




# Cross-worker joint modeling-based label integration for crowdsourcing

Can Pan, Liangxiao Jiang <sup>\*</sup>, Shanshan Si

School of Computer Science, China University of Geosciences, Wuhan 430074, China

## ARTICLE INFO

Dataset link: <https://github.com/jiangliangxiao/CJMLI>

### Keywords:

Crowdsourcing learning  
Label integration  
Cross-worker joint modeling

## ABSTRACT

In crowdsourcing learning, label integration algorithms are applied to infer each instance's integrated label from its multiple noisy label set. Recent advancements have demonstrated that worker modeling is an effective approach to improving label integration's performance. In real-world crowdsourced scenarios, however, each worker often annotates a few instances only, leading to insufficient worker modeling. To address this issue, we propose a novel cross-worker joint modeling-based label integration (CJMLI) algorithm. Different from existing algorithms that focus on modeling individual workers solely, CJMLI exploits cross-worker joint modeling to effectively mitigate the impact of insufficient worker modeling. Specifically, we first employ majority voting to get initial integrated labels and then apply them to estimate worker qualities. Subsequently, for each instance, we randomly select a subset of workers to estimate its class membership probabilities and then generate a weighted instance for each class. Next, we use the weighted instances to train a classifier. This process is executed several times to get multiple classifiers. Finally, we use weighted majority voting to fuse their predicted labels to infer the final integrated label of each instance. Extensive experiments demonstrate that CJMLI significantly outperforms all its competitors.

## 1. Introduction

Supervised learning has become a prevalent approach in machine learning and finds applications across a broad range of fields [16,18,28]. With this paradigm, access to substantial annotated data plays a key role in enhancing model performance [13,5]. As machine learning technology advances rapidly, especially with the extensive application of deep learning, the demand for annotated data has risen substantially [27,29]. However, obtaining such annotated data presents a significant challenge. Traditional approaches rely on domain experts to undertake the annotation tasks, which are time-consuming and expensive [9,17]. Therefore, some scholars have attempted to learn from limited or insufficient training data, which has alleviated the problem of difficult annotation to a certain extent [10–12].

Fortunately, crowdsourcing provides a resource-efficient way to access annotated data [22,8,33]. As crowdsourcing platforms have gained popularity [2], such as AMT<sup>1</sup> and Clickworker,<sup>2</sup> employers are able to submit unannotated classification tasks and recruit crowd workers to annotate them. However, due to the varying expertise levels of crowd workers, the labels collected from crowd

<sup>\*</sup> Corresponding author.

E-mail addresses: [cpan@cug.edu.cn](mailto:cpan@cug.edu.cn) (C. Pan), [ljiang@cug.edu.cn](mailto:ljiang@cug.edu.cn) (L. Jiang), [siss@cug.edu.cn](mailto:siss@cug.edu.cn) (S. Si).

<sup>1</sup> <http://www.mturk.com>.

<sup>2</sup> <http://www.clickworker.com>.

workers contain a lot of noise [25,3]. A common way to deal with this issue is *repeated labeling* by employing multiple different crowd workers. After that, label integration is applied to infer each instance's integrated label. As a result, label integration has become a topic of considerable research interest.

In recent years, a wide range of label integration works have been developed and studied extensively from different perspectives [21,36]. These algorithms can be divided into two categories depending on whether they explicitly model the workers. The first category is non-worker modeling algorithms, which ignore the differences among workers and typically rely on simple aggregation strategies to infer the true labels. This category includes ground truth inference using clustering (GTIC) [35], multiple noisy label distribution propagation (MNLDP) [14], label augmented and weighted majority voting (LAWMV) [4], attribute augmentation-based label integration (AALI) [38], instance redistribution-based label integration (IRLI) [39], and three-way decision-based label integration (TDLI) [24], etc. The second category is worker modeling-based algorithms, which explicitly estimate worker reliability, preferences, or confusion matrices to infer the true labels. This category includes Dawid-Skene (DS) [6], the generative model of labels, abilities, and difficulties (GLAD) [30], iterative weighted majority voting (IWMV) [19], decision tree-based weighted majority voting (DTWMV) [31], label consistency-based ground truth inference (LCGTI) [20], and dual-view learning from crowds [32], etc.

Based on our observations, recent advancements have demonstrated that worker modeling is often an effective approach to improving label integration's performance. However, existing worker modeling-based algorithms focus on modeling individual workers solely. In real-world crowdsourced scenarios, each worker typically annotates only a small number of instances, leading to insufficient worker modeling. This poses a significant challenge for existing label integration algorithms, as they often rely on sufficient labels to effectively model workers and infer true labels. To address this issue, we propose a novel cross-worker joint modeling-based label integration (CJMLI) algorithm. CJMLI exploits cross-worker joint modeling to effectively mitigate the impact of insufficient worker modeling, which is different from existing algorithms that focus on modeling individual workers solely. Overall, the core contributions of our this paper include:

1. We reveal the limitations of existing algorithms that exploit worker modeling to improve label integration. In fact, each worker often annotates a few instances only, leading to insufficient worker labels and thus limiting worker modeling and label integration.
2. We propose a novel cross-worker joint modeling-based label integration (CJMLI) algorithm. CJMLI exploits cross-worker joint modeling to effectively mitigate the impact of insufficient worker modeling, which is different from the existing algorithms that focus on modeling individual workers solely.
3. We experimentally validate CJMLI's effectiveness on simulated and real-world crowdsourced datasets. The results demonstrate that CJMLI significantly outperforms all the other advanced label integration algorithms.

The structure of the paper is arranged as follows. Section 2 describes existing work related to label integration. Section 3 proposes our CJMLI algorithm. Section 4 provides the detailed experimental results. Section 5 summarizes the paper and discusses potential directions.

## 2. Related work

During the last several years, scholars have actively explored label integration and introduced a variety of label integration algorithms based on different theoretical viewpoints [37]. These algorithms can be broadly divided into two categories according to whether they explicitly model the workers.

