

This is an Open Access document downloaded from ORCA, Cardiff University's institutional repository: <https://orca.cardiff.ac.uk/id/eprint/141993/>

This is the author's version of a work that was submitted to / accepted for publication.

Citation for final published version:

Chen, Changjian, Wang, Zhaowei, Wu, Jing , Wang, Xiting, Guo, Lan-Zhe, Li, Yu-Feng and Liu, Shixia 2021. Interactive graph construction for graph-based semi-supervised learning. IEEE Transactions on Visualization and Computer Graphics 27 (9) , pp. 3701-3716. 10.1109/TVCG.2021.3084694

Publishers page: <http://dx.doi.org/10.1109/TVCG.2021.3084694>

Please note:

Changes made as a result of publishing processes such as copy-editing, formatting and page numbers may not be reflected in this version. For the definitive version of this publication, please refer to the published source. You are advised to consult the publisher's version if you wish to cite this paper.

This version is being made available in accordance with publisher policies. See <http://orca.cf.ac.uk/policies.html> for usage policies. Copyright and moral rights for publications made available in ORCA are retained by the copyright holders.



Interactive Graph Construction for Graph-Based Semi-Supervised Learning

Changjian Chen, Zhaowei Wang, Jing Wu, Xiting Wang, Lan-Zhe Guo, Yu-Feng Li, Shixia Liu

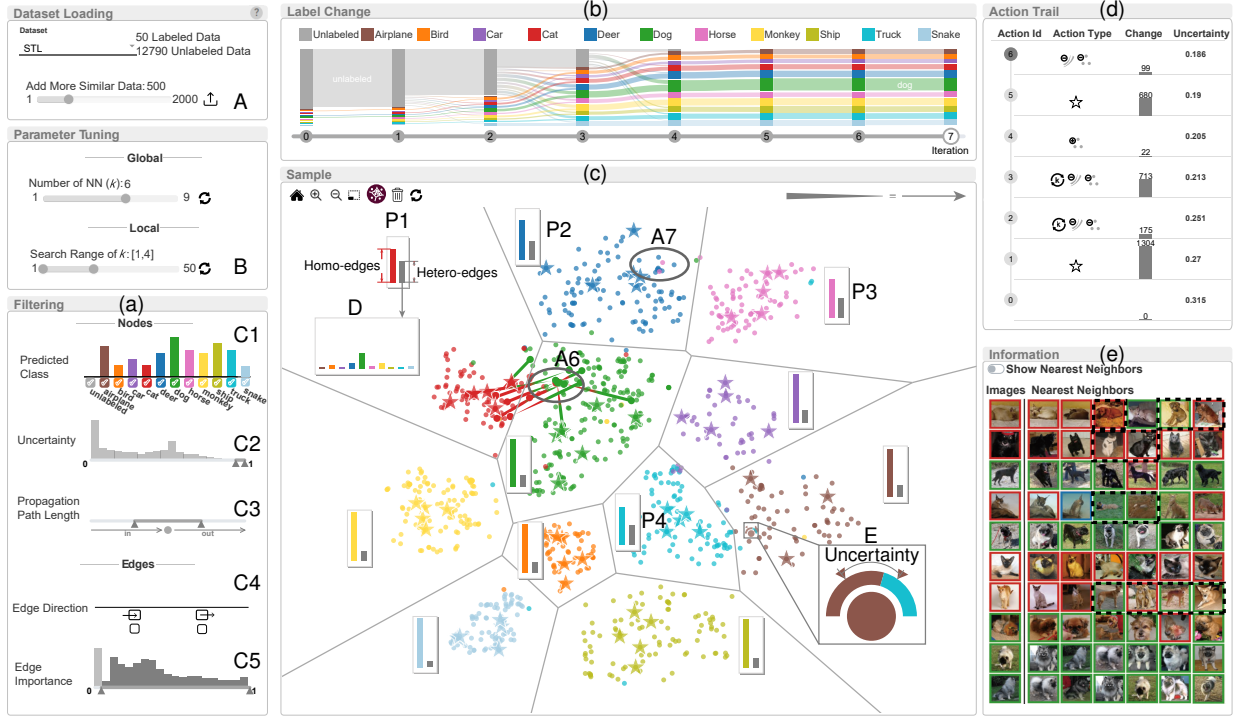


Fig. 1: DataLinker: (a) the Filtering panel helps focus on nodes and edges of interest; (b) the Label Change view shows label changes as an evolving river; (c) the Sample view displays samples as a combination of a scatterplot, a node-link diagram, and a bar chart; (d) the Action Trail records the modification history; (e) the Information view shows the image content of selected samples and their nearest neighbors.

Abstract—Semi-supervised learning (SSL) provides a way to improve the performance of prediction models (e.g., classifier) via the usage of unlabeled samples. An effective and widely used method is to construct a graph that describes the relationship between labeled and unlabeled samples. Practical experience indicates that graph quality significantly affects the model performance. In this paper, we present a visual analysis method that interactively constructs a high-quality graph for better model performance. In particular, we propose an interactive graph construction method based on the large margin principle. We have developed a river visualization and a hybrid visualization that combines a scatterplot, a node-link diagram, and a bar chart to convey the label propagation of graph-based SSL. Based on the understanding of the propagation, a user can select regions of interest to inspect and modify the graph. We conducted two case studies to showcase how our method facilitates the exploitation of labeled and unlabeled samples for improving model performance.

Index Terms—Semi-supervised learning, unlabeled samples, graph quality

1 INTRODUCTION

The success of supervised learning relies on a large number of labeled samples. However, in many applications (e.g., medical

image classification), the labeling process is often too tedious to keep up with the rate of data acquisition, which results in a large amount of data with only a small number of labels. Semi-supervised learning (SSL) provides a way to improve machine learning performance via the usage of unlabeled samples. An effective and widely used method involves constructing a graph to describe the relationship between labeled and unlabeled samples so that label information can be propagated from labeled samples to unlabeled samples [1]. Graph-based SSL (GSSL) has been continually

- C. Chen, Z. Wang, and S. Liu are with School of Software, BNRist, Tsinghua University. S. Liu is the corresponding author.
- J. Wu is with Cardiff University.
- X. Wang is with Microsoft Research Asia.
- L.-Z. Guo and Y.-F. Li are with Nanjing University.

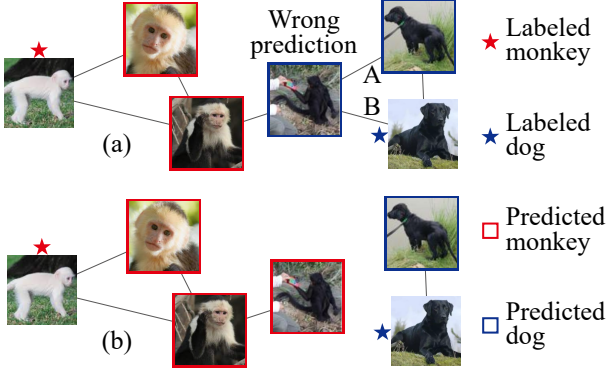


Fig. 2: (a) An example of a k NN graph where $k = 2$. The black monkey in the middle is predicted to be a dog due to the wrong edges A and B; (b) the wrong edges are removed when the k values of dogs and the misclassified monkey are set as 1.

improved since it was first proposed. For example, methods based on deep neural network models (Planetoid [2], graph convolutional network [3], etc.) have achieved state-of-the-art performance [4], [5]. For GSSL methods, graph quality is widely recognized as a key factor that significantly affects the learning performance [1], [4], [6]. In a high-quality graph, the edges correctly capture the similarity relationships between samples, and thus give the model strong generalization ability. Moreover, the model always performs better than direct supervised learning with labeled samples only. By comparison, in a low-quality graph, there are a significant number of wrong edges that connect samples in different classes (e.g., Figs. 2A and 2B). And this will adversely affect the model performance. Although the importance of graph quality is widely recognized, previous studies either assume that the graph structure is accurate or require a lot of labeled samples and computational resources to search for the graph structure [4]. It is still an unsolved problem to construct high-quality graphs in a label-efficient manner.

There are two challenges to solve this problem. First, the small number of labeled samples is insufficient to provide a reliable judgment of the graph quality. There are a large number of graphs that can fit the limited number of labeled samples but behave very differently on unlabeled ones [1]. Although some criteria, such as the large margin principle [7], can automatically exclude some high-risk graphs, the search space for graphs is still extensively large. Second, automatic algorithms are largely based on assumptions (e.g., nearest neighbors having similar labels) and work like a “black box.” With the increase in data quantity and complexity, these assumptions can hardly hold true for all data. For example, in a k NN graph, a global k is applied to all training data, but local regions often require different k values. It is more appropriate to adaptively set the k values based on local properties. For example, setting $k = 2$ for all the samples in Fig. 2 results in the wrong connections between the dogs and the black monkey (Fig. 2(a)). Locally changing the k values of these three samples to 1 will remove the wrong connections and result in a better quality graph (Fig. 2(b)). However, automatic GSSL lacks the flexibility for this kind of graph construction. The large search space, coupled with automatic algorithms, results in little control over the learning results. It is desirable to 1) open the “black box” to understand how the graph structure affects model performance, and; 2) leverage expert knowledge in the construction of high-quality graphs.

To achieve these goals, we have developed DataLinker, an

interactive visual analysis tool that helps machine learning experts 1) explore the graph structure and understand the label propagation in GSSL, and 2) participate in the graph construction process. To better understand the graph structure and how it affects model performance, both the propagation of labels and the spatial distribution of samples are required. Thus, we provide two coordinated views: a Label Change view (Fig. 1(b)) that shows an overview of the label propagation as an evolving river, and a Sample view (Fig. 1(c)) that shows the spatial distribution of samples in a hybrid visualization consisting of a scatterplot, a node-link diagram, and bar charts. The two views work together, with the Label Change view providing guidance on problem-prone samples that, upon selection, will be highlighted in the Sample view for further examination and modification. A Filtering panel (Fig. 1(a)) is also provided to filter edges/nodes according to their attributes, such as edge importance, node uncertainty, etc., to help quickly identify the important parts. By a coarse-to-fine exploration strategy, the whole interface (Fig. 1) facilitates experts to identify which part of the graph may cause performance deterioration and modify the graph structure locally. This greatly reduces the search space for high-quality graphs.

We conducted two case studies with experts on the STL-10 dataset for image classification [8] and on retinal OCT images for medical diagnosis [9]. Both case studies show that DataLinker allows experts to gain a better understanding of the data and data relationships, and in turn to make informed changes to the graph structure. With the constructed high-quality graph, model performance is improved. The demo and the source code are available at <http://datalinker.thuvis.org/>.

The main contributions of this work are:

- **A coordinated and hybrid visualization** to support the exploration of large scale graphs and facilitate the understanding of label propagation in GSSL.
- **An interactive graph construction method** that takes into consideration local data properties and leverages expert knowledge to construct high-quality graphs.
- **A visual analysis tool** based on hybrid visualization, interactive graph construction, and GSSL for improving model performance.

2 RELATED WORK

Our work is related to integrating interactive visualization with machine learning methods to acquire high-quality data. In the field of visualization, many methods have been proposed to improve data quality. Based on whether the data is labeled, the relevant work can be classified into two categories: improving the quality of noisy labeled samples and unlabeled samples [10].

