Interactive Reweighting for Mitigating Label Quality Issues

Weikai Yang, Yukai Guo, Jing Wu, Zheng Wang, Lan-Zhe Guo, Yu-Feng Li, and Shixia Liu

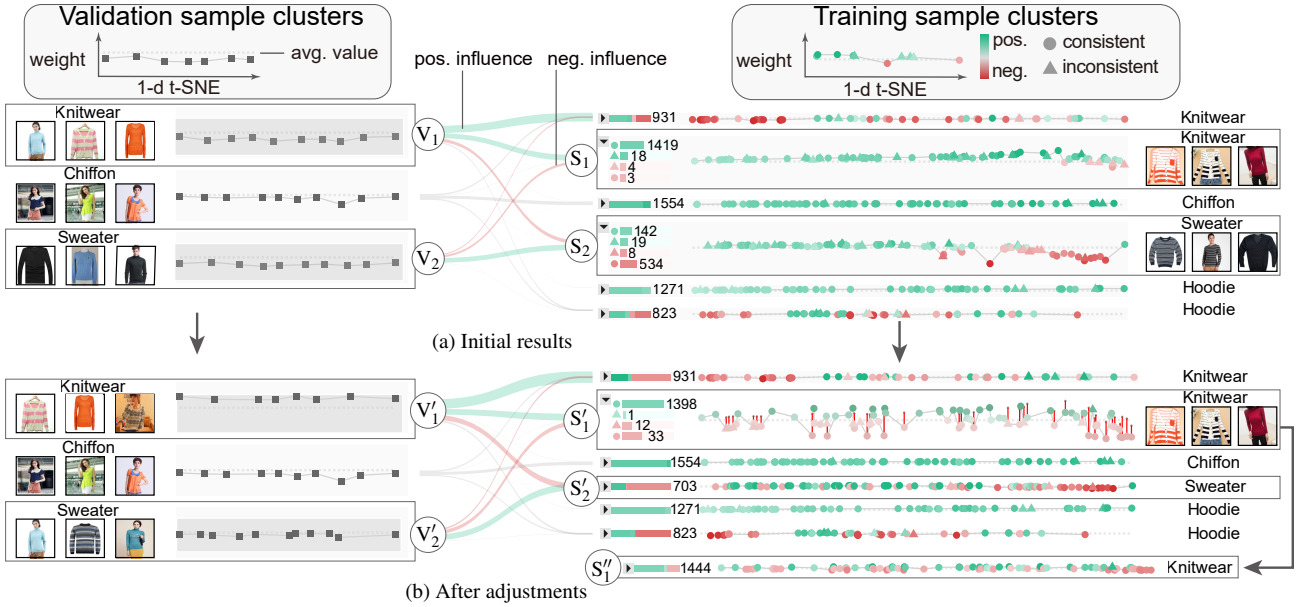


Fig. 1: (a) The reweighting relationships between 3 (out of 14) validation sample clusters and 6 (out of 35) training sample clusters. V_1 and V_2 contain low-quality validation samples, resulting in many inconsistent training samples in S_1 and S_2. (b) After correcting the noisy labels of low-quality validation samples, increasing the weights of high-quality validation samples, and verifying inconsistent training samples, the reweighting results are improved (S_1'' and S_2').

Abstract—Label quality issues, such as noisy labels and imbalanced class distributions, have negative effects on model performance. Automatic reweighting methods identify problematic samples with label quality issues by recognizing their negative effects on validation samples and assigning lower weights to them. However, these methods fail to achieve satisfactory performance when the validation samples are of low quality. To tackle this, we develop Reweighter, a visual analysis tool for sample reweighting. The reweighting relationships between validation samples and training samples are modeled as a bipartite graph. Based on this graph, a validation sample improvement method is developed to improve the quality of validation samples. Since the automatic improvement may not always be perfect, a co-cluster-based bipartite graph visualization is developed to illustrate the reweighting relationships and support the interactive adjustments to validation samples and reweighting results. The adjustments are converted into the constraints of the validation sample improvement method to further improve validation samples. We demonstrate the effectiveness of Reweighter in improving reweighting results through quantitative evaluation and two case studies.

Index Terms—Training data quality, sample reweighting, bipartite graph visualization

1 Introduction

Addressing label quality issues, such as noisy labels and imbalanced class distributions is critical due to their pervasive presence in real-world applications. For example, insufficient labeled data often compels model developers to resort to pseudo-labeling techniques for easier model training [1]. However, this workaround comes at a steep cost by introducing label quality issues that can significantly downgrade model performance and reliability. This casts a shadow on its applicability in real-world scenarios. Identifying and mitigating these label quality issues becomes difficult and even impossible in the era of deep learning. A more practical solution is to reduce the negative effects of problematic training samples with label quality issues on model training. Validation-sample-based reweighting methods [2], [3] have achieved state-of-the-art performance in reducing such negative effects. These methods first measure the model performance on a small set of samples, namely validation samples, which have clean labels

- *W. Yang, Y. Guo, Z. Wang, and S. Liu are with the School of Software, BNRist, Tsinghua University.*
- *J. Wu is with Cardiff University.*
- *L. Guo and Y. Li are with Nanjing University.*

and well represent training data. Then a higher/lower weight is assigned to a training sample if it increases/decreases the model performance on the validation samples. This forms the reweighting relationships between validation samples and training samples. Since the validation samples determine the reweighting results, it is crucial to ensure their quality for the performance of these reweighting methods [4].

High-quality validation samples should have clean labels and well represent training data [5], [6]. In response to this need, state-of-the-art reweighting methods automatically select training samples that are likely to have clean labels as validation samples based on their confidence metrics [3], [7]. While this automatic selection offers advantages in terms of speed and reduced manual interference, it comes with two issues. First, these automatically selected validation samples, despite being selected based on confidence metrics, can still contain noisy labels. Second, an over-reliance on these confidence metrics might lead to the omission of samples that represent the diverse nature of the training data, thereby creating a skewed or biased representation. This, in turn, can degrade the reweighting performance [4]. To ensure the cleanliness of the validation samples, model developers need to identify the validation samples with noisy labels. To ensure their representativeness, they need to repetitively compare the distributions of validation and training samples to identify the less represented samples [8]. This debugging process is labor-intensive and time-consuming [9]. Upon identifying the quality issues, model developers can make direct adjustments if they are familiar with the application domain and data. However, in situations where the domain is unfamiliar, collaboration with domain experts and/or crowd workers becomes essential [10]. Such collaboration also requires a visual analysis tool to efficiently convey the identified quality issues and guide appropriate adjustments. The challenging nature of the debugging process and the specific needs for the collaboration motivate us to develop Reweighter. We model the reweighting relationships between validation samples and training samples as a bipartite graph. Based on this graph, a validation sample improvement method is developed to improve the quality of validation samples. Since the automatic improvement may not always be perfect, a co-cluster-based bipartite graph visualization is developed to illustrate the reweighting relationships. By offering clear insights into how the reweighting results are derived, this visualization facilitates more informed adjustments to the validation samples and reweighting results. These adjustments are converted into the constraints of the validation sample improvement method to improve the quality of validation samples. This, in turn, leads to better reweighting results. The aforementioned analysis process is repeated iteratively. Upon satisfaction, the reweighting results are utilized for model training.

The effectiveness of the validation sample improvement method is demonstrated using three numerical experiments on four different datasets. Two case studies are conducted to demonstrate the usefulness of Reweighter in iteratively improving the quality of validation samples and generating better reweighting results. The main contributions of this work are:

- A visual analysis tool that helps generate better reweighting results by iteratively improving validation samples.
- A method for automatically improving validation samples based on the reweighting relationships.
- A bipartite graph visualization that illustrates the reweighting relationships and helps make informed adjustments.

2 Related Work

2.1 Sample Reweighting

Existing reweighting efforts can be classified into two categories [11]: distribution-based methods and validation-sample-based methods.

Distribution-based methods assign weights to training samples based on the data distribution. For example, Liu *et al*. [12] first introduced the importance reweighting method into binary classification to tackle the challenges posed by noisy labels. The weight of a sample is set as the probability of this sample being clean over the probability of it being noisy. They also extended this method to multi-class classification [13]. Instead of estimating the data distribution, later work treated the sample weights as latent variables of a learning model, such as a Bayesian model [14] or a deep learning model [15], and then inferred the weights of samples by this model. However, these methods require manually designing the reweighting function or the learning model. This makes it difficult to fit different datasets.

To address this challenge, validation-sample-based methods assign a weight to each training sample with the goal of optimizing the model performance on validation samples. These methods effectively reduce performance degradation caused by both noisy labels and imbalanced class distributions. Ren *et al*. [2] calculated the weights of the training samples using the loss gradients in the validation samples. A negative weight is assigned to a sample if its associated gradient is positive. Instead of calculating the weights from gradients, Meta-Weight-Net [11] trains an extra model to learn the weights. However, these methods require a pre-defined set of validation samples with clean labels and balanced class distributions. To be more flexible, Zhang *et al*. [3] automatically selected training samples that are likely to have clean labels as validation samples. Although this method saves the efforts for pre-defining validation samples, the performance is downgraded if the selected samples contain noisy labels and/or fail to represent the diversity of the training data [4]. Our method seeks to address these limitations by utilizing the reweighting relationships between validation samples and training samples. With these relationships and the corresponding visualization, experts can 1) easily identify and correct the noisy labels, and 2) identify the uncovered training samples and add some of them to increase the representativeness of these validation samples.

2.2 Visual Quality Improvement of Training Data

Many visualization methods have been proposed to improve the quality of training data [16], [17]. They are classified into two groups [18], [19]: collecting more annotated data and correcting annotation noise.

Collecting more annotated data. A variety of methods have been proposed to improve the efficiency of the annotation process. For example, Moehrmann *et al*. [20] employed a self-organizing map to allow the selection and simultaneous annotations of multiple similar images. The strategy of placing similar samples closer together is then applied to annotate different types of data [21], [22], [23], [24]. Active learning algorithms are also integrated to recommend informative samples for efficient annotation [25], [26], [27], [28], [29], [30], [31], [32]. Beyond efficiency, other methods seek to address the issue of dataset bias, which is introduced due to the distribution difference between training data and test data. Chen *et al*. [33] developed OoDAnalyzer to visually detect Out-of-Distribution samples that are not covered by training data. Yang *et*

al. [34] used the energy distance to measure the magnitude of change in data distribution when test data streams in. The idea of comparing the data distributions has also been adopted by many other visual analysis work [16], [35], [36], [37]. Different from these methods, our work focuses on correcting annotation noise in validation samples and making them well represent training data.