The first category is non-worker modeling algorithms, which ignore the differences among workers and typically rely on simple aggregation strategies to infer the true labels. For example, GTIC [35] begins by clustering instances based on features derived from multiple noisy label sets, and subsequently assigns each resulting cluster to a corresponding class. MNLDP [14] transforms each instance's multiple noisy label distribution as a probabilistic label representation and then propagates it across nearest neighbors, allowing each instance to absorb label information from similar instances and refine its integrated label. LAWMV [4] uses KNN to absorb each instance's multiple noisy label set to construct an enhanced multiple noisy label set, and subsequently infers the final integrated labels by weighted majority voting. AALI [38] enriches the original attribute space via attribute augmentation, uses majority voting to initialize integrated labels of reliable instances, and then uses cross-validation to build multiple component classifiers to predict all instances. TDLI [24] first segments the entire crowdsourced dataset into three exclusive subsets using three-way decision theory, and then infers the final integrated labels by applying specific integration strategies to each subset accordingly.

The second category is worker modeling-based algorithms, which explicitly estimate worker reliability, preferences, or confusion matrices to infer the true labels. For example, DS [6] proposed an approach based on expectation maximization, which first applies crowdsourced labels to model a confusion matrix and then represents the probability of each crowd worker annotating each instance for each class. GLAD [30] builds a probabilistic model based on crowd labels to jointly infer instances' integrated labels, their difficulty levels, and the expertise of individual crowd workers. IWMV [19] utilizes the multiple noisy label sets and the initial integrated labels by majority voting to evaluate the annotating quality of each crowd worker, then it refines error rate constraints to identify trustworthy crowd workers until convergence or the algorithm reaches the maximum iteration number. DTWMV [31] treats each crowd worker as a feature and uses decision trees to evaluate the importance of each crowd worker, assigning a weight based on its contribution to classification. LCGTI [20] estimates each crowd worker's annotating quality by combining the crowd worker's label consistency with others and their variance on similar instances, and selects the label of the highest-quality crowd worker as the true label. DVLFC [32] first selects workers with sufficient labels to build worker models, augments the noisy label sets by predicting pseudo labels for unannotated data, infers integrated labels, and then trains a supervised model for each view.

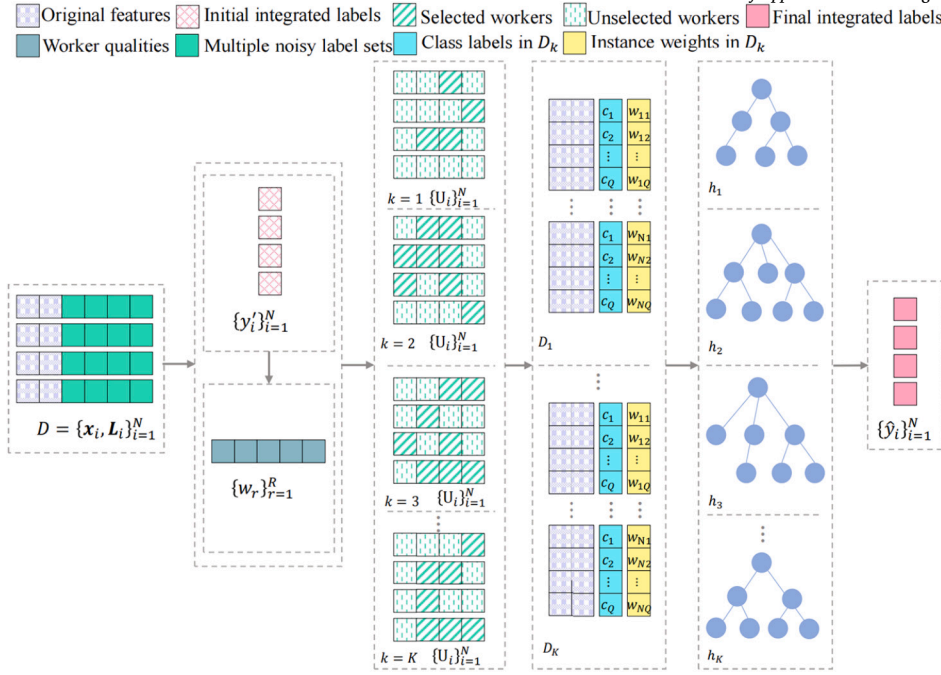


Fig. 1. Overall framework of CJMLI.

### 3. Cross-worker joint modeling-based label integration

#### 3.1. Motivation

From the description above, we can draw a conclusion that worker modeling has been demonstrated to be an effective approach to improving the performance of label integration. However, existing worker modeling-based algorithms focus on modeling individual workers solely. In real-world crowdsourced scenarios, each worker typically annotates only a small number of instances, and few labels are typically collected for each instance to reduce cost, resulting in a highly sparse crowdsourced label matrix. This fact leads to insufficient worker modeling, thereby limiting the practical effectiveness of worker modeling-based algorithms. Specifically, when workers annotate only a small number of instances, the statistical characteristics of their annotation behaviors become extremely unstable due to insufficient labels. This instability directly leads to severe biases in the estimation of workers' reliability, confusion matrices, or professional capabilities. For example, skilled workers may be underestimated in their true abilities due to an occasional single mistake, while unskilled workers may be overestimated due to coincidental performance in a small number of instances. Such biases will further affect the label integration, causing the final integrated labels to deviate from the true labels. Meanwhile, individual worker modeling algorithms have inherent limitations, making it difficult to capture the collective patterns existing in the worker group, such as the common biases of multiple workers towards specific categories or consistent performance on specific instances, which further exacerbates the difficulty of label integration.

This fact raises a key question: how can we effectively perform worker modeling when each worker provides insufficient labels? To address this question, we explore a new strategy called cross-worker joint modeling and then propose the CJMLI algorithm. Different from existing algorithms that focus on modeling individual workers solely, CJMLI exploits cross-worker joint modeling to effectively mitigate the impact of insufficient worker modeling. In the next several subsections, we first describe the problem definition and overall framework of CJMLI, then we describe the detailed processes of CJMLI.

