag) 2.0," *WWW Consortium (W3C)*, 2008.
[59] R. Rossi and N. Ahmed, "The network data repository with interactive graph analytics and visualization," in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.
[60] D. H. Kim, E. Hoque, and M. Agrawala, "Answering questions about charts and generating visual explanations," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–13.
[61] C. Lai, Z. Lin, R. Jiang, Y. Han, C. Liu, and X. Yuan, "Automatic annotation synchronizing with textual description for visualization," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–13.
[62] B. Saleh, M. Dontcheva, A. Hertzmann, and Z. Liu, "Learning style similarity for searching infographics," in *Proceedings of the 41st Graphics Interface Conference*, 2015, pp. 59–64.
[63] M. Bostock, V. Ogievetsky, and J. Heer, "D^3 data-driven documents," *IEEE transactions on visualization and computer graphics*, vol. 17, no. 12, pp. 2301–2309, 2011.
[64] J. D. Hunter, "Matplotlib: A 2d graphics environment," *IEEE Annals of the History of Computing*, vol. 9, no. 03, pp. 90–95, 2007.
[65] S. Van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu, "scikit-image: image processing in python," *PeerJ*, vol. 2, p. e453, 2014.
[66] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao, "Detecting text in natural image with connectionist text proposal network," in *European conference on computer vision*. Springer, 2016, pp. 56–72.
[67] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, pp. 91–99, 2015.
[68] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 11, pp. 2298–2304, 2016.
[69] F. Böschen, T. Beck, and A. Scherp, "Survey and empirical comparison of different approaches for text extraction from scholarly figures," *Multimedia Tools and Applications*, vol. 77, no. 22, pp. 29 475–29 505, 2018.
[70] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of operations research*, vol. 134, no. 1, pp. 19–67, 2005.
[71] X. Li, X. Sun, Y. Meng, J. Liang, F. Wu, and J. Li, "Dice loss for data-imbalanced nlp tasks," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 465–476.
[72] R. L. Wilder, "Evolution of the topological concept of "connected"," *The American Mathematical Monthly*, vol. 85, no. 9, pp. 720–726, 1978.
[73] R. M. Haralick, S. R. Sternberg, and X. Zhuang, "Image analysis using mathematical morphology," *IEEE transactions on pattern analysis and machine intelligence*, no. 4, pp. 532–550, 1987.
[74] B. Alper, B. Bach, N. Henry Riche, T. Isenberg, and J.-D. Fekete, "Weighted graph comparison techniques for brain connectivity analysis," in *Proceedings of the SIGCHI conference on human factors in computing systems*, 2013, pp. 483–492.
[75] K.-P. Yee, D. Fisher, R. Dhamija, and M. Hearst, "Animated exploration of graphs with radial layout," in *Proc. IEEE InfoVis 2001*, 2001, pp. 43–50.
[76] C. C. Gramazio, D. H. Laidlaw, and K. B. Schloss, "Colorgorical: Creating discriminable and preferable color palettes for information visualization," *IEEE transactions on visualization and computer graphics*, vol. 23, no. 1, pp. 521–530, 2016.
[77] C. Cook, F. Heath, R. L. Thompson, and B. Thompson, "Score reliability in webor internet-based surveys: Unnumbered graphic rating scales versus likert-type scales," *Educational and Psychological Measurement*, vol. 61, no. 4, pp. 697–706, 2001.
[78] M. Liu and F. G. Conrad, "Where should i start? on default values for slider questions in web surveys," *Social Science Computer Review*, vol. 37, no. 2, pp. 248–269, 2019.
[79] F. Chollet *et al.*, "Keras," https://github.com/keras-team/keras, 2015.
[80] D. Zhou, J. Fang, X. Song, C. Guan, J. Yin, Y. Dai, and R. Yang, "Iou loss for 2d/3d object detection," in *2019 International Conference on 3D Vision (3DV)*. IEEE, 2019, pp. 85–94.
[81] M. A. Rahman and Y. Wang, "Optimizing intersection-over-union in deep neural networks for image segmentation," in *International symposium on visual computing*. Springer, 2016, pp. 234–244.
[82] Q. Wang, Z. Chen, Y. Wang, and H. Qu, "A survey on ml4vis: Applying machinelearning advances to data visualization," *IEEE Transactions on Visualization and Computer Graphics*, 2021.
[83] L. Yuan, W. Zeng, S. Fu, Z. Zeng, H. Li, C.-W. Fu, and H. Qu, "Deep colormap extraction from visualizations," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2021.
[84] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.
[85] W. S. Cleveland, *The elements of graphing data*. Wadsworth Publ. Co., 1985.

Sicheng Song received his B.Eng. from Hangzhou Dianzi University, China, in 2019. He is working toward the Ph.D. degree with East China Normal University, Shanghai, China. His main research interests include information visualization and visual analysis.



Chenhui Li received Ph.D. from the Department of Computing at Hong Kong Polytechnic University, in 2018. He is an associate professor with the School of Computer Science and Technology at East China Normal University. He received ICCI*CC Best Paper Award (2015) and SIGGRAPH Asia Sym. Vis. Best Paper Award (2017). He has served as a local chair in VINCI2019. He works on the research of information visualization and computer graphics.

Dong Li received her B.Eng. from Zhejiang University of Technology, in 2019. He received his Master degree from East China Normal University, in 2022. His main research interests include information visualization and visual analysis.

Juntong Chen received his B.Eng. from East China Normal University, in 2022. He is working toward the Master degree with East China Normal University, Shanghai, China. His main research interests include information visualization and visual analysis.

Changbo Wang is a professor with the School of Computer Science and Technology, East China Normal University. He received his Ph.D. degree at the State Key Lab of CADCG of Zhejiang University in 2006. He was a post-doctor of the State University of New York in 2010. His research interests mainly include computer graphics, information visualization, visual Analytics, etc. He is serving as the Young AE of Frontiers of Computer Science, and PC member for several international conferences.