Improving the quality of noisy labeled samples. A cost-effective way to collect labels is crowdsourcing. As the crowdsourced annotations are usually noisy [11], [12], various methods have been proposed to improve their quality. Park *et al.* [13] developed C^2A , a visual analysis tool to facilitate the detection of polyps in virtual colonoscopy videos by leveraging crowdsourced labels. By interactively exploring crowdsourced labels and worker behavior with C^2A , doctors can discard most polyp-free video segments and focus on those that are likely to have polyps. Willett *et al.* [14] proposed a crowd-assisted clustering method to detect and remove redundant explanations provided by crowd workers. CRICTO [15] was developed for the sensemaking of text data by constructing a graph that represents entity connections annotated by workers. Analysts can generate cohesive hypotheses by interactively

exploring the graph. LabelInspect [16] focuses on identifying uncertain instances and unreliable workers to improve the quality of crowdsourced annotations. Based on expert verification, more instances and workers are recommended for validation by an interactive and iterative procedure. In practice, many datasets (e.g., ImageNet [17]) have no information about crowd workers. To handle data quality problems not restricted to crowdsourcing settings, Xiang *et al.* [18] developed a scalable data correction algorithm to propagate the labels of trusted items to other unverified items. A hierarchical visualization was proposed to facilitate the exploration and identification of trusted items.

Improving the quality of unlabeled samples. All the methods mentioned above require labels that are noisy. However, many datasets do not even have noisy labels. To tackle this, various methods have been proposed to improve the quality of unlabeled samples. A SOM-based visualization [19] was developed where similar images are placed together. Users can label multiple similar images at the same time. VASSL [20] was proposed to detect and label social spambot groups on social media. Five coordinated views are utilized to display similarities between accounts from different perspectives to facilitate identifying clusters with anomalous behavior. Recently, a variety of methods have been proposed for integrating learning models with visualizations to promote interactive visual labeling [21]. Inter-active labeling was first proposed by Hoferlin *et al.* [22] to enhance active learning with human knowledge. It not only enables users to query data for labeling via active learning, but also allows better understanding and refining of the classification model via visualizations. Dennig *et al.* [23] provided FDIVE to detect the best-fitting features and distance functions based on labels provided by users. The features and distance functions are then used to train a SOM-based relevance model, which is visually explorable and can be further refined by providing more labels. VIANA [24] developed a language model to recommend text fragments for annotation in argumentation mining tasks. In this work, layered visual abstractions were designed to support five relevant analysis tasks required for text fragment annotation. A unified visual interactive labeling process was proposed by Bernard *et al.* [21]. Experiments conducted by Bernard *et al.* [25] showed the superiority of user-centered visual interactive labeling over model-centered active learning. Bernard *et al.* [26] also ran a quantitative analysis of user strategies for selecting samples in the labeling process. Results show that data-based user strategies (clusters, dense areas) work well in early phases, and model-based user strategies (e.g., class separation) perform better in later phases.

Although these active-learning-based methods improve the data quality to some extent, they may deteriorate the performance due to some wrong similarity relationships between samples. In our work, the proposed interactive graph construction method complements these methods by providing high-quality relationships between samples. In particular, we designed a river visualization and a hybrid visualization to facilitate the understanding of how the graph structure, both nodes and edges, affects GSSL model performance. An interactive graph construction method is also provided to help experts improve graph quality in a coarse-to-fine manner.

3 BACKGROUND: GSSL MODELING

There are two steps in GSSL modeling: 1) graph construction that builds a k NN graph to describe the relationships between labeled and unlabeled samples; and 2) label propagation that propagates labels from labeled samples to unlabeled ones along graph edges.

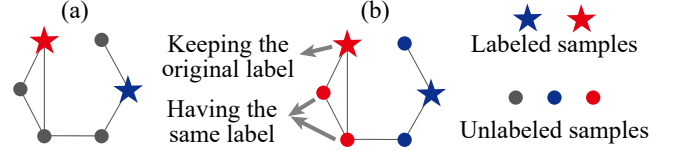


Fig. 3: An example of label propagation: (a) a graph with two labeled samples; (b) the final prediction.

k NN graph construction. Several automatic, adaptive k NN graph construction methods [1], [27], [28], [29], [30], [31] have been developed, which adaptively choose nearest neighbors for local regions. However, these methods are easily overfitted when labeled data is scarce [27], [31]. Previous studies have shown that the traditional k NN graph construction method is more robust [27], [31] and widely used [1]. Therefore, we employ the traditional method in DataLinker. Given n samples where l samples are labeled, and the rest u samples ($u = n - l$) are unlabeled, an initial k NN graph is constructed based on the similarities between them. Similarities between samples are measured by the cosine similarities between the sample features. State-of-the-art deep neural network models, such as the pretrained models on ImageNet [32] or deep SSL models [33], are used for feature extraction, which has been shown to achieve state-of-the-art performance [4], [5]. With the calculated similarities, each sample's k nearest neighbors are determined, and the k NN graph is constructed. The weights of edges are typically set to the sample similarities or a constant 1. In practice, the two approaches have achieved similar results [34], and we pick up the later one as our implementation. A common way of achieving high-quality graph construction is to maintain sparsity and make sure that each connected component in the graph has at least one labeled sample [34]. Thus, in our implementation, parameter k is set to be the smallest integer that ensures every connected component has at least one labeled sample.

Label propagation. With the constructed graph, label propagation assigns labels to samples along the graph edges. It aims to ensure that 1) similar samples have the same label and 2) the labeled samples keep their original labels, as shown in Fig. 3. The predicted labels are represented by a label matrix $\mathbf{F}_{n \times c}$ where each row \mathbf{F}_i is the label vector of sample i . The j -th element of \mathbf{F}_i represents how likely sample \mathbf{x}_i belongs to class j . Label propagation is achieved by optimizing the following cost function:

$$\min_{\mathbf{F}} \frac{1}{2} \left(\sum_{i,j=1}^{l+u} \mathbf{W}_{ij} \left\| \frac{1}{\sqrt{d_i}} \mathbf{F}_i - \frac{1}{\sqrt{d_j}} \mathbf{F}_j \right\|^2 \right) + \mu \sum_{i=1}^l \|\mathbf{F}_i - \mathbf{Y}_i\|^2 \quad (1)$$

The first term is a manifold regularization requiring similar samples to have similar predictions. The second term ensures the predicted labels of labeled samples are consistent with their provided ones. μ is a weight to balance the two terms. Matrix $\mathbf{W}_{n \times n}$ records the edge weights. $\mathbf{W}_{ij} = 1$ if edge $\mathbf{x}_i \mathbf{x}_j$ exists, otherwise $\mathbf{W}_{ij} = 0$. d_i is the sum of the i -th row of \mathbf{W} , which is used to normalize the label vector of sample i . \mathbf{Y} is the ground truth label matrix for labeled samples. This optimization problem has a closed-form solution: $\mathbf{F} = (1 - \alpha \mathbf{S})^{-1} \mathbf{Y}$, where $\alpha = (1 + \mu)^{-1}$, and \mathbf{S} is the symmetric normalized Laplacian matrix of \mathbf{W} . However, when matrix \mathbf{W} is large and dense, the matrix inversion in the closed-form solution is time-consuming. A mathematically equivalent solution is to solve it iteratively [35]: $\mathbf{F}^{(t+1)} = \alpha \mathbf{S} \mathbf{F}^{(t)} + (1 - \alpha) \mathbf{Y}$. \mathbf{F} is initialized to \mathbf{Y} . At each iteration, the label of each sample is propagated to its

nearest neighbors. This iterative process repeats until it converges. According to the previous study [36], the iterative solution is faster when \mathbf{W} is dense and no slower than the closed-form solution when \mathbf{W} is sparse. Thus, the iterative solution is a safer choice and is adopted in our implementation.

4 DESIGN OF DATALINKER

We designed DataLinker during an eight-month collaboration with two groups of machine learning experts. Based on interviews and discussions, we gained an understanding of the current practices, major challenges, and user needs in GSSL. We then distilled the requirements and designed the system with the experts.

4.1 Requirement Analysis

To derive the requirements, we closely worked with two groups of experts. The groups varied in terms of experience in GSSL and their application goals. The first group consisted of two experts (E_1 and E_2) specializing in GSSL. E_1 is a professor and has studied GSSL for over ten years. He has developed state-of-the-art GSSL methods and published papers in top-tier machine learning conferences. E_2 is a Ph.D. student who has studied GSSL for three years. E_1 and E_2 are familiar with the most advanced methods for exploiting unlabeled samples and are eager to extend the horizon of GSSL by finding new insights. The second group included two machine learning researchers (E_3 and E_4) who have a basic understanding of GSSL. Focusing on natural language processing and computer vision, respectively, the two experts want to leverage unlabeled samples to improve model performance. For simplicity, we took the k NN graph as an example during the requirement analysis. Despite the diverse experience in GSSL and different goals, we identified two common challenges:

C1. How does one determine which unlabeled samples are useful for graph construction and when labels must be provided? All experts mentioned that the quality of unlabeled samples is crucial for building a high-quality graph. However, since the unlabeled samples are crawled from the website, they are usually of poor quality, *e.g.*, contain noises that are irrelevant to the task. For example, E_1 said, “When the task is to classify whether an image contains a dog or a cat, it is very likely that the unlabeled samples will contain images of rabbits or tigers. This results in degenerated model performance.” It is also difficult to know when unlabeled samples can benefit graph construction and when labels must be added to improve graph quality (E_1 to E_3).

C2. How does one debug and refine the graph structure effectively? Currently, graph refinement is still based on a time-consuming trial-and-error procedure (E_1 to E_4). There is no way to fully understand how the key parameter k impacts the graph, what the learned edges are, and how the learned edges impact the final predictions. Compared with supervised learning, model (graph) refinement is even more difficult for GSSL, since there is no effective quantitative criterion for thoroughly measuring the graph quality (E_1 to E_3).

Both challenges require a better understanding and more effective refinement of the learned graph, whose structure is influenced by its nodes, *i.e.*, unlabeled or labeled samples ($C1$), edges, and the key parameter k ($C2$). Based on these observations, we then derived the following requirements through six 60-minute participatory design sessions with the experts. We divided the requirements into two groups, according to whether they focus on understanding and debugging or graph refinement.

To better **understand and debug** the graph, the experts wanted to understand how the graph structure affects the final prediction (understanding), with an emphasis on the problematic aspects (debugging). For example, E_2 said, “I hope that the system can show the deficiency of the graph, for example, which samples cannot be easily classified and which parts of the graph lead to the issue.” To achieve this goal, we need to display the graph in terms of both temporal influence and spatial structure, *i.e.*, how the final prediction is derived by propagating labels along the graph edges during iterations (temporal), and how the distribution of the nodes and edges impact the prediction results (spatial). Accordingly, we distilled two requirements:

R1. Displaying how labels propagate along the graph during iterations. All experts agreed that it is important to understand the temporal change in labels, *i.e.*, how the labels propagate along the graph edges. E_1 and E_2 said that this could help identify some problematic samples, *e.g.*, samples whose labels frequently change during iterations.

R2. Revealing how the graph structure impacts the final results at different levels of detail. In particular, the experts need to know how the distribution of nodes (both unlabeled and labeled) impacts the edges, and how both node and edge distributions contribute to the final prediction results. Since the number of nodes and edges is large, it is important that both overview and details can be obtained on demand. More importantly, problematic samples (*e.g.*, samples with high uncertainty) and important edges should be highlighted so that the experts can debug quickly. For example, E_1 said, “I would like to know which samples are the riskiest and why.”