#### 3.2. CJMLI

We first define our problem and some relevant symbols in crowdsourcing. In crowdsourcing scenarios, a dataset can be formulated in the form of a set  $D = \{(x_i, L_i)\}_{i=1}^N$ , where  $N$  is the number of instances. Here,  $x_i$  denotes the  $i$ -th instance in  $D$ , and  $L_i = \{l_{ir}\}_{r=1}^R$  represents its multiple noisy label set, where  $l_{ir}$  is the label of  $x_i$  annotated by the  $r$ -th worker  $u_r$  ( $r = 1, 2, \dots, R$ ), which takes a value from a fixed set  $\{-1, c_1, c_2, \dots, c_Q\}$ .  $Q$  indicates the total count of classes and  $-1$  signifies that  $u_r$  does not annotate  $x_i$ . Label integration aims to infer each instance's integrated label  $\hat{y}_i$  from its multiple noisy label set.

Now, CJMLI's overall framework can be graphically shown in Fig. 1. Given a crowdsourced dataset  $D = \{(x_i, L_i)\}_{i=1}^N$ , for each instance  $x_i$  in  $D$ , we first get its initial integrated label  $y'_i$ , and use  $y'_i$  to estimate the quality  $w_r$  of each worker  $u_r$ . For each iteration, we construct an empty dataset  $D_k$ , and randomly initialize a boolean worker selection vector  $U_i$  for each instance. Based on the selected workers, we estimate the class membership probabilities of each instance and generate a weighted instance for each class.

Then we add instance  $\mathbf{x}_i$  along with class  $c_q$  and weight  $w_{iq}$  to  $D_k$ , train a classifier  $h_k$  on  $D_k$ , and finally aggregate the outputs of all classifiers to get the final integrated labels  $\{\hat{y}_i\}_{i=1}^N$ .

Specifically, for instance  $\mathbf{x}_i$  in  $D$ , we first employ majority voting to get its initial integrated label  $y'_i$ , which is defined as:

$$y'_i = \arg \max_{c_q \in \{c_1, c_2, \dots, c_Q\}} \sum_{r=1}^R I(l_{ir} = c_q), \quad (1)$$

where  $I(\cdot)$  is a binary function that returns 0 if the condition inside the parentheses is false, and 1 otherwise.

Then, we apply the initial integrated label  $y'_i$  to estimate the quality  $w_r$  of worker  $u_r$ . It is calculated based on the agreement among the worker's labels and the initial integrated labels, defined as:

$$w_r = \frac{\sum_{i=1}^N I(l_{ir} = y'_i)}{\sum_{i=1}^N I(l_{ir} \neq -1)}. \quad (2)$$

After getting  $y'_i$  and  $w_r$ , we perform an iterative process for  $K$  rounds to train multiple classifiers. Specifically, for each iteration, we construct an empty dataset  $D_k$ . For instance  $\mathbf{x}_i$ , we randomly initialize a boolean worker selection vector  $\mathbf{U}_i = \{u_{ir}\}_{r=1}^R$ , where  $u_{ir}$  is a boolean indicator that equals 1 if worker  $u_r$  is selected and 0 otherwise. If worker  $u_r$  does not annotate  $\mathbf{x}_i$ , then  $u_{ir}$  is also equal 0. If  $\mathbf{U}_i$  is not equal  $\mathbf{0}$ , we calculate the weight  $w_{iq}$  of instance  $\mathbf{x}_i$  for each class  $c_q$  based on the worker quality  $w_r$ , defined as:

$$w_{iq} = \frac{\sum_{r=1}^R w_r \cdot I(l_{ir} = c_q \wedge u_{ir} = 1)}{\sum_{r=1}^R w_r \cdot I(u_{ir} = 1)}. \quad (3)$$

If  $w_{iq}$  is not equal 0, the instance  $\mathbf{x}_i$  will be added to the dataset  $D_k$  along with class  $c_q$  and weight  $w_{iq}$ . Finally, the dataset  $D_k$  is represented as:

$$D_k = \{(\mathbf{x}_i, c_q, w_{iq}) \mid i = 1, \dots, N; q = 1, \dots, Q; w_{iq} > 0\} \quad (4)$$

Then, we use  $D_k$  to train a classifier  $h_k$ . Through the  $K$  iterations, we can obtain multiple classifiers. Finally, to infer the final integrated labels, we use weighted majority voting to fuse the predicted labels of these classifiers by:

$$\hat{y}_i = \arg \max_{c_q \in \{c_1, c_2, \dots, c_Q\}} \sum_{k=1}^K I(h_k(\mathbf{x}_i) = c_q) \cdot P_{h_k}(c_q \mid \mathbf{x}_i), \quad (5)$$

where  $h_k(\mathbf{x}_i)$  denotes the predicted label for instance  $\mathbf{x}_i$  by  $h_k$ ,  $P_{h_k}(c_q \mid \mathbf{x}_i)$  denotes the probability that classifier  $h_k$  predict instance  $\mathbf{x}_i$  as  $c_q$ .

### 3.3. Core mechanism of CJMLI

#### 3.3.1. Iterative dataset construction

This mechanism addresses the core pain point of insufficient annotations by a single worker. It transforms discrete and sparse individual annotations into continuous group knowledge features via random subset exploration and quality-weighted aggregation. For each instance  $\mathbf{x}_i$ , a boolean worker selection vector  $\mathbf{U}_i = \{u_{ir}\}_{r=1}^R$  is constructed. Randomness is utilized to force the algorithm to traverse the search space of different worker combinations-annotation patterns, preventing the model from falling into local optimality due to the sparse annotations of fixed workers. When  $\mathbf{U}_i \neq \mathbf{0}$ , the weight of instance  $\mathbf{x}_i$  belonging to category  $c_q$  is calculated using Equation (3). Eventually,  $w_{iq}$  reflects the probabilistic consensus of the instance belonging to  $c_q$  from the group perspective, converting discrete annotations into continuous features. Only instance-category-weight triples with  $w_{iq} > 0$  are retained to construct  $D_k = \{(\mathbf{x}_i, c_q, w_{iq}) \mid i = 1, \dots, N; q = 1, \dots, Q; w_{iq} > 0\}$ . This injects weighted annotation knowledge into classifier training, enabling the model to integrate worker quality and group consensus, and adapting to feature learning needs under the sparse annotation scenario.