Correcting annotation noise in training data. Annotation noise is commonly present in real-world training data [6]. Many methods have been proposed to correct them. For training data with crowd information, crowd annotations and worker behavior are utilized to detect the annotation noise for correction [38], [39], [40]. For example, LabelInspect [39] facilitates the verification of uncertain samples and unreliable workers based on crowd information analysis. For training data without crowd information, the annotation noise is usually detected based on the difference between the sample annotations and their predictions made by a learning model [5], [17], [41], [42], [43], [44]. Among them, the most relevant one is DataDebugger [5], which develops a label propagation algorithm to identify and correct label noise based on the difference between their predictions and the expert-selected trusted items. Although this method improves the quality of training data, it has two issues. First, it requires to provide the exact label for each trusted item. This takes more time since it needs the experts to examine the sample and even compare it with similar samples. Second, it only handles the issue of label noise and does not work well for other label quality issues, such as imbalanced class distributions. Compared with this method, our work only requires the experts to indicate whether the label is clean or noisy, which demands less expertise and is more efficient. Moreover, our work handles not only the issue of label noise, but also the issue of imbalanced class distributions, which often occur in real-world training data.

3 Design of Reweighter

3.1 Requirement Analysis

We collaborated with four experts (E1-E4) to develop Reweighter. E1 is a software developer with five years of experience in improving the quality of training data. E2 and E3 are two Ph.D. students with more than four years of experience in diagnosing quality issues in training data and improving model performance. E4 is a postdoc researcher with seven years of experience in few-shot learning, which also requires high-quality training data. None of them are the co-authors of this work. To understand how they improve the reweighting results in their work, we interviewed each of them for about 45-60 minutes. Based on the interviews and literature review, we summarized the following requirements for improving reweighting results.

R1. Examining validation samples and training samples. A previous study has shown that the quality of validation samples plays a critical role in the reweighting process [3]. The experts also confirmed this and expressed the need to improve the validation samples. E3 said that he would like to examine the quality of the validation samples and then check whether they positively influenced the reweighting results. In addition to examining the quality of validation samples, the experts mentioned that examining the training samples and the reweighting results are also useful, as they may reflect quality issues in validation samples. E1 said, "I would like to spend more time examining training samples with incorrect weights, such as low-quality samples with positive weights, to identify the quality issues in validation samples." However, examining all the samples is time-consuming. Thus, it is desirable to provide an informative overview of validation samples and training samples, and highlight the samples of interest.

R2. Understanding the reweighting relationships. After identifying the validation/training samples of interest, the experts need to understand the reweighting relationships related to these samples. Without a comprehensive understanding of such relationships, it is difficult to make proper improvements and generate better reweighting results. For example, E3 commented that he would be more confident in adjusting validation samples after understanding how each validation sample influences the reweighting results. E1 added, "When I find a low-quality training sample with a positive weight, I would like to know which validation samples influence its weight and whether they also influence the weights of other training samples." However, it is difficult to examine all the relationships. To simplify the analysis process, there is a need for a method that clusters the relationships for an overview and displays the details upon request.

R3. Improving the quality of validation samples. All the experts expressed the need to explicitly improve the quality of validation samples and hence generate better reweighting results through simple interactions. For example, E2 said that he would like to correct the labels of noisy validation samples because they would lead to incorrect reweighting results. Meanwhile, the validation samples that positively influence the reweighting results should be strengthened. When the validation samples cannot well represent training samples, he also wanted to add more representative validation samples for better coverage. Since the experts often found some incorrect reweighting results in their analysis, they also expressed the requirements for implicitly improving the quality of validation samples through adjustments to reweighting results.

R4. Comparing the reweighting results before and after adjustments. Recent studies have shown that not all adjustments improve the quality of validation samples [2], [3]. The experts also confirmed this. For example, E3 noted that while certain unsuitable adjustments might benefit the reweighting results of the training samples that he was analyzing, they could negatively impact other training samples. Consequently, the experts highlighted the importance of comparing the reweighting results before and after the adjustments. For example, E4 commented, "The samples with large changes in their weights should be examined carefully. If the adjustments result in a lot of unfavorable changes, I would like to reverse them and reanalyze the corresponding reweighting results." Therefore, our tool is expected to emphasize the differences before and after the adjustments to simplify the comparison process.

3.2 System Overview

Motivated by the identified requirements, we develop Reweighter to support the interactive improvement of the reweighting results. As shown in Fig. 2, it contains three modules: data preparation, validation sample improvement, and visualization.

Given a set of validation samples and a set of training samples, the **data preparation module** initially extracts the reweighting relationships using a reweighting method (**R2**). Theoretically, any validation-sample-based reweighting method can be directly used in Reweighter. For our purpose, we employ the state-of-the-art reweighting method, Fast Sample Reweighting (FSR) [3]. The **validation sample improvement module** models the extracted reweighting relationships between validation samples and training samples as a bipartite graph (**R2**). Based on the bipartite graph, the

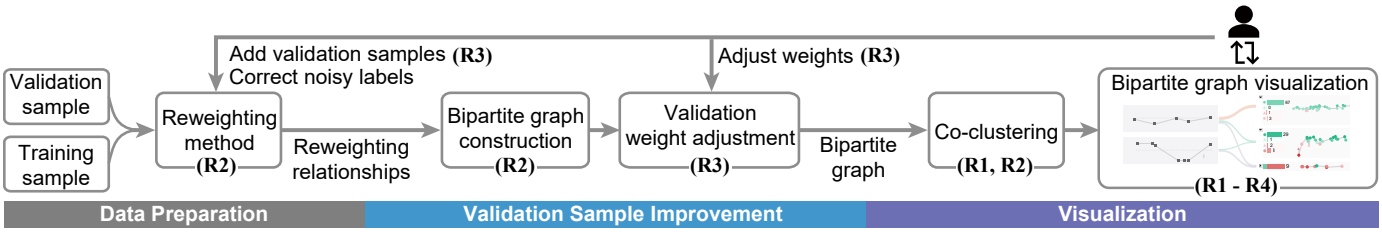


Fig. 2: System overview. The data preparation module extracts the reweighting relationships between validation samples and training samples. The validation sample improvement module models the reweighting relationships as a bipartite graph and improves the quality of validation samples. The visualization module facilitates interactive exploration and improvement on the validation samples.

weights of validation samples are automatically adjusted for generating better reweighting results (**R3**). Next, the **visualization module** employs a co-clustering algorithm [45] to simultaneously group similar validation samples and training samples based on the constructed bipartite graph. The co-clusters are visualized as a node-link diagram to simplify the exploration of the individual samples (**R1**) and their reweighting relationships (**R2**). Using the interactive visualization, model developers can adjust the validation samples and the reweighting results (**R3**). The adjustments are then used to improve the quality of validation samples by updating the reweighting relationships in the data preparation module and adjusting the weights of the validation samples in the validation sample improvement module. Furthermore, model developers can compare the reweighting results before and after the improvement to see if the adjustments are beneficial (**R4**).

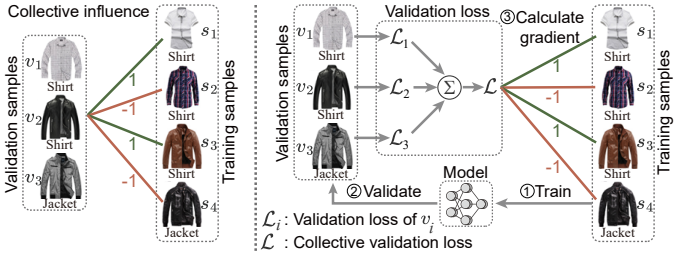


Fig. 3: The collective influence on the reweighting results (left) is computed as the gradient of collective validation loss with regard to the weights of training samples (right).

4 Validation Sample Improvement

Since the quality of validation samples is critical in validation-sample-based methods, we introduce a method for improving validation samples. Common quality issues associated with validation samples include noisy labels and the lack of representative samples. To address these issues, users usually correct the noisy labels and add the necessary samples. The updated validation samples are then used to reweight the training samples. During this process, a mutual influence between validation samples and training samples is observed. On the one hand, the quality of validation samples influences the reweighting results of training samples. On the other hand, the reweighting results may reflect the quality issues of validation samples. Given this, we seek to capture and streamline the validation sample improvement process by modeling this influence as a bipartite graph. With the bipartite graph, the weights of the validation samples can be improved based on their impact on the reweighting results. This improvement process thus unfolds in two phases: 1) constructing a bipartite graph between validation samples and training samples; 2) adjusting the weights of validation samples based on this graph.

4.1 Bipartite Graph Construction

The state-of-the-art reweighting methods [2], [3] treat the validation samples as a whole when computing their influence on the weight of each training sample s_j. As shown in Fig. 3, the influence is computed as the gradient of the collective validation loss with regard to the weight of sample s_j. However, mixing all validation samples together makes it difficult to identify the low-quality ones (Fig. 4(a)). To address this issue, the collective influence from the validation sample set should be decomposed into the influence of individual samples (**R2**). The decomposition is achieved by building a bipartite graph between validation samples and training samples (Fig. 4(b)).