To better **refine** the model, the experts required to directly make modifications in the visualization. According to the discussions on challenges, the most desirable functions are related to the refinement of graph nodes ($C1$), edges, and the key parameter k ($C2$).

R3. Guided refinement of graph nodes, including both unlabeled and labeled samples. According to the experts, three types of modifications on the graph nodes are needed: adding labels, adding unlabeled samples of specific classes, and deleting noisy samples. Since the number of samples is very large, it is important that this refinement process is guided by the designed visualization.

R4. Efficient modification of graph edges. Graph edges have a large impact on the final results. According to the experts, it is very possible that some edges wrongly connect nodes in different classes in a k NN graph or are missing from the graph. E_3 said, “It will be very cool if we can quickly identify the incorrect edges and remove their influence on the prediction.” To ensure that the edges can be modified efficiently, we need to support different granularities of interactions, *i.e.*, both local and individual edge modification.

R5. Interactive tuning of the key parameter k and observe its influence on the graph structure. In current practice, experts can only tune k globally (*i.e.*, for all samples). They are intrigued by being allowed to directly change k in the system both globally and locally for a group of samples.

4.2 System Overview

Motivated by these requirements, we designed a visual analysis tool, DataLinker, to assist the construction of a high-quality graph. Fig. 4 shows the overview of DataLinker. It contains two modules: GSSL modeling and visualization. The two modules work together to support a coarse-to-fine strategy for constructing a high-quality graph.

Given a set of labeled and unlabeled samples, state-of-the-art deep neural networks, such as the pretrained models on

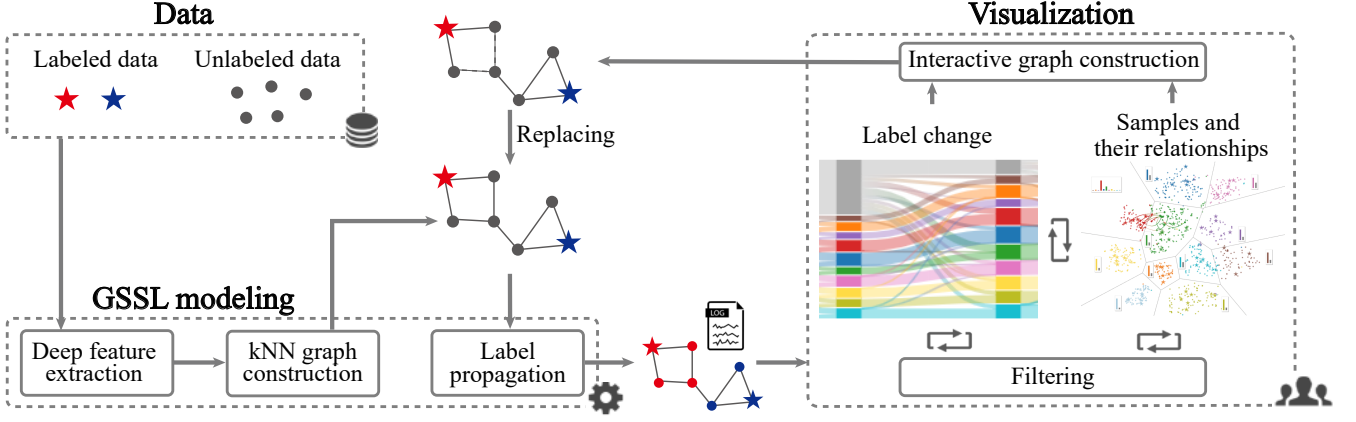


Fig. 4: DataLinker overview. The GSSL modeling module constructs an initial k NN graph and propagates the labels from labeled samples to unlabeled ones; the visualization module helps understand how the graph structure influences the final predictions and interactively construct a high-quality graph.

ImageNet [32], are used to extract features from input data. Then based on the cosine similarities between the extracted features of samples, an initial k NN graph is constructed, and labels are propagated through the graph. To help experts better understand how the graph structure influences the final predictions and interactively construct a high-quality graph, a visualization module is provided. By examining how labels propagate through iterations in the Label Change view ($R1$), experts can select problematic samples, such as those propagated to at later iterations, and turn to the Sample view for further analysis. The Sample view utilizes a hybrid visualization to present the distributions of both the nodes and the edges of the graph. Filters are provided to help experts focus on important information, *e.g.*, edges that have large impacts on the final predictions. The two views, along with the filters, help experts explore from a global overview to local details and understand how labels are propagated through the graph ($R2$).

Based on the understanding, a coarse-to-fine strategy is supported to construct a high-quality graph interactively. Experts can adjust labels and augment unlabeled samples to make sure that the graph nodes are sufficient and appropriate ($R3$). After that, the key parameter k controlling the graph structure can be modified in local regions with the proposed interactive graph construction method ($R5$). The graph can be further refined by modifying individual nodes and edges ($R3$, $R4$). The process of interactive graph construction, label propagation, and visual exploration iterate until a high-quality graph is constructed.

5 DATALINKER VISUALIZATION

In this section, we describe the visual design of DataLinker and the interactions for exploring and modifying graph structures.

5.1 Visual Design

The visualization consists of two major components: 1) a visualization of label changes to give an overview of the label propagation process ($R1$); 2) a hybrid visualization of samples to reveal both the samples and their relationships in the graph structure, and how they influence the label propagation process ($R2$). An Information view is also provided to assist the exploration. The Information view displays the selected samples and their nearest neighbors.

5.1.1 Label Change

As discussed in Sec. 4.1, experts expressed their need to understand the label propagation process, which can help identify the problematic samples. For example, a frequent label change of a sample indicates that the label of this sample is influenced by the samples from different classes. In such a case, this sample may have wrong edges that lead to misclassification [34]. Thus, the experts expressed their need to examine such samples. The iterative solution, as mentioned in Sec. 3, can naturally reveal the temporal changes in labels, which is a further advantage over the closed-form solution. With the iterative solution, the visualization needs to present 1) how the samples are distributed among the classes at each iteration, and 2) how the distribution changes through iterations. To this end, we propose to use a river flow metaphor [37], which has been used to present dynamic change of topics in visual text analysis [38], [39].

At each iteration, a stacked bar (Fig. 5A) is used to represent the distribution of samples among classes, where the bar color indicates the class, and the bar height encodes the number of samples. The top bar with the color “gray” represents samples that haven’t been propagated to.

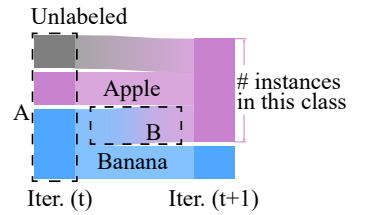


Fig. 5: Label change.

Between consecutive iterations, connections between stacked bars represent the exchange of samples, and the width of the connection encodes the number of samples. For example, the highlighted connection (Fig. 5B) represents samples that change their labels from “banana” to “apple” from iteration (t) to ($t+1$). The connection color is a blending of colors from one end to the other. From this visualization, experts can have an overview of the label propagation process and quickly identify problematic samples, *e.g.*, samples that are propagated to at later iterations and/or frequently change during iterations.

5.1.2 Samples and Their Relationships

To help experts understand the impact of samples and their relationships on label propagation, both the sample distribution and the graph structure should be displayed simultaneously. We propose a hybrid visual representation that combines a scatterplot,


a node-link diagram, and a bar chart (Fig. 1(c)). In addition, samples are hierarchically organized to support the exploration of a large amount of data.

Hybrid visual representation. The hybrid visual representation consists of a scatterplot, a node-link diagram, and a bar chart. In the visualization, samples are presented as nodes in a 2D scatterplot, while edges reflect the relationships between samples. To facilitate experts in quickly identifying problematic samples, an uncertainty value is calculated for each sample and visually displayed in the **scatterplot**. To effectively display a large number of graph edges, we design the edge visualization at two levels: 1) to individually display edges between samples in local regions of interest (**node-link diagram**), and 2) to statistically convey the overall distribution of edges (**bar chart**). In the following, we will introduce the key components in the hybrid visual representation, *i.e.*, uncertainty-aware scatterplot, individual edges, and edge distribution.

Uncertainty-aware scatterplot. An intuitive way to visualize the samples and their relationships is to use graph layouts, such as force-directed graph layouts. However, our experiments have shown that the sample distribution generated by the force-directed graph layout is cluttered (Fig. 6(a)). Samples in different classes are mixed together, failing to convey the data distribution as required by the experts. This result is also consistent with the findings of previous work that the force-directed graph layout can easily become stuck in a bad local minimum [40], [41]. Moreover, it does not utilize the predicted classes of samples to get better class separation. Another popular method for visualizing sample distribution, t-SNE-based projection, uses an early exaggeration strategy to attract similar samples to be gathered into clusters during initial iterations, which avoids a bad local minimum [42]. Existing studies have shown that it is effective for preserving neighborhoods and clusters [43]. In our implementation, we use a supervised t-SNE [44], which utilizes the predicted classes to reduce the distance between the projected samples of the same class. As shown in Fig. 6, using the supervised t-SNE results in better class separation than using the force-directed graph layout.

During the interviews, experts expressed the need to examine not only the predicted classes of samples but also their prediction uncertainties (R2). Uncertainty is a widely used measure to indicate problematic unlabeled samples when ground truth labels are not available [45]. It has proven to be useful in many tasks [46], [47]. According to the widely used large margin principle [7], low uncertainty usually means better prediction results. Inspired by the uncertainty glyphs in [48], [49], here we also encode uncertainty

in glyphs and display them in place of the nodes in the scatterplot when required. The glyph design is shown in Fig. 1E. The color of the central dot encodes the predicted class. The semicircular ring consisting of multiple colored arcs represents the predicted class distribution. The length of each arc encodes the probability of belonging to the corresponding class. The angle between the two sliders encodes the degree of uncertainty, which is calculated as the entropy of the predicted probabilities. The larger the angle, the higher the uncertainty.

Individual edges. Edges are displayed on demand upon selecting a region of interest. We use tapered edges [50] () to encode their directions where an edge starts at the wide side and ends at the narrow side. The color of an edge is the blending of the predicted class colors at its two ends. As the number of edges is usually large, nearby edges of the same type, *i.e.*, edges with the same starting classes and ending classes, can be grouped to reduce visual clutter and facilitate tracing. We thus turn to the divided edge bundling [51]. It is a force-directed-based method where edges in the same direction attract each other, and edges in opposite directions repel each other. To group edges by types, the same as grouping edges by directions, we utilize the divided edge bundling method to attract edges of the same type and repel edges of different types.

Edge distribution. For each class, the edges connected to samples can be categorized into homogeneous edges (**homo-edges**) connecting samples of the same class and heterogeneous edges (**hetero-edges**) connecting samples of different classes. A higher ratio of hetero-edges indicates a heavier confusion of this class with others. To give an overview of the edge distribution, a bar-chart-based distribution visualization and a Voronoi-based space partition are combined. A bar chart (Fig. 1P1) is displayed for each class showing the homo/hetero-edges distribution within the class. The height of each bar encodes the ratio of homo/hetero-edges, and the color indicates the type (colored: homo-edges, gray: hetero-edges). The hetero-edge bar can be further expanded to show the detailed distribution of edges connecting each of the other classes (Fig. 1D). Clicking on a bar will display the corresponding edges in the Sample view. The Voronoi-based partition aims to allow easy perceptual separation of different classes and link the classes with corresponding bar charts. Each partition should 1) try to include samples of only one specific class; and 2) following Gestalt theory [52], be as convex as possible to make clusters more evident. To satisfy these two constraints, we propose a two-step partition method based on the Voronoi tessellation [53].