#### 3.3.2. Ensemble learning with random subsets

This mechanism leverages the diversity of random worker subsets to enhance the robustness of label inference. Multiple classifiers  $h_1, h_2, \dots, h_K$  are trained using datasets  $D_1, D_2, \dots, D_K$  constructed from different random subsets. Due to the randomness in subset selection, each classifier captures distinct annotation patterns from varying combinations of workers, which helps reduce the variance caused by individual sparse annotations. The final integrated label  $\hat{y}_i$  is determined by weighted majority voting of the predictions from these classifiers, as defined in Equation (5). By incorporating  $P_{h_k}(c_q \mid \mathbf{x}_i)$  into the fusion process, this mechanism prioritizes more reliable predictions, enabling the integrated result to converge to the true group consensus. This effectively suppresses prediction errors caused by sparse or noisy annotations, significantly improving the accuracy and stability of label integration in crowdsourcing scenarios.

**Algorithm 1** CJMLI.

---

**Input:**  $D = \{(\mathbf{x}_i, \mathbf{L}_i)\}_{i=1}^N$  - a crowdsourced dataset;  $K$  - the number of classifiers

**Output:**  $\{\hat{y}_i\}_{i=1}^N$  - the integrated labels

```

1: for  $i = 1$  to  $N$  do
2:   Get the initial integrated label  $y'_i$  of  $\mathbf{x}_i$  by Eq. (1)
3: end for
4: for  $r = 1$  to  $R$  do
5:   Estimate the quality  $w_r$  of worker  $u_r$  by Eq. (2)
6: end for
7: for  $k = 1$  to  $K$  do
8:   Construct an empty dataset  $D_k$ 
9:   for  $i = 1$  to  $N$  do
10:    Randomly initialize a boolean worker selection vector  $\mathbf{U}_i = \{u_{ir}\}_{r=1}^R$ 
11:    for  $r = 1$  to  $R$  do
12:      if  $l_{ir} = -1$  then
13:         $u_{ir} = 0$ 
14:      end if
15:    end for
16:    if  $\mathbf{U}_i \neq \mathbf{0}$  then
17:      for  $q = 1$  to  $Q$  do
18:        Estimate the weight  $w_{iq}$  of instance  $\mathbf{x}_i$  with class  $c_q$  by Eq. (3)
19:        if  $w_{iq} \neq 0$  then
20:          Copy instance  $\mathbf{x}_i$  with class  $c_q$  and weight  $w_{iq}$  into  $D_k$ 
21:        end if
22:      end for
23:    end if
24:  end for
25:  Train a classifier  $h_k$  on  $D_k$ , represented by Eq. (4)
26: end for
27: for  $i = 1$  to  $N$  do
28:   Infer the integrated label  $\hat{y}_i$  of  $\mathbf{x}_i$  by Eq. (5)
29: end for
30: Return  $\{\hat{y}_i\}_{i=1}^N$ 

```

---

**Table 1**  
Time complexity comparison of label integration algorithms.

Algorithm	Time complexity
GTIC	$O(N^2 + TNC)$
AALI	$O(QN + JNMt_1)$
TDLI	$O(N^2(\log N + F) + t_1 +  D_3 t_2 +  D_3 Q)$
DS	$O(TNRQ^2)$
IWMV	$O(TNR)$
LCGTI	$O(N^2R(\log N + F))$
CJMLI	$O(K(2NR + NQR + t_1 + Nt_2))$

### 3.4. Time complexity analysis

The complete procedure of CJMLI is presented in Algorithm 1. According to Algorithm 1, lines 1-3 calculate the initial integrated labels for all instances, the complexity is  $O(N(RQ))$ . Lines 4-6 calculate the worker qualities, the time complexity is  $O(RN)$ . Lines 7-26 describe the iterative process of training  $K$  classifiers. Among them, line 10 randomly initializes a boolean worker selection vector  $\mathbf{U}_i$ , the time complexity is  $O(R)$ . Lines 11-15 iterate over each worker  $u_r$  and check whether the worker has annotated the instance  $\mathbf{x}_i$ , its complexity is  $O(R)$ . Lines 16-23 iterate over each class  $c_q$ , calculate the weight  $w_{iq}$  of instance  $\mathbf{x}_i$ , and add the instance along with class and weight to the dataset  $D_k$ , its complexity is  $O(QR)$ . Therefore, the total complexity of lines 9-24 is  $O(N(R + R + QR))$ . Line 25 trains a classifier  $h_k$  on  $D_k$ . If the training complexity of  $h_k$  is  $O(t_1)$ , then the complexity of line 25 is  $O(t_1)$ . So the total complexity of lines 7-26 is  $O(K(N(R + R + QR) + t_1))$ . Line 28 infers the integrated label  $\hat{y}_i$  of  $\mathbf{x}_i$ , if the prediction complexity of  $h_k$  is  $O(t_2)$ , then the complexity of lines 27-29 is  $O(N(Kt_2 + Q))$ . Therefore, the overall complexity of CJMLI is  $O(NRQ + RN + KN(2R + QR) + Kt_1 + N(Kt_2 + Q))$ , and when considering only the dominant term, the total complexity of CJMLI simplifies to  $O(K(2NR + NQR + t_1 + Nt_2))$ .

To further verify the practicality of CJMLI, we summarize the time complexity of each comparison algorithm. As shown in Table 1, the time complexity of each algorithm is presented, where  $T$  is the number of clustering iterations,  $C$  is the number of cluster centers,  $J$  is the number of cross-validation folds,  $M$  is the number of component classifiers,  $F$  is the number of features, and  $|D_3|$  is the size of the negative set.