The construction of the bipartite graph is fundamentally grounded on two key properties: 1) the validation loss is the sum of the loss on each validation sample; 2) the gradient operator is linear ($\nabla(f+g)=\nabla f+\nabla g$). With these properties, the collective influence of the validation sample set on training samples can be decomposed into a set of influences $\{g_{ij}|1\le i\le m, 1\le j\le n\}$ [46]. Here, g_{ij} is the influence of the validation sample v_i on

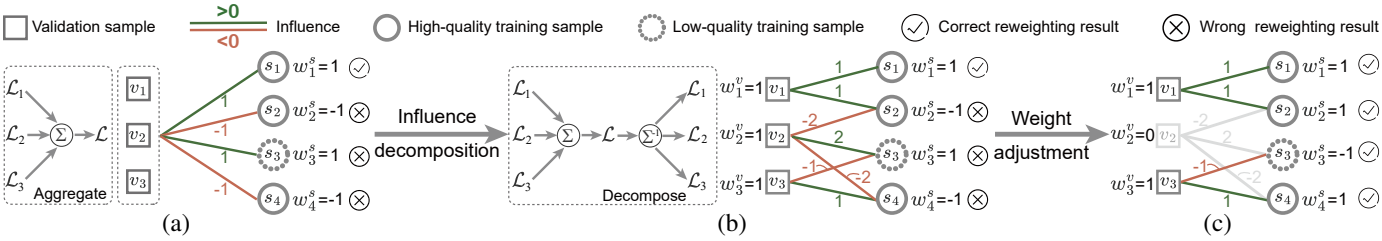


Fig. 4: The benefit of introducing the bipartite graph in the validation sample improvement method: (a) without the bipartite graph, it is hard to diagnose why the reweighting results on s_2-s_4 are incorrect (high-quality samples s_2 and s_4 have negative weights; while low-quality sample s_3 have a positive weight); (b) with the bipartite graph, it is identified that v_2 leads to the incorrect results; (c) after reducing w_2^v to zero, the reweighting results are improved.

the training sample s_j. m and n are the numbers of validation samples and training samples, respectively. The bipartite graph is then constructed by computing all g_{ij}s. With this graph, each validation sample v_i is represented by a n-dimensional vector, $v_i = [g_{i1}, g_{i2}, \ldots, g_{in}]^\intercal$. Each training sample s_j is represented by a m-dimensional vector, $s_i = [g_{1j}, g_{2j}, \ldots, g_{mj}]^\intercal$.

4.2 Validation Sample Weight Adjustment

The bipartite graph models how each validation sample influences the reweighting results. Accordingly, the reweighting results can be improved by assigning lower/higher weights to the validation samples that have negative/positive effects on the reweighting results (**R3**). For example, in Fig. 4(b), s_2 and s_4 are high-quality training samples with negative weights, and s_3 is a low-quality training sample with a positive weight. Such reweighting results are incorrect. Examining the reweighting relationships in the bipartite graph reveals that the validation sample v_2 leads to these incorrect results. It incorrectly assigns negative weights to the high-quality training samples (s_2, s_4) and a positive weight to the low-quality training sample (s_3). By reducing its weight to zero, the reweighting results are improved (Fig. 4(c)). From this example, it can be seen that the key to improving the reweighting results is to evaluate the quality of the validation samples and adjust their weights. Next, we introduce how to evaluate the quality of validation samples by two measures: correctness and balancedness.

Correctness. A previous study has indicated that correct validation samples should assign positive weights to high-quality training samples and negative weights to low-quality training samples [5]. However, in practice, we do not know which training samples are of *high quality* and which ones are of *low quality*. To solve this problem, we initially regard *high-confidence* samples as high-quality and *low-confidence* samples as low-quality. The confidence of a sample reflects how likely it is of high quality [3]. Then we assess the correctness by examining whether the validation samples generate correct reweighting results on high-quality samples and low-quality samples. This can be regarded as a binary classification problem. We thus employ the binary cross-entropy, a commonly used loss for binary classification [47], to measure the deviation from the correct reweighting results:

$$\mathcal{L}_{\mathrm{c}}(w_1^s, \ldots, w_n^s) = \sum_{s_j \in S_+} -\log \phi(w_j^s) + \sum_{s_j \in S_-} -\log(1 - \phi(w_j^s)), \tag{1}$$

where w_j^s is the weight of training sample s_j, S_+ and S_- are the sets of high-quality and low-quality samples, respectively. $\phi(w) = 1/(1+e^{-w})$ is a sigmoid function to normalize the weight to $(0,1)$. When minimizing the loss defined in Eq. (1), the first term encourages positive weights on high-quality samples, while the second term encourages negative weights on low-quality samples.

Balancedness. According to the study of He *et al*. [48], balanced validation samples assign equal weights to high-quality training samples across different classes. This indicates that the weight distribution over classes is uniform. In our implementation, the weight distribution is computed on high-quality samples. Here, only high-quality samples are considered because their labels are more reliable. Let C be the number of classes and S_{c+} be the set of high-quality samples of class c $(1 \le c \le C)$. The weight distribution is obtained by summing the sample weights in each class c and then dividing by the total weights: $p_c = (\sum_{s_j \in S_{c+}} w_j^s)/(\sum_{s_j \in S_+} w_j^s)$ $(1 \le c \le C)$. Accordingly, the balancedness is measured by the deviation of the weight distribution over all classes from the uniform distribution. It is computed by the Shannon entropy loss [49]:

$$\mathcal{L}_{\mathrm{b}}(w_j^s, \ldots, w_j^s) = \sum_{c=1}^{C} p_c \log p_c. \tag{2}$$

A smaller entropy loss reflects a more balanced distribution.

With these two quality measures, the validation sample weight adjustment is formulated as a multi-task learning problem, which aims to minimize the combination of the two corresponding measures. However, it is prohibitively expensive and difficult to manually tune the optimal weighting parameters for combining them. To tackle this issue, we adopted multi-task learning [50] to automatically adjust the weighting parameters. The basic idea is to reduce the contribution of the measure with higher uncertainty. Let σ_c and σ_b denote the uncertainty of $\mathcal{L}_c$ and $\mathcal{L}_b$, respectively, the optimization goal is defined as:

$$\begin{aligned} \underset{w_i^v, \sigma_c, \sigma_b}{\text{minimize}} \quad & \frac{1}{\sigma_c^2}\mathcal{L}_{\mathrm{c}}(w_1^s, \ldots, w_n^s) + \frac{1}{\sigma_b^2}\mathcal{L}_{\mathrm{b}}(w_1^s, \ldots, w_n^s) + \log \sigma_c \sigma_b \\ \text{s.t.} \quad & w_j^s = \sum_{i=1}^{m} w_i^v g_{ij}, \ \forall j; \qquad w_i^v \ge 0, \ \forall i, \end{aligned} \tag{3}$$

where w_i^v is the weight of validation sample v_i. The first term and the second term correspond to the correctness and the balancedness, respectively. The last term penalizes too large settings of σ_c or σ_b. The values of σ_c and σ_b are learned during the optimization.

To ensure the non-negative constraints ($w_i^v \ge 0$) in the optimization process, we employ Projected Gradient Descent [51] to solve this optimization problem. The basic idea is to compute the gradient and project it to the subspace tangent to the constraints. This ensures the projected gradient satisfies the constraints.

5 Visualization

Better understanding the reweighting relationships between validation samples and training samples facilitates the adjustments to the validation samples and reweighting results. To effectively reveal a large number of such relationships, we first co-cluster the

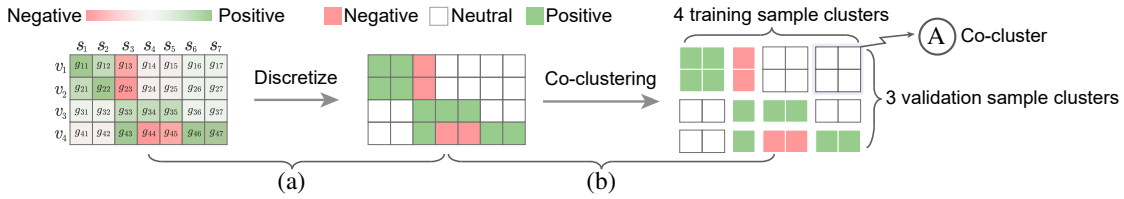


Fig. 5: The co-clustering pipeline: (a) the influence values are discretized into positive, neutral, and negative categories; (b) validation samples (rows) and training samples (columns) are grouped into clusters simultaneously.

validation samples and training samples based on the constructed bipartite graph. Then a bipartite graph visualization is developed to visually illustrate the reweighting relationships.

5.1 Co-Clustering

The bipartite graph usually contains thousands of samples and hundreds of thousands of reweighting relationships. Displaying all of them will cause severe visual clutter. To facilitate the exploration, we group similar validation samples and similar training samples simultaneously through co-clustering. The most widely used co-clustering algorithm is Spectral Co-clustering [17], [52]. A drawback of this algorithm is that it involves costly matrix decomposition whose time complexity is $O((n+m)^3)$. Here, m and n are the numbers of validation samples and training samples, respectively. Thus, it is intractable for a large graph. A method to solve this problem is Fully Automatic Cross-Associations (FACA) [53], which is an information-theory-based method. This method can reduce the time complexity to $O(nm)$. However, to use this method, the influence values (g_{ij}) need to be discrete. In our case, although the influence values are continuous, users are more interested in the non-zero influence values because they affect the reweighting results mostly. Therefore, we discretize the continuous values into three categories: positive, neutral, and negative (Fig. 5(a)). To decide the threshold ε among the positive ($\geq \varepsilon$), neutral (between $-\varepsilon$ and ε), and negative categories ($\leq -\varepsilon$), we have experimented with four datasets. The experimental results show that $\varepsilon = 0.05$ is the best value (see supplemental material for more details).

With the discretization, we then employ the FACA method [53] to build the co-clusters (Fig. 5(b)). A co-cluster consists of a pair of highly relevant validation sample cluster and training sample cluster (Fig. 5A). The basic idea of FACA is to greedily group samples into clusters while maintaining the high purity of each co-cluster. The purity is defined as the fraction of influence values belonging to the dominant influence category within each co-cluster. Take Fig. 5 as an example, the influence category of each co-cluster only belongs to one category (positive, neutral, or negative), and is therefore pure. In addition, FACA automatically determines the optimal number of clusters based on the Minimum Description Length principle [54].

In practice, the number of validation samples is typically small, often in the hundreds. In contrast, the number of training samples is much larger, reaching tens of thousands or even more. To better convey these training samples, we hierarchically cluster them by using the hierarchical clustering method developed by Chen *et al*. [17]. Specifically, we fix the validation sample clusters and recursively apply FACA to divide each high-level training sample cluster into sub-clusters.