The **first step** is to generate an initial partition based on the Voronoi tessellation (Figs. 7(b) and 7(c)). A straightforward way is to generate the Voronoi tessellation on all samples and then merge adjacent Voronoi cells of the same class for generating the partitions. However, this method suffers from high computational costs. To accelerate the process, instead of generating the Voronoi tessellation on all samples, we only use the ones on the boundary of each class for the tessellation. Here we use α -hulls [54] to represent the boundaries (Fig. 7A). The α -hull is a generalization of the convex hull, which can capture the shape more accurately than the convex hull by allowing internal angles to be larger than 180 degrees. As the α -hull is sensitive to outliers [55], we utilize the local outlier factor algorithm [56], one of the most widely-used outlier detection methods [57], to remove outliers for each class before generating its α -hull. With this strategy, the number of samples for the Voronoi tessellation is largely reduced. However, as shown in Fig. 7(c), this step cannot guarantee the convexity of

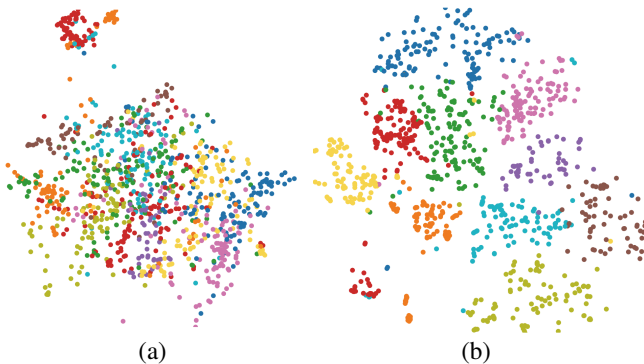


Fig. 6: Sample distributions generated by (a) the force-directed graph layout and (b) the supervised t-SNE.

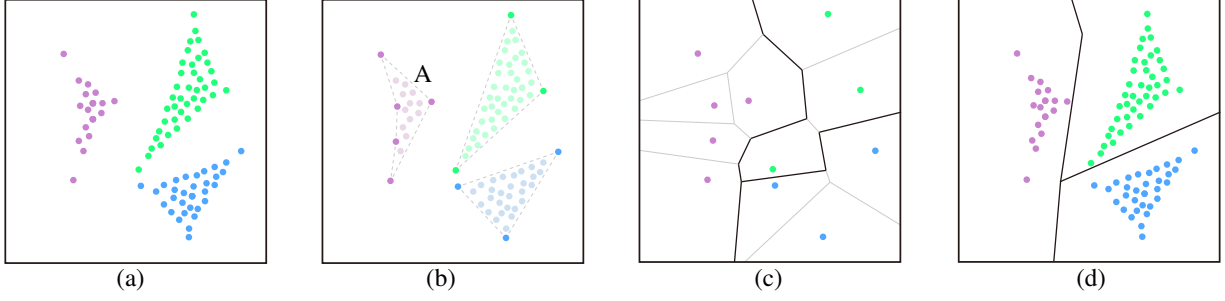


Fig. 7: Illustration of the Voronoi-based partition method: (a) positions of all samples; (b) finding samples on the boundary of each class; (c) generating the Voronoi tessellation for samples on the boundaries and merging adjacent cells of the same class; (d) approximating the polyline-based borders by a set of polylines with a smaller number of vertices.

partitions. The **second step** thus follows to improve the convexity. A direct solution is to replace the polyline-based border of two partitions with a line segment. However, it may introduce *errors* to the partitions, *i.e.*, each partition may contain samples other than of the main class. To balance between error and convexity, we propose to use a polyline with a smaller number of vertices (Fig. 7(d)) as a balance between the original polyline and a line segment. In particular, we define the partition cost as

$$C(P) = \sum_{i=1}^c \{E(P_i) + \beta(1 - V(P_i))\}, \quad (2)$$

where c is the number of partitions; $E(P_i)$ measures the error of partition P_i by the number of misplaced samples in it; $V(P_i)$ is the measure of convexity in P_i using the measure defined in [58]. The idea behind is that the line segment between any two samples in the partition should lie within the partition as well if the partition is convex. The ratio of line segments that lie within measures the convexity of the partition. β is the balancing parameter and is empirically set as 2 in our tool to balance the magnitude of these two measures. Our goal is to minimize cost $C(P)$ by replacing groups of consecutive line segments on the border with straight lines. The optimal solution can be obtained using dynamic programming.

Hierarchy construction. To allow efficient exploration of a large number of samples, we organize the samples in a hierarchy. An uncertainty-biased sampling method [18], [59] is used to build the hierarchy. This sampling method takes both region density and classification uncertainty into consideration and increases the sampling ratio from dense regions with low uncertainty to sparse regions with high uncertainty. As such, it preserves more uncertain samples while maintaining the overall data distribution.

The hierarchy is built in a bottom-top manner. The bottom layer L_0 contains all samples. When sampling from layer L_{l-1} to layer L_l , the sampling rate is set as 25%, and each sample in layer L_{l-1} will be assigned as a child to its nearest sample in layer L_l . This process repeats until the number of samples in the top layer is less than a certain threshold (1,000 in our implementation). During exploration, experts can zoom in/out to navigate the hierarchy.

5.2 Interactive Exploration and Graph Construction

To help experts easily examine the samples of interest and make informed changes to the underlying graph structure, two main interactions, *i.e.*, graph filtering and interactive graph construction, are provided. The first interaction helps identify the samples of interest more efficiently. The second interaction follows a coarse-to-fine strategy and enables experts to make efficient changes to

the graph structure at different levels, including global ($R3$), local ($R5$), and individual levels ($R3$, $R4$).

5.2.1 Graph Filtering

A set of filters (Fig. 1(a)) are provided to help experts find the graph nodes and edges of interest. Scented Widgets [60] are used as the visual guidance for filtering out unimportant information and focusing on the information of interest. Information filtering can be carried out according to five attributes of nodes or edges:

- *Predicted class* allows experts to focus on specific classes of interest (Fig. 1C1);
- *Uncertainty* helps identify unreliable samples with higher prediction uncertainties (Fig. 1C2);
- *Path length* limits the length of the propagation path to be displayed (Fig. 1C3).
- *Edge direction* allows the selective display of edges that start or end in the selected region (Fig. 1C4).
- *Edge importance* enables the display of edges that contribute the most to the final predictions (Fig. 1C5).

To quantitatively measure edge importance, we utilize the AURORA algorithm described in [61]. This algorithm defines the edge importance as the derivative of the predictions with respect to an edge, which approximates the change in prediction scores after removing this edge.

5.2.2 Interactive Graph Construction

Efficient modifications of the graph structure at the global, local, and individual levels are the essential requirements of our experts. For adjusting labels and augmenting unlabeled samples at the global level, and removal of incorrect edges and noisy samples at the individual level, the interactions are straightforward, such as directly deleting incorrect edges from the graph. At the local level, however, interactively determining the parameter k to modify the local graph structures is nontrivial due to the limited labeled samples. To help experts tune parameter k in a local region, a local graph modification method based on the large margin principle [7] is proposed. Combining the visualization and the proposed local graph modification method, a coarse-to-fine strategy is provided to help experts interactively construct high-quality graphs. In addition, an Action Trail is developed to record the modification history and allows experts to return to a previous step.

Local graph modification. A straightforward method to determine the best k value for a local region is to choose the value that performs best on a given set of labeled samples. However, it is often difficult to obtain a certain number of labeled samples in practice. To solve this problem, we utilize the large margin principle [7],

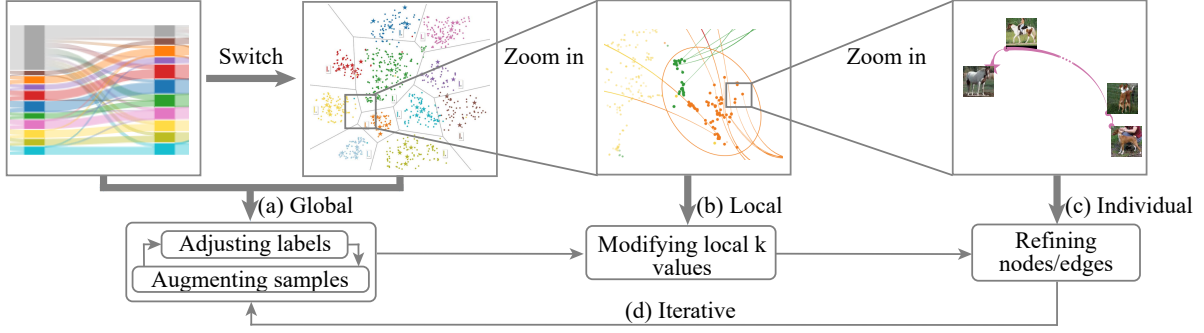


Fig. 8: A typical workflow of DataLinker is a coarse-to-fine graph construction process. Experts usually start from (a) adjusting graph nodes at the global level; then (b) modify local k values according to local properties; they (c) further refine individual nodes with high uncertainty; and (d) start a new round of modification.

which has been shown promoting GSSL model performance [6], to guide the search for the best k . The margin for a sample is the difference between the probabilities of its top two predicted classes. A larger margin indicates a more discriminative prediction. For multi-class classification problems, a large margin is achieved by minimizing the entropy of predictions [62]. Given a set of nodes \mathbf{X} in the local region and a search range $[k_0, k_1]$, a grid search is applied to determine the best k in the range. More specifically, for each k' in the range, a candidate graph is constructed with k' nearest neighbors of nodes in \mathbf{X} . Label propagation is then applied to obtain predictions and the corresponding entropy. The k' with the lowest entropy is the best k . The candidate graph may contain multiple components when the new k' is small. To tackle this issue, we apply a greedy method for connecting all components. Starting with the two largest components, we connect them by connecting the two nodes from each of them with the minimum distance. This process repeats until only one component exists.

Graph construction workflow. Fig. 8 illustrates a typical workflow of DataLinker to construct a high-quality graph interactively. With the initial k NN graph construction, the expert first examines the overall sample distribution to make sure each class has enough labeled and unlabeled samples (**global**). From both the Label Change view and the Sample view, the expert can find problematic samples that may need label adjustment and classes that are with insufficient unlabeled samples (Fig. 8(a)). After the adjustment of samples, the expert then turns to the local modification of graph edges (**local**). From the Sample view, regions with inappropriate k values can be identified (Fig. 8(b)). For example, a small cluster with limited connections to the major cluster of this class may need a larger k . The local graph modification is then used to find the best k for each of these regions and update the graph locally. Further individual refinement will be carried out for the remaining small number of samples with high uncertainty (**individual**). Their propagation paths are first examined (Fig. 8(c)) to understand why the GSSL model makes such predictions on them. Based on that, experts can identify and remove the problematic nodes/edges that result in the wrong predictions. The quality of nodes and edges are mutually influenced. The changes in one (e.g., the nodes) will affect the quality of the other. Therefore it is appropriate to carry out iterative modifications. The experts can start a new round of modifications to further improve the graph quality (Fig. 8(d)).