**Table 2**  
The comparison results on the uniform distribution.

Dataset	GTIC	AALI	TDLI	DS	IWMV	LCGTI	CJMLI
anneal	66.15	64.67	79.18	78.40	65.99	74.83	92.46
audiology	56.19	63.01	77.43	34.96	63.45	64.16	84.12
autos	57.56	60.63	60.00	37.56	57.46	54.63	75.56
balance-scale	58.40	58.34	62.40	71.36	59.74	58.40	80.93
biodeg	59.05	59.27	58.39	66.26	56.83	53.65	68.64
breast-cancer	62.59	59.13	61.54	70.28	55.38	54.90	69.86
breast-w	56.65	61.90	57.65	65.52	56.61	53.22	82.55
car	66.38	60.16	67.25	72.11	62.19	56.42	87.01
credit-a	57.25	63.35	60.58	55.51	60.58	59.28	73.83
credit-g	58.60	59.87	56.80	70.00	57.14	56.90	61.24
heart-c	68.65	61.88	68.32	17.49	59.01	66.01	78.09
heart-h	50.34	63.44	78.57	61.90	66.80	69.73	86.50
heart-statlog	58.89	65.04	57.78	44.44	56.30	52.22	65.48
hepatitis	52.90	53.55	59.35	79.35	60.32	50.97	50.84
horse-colic	52.99	57.20	54.89	36.96	57.53	54.08	60.52
hypothyroid	68.03	55.12	71.53	92.29	62.59	58.59	99.06
ionosphere	52.42	55.56	58.12	64.10	54.81	52.99	70.63
diabetes	61.46	62.20	61.85	65.10	58.01	56.12	65.10
iris	52.67	56.87	56.00	42.00	50.07	46.67	58.87
kr-vs-kp	53.85	54.47	55.73	47.78	53.95	52.47	61.65
labor	66.67	85.44	66.67	68.42	64.74	70.18	83.68
letter	62.38	64.97	79.00	63.72	63.34	71.57	95.29
lymph	60.81	64.19	69.59	60.14	60.54	63.51	79.46
mushroom	57.79	62.28	61.71	51.80	58.58	55.75	90.51
segment	53.33	60.30	71.90	14.29	55.29	63.38	89.35
sick	67.26	67.34	63.65	93.88	60.15	58.67	95.00
sonar	49.52	51.01	51.44	46.63	55.48	55.77	46.44
spambase	54.60	57.27	54.03	60.60	53.32	52.68	60.95
tic-tac-toe	64.82	63.37	64.93	65.34	61.21	59.19	71.97
vehicle	61.58	62.84	67.14	31.56	60.33	56.97	74.49
vote	59.77	64.51	59.31	61.38	57.31	56.32	78.30
vowel	63.13	62.65	77.17	53.74	62.39	62.73	87.69
waveform	59.26	63.61	62.06	33.06	59.61	55.44	79.76
zoo	60.40	60.69	72.28	35.64	62.28	65.35	89.21
<b>Average</b>	<b>59.19</b>	<b>61.36</b>	<b>64.24</b>	<b>56.28</b>	<b>59.10</b>	<b>58.64</b>	<b>76.32</b>

#### 4. Experiments and results

To demonstrate the performance of CJMLI, we conduct comparisons with six representative advanced competitors: GTIC [35], AALI [38], TDLI [24], DS [6], IWMV [19] and LCGTI [20] with respect to the integration accuracy, namely the proportion of the instances whose integrated labels are same as their true labels. Among them, GTIC, AALI and TDLI are non-worker modeling algorithms, whereas DS, IWMV and LCGTI are worker modeling-based algorithms. We implement CJMLI on the CEKA platform [34], and set the classifier to J48 (C4.5) [26] and  $K$  to 70. We take advantage of the existing codes of GTIC and DS on CEKA, as well as the codes of AALI, TDLI, IWMV and LCGTI provided by their authors. The parameters of these algorithms are set in accordance with the descriptions in the corresponding published papers.

##### 4.1. Experiments on 34 simulated datasets

To evaluate CJMLI's performance, we first conduct experiments on all 34 datasets from CEKA. To realistically simulate the crowdsourcing process, we first hide the true label of each instance and then simulate a group of crowd workers to annotate the instance. We simulate 20 workers and set the annotation ratio of each worker to 10%, i.e., each worker will only randomly select 10% of the instance to annotate, which simulates the sparsity in crowdsourcing scenarios. Besides, to reflect the diversity of workers, each crowd worker's annotation quality is randomly sampled from a uniform distribution over the interval  $[0.4, 0.7]$ . The annotating quality reflects the probability that the worker correctly annotates an instance. To reduce the influence of simulation randomness, we repeat each experiment ten times and present the averaged integration accuracy as the final accuracy.

Table 2 presents the detailed results. The bottom row summarizes the average results across 34 datasets. To evaluate whether the observed differences in Table 2 are statistically significant, we conduct pairwise comparisons using the Wilcoxon signed-rank test [7,15], as implemented in the KEEL software [1]. The Wilcoxon signed-rank test is a non-parametric alternative to the paired t-test [23] and compares the two algorithms by ranking their performance discrepancies on a per-dataset basis. Table 3 outlines the variations in rankings between algorithms using the Wilcoxon test on various datasets for uniform distribution. Each row and column corresponds to an algorithm, with the values in the cells showing the total of ranking differences between the two respective algorithms. In particular, the numbers below the diagonal represent the sum of ranks for positive differences (R+), indicating the sum



**Table 3**

Ranks computed on the uniform distribution using the Wilcoxon test.

	GTIC	AALI	TDLI	DS	IWMV	LCGTI	CJMLI
GTIC	-	137.0	76.0	323.0	336.5	355.0	4.0
AALI	458.0	-	192.0	375.0	456.0	436.0	17.0
TDLI	519.0	403.0	-	408.0	556.0	576.0	18.0
DS	272.0	220.0	187.0	-	272.0	278.0	41.0
IWMV	258.5	139.0	39.0	323.0	-	376.0	16.0
LCGTI	240.0	159.0	19.0	317.0	219.0	-	8.0
CJMLI	591.0	578.0	577.0	554.0	579.0	587.0	-

**Table 4**

The comparison results of Wilcoxon tests on the uniform distribution.

	GTIC	AALI	TDLI	DS	IWMV	LCGTI	CJMLI
GTIC	-	◦	◦				◦
AALI	•	-	◦		•	•	◦
TDLI	•		-	•	•	•	◦
DS				-			◦
IWMV		◦	◦		-		◦
LCGTI		◦	◦			-	◦
CJMLI	•	•	•	•	•	•	-

of ranks where the algorithm in the row outperforms the algorithm in the column. The numbers above the diagonal represent the sum of ranks for negative differences (R-), indicating the sum of ranks where the algorithm in the column is worse than the algorithm in the row. Based on the critical values table for the Wilcoxon test, for datasets with a confidence level of  $\alpha = 0.05$  and  $N = 34$ , if the smaller value between R+ and R- is 182 or less, the two algorithms are considered to have significant differences, leading to the rejection of the null hypothesis. A comprehensive overview of the Wilcoxon test outcomes is presented in Table 4, where the symbol • denotes that the row algorithm performs significantly better than the column algorithm, and the symbol ◦ denotes that the column algorithm performs significantly better than the row algorithm. The adopted significance level for the lower diagonal is  $\alpha = 0.05$  while the upper diagonal is  $\alpha = 0.1$ . According to the comparative findings, we note that:

1. The average result of GTIC (59.19%), AALI (61.36%), and TDLI (64.24%) are significantly higher than those of DS (56.28%), IWMV (59.10%) and LCGTI (58.64%). The result demonstrates that insufficient worker labels leading to insufficient worker modeling, thereby limiting the effectiveness of worker modeling-based label integration algorithms.
2. The average accuracy of CJMLI across all datasets is 76.32%, which is significantly higher than those of GTIC (59.19%), AALI (61.36%), TDLI (64.24%), DS (56.28%), IWMV (59.10%) and LCGTI (58.64%). CJMLI achieves the highest average integration accuracy, which indicates that CJMLI is more effective in crowdsourced scenarios with insufficient worker labels compared to the existing algorithms.
3. Based on the results of the Wilcoxon signed-rank test, CJMLI visibly outperforms all these competitors at the significance levels of 0.05 and 0.1, which provides strong evidence supporting the effectiveness of CJMLI in crowdsourced scenarios with insufficient worker labels.

Furthermore, to examine the effectiveness of CJMLI under different quality distributions, we use a Gaussian distribution with  $N(0.55, 0.05^2)$  to randomly generate each worker's annotation quality. Tables 5-7 report the corresponding comparison results, which demonstrate that CJMLI achieves significantly better performance than GTIC, AALI, TDLI, DS, IWMV, and LCGTI.

#### 4.2. Experiments on a real-world dataset

To additionally evaluate CJMLI's effectiveness, we perform experiments using the real-world crowdsourced dataset "Labelme" collected via the AMT platform. It is a classic multi-class crowdsourced dataset containing 8 distinct classes, 2547 labels, and 1000 instances. Each instance includes 64 numeric features. The annotation process involved 59 workers, resulting in each instance being annotated by an average of only 2.5 different workers. Thus, this dataset conforms to the crowdsourcing scenarios with insufficient worker labels we discussed.

To mitigate the impact of randomness on the results, we independently repeat each experiment 10 times on each dataset. Fig. 2 shows the integration accuracy comparisons of seven label integration algorithms on the "Labelme" dataset. The results demonstrate that CJMLI substantially outperforms its competitors, achieving an accuracy of 81.15%, which exceeds those of GTIC (76.70%), AALI (78.92%), TDLI (80.10%), DS (74.70%), IWMV (77.32%), and LCGTI (77.40%). These results further confirm CJMLI's effectiveness.

**Table 5**  
The comparison results on the Gaussian distribution.

Dataset	GTIC	AALI	TDLI	DS	IWMV	LCGTI	CJMLI
anneal	61.25	58.52	72.83	79.96	60.92	70.94	88.26
audiology	63.27	62.39	75.66	29.65	64.20	59.29	83.10
autos	54.63	58.73	69.76	38.05	61.71	64.39	75.17
balance-scale	59.04	62.72	65.60	46.08	60.14	56.96	82.74
biodeg	56.40	57.21	55.83	66.26	52.08	52.42	58.84
breast-cancer	56.99	57.24	54.90	70.28	53.46	53.15	70.87
breast-w	58.94	63.83	57.22	65.52	54.82	56.51	79.03
car	63.89	57.49	65.39	72.74	59.63	56.08	84.10
credit-a	50.87	53.71	53.91	55.51	52.83	52.03	57.88
credit-g	62.50	62.75	60.70	70.00	60.20	57.00	66.00
heart-c	46.20	60.63	66.01	43.56	57.49	66.34	79.67
heart-h	60.88	53.30	63.95	50.34	53.64	59.52	79.97
heart-statlog	58.52	59.48	57.04	55.56	56.52	58.52	66.04
hepatitis	52.90	58.84	57.42	79.35	56.90	54.84	60.13
horse-colic	51.09	54.97	55.98	36.96	54.62	53.80	64.84
hypothyroid	62.91	51.12	66.20	92.36	56.60	52.73	97.19
ionosphere	49.00	54.02	51.57	35.90	49.49	49.29	31.88
diabetes	59.51	59.40	58.59	65.10	55.46	54.69	65.13
iris	60.67	63.27	68.67	44.67	59.00	58.67	90.13
kr-vs-kp	58.76	61.92	60.61	47.78	58.31	57.23	80.65
labor	54.39	54.74	57.89	59.65	57.19	45.61	64.91
letter	61.73	63.49	78.39	61.95	62.38	71.18	94.77
lymph	59.46	66.01	70.95	33.78	61.49	59.46	78.78
mushroom	55.47	63.16	58.86	51.80	54.75	54.99	80.05
segment	61.17	66.58	76.58	14.29	61.01	69.65	94.64
sick	61.88	60.49	56.52	93.88	54.18	53.15	84.88
sonar	55.77	56.83	62.02	53.37	56.54	55.77	67.79
spambase	59.07	65.12	59.86	60.60	57.58	56.03	82.15
tic-tac-toe	58.35	58.09	58.66	65.34	55.96	54.59	59.44
vehicle	62.65	62.42	69.39	25.41	61.00	58.27	75.46
vote	60.00	67.70	61.15	38.62	59.43	58.16	83.15
vowel	65.25	64.28	79.19	47.68	63.44	65.35	86.62
waveform	58.44	61.88	62.04	33.06	58.51	56.02	79.62
zoo	60.40	64.46	79.21	30.69	58.81	66.34	95.05
<b>Average</b>	<b>58.30</b>	<b>60.20</b>	<b>63.78</b>	<b>53.40</b>	<b>57.66</b>	<b>57.91</b>	<b>76.15</b>

**Table 6**  
Ranks computed on the Gaussian distribution using the Wilcoxon test.

	GTIC	AALI	TDLI	DS	IWMV	LCGTI	CJMLI
GTIC	-	144.0	61.0	377.0	380.5	373.0	13.0
AALI	451.0	-	162.0	399.0	505.0	440.0	24.0
TDLI	534.0	433.0	-	432.0	595.0	591.0	26.0
DS	218.0	196.0	163.0	-	222.0	224.0	53.0
IWMV	214.5	90.0	0.0	373.0	-	336.0	15.0
LCGTI	222.0	155.0	4.0	371.0	259.0	-	14.0
CJMLI	582.0	571.0	569.0	542.0	580.0	581.0	-