5.2 Bipartite Graph Visualization

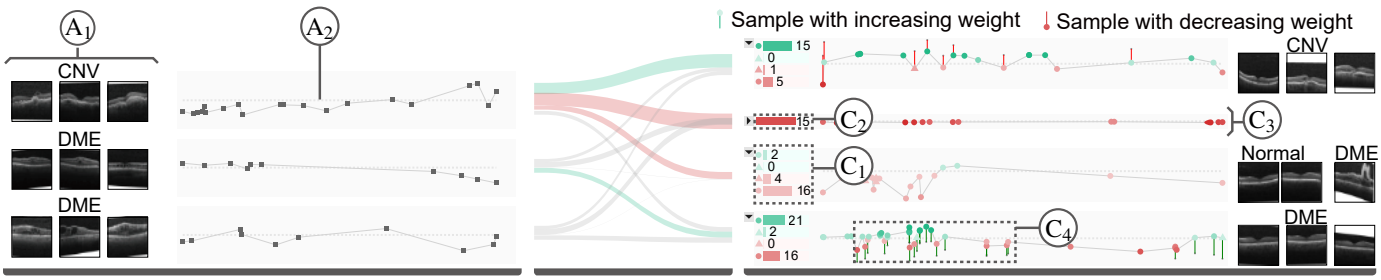
The visualization consists of two views: 1) a cluster view to provide an overview of reweighting relationships and help select the samples of interest; 2) a sample view to reveal the critical training samples that are influenced by the selected validation samples or the critical validation samples that influence the selected training samples.

5.2.1 Cluster View

A node-link diagram is employed in the cluster view to visualize the bipartite graph based on the co-clustering result (Fig. 6). In the diagram, each node represents a validation sample cluster (Fig. 6(a)) or a training sample cluster (Fig. 6(c)), and each link represents the influence between a validation sample cluster and a training sample cluster (Fig. 6(b)). We choose the node-link diagram because of its intuitiveness for understanding the reweighting relationships [55].

Validation sample cluster (**R1**). In each cluster, a dark gray square ■ represents a validation sample. The y-position of a square encodes the weight (w_i^y) of the validation sample. A dotted line in the middle of each cluster (Fig. 6A$_2$) represents the average weight of all validation samples. Squares above/below the line are the samples with larger/smaller weights. Along the x-axis, placing similar samples together facilitates faster identification of related samples by examining the neighbors of a known one, thus streamlining the exploration process. In light of this, t-SNE is employed to project validation samples onto a one-dimensional space because of its effectiveness in preserving the neighborhood relationships between samples [56]. As the number of samples in each cluster can vary widely, the perplexity in the t-SNE projection is adaptively set as the square root of the number of samples in the cluster, which gives satisfactory results in practice [57]. Other hyperparameters used in t-SNE are fixed with the default values in Scikit-learn. To better convey the trend of the weight change, the validation samples are connected by a polyline along the x-axis [58]. To help users quickly understand the content of each cluster, we sample representative images and display them on the left side of the cluster (Fig. 6A$_1$).

Training sample cluster (**R1**). The training sample cluster employs a similar design to that of the validation cluster. Here we describe the differences. In each cluster, the dotted line in the middle is the line of weight zero. The samples positioned above the line have positive weights, whereas those below it bear negative weights. To enhance the clarity of this distinction, we have employed a double-encoding method that combines vertical positioning with a diverging color scheme. In this scheme, green represents positive weights, and red represents negative weights.



(a) Validation sample cluster (node) (b) Influence (link) (c) Training sample cluster (node)

Fig. 6: The visual design of the cluster view.

Moreover, the lightness of the color represents the absolute value of the weight $|w|$. Darker colors indicate larger absolute values. Each sample is represented by a circle or a triangle glyph to indicate whether its weight is consistent with the associated confidence value (●) or not (▲). For example, an inconsistent sample with a high-confidence value but a negative weight is denoted as ▲. To facilitate the examination of inconsistent training samples, a bar chart (Fig. 6C$_1$) is used to show the number of samples of different types (●, ▲, ▲, and ●). By default, the clusters with fewer inconsistent samples are collapsed (Fig. 6C$_3$) by stacking the bars horizontally and removing the sample images (Fig. 6C$_2$).

When a cluster contains too many samples, we select a subset of representative ones, giving priority to high-quality samples S_+ and low-quality samples S_- due to their importance in evaluating the quality of validation samples. Motivated by the outlier-biased sampling method [5], [59], we try to preserve the sample distribution while prioritizing the sampling of training samples in S_+ and S_-. Specifically, the sampling probability of a sample x is proportional to $1/\rho(x)+1[x \in S_+ \cup S_-]$, where $\rho(x)$ is the density of its local region, and $1[x \in S_+ \cup S_-]$ indicates whether it is a high-quality sample or a low-quality sample.

Links (**R2**). A link between a validation sample cluster and a training sample cluster is represented by a line strip. The width of the strip encodes the sum of the influence values between the associated validation samples and training samples. Green strips and red strips represent positive and negative influence values, respectively. To reduce visual clutter, we first minimize the strip crossings by reordering the validation sample clusters and the training sample clusters. The ordering is initialized by the widely-used barycenter heuristic [60] and then improved by swapping adjacent clusters to further reduce the number of crossings [61]. Next, the strips with small influence values are shown with light gray and serve as context. They will be highlighted when hovering over the associated clusters.

5.2.2 Sample View

After selecting the samples of interest in the cluster view, users can examine the associated validation/training samples and their detailed reweighting relationships in the sample view (**R1**, **R2**). We adopt the adjacency-list design because it can compactly represent the reweighting relationships between the training samples and validation samples and their content [62]. When the exploration starts by **selecting training samples**, each row corresponds to a selected training sample. As shown in Fig. 7A, its image content is shown at the beginning of the row, followed by the top-3 positively contributing validation samples and the top-3 negatively contributing validation samples. For each training sample s_j, its weight (w_j^s) is placed under the image. For each contributing validation sample v_i, its weighted influence value ($w_i^v g_{ij}$) is placed to directly reveal how much v_i contributes to w_j^s. The circle/triangle glyphs representing the sample types (e.g., an inconsistent sample ▲/▲) are displayed beside the weights/values. When the exploration starts by **selecting validation samples**, each row corresponds to a selected validation sample (Fig. 7B). For each validation sample v_i, its weight (w_i^v) is placed under the image content. The top-3 training samples that are positively/negatively influenced by it and their associated influence values are displayed in the corresponding row.

5.3 Interactive Adjustment and Comparison

Reweighter allows users to improve the quality of validation samples by interactively adjusting validation samples and training samples. After an adjustment, the users can also compare the reweighting results before and after the adjustment.

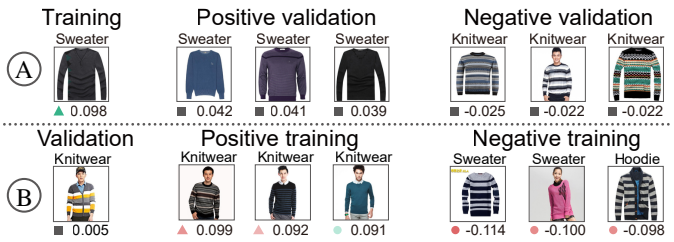


Fig. 7: The visual design of the sample view.

Interactive adjustment (**R3**). Reweighter supports the adjustments to both validation samples and training samples.

Adjusting the validation samples. Reweighter provides several ways to adjust the validation samples. If the users find a mislabeled validation sample, they can right-click the sample and relabel (✎) it in the pop-up menu. If the users identify some training samples that are not covered by the validation samples, they can add (⊕) them to the validation samples to improve the coverage. The influence values in the reweighting relationships will be updated accordingly after the two adjustments. In addition, the users can adjust the weights of validation samples if they are not appropriate. Since it is difficult for the users to specify the exact weights, we allow them to simply indicate the direction of the weight adjustment by dragging the samples up (increase) or down (decrease). The adjustment is formulated as the inequality constraint of the weight adjustment optimization problem defined in Eq. (3). In our implementation, the inequality constraint is set as $w_i^v \geq (1+\gamma)\tilde{w}_i^v$ when increasing the weight and as $w_i^v \leq (1-\gamma)\tilde{w}_i^v$ when decreasing the weight. Here $\tilde{w}_i^v$ is the previous weight, and $\gamma = 0.1$. The details regarding this default value can be found in supplemental material.

Adjusting the training samples. As the quality measures of validation samples are calculated based on the high-quality sample set S_+ and the low-quality sample set S_-, users can implicitly improve the quality of validation samples by refining S_+ and S_-. To this end, users first select a set of samples of interest by brushing them in a training sample cluster or clicking a bar in a bar chart associated with a cluster. Then they can right-click the selected samples and verify them as high-quality (👍) or low-quality (👎) in the pop-up menu. In addition, they can verify samples as high-quality by increasing their weights or as low-quality by decreasing their weights. The user-verified high-quality and low-quality samples are added to S_+ and S_-, respectively. With the refined S_+ and S_-, the calculation of the correctness (Eq. (1)) and balancedness (Eq. (2)) are updated accordingly, which affects the optimization results in Eq. (3).

Comparison of the reweighting results (**R4**). After the users finish their adjustments, the reweighting results are updated accordingly by re-solving the optimization problem defined in Eq. (3). To facilitate the result comparison between before and after the adjustments, the users can switch to the "diff" mode, which highlights significant changes. In this mode, the samples are positioned based on their current reweighting results. For samples with significant weight changes, each has a green or red line connecting its previous and current positions to highlight the change (Fig. 6C$_4$). The color indicates whether the weight is increased or decreased. By default, the samples with weight changes among the top 10% are considered to have significant changes, and the percentage (10%) can be modified by the users. After examining

the weight changes, they can undo the adjustments if the changes are unsatisfactory.

6 Evaluation

We evaluated our validation sample improvement method using three separate experiments. These experiments were conducted automatically, requiring no human intervention. Moreover, through two case studies, we demonstrate how Reweighter enables users to interactively improve validation samples, leading to better reweighting results.

6.1 Quantitative Evaluation

This section evaluates the effectiveness of the improved validation samples in the separate scenarios of noisy labels and imbalanced class distributions, as well as the scenario of their combination.