Action Trail. We provide an Action Trail (Fig. 1(d)) to record the expert’s modification actions and allow them to roll back to a previous step. Each row represents a step that the expert has

made, and the rows are ordered by time stamps. For each step, three kinds of information are recorded: 1) the action types, 2) the number of samples with changed predictions, and 3) the average uncertainty of all samples. With the Action Trail, experts know what kinds of actions they have made and what changes these actions bring. Based on the understanding, they can then stop the modification when there are little changes in the prediction result.

6 CASE STUDY

Two case studies have been conducted to evaluate how DataLinker helps understand the influence of the graph structure on GSSL and improve the learned models by making informed changes to the underlying graphs. We invited two machine learning experts, E_2 and E_4 , to conduct case studies on two datasets, the STL-10 [8] and the OCT (Optical Coherence Tomograph) [9]. STL-10 is a popular dataset for SSL [63], which E_2 has often used in his research. As a researcher in SSL, E_2 has a focus on analyzing the influence of graph structures on final results. E_4 is currently working on a collaborative project on OCT image classification and is familiar with the OCT dataset. His analysis is more performance-driven.

In the two cases, we constructed the preliminary graph structures using the features obtained by deep neural network models. In particular, for the first case with natural images, we employed pre-trained supervised learning models on ImageNet for feature extraction, where ImageNet is a well-known large-scale natural image dataset with broad and diverse image coverage. The features extracted by the pre-trained models on ImageNet have proven helpful with natural image classification tasks [64]. However, for image datasets that are significantly different from the ImageNet dataset, such as OCT images, the extracted features will actually hurt performance [64]. As a result, for the second case involving OCT images, we instead train MixMatch [33], a recent state-of-the-art deep SSL model, on OCT images for feature extraction.

6.1 Case study on the STL-10 dataset

STL-10 consists of 105,000 training images (5,000 labeled images and 100,000 unlabeled images) and 8,000 test images [8]. In this case study, a subset of 12,840 training images (50 labeled images, 12,790 unlabeled images) from this dataset was used for training. Our images comprise 11 classes (airplane, bird, car, cat, deer, dog, horse, monkey, ship, truck, snake), with only five labeled images in each of the first ten classes. Our test images include: 1) all the test images in STL-10 (800 for each of the first ten

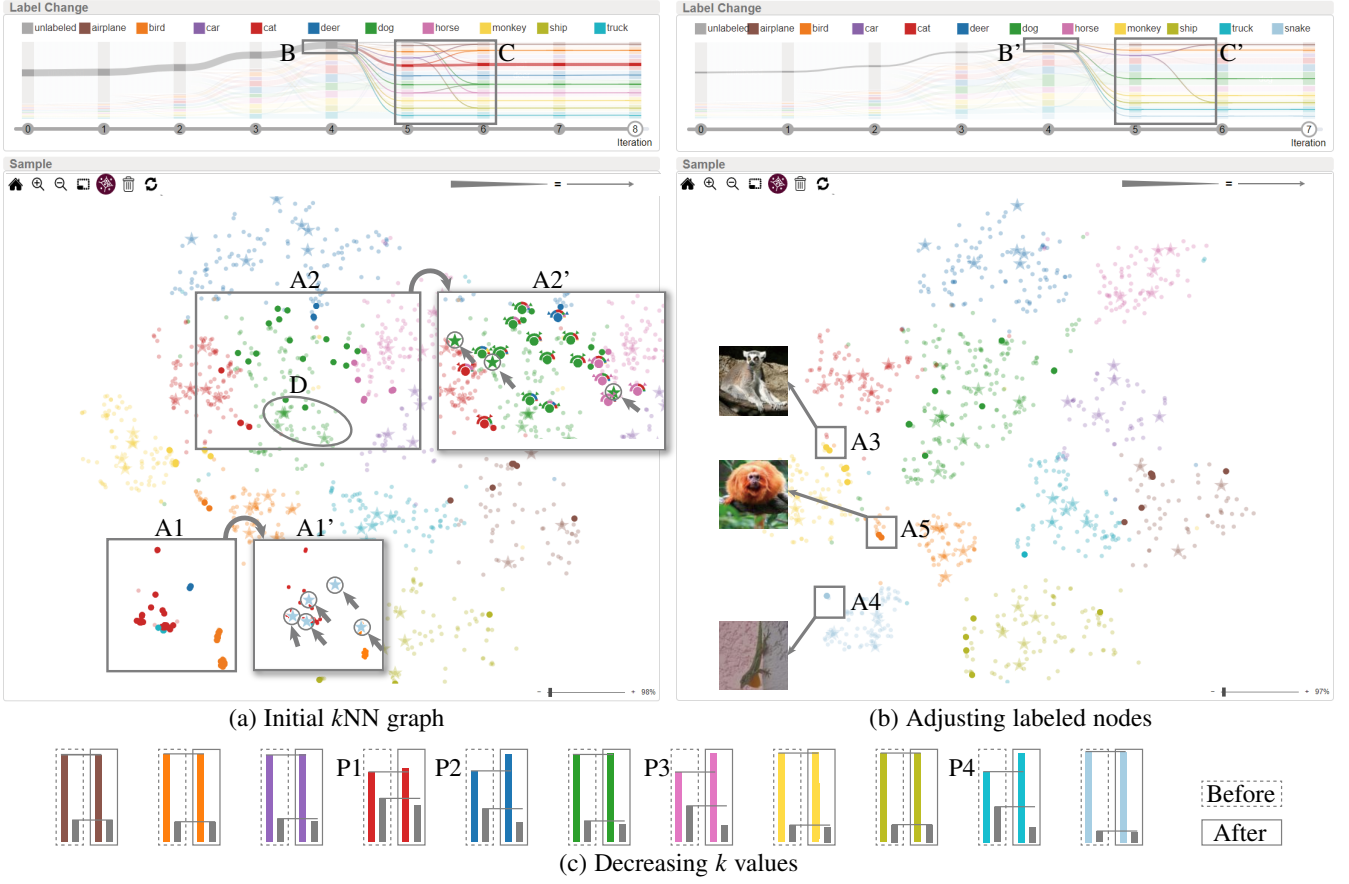


Fig. 9: Label Change view and Sample view after (a) constructing the initial graph; (b) adjusting labels; (c) edge distributions before and after decreasing k values. The ratios of hetero-edges decrease after decreasing k values, but still noticeably high in partition P1.

classes; and 2) additional 100 randomly selected images from the remaining snake class training images. The low ratio of the labeled samples (50/12840) and noisy samples (samples of other classes) makes classification on this dataset a challenging task. Initially, an accuracy of 88.91% was achieved using a state-of-the-art GSSL model developed by Iscen *et al.* [5], which was not satisfying. Thus, E_2 would like to use DataLinker to examine and modify the underlying graph for improving classification accuracy.

Overview of label propagation (requirements $R1$ and $R2$). An initial graph was constructed with $k = 6$, the smallest integer that ensures every connected component in the graph has at least one labeled sample. Labels were propagated from the labeled nodes to unlabeled ones along the constructed graph. To know how the propagation progressed through iterations, E_2 first looked at the Label Change view. He noticed that most samples were propagated to within four iterations except for a small number (Fig. 9B). He also noticed a high ratio of label changes among these samples in later iterations (Fig. 9C), indicating some uncertainty in the predicted labels. E_2 considered these samples might be misclassified and would like to find where they resided in the graph. He selected these samples in the Label Change view and turned his attention to the Sample view.

Adjusting the distribution of labeled nodes (requirements $R2$ and $R3$). The Sample view shows the spatial distribution of samples (graph nodes) in a 2D scatterplot. An even distribution of labeled nodes in each class is expected to construct a high-quality graph. Based on this expectation, E_2 examined the distributions of the

selected samples (highlighted) and the labeled ones (starred) for each class and observed a noticeable deficiency of dominant labeled samples in two regions, as shown in Figs. 9A1 and 9A2. In A1, there were no labeled samples for this cluster. Checking the sample images in A1, E_2 found they were images of snakes, a class that hadn't been considered before. As there were not any labeled samples for this class, these samples were all wrongly predicted. E_2 thus manually labeled five samples (stars with circles in A1') in this region to include this new label into the propagation process. In A2, the highlighted green (dog) samples were relatively far from their labeled ones, which were clustered at a corner of the class (Fig. 9D). "This elongated the propagation paths and reduced the effective information passing," pointed out by E_2 . The high uncertainties (0.6 - 1.0) of these highlighted samples in this region confirmed E_2 's speculation. E_2 then evenly labeled three more dog samples (stars with circles in A2') in region A2. Upon update, most samples were labeled within four iterations, and label changes were also largely reduced (Figs. 9B' and 9C'). The Action Trail showed a reduced overall uncertainty from 0.315 to 0.27 (Fig. 1(d)).

Selecting the remaining unlabeled samples in B' (Fig. 9(b)) and checking their images in the Information view, E_2 found a few more samples from other classes (A3, A4 in Fig. 9(b)). As E_2 considered these samples to be noise, he deleted them directly.

Coarse-to-fine modification of graph edges (requirements $R2$, $R4$, and $R5$). The following steps are executed to interactively modify the graph edges from coarse to fine.

Increasing k . E_2 noticed some monkey images were wrongly predicted. (Fig. 9A5). To find the cause, E_2 zoomed into this region

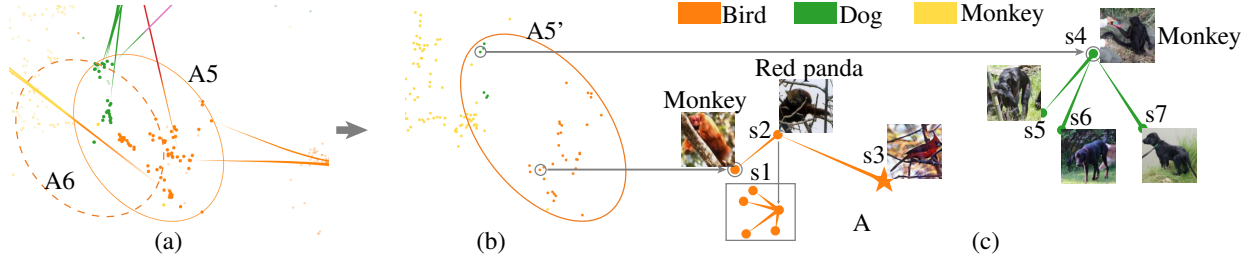


Fig. 10: Local and individual modification: sample and edges of the selected region (a) before and (b) after increasing the local k value; (c) propagation paths of selected samples.

to examine how the labels were propagated in this local region. He noticed two clusters that were all monkey images but wrongly predicted as dogs (green) or birds (orange), as shown in Fig. 10A5. To examine the reason for the incorrect label propagation, E_2 displayed the important edges going into A5. He found a relatively small number of important edges from the monkey (yellow) class compared to those from the bird (orange) or dog (green) classes, indicating a smaller influence. E_2 reasoned that the insufficient influence from the main monkey cluster to this region resulted in the wrong predictions. To increase the influence, E_2 considered increasing the connectivity between the main monkey cluster and the clusters in A5. He thus selected the region around the boundary between A5 and the monkey cluster (Fig. 10A6) and increased the k value in this local region. $k = 26$ was finally determined by a grid search in the range of $[7, 40]$.