**Table 7**  
The comparison results of Wilcoxon tests on the Gaussian distribution.

	GTIC	AALI	TDLI	DS	IWMV	LCGTI	CJMLI
GTIC	-	o	o				o
AALI	•	-	o	•	•	•	o
TDLI	•	•	-	•	•	•	o
DS			o	-			o
IWMV		o	o		-		o
LCGTI		o	o			-	o
CJMLI	•	•	•	•	•	•	-

#### 4.3. Comparisons in running time

We further investigate the time consumption of CJMLI, comparing with others. The running time of CJMLI is mostly spent on iteratively training multiple classifiers and fusing their predicted results. Table 8 lists their running time in milliseconds on “Labelme” dataset.



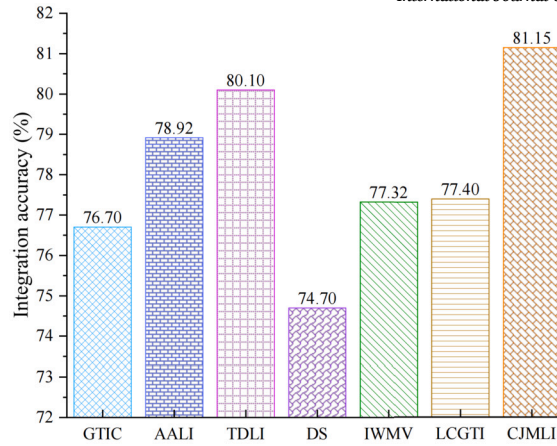


Fig. 2. The comparison results on the "Labelme" dataset.

Table 8

The comparison results in running time (in millisecond).

Dataset	GTIC	AALI	TDLI	DS	IWMV	LCGTI	CJMLI
Labelme	51.10	39936.60	2941.20	3481.10	1076.90	1556.20	29371.50

From the running time shown in Table 8, among the non-worker modeling algorithms, GTIC is the most efficient because it simplifies clustering operations and avoids the overhead of worker modeling. AALI takes a long time, which is due to its attribute augmentation and cross-validation processes. Overall, worker modeling-based algorithms take relatively more time as they require modeling and calculation of worker quality and other aspects, reflecting the time cost incurred by worker modeling in improving quality. Among these, CJMLI takes a relatively high amount of time due to operations such as integrated classifier training, but it is still lower than AALI.

#### 4.4. Discussion and analysis

##### 4.4.1. Effect and influence of the initial integrated labels

In CJMLI, we employ majority voting to get each instance's initial integrated label  $y'_i$  and further apply  $y'_i$  to estimate the quality  $w_r$  of worker  $u_r$ . However, it is susceptible to noisy labels or group biases among workers. To alleviate this problem, CJMLI through the  $K$  iterations to obtain multiple classifiers, and use weighted majority voting to fuse the predicted labels to weaken the impact of the initial integrated label. Extensive experiments demonstrate that the average accuracy of CJMLI is significantly better than all comparison algorithms.

##### 4.4.2. Parameter sensitivity analysis of CJMLI

In CJMLI, the only adjustable parameter is the number of classifiers  $K$ . To investigate how different values of  $K$  affect the performance of CJMLI, we design a group of experiments to analyse its variation in the integration accuracy. Fig. 3 shows CJMLI's performance with different  $K$  values (from 10 to 120). From Fig. 3, we can see that CJMLI is not sensitive to the choice of  $K$  as long as  $K$  is not too small. Notably, when  $K$  gets to 70, CJMLI reaches optimal. Therefore,  $K$  is set to 70 in our experiments.

## 5. Conclusions and future work

This paper reveals the limitations of existing algorithms that exploit worker modeling for label integration, where the insufficient annotated instances per worker lead insufficient worker modeling and thus limiting label integration's performance. To address this issue, this paper proposes a cross-worker joint modeling-based label integration (CJMLI) algorithm. Specifically, we first estimate worker qualities using initial integrated labels. Then, we randomly select subsets of workers to generate multiple weighted datasets, and train a classifier on each dataset. Finally, we employ weighted majority voting to fuse the predictions of these classifiers and infer the final integrated label of each instance. Different from existing algorithms that focus on modeling individual workers solely, CJMLI exploits cross-worker joint modeling to effectively mitigate the impact of insufficient worker modeling. The experiments validate the effectiveness of CJMLI.

Currently, we estimate workers' qualities based on their agreement with the initial integrated labels. However, this approach relies on the initial integrated labels, which may be inaccurate and thus affect the reliability of the estimated worker qualities. In future work, we will consider more reliable approaches for worker quality estimation, such as incorporating entropy-based uncertainty estimation from logit prototype learning to quantify annotation stability, and leveraging global-local feature alignment ideas from progressive learning vision transformer to assess consistency with sample features.

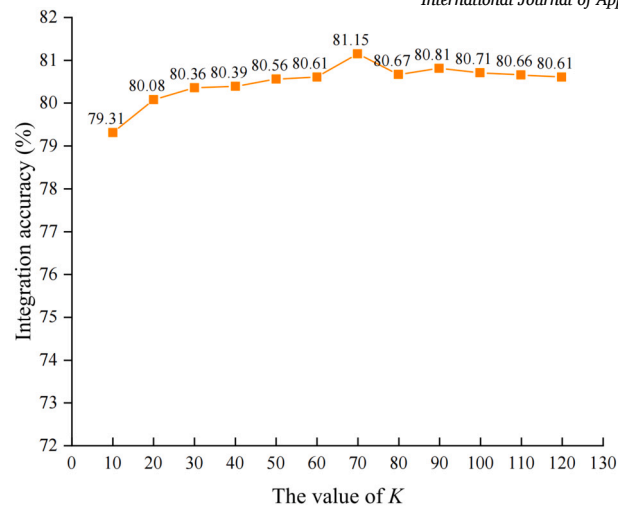


Fig. 3. The comparison results with different  $K$  values.