Datasets. Four datasets are used in the evaluation. The first two, **CIFAR10** and **CIFAR100**, are commonly used benchmark datasets for evaluating sample reweighting methods [2], [3]. CIFAR10 contains 60,000 samples in 10 classes, with 5,000 training samples and 1,000 test samples per class. Similarly, CIFAR100 contains 60,000 samples in 100 classes, with 500 training samples and 100 test samples per class. These two datasets are clean and balanced. We thus simulated noisy labels and imbalanced class distributions for these two datasets. The third one, **Clothing** dataset [5], contains 37,497 samples in 14 different classes (T-shirt, Shirt, Knitwear, *etc*.). We used 36,000 samples for training and 1,497 samples for testing. As the sample labels are automatically extracted from the associated text descriptions, this dataset is noisy with 38.3% of the samples being mislabeled. The fourth one, **OCT** (Optical Coherence Tomograph) dataset [63], contains 20,000 retinal OCT images (17,000 for training and 3,000 for test). The images are of four classes: Normal, Choroidal NeoVascularization (CNV), DiabeticMacular Edema (DME), and Drusen. This dataset is imbalanced as there are only 1,000 training samples of DME and 1,000 of Drusen, compared to 10,000 of Normal and 5,000 of CNV. The imbalance factor, which is the ratio between the number of training samples in the largest class and that in the smallest class, is $10000/1000 = 10$.

Experimental settings. To evaluate the effectiveness of the developed validation sample improvement method, we combined it with a state-of-the-art reweighting method, FSR [3]. We fed the improved validation samples into FSR and evaluated whether the generated weights further improved model performance. For comparison, we used Uniform (no reweighting) and FSR [3] as the baseline methods. Following the settings of FSR, we utilized WideResNet-28-10 [64] for low-resolution images (CIFAR10 and CIFAR100) and ResNet50 [65] for high-resolution images (Clothing and OCT). The models were also trained for 255 epochs with a cosine learning rate decay. The numbers of validation samples are 100, 200, 140, and 80 for CIFAR10, CIFAR100, Clothing, and OCT datasets, respectively. For evaluation in the scenario of noisy labels, the Clothing dataset with real-world label noise is used. In addition, CIFAR10 and CIFAR100 with simulated noisy labels are used to evaluate the classification accuracy under different noise ratios. Pseudo labeling utilizes a trained model to generate labels for unlabeled samples and then incorporates these newly labeled samples back into the training set to refine the model further. This method closely simulates human-like labeling errors [66]. Using this method, we first randomly selected and labeled $k \in \{10, 20, 50, 100\}$ samples for each class in CIFAR10 and CIFAR100 and built four classification models [32]. The pseudo labels of the remaining samples in the two datasets are then derived using these classification models. The noise ratios of the pseudo labels are summarized in Table 1. For evaluation in the scenario of imbalanced class distributions, the OCT dataset is used. Similarly, we used CIFAR10 and CIFAR100 with simulated imbalanced class distributions to evaluate the accuracy under different imbalance factors. The simulation was performed by sampling a subset of training samples following the Pareto distribution with imbalance factors $\lambda \in \{5, 10, 20, 50\}$. In the combined scenario, we simulated label noise using different numbers of labeled samples per class (k) in the OCT dataset, and simulated imbalanced class distributions with different imbalance factors (λ) on the Clothing dataset. For CIFAR10 and CIFAR100, we evaluated the accuracy under all the combinations of k and λ. We repeated each experiment three times and reported the average accuracy on the test set.

TABLE 1: The noise ratios of the simulated noisy labels.

Dataset	# labeled samples per class			
	10	20	50	100
CIFAR10	0.48	0.42	0.36	0.32
CIFAR100	0.70	0.65	0.55	0.45

TABLE 2: Test set accuracy under noisy labels.

Dataset	Method	# labeled samples per class			
		10	20	50	100
CIFAR10	Uniform	56.3%	62.0%	67.1%	70.5%
	FSR	67.6%	74.7%	78.6%	81.2%
	Ours	**68.4%**	**75.4%**	**79.0%**	**81.4%**
CIFAR100	Uniform	30.4%	35.9%	43.9%	50.4%
	FSR	37.0%	43.5%	53.4%	58.8%
	Ours	**38.1%**	**44.2%**	**53.7%**	**59.1%**

(a) Simulated noisy labels.

Dataset	Noise ratio	Method	Accuracy
Clothing	0.383	Uniform	57.8%
		FSR	70.5%
		Ours	**71.9%**

(b) Real-world noisy labels.

Results. Table 2 shows that our method achieves better performance in the noisy label scenario. The performance gain increases with the decreasing number of labeled samples, which demonstrates the effectiveness of our method in handling more label noise. We also noticed that the gain on the Clothing dataset is larger than the gain on the CIFAR datasets. The clearer differences between the classes of the CIFAR datasets enable FSR to easily select high-quality validation samples. As a result, there is less room for additional improvements. Unlike the CIFAR datasets, the Clothing dataset exhibits more subtle differences between its classes, making classification challenging. This type of task is known as fine-grained classification [67]. These subtle differences pose difficulties for the FSR method in selecting high-quality validation samples. In comparison, our method consistently im-

TABLE 3: Test set accuracy under imbalanced class distributions.

Dataset	Method	Imbalance factor 5	10	20	50
CIFAR10	Uniform	92.2%	90.1%	87.1%	81.2%
	FSR	93.4%	90.1%	87.2%	81.7%
	Ours	**93.6%**	**90.3%**	**87.8%**	**82.1%**
CIFAR100	Uniform	63.8%	31.2%	32.6%	34.9%
	FSR	69.8%	63.9%	56.8%	48.9%
	Ours	**70.2%**	**64.5%**	**57.2%**	**49.4%**

(a) Simulated imbalanced class distributions.

Dataset	Imbalance factor	Method	Accuracy
OCT	10	Uniform	78.1%
		FSR	82.8%
		Ours	**83.4%**

(b) Real-world imbalanced class distributions.

TABLE 4: Test set accuracy under combined noise.

Dataset	# labels per class	Imbal. factor	Uniform	FSR	Ours
CIFAR10	10	5	55.1%	57.4%	**58.0%**
	10	10	54.2%	56.3%	**56.9%**
	20	5	62.4%	62.8%	**63.2%**
	20	10	61.2%	62.1%	**62.7%**
CIFAR100	10	5	29.5%	30.4%	**30.8%**
	10	10	28.2%	29.0%	**29.7%**
	20	5	34.7%	35.0%	**35.2%**
	20	10	32.9%	33.4%	**33.8%**
Clothing	N/A	5	55.9%	59.8%	**62.1%**
	N/A	10	52.6%	57.4%	**59.8%**
OCT	10	N/A	49.5%	57.4%	**59.6%**
	20	N/A	53.0%	63.8%	**65.2%**

proves the quality of validation samples and thus brings more performance gain. In the imbalanced class distribution scenario, our method improves the classification accuracy on both the simulated and real-world datasets (Table 3). For the retinal OCT images that are hard to classify correctly, our method still achieves better performance (Table 3(b)). Table 4 shows that in scenarios where both noisy labels and imbalanced class distributions are present, our method achieves a larger performance gain. The performance gain on the OCT dataset and the Clothing dataset reached around 2%, exceeding that on the CIFAR datasets. This further demonstrates that our method is more effective to improve validation samples in fine-grained classification, a task frequently encountered in real-world applications. Due to the page limit, we only present the results with small numbers of labeled samples per class and small imbalance factors. The full results are in supplemental material.

6.2 Case Studies

To demonstrate how Reweighter facilitates the human-AI collaboration in improving the reweighting results, we conducted two case studies. They started from the output of the automatic reweighting method. The case study results showed that users could further improve the accuracy by interactively providing a small amount of feedback.

6.2.1 *Interactively Reweighting the Clothing Dataset*

In this case study, we collaborated with E1 to interactively improve the reweighting results of the Clothing dataset. This dataset, sourced from real-world data, contains many noisy labels. The initial accuracy without human intervention was 71.9% (Table 2(b)).

Overview (**R1**). After co-clustering, there were 14 validation sample clusters and 35 training sample clusters. Fig. 1(a) shows three validation sample clusters and six training sample clusters among them. E1 first looked at the validation sample clusters and noticed that in each cluster, there was no big difference in the y-positions of the validation samples. This is in line with the expectation that each validation sample should favorably contribute to the reweighting results. However, the y-positions of the validation samples in clusters V_1 and V_2 are lower than the dotted line (the average value), indicating that their contributions are relatively low. Thus, E1 decided to examine them first.

Identifying the low-quality samples (**R1**, **R2**). E1 first selected all the 20 validation samples in clusters V_1 and V_2. These validation samples and the training samples that were highly influenced by them were displayed in the sample view. After examining the image content, he found that some validation samples in V_1 were actually "sweater" but mislabeled as "knitwear" (Fig. 8A), while some samples in V_2 were actually "knitwear" but mislabeled as "sweater" (Fig. 8B). Similar mislabeling issues were also identified in their associated training samples. For example, the samples in Fig. 8A_P were "sweater" but mislabeled as "knitwear." To explore more training samples that were highly influenced by the selected validation samples, E1 turned to examine the training sample clusters. He found that clusters S_1 and S_2 were highly influenced and contained the most inconsistent samples ▲ and ▲ (Fig. 1(a)). E1 selected these inconsistent samples to examine their content and that of their associated validation samples in the sample view. Fig. 8C shows two inconsistent samples ▲ with positive weights but low-confidence values in cluster S_2. They were in fact "knitwear" but mislabeled as "sweater," so as their most positively contributing validation samples (Fig. 8C_P). The wrong labels of these validation samples explain the positive weights of low-quality samples in Fig. 8C. E1 concluded that the label noise in V_1 and V_2 led to the inconsistencies in S_1 and S_2.

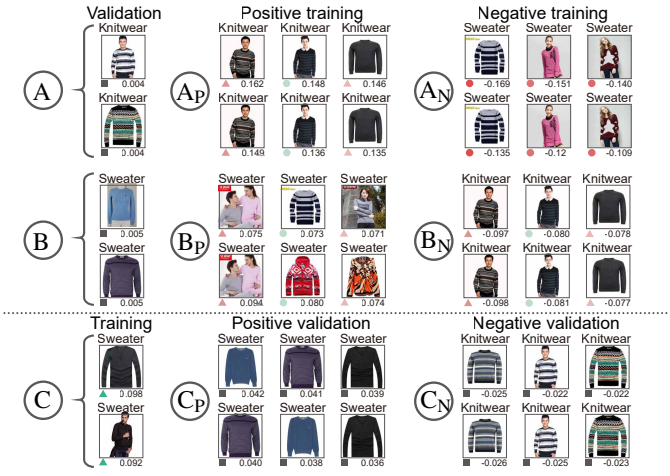


Fig. 8: Analyzing low-quality validation samples in clusters V_1 (A) and V_2 (B) and inconsistent training samples in cluster S_2 (C).