Refining. The operation of increasing k locally reduced the number of wrongly predicted monkey images (Fig. 10A5'). However, a few still existed. Given the small number, E_2 decided to examine their label propagation paths individually. Selecting s1, a monkey image wrongly classified as a bird, he noticed its label was propagated from s2, which was further propagated from s3, a labeled bird image. Looking for similarities among the three images, E_2 speculated the same state of “on a tree” connected the three samples and wrongly propagated s3’s label (bird) to s2 and to s1. E_2 wanted to break the connection by either deleting nodes or deleting edges. s2 is a noisy sample whose class (red panda) is not of interest. But checking the important edges starting from s2 (Fig. 10A), E_2 found s2 adversely affected the labeling of several other monkey images. To this end, E_2 deleted s2 to break the connection. s4 in the green cluster was another sample checked by E_2 . It is a black monkey but had its label propagated from two black dogs (s6 and s7). This time, E_2 speculated the same “black” color resulted in the wrong propagation. Checking the important

edges starting from s4, E_2 found s4 further passed the incorrect label to s5, an image of two black monkeys. As labeling s4 would lead to wrong label propagation from s4 to s6 and s7, E_2 decided to delete the incorrect edges from s6 to s4 and from s7 to s4. E_2 continued checking the propagation paths in the two clusters and deleted nodes/edges for another six wrongly predicted nodes. Upon update, the overall uncertainty was reduced from 0.270 to 0.251.

Decreasing k . After the above step, E_2 zoomed back to the top Sample view. He found the confusions between classes were still observed, especially around the cluster boundaries (e.g., Fig. 1A7). To better examine the cause of the confusion, E_2 displayed the partitions and the bar charts. From the bar charts, E_2 found four partitions (P1, P2, P3, P4) had a relatively higher ratio of hetero-edges, indicating heavier confusions. E_2 wanted to find the confused regions, examine their local graphs, and make proper corrections. He started with partition P1, which had the highest ratio of hetero-edges. The detailed distribution of hetero-edges in P1 (Fig. 1D) showed that the cat (red) and the dog (green) classes had the highest confusion. The accordingly displayed important hetero-edges guided E_2 to find where the confusion occurred (Fig. 1A6). E_2 selected the confused region and zoomed in (Fig. 11(a)). From the edge bar chart inside this region (Fig. 11A), he found that the ratio of the hetero-edges was much higher than that of the red class, and many of the samples had high uncertainty (0.6 - 1.0). Checking the nearest neighbors of these uncertain samples in the Information view (Fig. 1(e)), E_2 further observed that many of these samples were wrongly predicted, and they tended to have lower-ranked neighbors in wrong classes (marked with black borders). From these observations, E_2 concluded that $k = 6$ was too large for this local region, and he wanted to reduce the k value in this local region. $k = 2$ was finally set by a grid-search in the range of $[1, 5]$.

Refining. For the small number of remaining uncertain samples shown in Fig. 11(b), E_2 again checked their individual propagation paths and deleted nodes/edges for these samples in the same way as the refining step before.

In the same way, E_2 analyzed the confusions in P2, P3, and P4, and reduced the k values in these local regions. The remaining uncertain samples were further refined. As shown in Fig. 9(c), after this step, hetero-edges in these partitions were reduced. The overall uncertainty was reduced from 0.251 to 0.213.

Incremental exploitation of unlabeled samples (requirement R3). After the modification of labeled nodes and graph edges, a cleaner data distribution with fewer confusions and uncertainties was achieved. However, the bar charts showed the ratio of hetero-edges in partition P1 was still noticeably higher than others (Fig. 9(c)). As the ratio was determined by both the number of hetero-edges and the number of homo-edges, E_2 speculated that the reason might lie in the small number of homo-edges at this

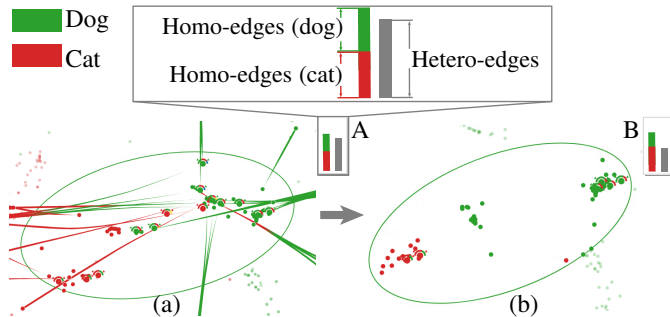


Fig. 11: Sample and edge distributions of the selected region (a) before and (b) after decreasing the local k value.

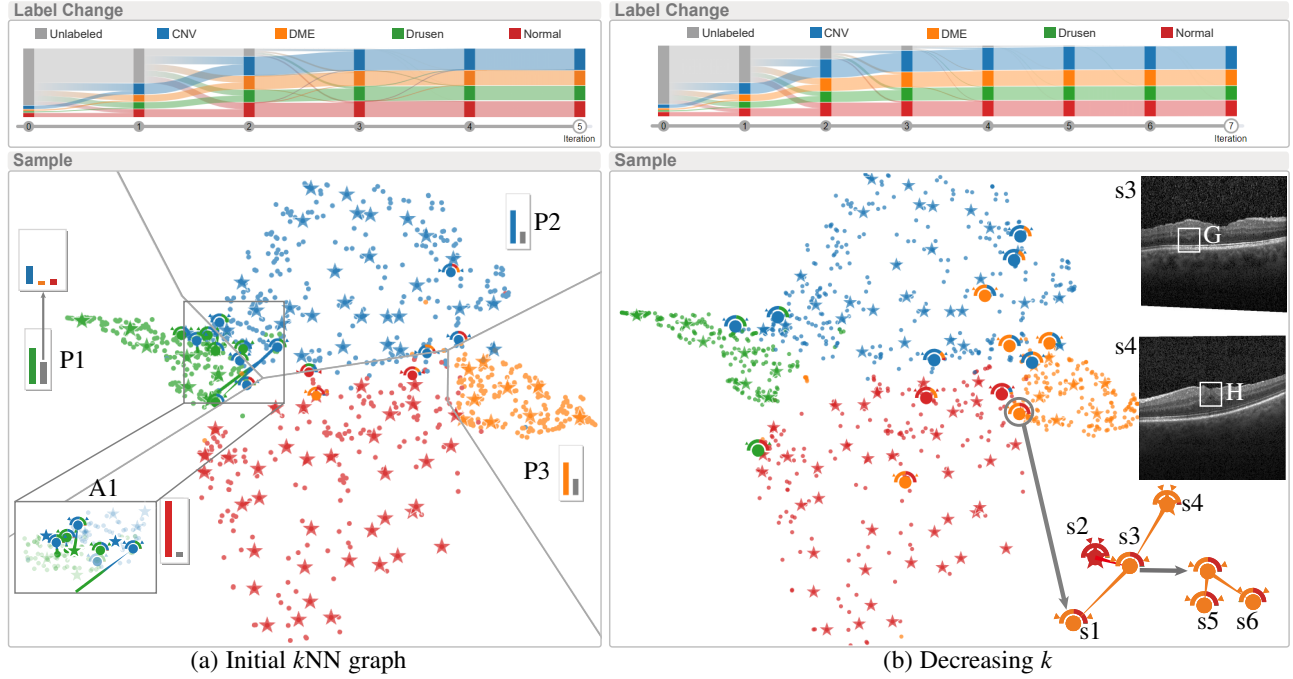


Fig. 12: Propagation overview of the OCT dataset: Label Change view and Sample view after (a) constructing the initial graph and (b) decreasing local k values.

step. The relatively small number of cat (red) samples (Fig. 1C1) confirmed his speculation. A balanced number of samples among classes was needed for classification. When there is a suspicion that some classes have fewer samples, it is a common practice to find more data from other sources, *e.g.*, on the internet [63] or in a larger repository of unlabeled samples. Semi-supervised learning is no exception. Although the labels were unknown, more samples can be added to a cluster based on the commonly used sample similarity measures (*e.g.*, the cosine similarity of two samples). As a large number of unlabeled samples in STL-10 were not used as training data in the case study, E_2 decided to search in this large repository and add more unlabeled samples to the predicted cat cluster. He selected a large set of samples in the red cluster and loaded 500 more unlabeled samples that were similar (*i.e.*, nearest neighbors) to the selected ones. With the update, the ratio of hetero-edges in P1 was reduced, and the overall uncertainty, as shown in the Action Trail, was reduced from 0.213 to 0.205.

The increase in nodes inevitably caused some local changes in the graph structure. E_2 thus carried out a second round of modifications, starting by checking the distribution of labeled nodes. After the second round, the overall uncertainty was further improved from 0.205 to 0.186. Although the uncertainty and the decrease were small, misclassified samples might still exist. To check whether there were many such samples, E_2 further sampled 1% of samples from the test data and labeled them. The accuracy of the sampled data was 95.06%. He was satisfied with the results and ended the process.

6.2 Case study on the OCT dataset

The OCT dataset [9] contains the OCT images of the retina, which are of 4 classes: Normal, Choroidal NeoVascularization (CNV), Diabetic Macular Edema (DME), and Drusen. Correct classification of OCT images is important for guiding the diagnosis and treatment of eye diseases. For this case study, 10,000 images (3,458 CNV, 1,013 DME, 794 Drusen, and 4,735 Normal) were randomly

selected from the dataset, among which 1,000 (370 CNV, 109 DME, 81 Drusen, and 440 Normal) were labeled. As mentioned, the state-of-the-art deep SSL model, MixMatch [33], was trained for feature extraction. Another 10,000 images (3,539 CNV, 972 DME, 792 Drusen, and 4697 Normal) were used as the test dataset to evaluate the performance of the GSSL model. Initially, 92.90% test accuracy was obtained.

Modifying the graph at the local level (requirements $R2$ and $R5$). An initial graph was first constructed with $k = 5$, which is the smallest k that ensures every connected component in the graph has at least one labeled sample. From the Label Change view, E_4 was satisfied that most unlabeled samples had been propagated to within three iterations. Considering the relatively large ratio of labeled samples (10%) and their balanced distribution in the Sample view, E_4 believed it was unnecessary to adjust the labeled samples and moved on to check the edge distribution in the Sample view. From the bar charts (Fig. 12(a)), it was observed that the green (P1), blue (P2), and orange (P3) classes had relatively higher ratios of hetero-edges. He started his analysis with the green one (P1) that had the largest ratio. The detailed distribution of hetero-edges showed that most hetero-edges existed between the green and blue classes and were around the boundary (Fig. 12A1). Selecting this region where the hetero-edges resided, higher uncertainties were observed for samples in these regions. The Information view further demonstrated that these uncertain samples tended to have lower-ranked neighbors in the wrong classes (Fig. 13(b)). As with the STL-10 case, E_4 decided to decrease the k values for this region and the others in partitions P2 and P3. Upon update, the Sample view showed an obvious reduction of uncertain samples, and the Action Trail confirmed the improvement with the overall uncertainty reduced from 0.029 to 0.014 ((Figs. 12(b) and 13(a))).

Refining the graph at the individual level (requirements $R2$ and $R4$). For the remaining uncertain samples, E_4 zoomed in locally and further refined the local graph structures by checking and editing the nodes and edges individually. During the process,

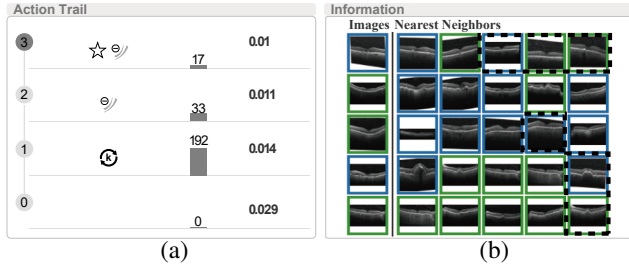


Fig. 13: The OCT dataset: (a) Action Trail; (b) Information view.