Improving the quality of validation samples (**R3**, **R4**). From these observations, E1 decided to 1) correct the noisy labels in clusters V_1 and V_2, 2) increase the weights of the high-quality validation samples in V_1 and V_2, and 3) examine the inconsistent training samples ▲ and ▲ in clusters S_1 and S_2 and verify them as high/low-quality samples. In total, he corrected the labels of 9 validation samples, increased the weights of 11 validation samples, and verified 49 training samples. After the adjustments (Fig. 1(b)), the inconsistency in cluster S_2 was eased, and hence the cluster was collapsed automatically (Fig. $1S_2'$). However, as shown in the bar chart on the left of Fig. $1S_1'$, there were more inconsistent samples ▲ with high-confidence values but negative weights in the training sample cluster S_1'. To understand the reason, E1 compared the reweighting results before and after the adjustments in the "diff" mode. On the right of Fig. $1S_1'$, he found many red lines. This indicated that many high-confidence samples in S_1 had a large drop in their weights and thus resulted in more inconsistency. Examining the image content of these samples, he found that they were samples of other classes but mislabeled as "knitwear." These low-quality samples were not identified in the previous step due to their incorrect positive weights coinciding with their incorrect high-confidence values. The adjustments he made shifted their weights to the negative side, revealing the inconsistency. E1 then verified these 12 samples as low-quality samples to reduce their confidence values. As shown in Fig. $1S_1''$, there are fewer inconsistent samples ▲. The adjustments were used to fine-tune the model, leading to an accuracy boost from 71.9% to 72.7%.

Similar mislabeling issues were observed in "down coat," "jacket," "windbreaker," "vest," and "dress." E1 corrected the labels of 13 validation samples and verified 71 training samples as high/low-quality samples. Upon satisfaction, he obtained a set of better-quality validation samples and a set of verification of high/low-quality training samples, which were used to fine-tune the model. The accuracy was boosted from 72.7% to 75.4%. In summary, E1 further improved the model accuracy from 71.9% to 75.4% (+3.5%) by correcting the labels of 22 validation samples and verifying 132 training samples as high/low-quality samples.

6.2.2 *Interactively Reweighting the OCT Dataset*

In this case study, we collaborated with E3 to improve the reweighting results of the OCT dataset with added noisy labels (generated from 20 labeled samples per class), which was used in the aforementioned combined scenarios. This dataset is more complex because it contains both noisy labels and imbalanced class distributions. E3 has experience in developing models for diagnosing retinal edema. We also invited D1, an ophthalmology clinical doctor, to examine medical images and explain his judgments based on medical expertise. The initial accuracy without human intervention was 65.2% (the last row in Table 4).

Overview (**R1**). After co-clustering, there were 6 validation sample clusters and 11 training sample clusters. E3 noticed that the weights of validation samples in Fig. 9A and Fig. 9B are lower than other validation samples in the same clusters. This indicated that these samples did not favorably contribute to the reweighting results. He decided to analyze them first.

Diagnosing & Improving low-weight validation samples (**R1**, **R2**, **R3**). As the two validation samples in Fig. 9A (with the label "Normal") had the lowest weight, E3 examined them first. By examining their image content (Fig. $9A_1$) and the associated training samples, he found that these two validation samples and two of the training samples (Fig. $9A_2$ and Fig. $9A_3$) contained cystoid space (the dark chamber in the middle). This makes them different from other "Normal" samples. He then consulted with doctor D1, who confirmed that both the validation samples and the two training samples were "DME" but mislabeled as "Normal." He explained that the cystoid space was a typical symptom of "DME." However, the green circles ● indicated that these two training samples were wrongly assigned positive weights. He corrected the labels of these two validation samples and verified the two training samples as low-quality samples. Next, E3 analyzed the validation samples in Fig. 9B in a similar way. D1 confirmed that these two samples were "DME" but mislabeled as "Drusen." The hyporeflective cysts between the inner plexiform layer and the outer plexiform layer form a triangle with white borders (Fig. $9B_1$), which is the symptom of "DME." After examining the highly influenced training samples, he found two inconsistent training samples ▲ with similar symptoms (Fig. $9B_2$, B_3) were also "DME" but mislabeled as "Drusen." These training samples appeared in Fig. $9B_4$, where he found more similar samples that were mislabeled as "Drusen." E3 corrected the labels of these two low-quality validation samples and verified seven training samples in Fig. $9B_4$ as low-quality samples. After these adjustments, the inconsistency in the training sample cluster "Drusen" was reduced, and the cluster was collapsed automatically. The accuracy was improved from 65.2% to 66.1%.

Adding missing validation samples (**R1**, **R2**, **R3**). After

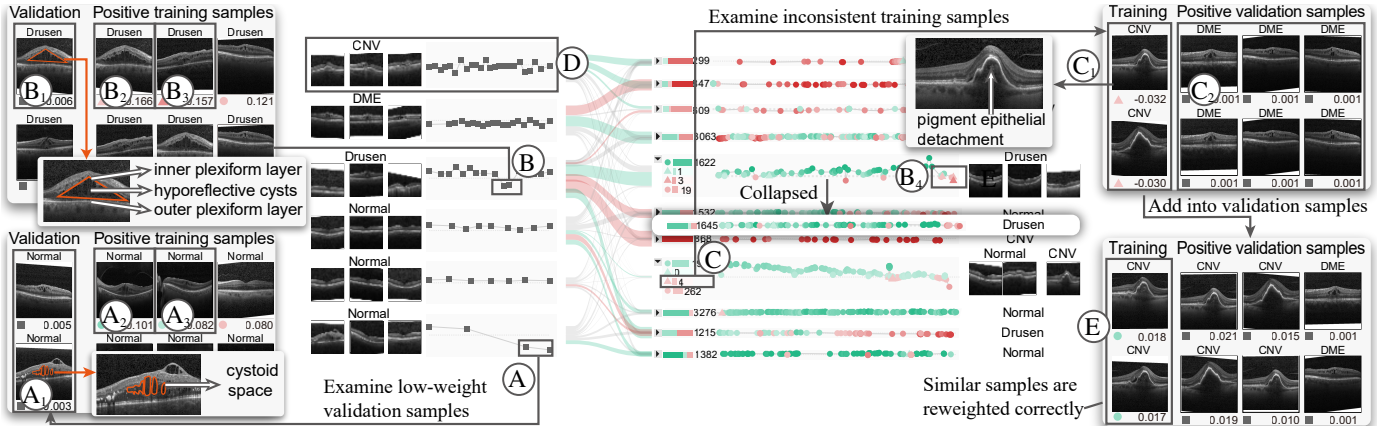


Fig. 9: Interactively improving the quality of validation samples in the OCT case study.

adjusting the validation samples, E3 turned to examine the training samples. He noticed that there were four inconsistent samples ▲ with high-confidence values but negative weights in a cluster containing both "Normal" and "CNV" samples (Fig. 9C). He clicked the bar to examine their content and the contributing validation samples. The image content (Fig. $9C_1$) showed that they were high-quality samples of "CNV." However, there were no validation samples of "CNV" that positively influenced them (Fig. $9C_2$). He then examined the validation samples in the validation sample cluster "CNV" (Fig. 9D) and found that these samples were not representative enough for the "CNV" class. According to doctor D1, the "CNV" training samples in Fig. 9C show serious pigment epithelial detachments adjacent to a small area of subretinal fluid (the sharp protrusion), which is one of the typical symptoms of "CNV." However, none of the existing validation samples in "CNV" have this symptom. This explained why these training samples received negative weights. To address this issue, E3 added two of them into the validation set and updated the reweighting results. The other two training samples (Fig. 9E) were then correctly assigned positive weights due to the added validation samples. The model was fine-tuned with these added validation samples, and the accuracy was increased from 66.1% to 68.1%. In summary, E3 further improved the model accuracy from 65.2% to 68.1% (+2.9%) by adding two validation samples, correcting the labels of four validation samples, and verifying nine training samples as high/low-quality samples.

6.3 Comparison with DataDebugger

To demonstrate the effectiveness of Reweighter, we compared it with DataDebugger [5], which is the state-of-the-art visual analysis tool for interactively correcting label noise in training samples. In DataDebugger, users need to examine samples and provide exact labels for them. The provided labels are then propagated to other samples with a label correction algorithm. However, since no additional information is considered when correcting the labels, more samples with exact labels are required to achieve acceptable performance. In contrast, Reweighter enables users to analyze the reweighting relationships between validation samples and training samples. This additional relationship information facilitates the correction of label noise in validation samples and the verification of high/low-quality training samples. These corrections and verification help improve the reweighting results and boost the model performance. In this experiment, we invited E1-E4 to use both tools and recorded their adjustments and time spent. All the experts are familiar with both tools. To reduce the potential bias resulting from the order in which the tools were used, E1 and E2 were asked to use DataDebugger first, while E3 and E4 were asked to use Reweighter first. The Clothing dataset was employed because they are more familiar with it, and the initial accuracy was 71.9% for both tools. Table 5 shows the results from each expert. It can be seen that to achieve comparable performance, Reweighter requires an average of 22.75 exact labels, which is much fewer than 297.25 labels that DataDebugger needs. This is because DataDebugger requires the experts to label training samples and then propagate them to other samples. In contrast, Reweighter only requires the experts to label validation samples, which are much fewer than training samples (140 compared to 36,000). Although Reweighter requires additional verification of high/low-quality training samples, the average number of verified samples (156.25) is still less than the average number of exact labels required by DataDebugger (297.25). Moreover, E1 pointed out that providing verification is easier than providing exact labels, particularly with 14 classes in the Clothing dataset. The verification effort is further simplified because high/low-quality samples are usually grouped together in Reweighter. As illustrated in Fig. 1, the 18 inconsistent training samples with low weights in cluster S_1 can be collectively verified as low-quality samples with just one click. In contrast, since their ground-truth labels are different ("knitwear," "sweater," "shawl," and "underwear"), the experts have to re-label each sample separately, which takes more time. On average, they spent 0.57 hours using Reweighter, which was shorter than the 1.49 hours they spent with DataDebugger to achieve comparable performance.

TABLE 5: The numbers of provided exact labels, verification, and time comparison between DataDebugger and Reweighter.

Method	Expert	# labels	# verification	Time	Accuracy
DataDebugger	E1	308	0	1.66h	75.0%
	E2	293	0	1.50h	74.7%
	E3	331	0	1.63h	75.1%
	E4	257	0	1.17h	74.6%
Reweighter	E1	22	132	0.54h	75.4%
	E2	26	179	0.62h	75.6%
	E3	25	171	0.66h	75.4%
	E4	18	143	0.47h	75.1%

7 Expert Feedback and Discussion

After the case studies, we conducted six semi-structured interviews with the four experts we collaborated with (E1-E4) and two newly invited ones (E5 and E6). For the experts who were not involved in the case studies, we first introduced Reweighter and then presented the case studies. Then the experts were asked to identify low-quality validation samples and make adjustments using the tool. Finally, we discussed with the experts about the strengths and limitations of the tool. The entire process lasted from 50 to 85 minutes. The experts were generally positive about the usability of Reweighter. A few limitations were also identified by the experts, from which we summarized several future research directions.

7.1 Usability

Adopting simple visual design. All the experts appreciated the simplicity of the visual design used in Reweighter. They indicated that the cluster view was intuitive and easy to understand, facilitating the identification of low-quality validation samples. E4 commented, "The visual encoding of validation samples is concisely explained in the legend. Guided by the explanation, I can quickly identify the low-weight validation samples and select them in the cluster view. Moreover, the bipartite graph provides a clear overview of their reweighting relationships, enabling me to conveniently trace the related training samples for more comprehensive examination." The experts also emphasized the usefulness of the sample view in identifying low-quality validation samples. Their prior experience with list-based visual designs made this view particularly accessible and streamlined their analysis. For example, E5 mentioned, "During an analysis, after selecting a few inconsistent training samples, I found that some validation samples appeared repeatedly in the sample view. More examination showed

that these repetitively occurring samples were of low quality and required curation." He added that this simple but intuitive design made these recurring samples stand out.

Providing an efficient way to diagnose quality issues. All the experts were able to efficiently identify low-quality validation samples and made appropriate adjustments. They agreed that Reweighter reduced their efforts in diagnosing quality issues of validation samples. E2 commented, "The glyphs ▲/▲ are useful in highlighting inconsistent training samples for further examination. These samples often reflect quality issues in validation samples." Meanwhile, E5 highlighted the practicality of the "diff" mode, noting, "It provides a convenient method for me to examine weight changes and evaluate the appropriateness of my adjustments." Interestingly, our observations revealed a divergence in expert strategies: a subset of experts (E1, E3, E4, E6) usually started their analysis from low-weight validation samples, while the others (E2, E5) preferred to start their analysis from inconsistent training samples. This indicates that different analysis pipelines offered by our tool meet the diverse analytical preferences of individual experts. After examining the quantity evaluation and detailed case studies, E1 concluded that Reweighter is effective at handling fine-grained classification, which aims to distinguish different classes that are closely related to each other (*e.g.*, "knitwear" and "sweater" in the Clothing dataset). Given the subtle differences in fine-grained classification, validation samples often have more label noise. Our tool helps effectively identify and correct such label noise, and thus leads to a larger performance gain. The experts were also satisfied with the performance gains on real-world datasets ranging from 1.4% to 3.2%. E6 commented, "In real-world applications, even a performance gain close to 1% is important and highly valued. For example, a slight improvement in accuracy can enhance short video recommendations. This, in turn, leads to increased user engagement and extended interaction duration on the platform."

Being easily integrated with different reweighting methods. Our work demonstrates how Reweighter improves the performance of FSR [3]. Representing reweighting relationships as a bipartite graph offers the potential to integrate different reweighting techniques. For example, E6 indicated that our tool can be easily integrated with different reweighting methods by modeling the reweighting relationships as a bipartite graph. He noted, "The tool directly supports validation-sample-based reweighting methods by extracting the reweighting relationships between validation samples and training samples using the corresponding reweighting methods." As distribution-based methods [12], [15] do not include validation samples in the reweighting process, they cannot directly apply Reweighter. In these situations, there is a need to understand their reweighting mechanism and then construct the corresponding bipartite graph. For example, the method proposed by Liu *et al*. estimates the sample weights by assessing the embedding similarity between training samples and a set of selected trusted samples [15]. These similarity relationships can be modeled as a bipartite graph, which enables the utilization of Reweighter.

7.2 Limitations and Future Work

How to provide in-context recommendation. Currently, the experts started their analysis after identifying the samples of interest (*e.g.*, low-quality validation samples). To save exploration efforts, E1, E3, and E5 expressed a preference for the tool to automatically recommend these samples to them. A straightforward solution is to recommend low-weight validation samples and inconsistent training samples. However, E3 was not satisfied with this solution, "I can conduct the same analysis by selecting validation samples with low y-positions and training samples with triangle shapes. Such recommendations do not offer me additional benefits." A more favorable solution is to dynamically recommend the samples of interest based on users' previous adjustments. As E1 mentioned, "If Reweighter can recommend more samples with similar quality issues after I have identified a few, it would greatly improve efficiency. Such a method will streamline the analysis process by maintaining a consistent context." The challenge lies in simultaneously considering the quality issues and context when making recommendations, which deserves further investigation.

How to support online monitoring. After using our tool, E6 suggested that it would be more helpful if Reweighter could analyze the reweighting relationships and results during the model training process, "The reweighting relationships and results will gradually change as the model is trained over epochs. If Reweighter supports tracking these dynamic changes, I can intervene and adjust validation samples to match current models better." This perspective was echoed by both E1 and E3. After a thorough discussion, we reached a consensus that our tool could partially support this functionality by automatically updating the reweighting relationships during model training. However, the analysis of these relationships and results across different epochs still presents challenges for model developers. How to design an informative overview that effectively summarizes the changes over epochs deserves further exploration. Moreover, in an online monitoring system, it is crucial to have an alert mechanism that can promptly report potential quality issues to model developers. How to design an alert mechanism that can accurately report the issues and reduce false alarms requires more investigation.

How to extend to other tasks. Although our tool is evaluated with image classification, it directly supports the classification tasks of other types of data, including text, video, and chart [68], [69]. This is because the extraction of reweighting relationships is not limited to specific types of data. The experts also discussed the potential of applying our tool to other tasks, such as object detection and image segmentation. The bipartite graph construction method, co-clustering algorithm, and visualization can be directly applied to other tasks because their design is inherently task-agnostic, focusing on reweighting relationships rather than specific task details. However, the validation sample improvement method needs adaptation to address the complexities of object detection and image segmentation tasks. Elaborating on this, E6 highlighted the differences between the tasks: "In classification, I only need to verify whether the labels are clean, which can be performed in batch mode. However, object detection and image segmentation require additional feedback on the bounding boxes and segmentation masks, making batch annotation adjustment unfeasible. Developing strategies to address the unique challenges of object detection and image segmentation in the context of validation sample improvement remains a promising avenue for future research."

8 Conclusion

We have developed Reweighter, a visual analysis tool for generating better reweighting results of training samples. The key is to model the reweighting relationships between validation samples and training samples as a bipartite graph. Based on this graph, we

develop a validation sample improvement method and a co-cluster-based bipartite visualization. They are tightly integrated together to support an interactive sample reweighting process, where the user adjustments are converted to the constraints of the validation sample improvement method. This process interactively improves the validation samples and hence generates better reweighting results. A quantitative evaluation and two case studies are conducted to demonstrate the effectiveness and usefulness of Reweighter in improving sample reweighting methods.

ACKNOWLEDGMENTS

This research is supported by the National Natural Science Foundation of China (No.s U21A20469, 61936002), the National Key R&D Program of China (No. 2020YFB2104100), grants from the Institute Guo Qiang, THUIBCS, and BLBCI. The authors would like to thank Yuxing Zhou for his efforts in the validation sample weight adjustment algorithm, Dr. Changjian Chen and Yusong Zhu for their engaging discussion about the co-clustering algorithm, and Lanxi Xiao, Jing Wang, and Hanjie Yu for their valuable comments on the visualization design.

REFERENCES

[1] B. Zhang, Y. Wang, W. Hou, H. Wu, J. Wang, M. Okumura, and T. Shinozaki, "Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling," in *Proc. Adv. Neural Inform. Process. Syst.*, 2021, pp. 18 408–18 419.

[2] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4334–4343.

[3] Z. Zhang and T. Pfister, "Learning fast sample re-weighting without reward data," in *Proc. Int. Conf. Comput. Vis.*, 2021, pp. 725–734.

[4] D. A. Hoang, C. Nguyen, B. Vasileios, and G. Carneiro, "Maximising the utility of validation sets for imbalanced noisy-label meta-learning," *CoRR*, 2022.

[5] S. Xiang, X. Ye, J. Xia, J. Wu, Y. Chen, and S. Liu, "Interactive correction of mislabeled training data," in *Proc. IEEE Conf. Vis. Anal. Sci. Technol.*, 2019, pp. 57–68.

[6] 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.

[7] Y. Xu, L. Zhu, L. Jiang, and Y. Yang, "Faster meta update strategy for noise-robust deep learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 144–153.

[8] S. Song, C. Li, D. Li, J. Chen, and C. Wang, "GraphDecoder: Recovering diverse network graphs from visualization images via attention-aware learning," *IEEE Transactions on Visualization and Computer Graphics*, 2022, to be published.

[9] X. Zhang, J. P. Ono, H. Song, L. Gou, K.-L. Ma, and L. Ren, "SliceTeller: A data slice-driven approach for machine learning model validation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 842–852, 2023.

[10] S. Robertson, Z. J. Wang, D. Moritz, M. B. Kery, and F. Hohman, "Angler: Helping machine translation practitioners prioritize model improvements," in *CHI*, 2023.

[11] J. Shu, Q. Xie, L. Yi, Q. Zhao, S. Zhou, Z. Xu, and D. Meng, "Meta-weight-net: Learning an explicit mapping for sample weighting," in *Advances in Neural Information Processing Systems*, 2019.

[12] T. Liu and D. Tao, "Classification with noisy labels by importance reweighting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 3, pp. 447–461, 2016.