E_4 noticed that sample s_1 had a high prediction uncertainty between the DME and Normal classes (Fig. 12(b)). Checking the propagation path to s_1 , it was found that the most important edge was from s_3 , which also had a high prediction uncertainty between the DME and Normal classes. E_4 consulted with his collaborated ophthalmologist, who confirmed that both s_1 and s_3 should be Normal. s_3 had its label propagated from both s_2 and s_4 , where s_2 was Normal, and s_4 was DME. Comparing the images of s_3 and s_4 , the domain expert pointed out that s_4 was DME because of the obvious cystoid cavity in the middle (Fig. 12H). The blurred black patch in the middle of s_3 is not considered typical for DME, but does make s_3 visually similar to s_4 . E_4 further checked the samples that were propagated from s_3 and found two more wrongly predicted samples, s_5 and s_6 . Based on the observations, E_4 deleted the wrong edge from s_4 to s_3 . In collaboration with the ophthalmologist, E_4 corrected the propagation paths for 18 more uncertain samples. After being updated, the overall uncertainty was further reduced from 0.014 to 0.011 (Fig. 13(b)).

As with the STL-10 case study, E_4 carried out a second round of modifications. After that, two more labels were provided, and 21 edges were deleted. The uncertainty was reduced from 0.011 to 0.010. Following E_2 in the first case study, the ophthalmologist also sampled 1% of the test data. The accuracy of the sampled data was 95.00%. Both E_4 and the ophthalmologist were satisfied with the result and ended the modification at this point.

6.3 Post Analysis and Expert Feedback

Post analysis for the case studies. To objectively evaluate the effectiveness of the interactive construction process using DataLinker, classification of test data was carried out after each step. Table 1 shows the step-by-step improvements of the classification accuracy on the test data. In both cases, the learned model improved performance with each step. As adjusting labels influenced the graph structure at a global level, it offered the greatest increase in classification accuracy. Adjusting the local k values and refining the nodes and edges modified the graph structures in local regions and offered a smaller yet steady improvement. Exploiting more unlabeled samples achieved further improvements on a smaller scale, as can be expected. A second round of modifications improved the accuracy further. Using DataLinker, the classification accuracy on STL-10 was improved from the initial 88.91% to 95.60%, while on OCT was improved from 92.90% to 94.27%, showing a larger gain on datasets with smaller ratios of labeled samples.

To evaluate the effectiveness of DataLinker to reduce expert effort, we also compared it with HSE [65], an effective graph-based active learning method, on STL-10. After labeling eight samples using both methods (HSE: samples recommended by the algorithm;

TABLE 1: Improvement of test accuracy after each step using DataLinker.

Step	STL-10		OCT	
	Accuracy	Gain	Accuracy	Gain
Initial	88.91%		92.90%	
Adjusting labels	92.60%	3.69%	-	-
Modifying graph	94.40%	1.80%	94.12%	1.22%
Augmenting samples	95.01%	0.61%	-	-
Second round	95.60%	0.59%	94.27%	0.15%

DataLinker: samples selected by the expert with the help of interactive visualization), the accuracy achieved with DataLinker (92.60%) was higher than that of HSE (89.99%). To achieve the same accuracy of 92.60%, 21 samples were needed to be labeled for HSE. DataLinker reduced expert effort by 61.90%. This is because the Label Change view and the Sample view effectively help experts find more problematic samples to label.

Expert feedback. We had a discussion with both experts after the case studies. They commented positively on the visualization for providing an effective and efficient way to improve their GSSL models. E_2 was especially impressed by the 6.69% gain in classification accuracy. Without visualizations, he usually selected uncertain samples to label. However, it usually takes more effort to label more samples for satisfactory performance, which is very time-consuming. “With the Label Change view and the Sample view, I can quickly identify regions lacking labeled samples. Labeling samples in such regions results in relatively large gains. To achieve similar accuracy, DataLinker reduces my labeling effort by around 60%,” E_2 said. He also pointed out that it would be hard to improve the prediction accuracy even though the local region had enough labeled samples but an incorrect graph structure. This is because the wrong connections between samples will lead to wrong label propagation and thus low prediction accuracy. For example, the misclassification of the black monkey in the first case was caused by the incorrect label propagation from two black dogs. However, labeling the black monkey will lead to misclassification of these two black dogs because the label can be propagated from the monkey to the dogs. “Modifying the graph structure seems to be a more effective method,” E_2 commented. In addition to effectiveness, both experts found the tool helped them efficiently locate problematic regions and identify samples/edges that lead to wrong predictions, thanks to the combination of Label Change view and Sample view, as well as edge filtering. Both experts also agreed that the tool helped identify the reasons for wrong predictions. For example, the interactive identification of outliers, such as the red panda image in the first case and the wrong edge connecting the DME and the Normal samples in the second case, were helpful for improving the graph quality. However, they also pointed out that they wanted to verify these suspected causes and gain insights into how to make use of these observations in refining the employed models. Overall, with DataLinker, the experts now have more confidence that the unlabeled samples have been used effectively.

7 DISCUSSION AND FUTURE WORK

DataLinker has been successfully applied to two real-world applications with satisfying results, which demonstrates its effectiveness and efficiency in improving GSSL models and increasing user confidence in exploiting unlabeled samples. Image data is used in these applications to illustrate the idea. However, DataLinker can be

easily generalized to handle other types of data, such as textual data, with suitable methods for feature extraction. Only minor changes in the Information view are needed to display samples of other types. Apart from generalization, several other improvements are also desired by experts and will be the subject of our future research:

Reasoning power. DataLinker can help identify causes for wrong predictions by tracing individual propagation paths. As desired by experts, one direction of research is to investigate how to go further from these discrete causes and provide more reasoning power to the tool. One potential avenue is to integrate automatic outlier detection methods into DataLinker for automatically detecting more outliers, which can then be presented to experts for a better understanding of the root cause of these occurrences. Based on this understanding, users can figure out the deficiencies in the employed GSSL model and improve it for better performance.

Streaming data. Currently, DataLinker handles a graph constructed with the labeled/unlabeled samples at hand. In real-world applications, unlabeled samples may come in over time in a streaming manner. In such a scenario, more timely updates of the learning models are desirable. An interesting direction for future research is to study how we can effectively visualize the dynamic changes in a graph structure and incrementally update GSSL models with the arrival of new unlabeled samples.

Action confidence. DataLinker currently incorporates expert action as ground truth. For example, it is assumed that labels provided by experts are correct, and the deleted edges are truly wrong. While this assumption is correct most of the time, sometimes experts can make mistakes. To make the tool more robust to the noise introduced by experts, a possible solution is to evaluate the confidence of each action and take this into consideration during model construction and visualization.

Learning curve. The visual metaphors used in DataLinker, such as river flow, bar chart, Scented Widgets, and Voronoi-based partition, are all common ones. Thus, the experts can quickly grasp these visual encodings. It usually takes the experts 10-30 minutes to become familiar with DataLinker, especially the interactions. The experts believe that the benefits brought by the tool via interactive visualization undoubtedly outweigh the learning cost. To further reduce the learning cost, we also added a tour function to help experts quickly become familiar with the visual encodings and interactions of DataLinker.

Generalization to other datasets. DataLinker can help experts improve the quality of graphs. However, participation in the graph construction for each dataset is still inefficient. It would be desirable to generalize expert effort from one graph to other datasets. A possible solution is to use the model trained on the refined graph to make predictions on unlabeled samples of other datasets. The predictions with high confidence can be treated as candidate ground truth labels to refine other models. This strategy is also widely used in the field of machine learning [66], [67].

8 CONCLUSION

In this paper, we have presented DataLinker, a visual analysis tool to help machine learning experts interactively construct a high-quality graph for better exploitation of unlabeled samples in GSSL. A coordinated and hybrid visualization has been developed to help experts understand label propagation along graph edges. To support the visualization of large graphs, hierarchical presentation and filtering of graph nodes and edges are provided. The visualization

is integrated with an interactive graph construction method and provides a coarse-to-fine strategy to progressively construct high-quality graphs. Two case studies were conducted to demonstrate the effectiveness of DataLinker.

ACKNOWLEDGMENTS

This research is supported by the National Key R&D Program of China (No. 2020YFB2104100), the National Natural Science Foundation of China (No. 61936002), grants from the Institute Guo Qiang and the Institute for Brain and Cognitive Science, Tsinghua University, and in part by Tsinghua-Kuaishou Institute of Future Media Data. The authors would like to thank Zhen Li and Shouxing Xiang for implementing the supervised t-SNE algorithm.

REFERENCES

- [1] J. E. Van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Machine Learning*, vol. 109, no. 2, pp. 373–440, 2020.
- [2] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *Proceedings of the International Conference on Machine Learning*, 2016, pp. 40–48.
- [3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the International Conference on Learning Representations*, 2016.
- [4] Q. Li, X.-M. Wu, H. Liu, X. Zhang, and Z. Guan, "Label efficient semi-supervised learning via graph filtering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9582–9591.
- [5] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, "Label propagation for deep semi-supervised learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5070–5079.
- [6] Y.-F. Li, S.-B. Wang, and Z.-H. Zhou, "Graph quality judgement: A large margin expedition," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2016, pp. 1725–1731.
- [7] V. Vapnik, *The nature of statistical learning theory*. Springer Science & Business Media, 2013.
- [8] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2011, pp. 215–223.
- [9] D. S. Kermany, M. Goldbaum, W. Cai, C. C. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, and F. Yan, "Identifying medical diagnoses and treatable diseases by image-based deep learning," *Cell*, vol. 172, no. 5, pp. 1122–1131, 2018.
- [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] M. Liu, L. Jiang, J. Liu, X. Wang, J. Zhu, and S. Liu, "Improving learning-from-crowds through expert validation," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2017, pp. 2329–2336.
- [12] T. Tian and J. Zhu, "Max-margin majority voting for learning from crowds," in *Proceedings of the Advances in Neural Information Processing Systems*, 2015, pp. 1621–1629.
- [13] J. H. Park, S. Nadeem, S. Mirhosseini, and A. Kaufman, "C²A: Crowd consensus analytics for virtual colonoscopy," in *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, 2016, pp. 21–30.
- [14] W. Willett, S. Ginosar, A. Steinitz, B. Hartmann, and M. Agrawala, "Identifying redundancy and exposing provenance in crowdsourced data analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2198–2206, 2013.
- [15] H. Chung, S. P. Dasari, S. Nandhakumar, and C. Andrews, "CRICTO: Supporting sensemaking through crowdsourced information schematization," in *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, 2017, pp. 139–150.
- [16] S. Liu, C. Chen, Y. Lu, F. Ouyang, and B. Wang, "An interactive method to improve crowdsourced annotations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 235–245, 2019.
- [17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