[13] R. Wang, T. Liu, and D. Tao, "Multiclass learning with partially corrupted labels," *TNNLS*, vol. 29, no. 6, pp. 2568–2580, 2017.

[14] Y. Wang, A. Kucukelbir, and D. M. Blei, "Robust probabilistic modeling with bayesian data reweighting," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3646–3655.

[15] Y.-P. Liu, N. Xu, Y. Zhang, and X. Geng, "Label distribution for learning with noisy labels," in *Proc. Int. Joint Conf. Artif. Intell.*, 2021, pp. 2568–2574.

[16] W. He, L. Zou, A. K. Shekar, L. Gou, and L. Ren, "Where can we help? a visual analytics approach to diagnosing and improving semantic segmentation of movable objects," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 1040–1050, 2022.

[17] C. Chen, J. Wu, X. Wang, S. Xiang, S.-H. Zhang, Q. Tang, and S. Liu, "Towards better caption supervision for object detection," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 4, pp. 1941–1954, 2022.

[18] S. E. Whang and J.-G. Lee, "Data collection and quality challenges for deep learning," *Proc. VLDB Endow.*, vol. 13, no. 12, p. 3429–3432, 2020.

[19] W. Yang, M. Liu, Z. Wang, and S. Liu, "Foundation models meet visualizations: Challenges and opportunities," *arXiv preprint arXiv:2310.05771*, 2023.

[20] 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 *Proc. Int. Conf. Hum.-Comput. Interact.*, 2011, pp. 618–627.

[21] K. Kurzhals, M. Hlawatsch, C. Seeger, and D. Weiskopf, "Visual analytics for mobile eye tracking," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 301–310, 2017.

[22] 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, 2020.

[23] G. Halter, R. Ballester-Ripoll, B. Flueckiger, and R. Pajarola, "VIAN: A visual annotation tool for film analysis," *Comput. Graph. Forum*, vol. 38, no. 3, pp. 119–129, 2019.

[24] J. Eirich, J. Bonart, D. Jäckle, M. Sedlmair, U. Schmid, K. Fischbach, T. Schreck, and J. Bernard, "Irvine: A design study on analyzing correlation patterns of electrical engines," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 11–21, 2022.

[25] A. S. Júnior, C. Renso, and S. Matwin, "Analytic: An active learning system for trajectory classification," *IEEE Comput. Graph. Appl.*, vol. 37, no. 5, pp. 28–39, 2017.

[26] F. L. Dennig, T. Polk, Z. Lin, T. Schreck, H. Pfister, and M. Behrisch, "FDive: Learning relevance models using pattern-based similarity measures," in *Proc. IEEE Conf. Vis. Anal. Sci. Technol.*, 2019, pp. 69–80.

[27] L. S. Snyder, Y.-S. Lin, M. Karimzadeh, D. Goldwasser, and D. S. Ebert, "Interactive learning for identifying relevant tweets to support real-time situational awareness," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 558–568, 2020.

[28] F. Sperrle, R. Sevastjanova, R. Kehlbeck, and M. El-Assady, "VIANA: Visual interactive annotation of argumentation," in *Proc. IEEE Conf. Vis. Anal. Sci. Technol.*, 2019, pp. 11–22.

[29] J. Choi, S.-E. Lee, Y. Lee, E. Cho, S. Chang, and W.-K. Jeong, "Dxplorer: A unified visualization framework for interactive dendritic spine analysis using 3d morphological features," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 2, pp. 1424–1437, 2023.

[30] S. Jia, Z. Li, N. Chen, and J. Zhang, "Towards visual explainable active learning for zero-shot classification," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 791 –801, 2022.

[31] Z. Zhao, P. Xu, C. Scheidegger, and L. Ren, "Human-in-the-loop extraction of interpretable concepts in deep learning models," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 780–790, 2022.

[32] W. Yang, X. Ye, X. Zhang, L. Xiao, J. Xia, Z. Wang, J. Zhu, H. Pfister, and S. Liu, "Diagnosing ensemble few-shot classifiers," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 9, pp. 3292–3306, 2022.

[33] C. Chen, J. Yuan, Y. Lu, Y. Liu, H. Su, S. Yuan, and S. Liu, "OoD-Analyzer: Interactive analysis of out-of-distribution samples," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 7, pp. 3335–3349, 2021.

[34] W. Yang, Z. Li, M. Liu, Y. Lu, K. Cao, R. Maciejewski, and S. Liu, "Diagnosing concept drift with visual analytics," in *Proc. IEEE Conf. Vis. Anal. Sci. Technol.*, 2020, pp. 12–23.

[35] X. Wang, W. Chen, J. Xia, Z. Chen, D. Xu, X. Wu, M. Xu, and T. Schreck, "Conceptexplorer: Visual analysis of concept drifts in multi-source time-series data," in *Proc. IEEE Conf. Vis. Anal. Sci. Technol.*, 2020, pp. 1–11.

[36] A. Yeshchenko, C. Di Ciccio, J. Mendling, and A. Polyvyanyy, "Visual drift detection for sequence data analysis of business processes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 8, pp. 3050 – 3068, 2022.

[37] L. Gou, L. Zou, N. Li, M. Hofmann, A. K. Shekar, A. Wendt, and L. Ren, "VATLD: a visual analytics system to assess, understand and improve traffic light detection," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 261–271, 2021.

[38] J. H. Park, S. Nadeem, S. Mirhosseini, and A. Kaufman, "C^2A: Crowd consensus analytics for virtual colonoscopy," in *Proc. IEEE Conf. Vis. Anal. Sci. Technol.*, 2016, pp. 21–30.

[39] 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.

[40] J. H. Park, S. Nadeem, S. Boorboor, J. Marino, and A. Kaufman, "CMed: Crowd analytics for medical imaging data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 6, pp. 2869–2880, 2021.

[41] J. G. S. Paiva, W. R. Schwartz, H. Pedrini, and R. Minghim, "An approach to supporting incremental visual data classification," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, no. 1, pp. 4–17, 2015.

[42] C. Chen, Z. Wang, J. Wu, X. Wang, L.-Z. Guo, Y.-F. Li, and S. Liu, "Interactive graph construction for graph-based semi-supervised learning," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 9, pp. 3701–3716, 2021.

[43] A. Bäuerle, H. Neumann, and T. Ropinski, "Classifier-guided visual correction of noisy labels for image classification tasks," *Comput. Graph. Forum*, vol. 39, no. 3, pp. 195–205, 2020.

[44] X. Zhang, X. Xuan, A. Dima, T. Sexton, and K.-L. Ma, "Labelvizier: Interactive validation and relabeling for technical text annotations," in *IEEE Pacific Visualization Symposium*, Seoul, 2023, pp. 167–176.

[45] D. Chakrabarti, S. Papadimitriou, D. S. Modha, and C. Faloutsos, "Fully automatic cross-associations," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2004, pp. 79–88.

[46] S. Chatterjee, "Coherent gradients: An approach to understanding generalization in gradient descent-based optimization," in *International Conference on Learning Representations*, 2020.

[47] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[48] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, 2009.

[49] R. M. Gray, *Entropy and information theory*. Springer Science & Business Media, 2011.

[50] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7482–7491.

[51] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[52] I. S. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2001, pp. 269–274.

[53] D. Chakrabarti and C. Faloutsos, *Graph mining: laws, tools, and case studies*. Morgan & Claypool Publishers, 2012.

[54] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[55] M. Ghoniem, J.-D. Fekete, and P. Castagliola, "On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis," *Inf. Vis.*, vol. 4, no. 2, pp. 114–135, 2005.

[56] Z. Li, X. Wang, W. Yang, J. Wu, Z. Zhang, Z. Liu, M. Sun, H. Zhang, and S. Liu, "A unified understanding of deep nlp models for text classification," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 12, pp. 4980–4994, 2022.

[57] I. Robinson and E. Pierce-Hoffman, "Tree-SNE: Hierarchical clustering and visualization using t-sne," *arXiv preprint arXiv:2002.05687*, 2020.

[58] B. Saket, A. Endert, and Ç. Demiralp, "Task-based effectiveness of basic visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 7, pp. 2505–2512, 2019.

[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] K. Sugiyama, S. Tagawa, and M. Toda, "Methods for visual understanding of hierarchical system structures," *IEEE Trans. Syst. Man Cybern.*, vol. 11, no. 2, pp. 109–125, 1981.

[61] E. R. Gansner, E. Koutsofios, S. C. North, and K.-P. Vo, "A technique for drawing directed graphs," *IEEE Trans. Softw. Eng.*, vol. 19, no. 3, pp. 214–230, 1993.

[62] M. Hlawatsch, M. Burch, and D. Weiskopf, "Visual adjacency lists for dynamic graphs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 11, pp. 1590–1603, 2014.

[63] D. S. Kermany, M. Goldbaum, W. Cai, C. C. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan *et al.*, "Identifying medical diagnoses and treatable diseases by image-based deep learning," *Cell*, vol. 172, no. 5, pp. 1122–1131, 2018.

[64] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[65] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, Las Vegas, 2016, pp. 770–778.

[66] K. Gu, X. Masotto, V. Bachani, B. Lakshminarayanan, J. Nikodem, and D. Yin, "An instance-dependent simulation framework for learning with label noise," *Machine Learning*, vol. 112, no. 6, pp. 1871–1896, 2023.

[67] K.-H. Lee, X. He, L. Zhang, and L. Yang, "Cleannet: Transfer learning for scalable image classifier training with label noise," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 5447–5456.

[68] A. Wu, Y. Wang, X. Shu, D. Moritz, W. Cui, H. Zhang, D. Zhang, and H. Qu, "AI4VIS: Survey on artificial intelligence approaches for data visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 12, pp. 5049–5070, 2022.

[69] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer, "Revision: Automated classification, analysis and redesign of chart images," in *Proceedings of the Annual ACM Symposium on User Interface Software and Technology*, 2011, pp. 393–402.

Weikai Yang is a Ph.D. candidate at Tsinghua University. His research interests include visual text analytics and interactive machine learning. He received a B.S. degree from Tsinghua University.

Yukai Guo is a Ph.D. student at Tsinghua University. His research interests include interactive machine learning and model evaluation. 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.

Zheng Wang is currently working toward the graduate degree at Tsinghua University.

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 a 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.