- [18] S. Xiang, X. Ye, J. Xia, J. Wu, Y. Chen, and S. Liu, "Interactive correction of mislabeled training data," in *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, 2019, pp. 57–68.
- [19] J. Moehrmann, S. Bernstein, T. Schlegel, G. Werner, and G. Heidemann, "Improving the usability of hierarchical representations for interactively labeling large image data sets," in *Proceedings of the International Conference on Human-Computer Interaction*, 2011, pp. 618–627.
- [20] M. Khayat, M. Karimzadeh, J. Zhao, and D. S. Ebert, "VASSL: A visual analytics toolkit for social spambot labeling," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 874–883, 2019.
- [21] J. Bernard, M. Zeppelzauer, M. Sedlmair, and W. Aigner, "VIAL: A unified process for visual interactive labeling," *The Visual Computer*, vol. 34, no. 9, pp. 1189–1207, 2018.
- [22] B. Höferlin, R. Netzel, M. Höferlin, D. Weiskopf, and G. Heidemann, "Inter-active learning of ad-hoc classifiers for video visual analytics," in *Proceedings of the Conference on Visual Analytics Science and Technology*, 2012, pp. 23–32.
- [23] F. L. Dennig, T. Polk, Z. Lin, T. Schreck, H. Pfister, and M. Behrisch, "FDive: Learning relevance models using pattern-based similarity measures," in *Proceedings of the Conference on Visual Analytics Science and Technology*, 2019, pp. 69–80.
- [24] F. Sperrle, R. Sevastjanova, R. Kehlbeck, and M. El-Assady, "VIANA: Visual interactive annotation of argumentation," in *Proceedings of the Conference on Visual Analytics Science and Technology*, 2019, pp. 11–22.
- [25] J. Bernard, M. Hutter, M. Zeppelzauer, D. Fellner, and M. Sedlmair, "Comparing visual-interactive labeling with active learning: An experimental study," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 298–308, 2017.
- [26] J. Bernard, M. Zeppelzauer, M. Lehmann, M. Müller, and M. Sedlmair, "Towards user-centered active learning algorithms," *Computer Graphics Forum*, vol. 37, no. 3, pp. 121–132, 2018.
- [27] A. Balsubramani, S. Dasgupta, Y. Freund, and S. Moran, "An adaptive nearest neighbor rule for classification," in *Proceedings of the Advances in Neural Information Processing Systems*, 2019, pp. 7577–7586.
- [28] L. Baoli, L. Qin, and Y. Shiwen, "An adaptive k-nearest neighbor text categorization strategy," *ACM Transactions on Asian Language Information Processing*, vol. 3, no. 4, pp. 215–226, 2004.
- [29] S. Sun and R. Huang, "An adaptive k-nearest neighbor algorithm," in *Proceedings of the International Conference on Fuzzy Systems and Knowledge Discovery*, 2010, pp. 91–94.
- [30] N. García-Pedrajas, J. A. R. Del Castillo, and G. Cerruela-García, "A proposal for local k values for k-nearest neighbor rule," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 2, pp. 470–475, 2015.
- [31] D. Wettschereck and T. G. Dietterich, "Locally adaptive nearest neighbor algorithms," *Proceedings of the Advances in Neural Information Processing Systems*, pp. 184–184, 1994.
- [32] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8697–8710.
- [33] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "MixMatch: A holistic approach to semi-supervised learning," in *Proceedings of the Advances in Neural Information Processing Systems*, 2019, pp. 5050–5060.
- [34] X. Zhu, "Semi-supervised learning with graphs," Ph.D. dissertation, Carnegie Mellon University, 2005.
- [35] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Proceedings of the Advances in Neural Information Processing Systems*, 2003, pp. 321–328.
- [36] W. Liu, J. He, and S.-F. Chang, "Large graph construction for scalable semi-supervised learning," in *Proceedings of the International Conference on Machine Learning*, 2010, pp. 1–8.
- [37] W. Cui, S. Liu, L. Tan, C. Shi, Y. Song, Z. Gao, H. Qu, and X. Tong, "TextFlow: Towards better understanding of evolving topics in text," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2412–2421, 2011.
- [38] S. Liu, J. Yin, X. Wang, W. Cui, K. Cao, and J. Pei, "Online visual analytics of text streams," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 11, pp. 2451–2466, 2015.
- [39] G. Sun, Y. Wu, S. Liu, T.-Q. Peng, J. J. Zhu, and R. Liang, "EvoRiver: Visual analysis of topic competition on social media," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1753–1762, 2014.
- [40] A. Damian, A. Costan, V. Cristea, and I. Legrand, "A visualisation technique for network topology transformation within monalisa monitoring framework," *International Journal of Grid and Utility Computing*, vol. 2, no. 2, pp. 119–129, 2011.
- [41] Z. Yang, J. Peltonen, and S. Kaski, "Optimization equivalence of divergences improves neighbor embedding," in *Proceedings of the International Conference on Machine Learning*, 2014, pp. 460–468.
- [42] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.
- [43] P. E. Rauber, S. G. Fadel, A. X. Falcao, and A. C. Telea, "Visualizing the hidden activity of artificial neural networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 101–110, 2016.
- [44] J. Choo, C. Lee, C. K. Reddy, and H. Park, "UTOPIAN: User-driven topic modeling based on interactive nonnegative matrix factorization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 1992–2001, 2013.
- [45] B. Settles, *Active learning literature survey*. University of Wisconsin-Madison, 2009.
- [46] Y. Yang, Z. Ma, F. Nie, X. Chang, and A. G. Hauptmann, "Multi-class active learning by uncertainty sampling with diversity maximization," *International Journal of Computer Vision*, vol. 113, no. 2, pp. 113–127, 2015.
- [47] H. H. Aghdam, A. Gonzalez-Garcia, J. v. d. Weijer, and A. M. López, "Active learning for deep detection neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3672–3680.
- [48] A. M. MacEachren, R. E. Roth, J. O'Brien, B. Li, D. Swingley, and M. Gahegan, "Visual semiotics & uncertainty visualization: An empirical study," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2496–2505, 2012.
- [49] X. Wang, S. Liu, J. Liu, J. Chen, J. Zhu, and B. Guo, "TopicPanorama: A full picture of relevant topics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 12, pp. 2508–2521, 2016.
- [50] D. Holten and J. J. Van Wijk, "A user study on visualizing directed edges in graphs," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2009, pp. 2299–2308.
- [51] D. Selassie, B. Heller, and J. Heer, "Divided edge bundling for directional network data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2354–2363, 2011.
- [52] W. Kanizsa and W. Gerbino, "Convexity and symmetry in figure-ground organization," in *Proceedings of the Vision and Artifact*, 1976, pp. 25–32.
- [53] M. De Berg, O. Cheong, M. Van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*. Springer, 2008.
- [54] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane," *IEEE Transactions on Information Theory*, vol. 29, no. 4, pp. 551–559, 1983.
- [55] D. Krasnoshechekov and V. Polishchuk, "Order-k α -hulls and α -shapes," *Information Processing Letters*, vol. 114, no. 1–2, pp. 76–83, 2014.
- [56] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2000, pp. 93–104.
- [57] H. Lin, S. Gao, D. Gotz, F. Du, J. He, and N. Cao, "RCLens: Interactive rare category exploration and identification," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 7, pp. 2223–2237, 2017.
- [58] A. Efrat, Y. Hu, S. G. Kobourov, and S. Pupyrev, "MapSets: visualizing embedded and clustered graphs," in *International Symposium on Graph Drawing*, 2014, pp. 452–463.
- [59] J. Yuan, S. Xiang, J. Xia, L. Yu, and S. Liu, "Evaluation of sampling methods for scatterplots," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 1720–1730, 2021.
- [60] W. Willett, J. Heer, and M. Agrawal, "Scented widgets: Improving navigation cues with embedded visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1129–1136, 2007.
- [61] J. Kang, M. Wang, N. Cao, Y. Xia, W. Fan, and H. Tong, "AURORA: Auditing PageRank on large graphs," in *IEEE International Conference on Big Data*, 2018, pp. 713–722.
- [62] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Proceedings of the Advances in Neural Information Processing Systems*, 2005, pp. 529–536.
- [63] A. Oliver, A. Odena, C. A. Raffel, E. D. Cubuk, and I. Goodfellow, "Realistic evaluation of deep semi-supervised learning algorithms," in *Proceedings of the Advances in Neural Information Processing Systems*, 2018, pp. 3235–3246.
- [64] E. Triantafyllou, T. Zhu, V. Dumoulin, P. Lamblin, U. Evci, K. Xu, R. Goroshin, C. Gelada, K. Swersky, P.-A. Manzagol et al., "Meta-dataset: A dataset of datasets for learning to learn from few examples," in *Proceedings of the International Conference on Learning Representations*, 2020.

- [65] O. Mac Aodha, N. D. Campbell, J. Kautz, and G. J. Brostow, "Hierarchical subquery evaluation for active learning on a graph," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 564–571.
- [66] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Proceedings of the ICML Workshop on Challenges in Representation Learning*, 2013.
- [67] X. Li, Q. Sun, Y. Liu, Q. Zhou, S. Zheng, T.-S. Chua, and B. Schiele, "Learning to self-train for semi-supervised few-shot classification," in *Proceedings of the Advances in Neural Information Processing Systems*, 2019, pp. 1–11.



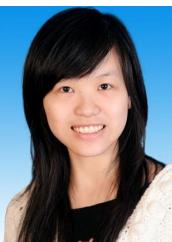
Changjian Chen is now a Ph.D. student at Tsinghua University. His research interests are in interactive machine learning. He received a B.S. degree from University of Science and Technology of China.



Zhaowei Wang is currently a Ph.D. student at Tsinghua University. His research interests are in explainable artificial intelligence. He received a B.S. degree from Tsinghua University.



Jing Wu is a lecturer in computer science and informatics at Cardiff University, UK. Her research interests are in computer vision and graphics including image-based 3D reconstruction, face recognition, machine learning and visual analytics. She received BSc and MSc from Nanjing University, and Ph.D. from the University of York, UK. She serves as a PC member in CGVC, BMVC, etc., and is an active reviewer for journals including Pattern Recognition, Computer Graphics Forum, etc.



Xiting Wang is a senior researcher at Microsoft Research Asia. Her research interests include explainable machine learning and visual text analytics. She has published academic papers on reputable international conferences and journals in her research area, such as KDD, TKDE, AAAI, IJCAI, TVCG and VAST. One of her first author papers has been chosen as the TVCG spotlight article for Dec. 2016. She is a senior program committee member of AAAI and is a program committee member of many top conferences.



Lan-Zhe Guo received the BSc degree in 2017. He is currently working toward the Ph.D. degree in the National Key Laboratory for Novel Software Technology at Nanjing University, China. His research interests include semi-supervised learning, label noise learning, distribution shift learning. He is/was a PC member of top conferences such as AAAI, IJCAI, and a reviewer of journals like TKDE.



Yu-Feng Li is an associate professor in Nanjing University. His research interests include weakly supervised learning, statistical learning and optimization. He has received outstanding doctoral dissertation award from China Computer Federation (CCF) and Microsoft Fellowship Award. He is/was served as an editorial board member of machine learning journal special issues, co-chair of ACML18 workshop and ACML19 tutorial, and a senior PC member of top-tier conferences such as IJCAI'19/17/15, AAAI'19.



Shixia Liu is an associate professor at Tsinghua University. Her research interests include visual text analytics, visual social analytics, interactive machine learning, and text mining. She worked as a research staff member at IBM China Research Lab and a lead researcher at Microsoft Research Asia. She received a B.S. and M.S. from Harbin Institute of Technology, a Ph.D. from Tsinghua University. She is a fellow of IEEE and an associate editor-in-chief of IEEE Trans. Vis. Comput. Graph.