### CRedit authorship contribution statement

**Can Pan:** Writing – original draft, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization.  
**Liangxiao Jiang:** Writing – review & editing, Supervision, Methodology, Funding acquisition, Formal analysis, Conceptualization.  
**Shanshan Si:** Writing – review & editing, Methodology, Formal analysis.

### Declaration of competing interest

I confirm that there is no conflict-of-interest in the submission, and the manuscript has been approved by all authors for publication.

### Acknowledgements

The work was partially supported by National Natural Science Foundation of China (No. 62276241) and Scientific and Technological Development Scheme of Jilin Province (No. 20240101371JC).

### Data availability

The source code of CJMLI and all the datasets used in this research can be found on the website: <https://github.com/jiangliangxiao/CJMLI>.

### References

- [1] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework, *J. Mult.-Valued Log. Soft Comput.* 17 (2011) 255–287.
- [2] M. Buhrmester, T. Kwang, S.D. Gosling, Amazon's mechanical turk: a new source of inexpensive, yet high-quality data? *Perspect. Psychol. Sci.* 6 (2011) 3–5.
- [3] P. Chen, H. Sun, Y. Fang, X. Liu, CONAN: a framework for detecting and handling collusion in crowdsourcing, *Inf. Sci.* 515 (2020) 44–63.
- [4] Z. Chen, L. Jiang, C. Li, Label augmented and weighted majority voting for crowdsourcing, *Inf. Sci.* 606 (2022) 397–409.
- [5] Z. Chen, L. Jiang, W. Zhang, C. Li, Weighted adversarial learning from crowds, *IEEE Trans. Serv. Comput.* 17 (2024) 4467–4480.
- [6] A.P. Dawid, A.M. Skene, Maximum likelihood estimation of observer error-rates using the em algorithm, *J. R. Stat. Soc., Ser. C, Appl. Stat.* 28 (1979) 20–28.
- [7] J. Demsar, Statistical comparisons of classifiers over multiple data sets, *J. Mach. Learn. Res.* 7 (2006) 1–30.
- [8] K.G. Dizaji, H. Gao, Y. Yang, H. Huang, C. Deng, Robust cumulative crowdsourcing framework using new incentive payment function and joint aggregation model, *IEEE Trans. Neural Netw. Learn. Syst.* 31 (2020) 4610–4621.
- [9] Y. Dong, L. Jiang, C. Li, Improving data and model quality in crowdsourcing using co-training-based noise correction, *Inf. Sci.* 583 (2022) 174–188.
- [10] Y. Fu, Z. Liu, J. Lyu, Reason and discovery: a new paradigm for open set recognition, *IEEE Trans. Pattern Anal. Mach. Intell.* 47 (2025) 5586–5599.
- [11] Y. Fu, Z. Liu, Z. Wang, Logit prototype learning with active multimodal representation for robust open-set recognition, *Sci. China Inf. Sci.* 67 (2024).
- [12] Y. Fu, Z. Liu, Z. Zhang, Progressive learning vision transformer for open set recognition of fine-grained objects in remote sensing images, *IEEE Trans. Geosci. Remote Sens.* 61 (2023) 1–13.
- [13] Y. Hu, L. Jiang, C. Li, Instance difficulty-based noise correction for crowdsourcing, *Expert Syst. Appl.* 212 (2023) 118794.
- [14] L. Jiang, H. Zhang, F. Tao, C. Li, Learning from crowds with multiple noisy label distribution propagation, *IEEE Trans. Neural Netw. Learn. Syst.* 33 (2022) 6558–6568.
- [15] L. Jiang, L. Zhang, C. Li, J. Wu, A correlation-based feature weighting filter for naive Bayes, *IEEE Trans. Knowl. Data Eng.* 31 (2019) 201–213.
- [16] L. Jiang, L. Zhang, L. Yu, D. Wang, Class-specific attribute weighted naive Bayes, *Pattern Recognit.* 88 (2019) 321–330.
- [17] C. Li, L. Jiang, W. Xu, Noise correction to improve data and model quality for crowdsourcing, *Eng. Appl. Artif. Intell.* 82 (2019) 184–191.
- [18] H. Li, L. Jiang, S. Xue, Neighborhood weighted voting-based noise correction for crowdsourcing, *ACM Trans. Knowl. Discov. Data* 17 (2023) 96.

- [19] H. Li, B. Yu, Error rate bounds and iterative weighted majority voting for crowdsourcing, CoRR, arXiv:1411.4086, 2014.
- [20] J. Li, L. Jiang, W. Zhang, Label consistency-based ground truth inference for crowdsourcing, IEEE Trans. Neural Netw. Learn. Syst. 36 (2025) 9408–9421.
- [21] J. Li, H. Sun, J. Li, Beyond confusion matrix: learning from multiple annotators with awareness of instance features, Mach. Learn. 112 (2023) 1053–1075.
- [22] S. Li, Y. Jiang, N.V. Chawla, Z. Zhou, Multi-label learning from crowds, IEEE Trans. Knowl. Data Eng. 31 (2019) 1369–1382.
- [23] C. Nadeau, Y. Bengio, Inference for the generalization error, Mach. Learn. 52 (2003) 239–281.
- [24] C. Pan, L. Jiang, C. Li, Three-way decision-based label integration for crowdsourcing, Pattern Recognit. 158 (2025) 111034.
- [25] F. Rodrigues, F.C. Pereira, Deep learning from crowds, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2018, pp. 1611–1618.
- [26] S. Salzberg, Book review: C4.5: programs for machine learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993, Mach. Learn. 16 (1994) 235–240.
- [27] L. Sun, H. Liang, W. Ding, J. Xu, Granular ball fuzzy neighborhood rough sets-based feature selection via multiobjective mayfly optimization, IEEE Trans. Fuzzy Syst. 32 (2024) 6112–6124.
- [28] L. Sun, F. Xu, W. Ding, J. Xu, AFIFC: adaptive fuzzy neighborhood mutual information-based feature selection via label correlation, Pattern Recognit. 164 (2025) 111577.
- [29] L. Sun, Q. Zhang, W. Ding, T. Wang, J. Xu, FCPFS: fuzzy granular ball clustering-based partial multilabel feature selection with fuzzy mutual information, IEEE Trans. Emerg. Top. Comput. Intell. 9 (2025) 590–606.
- [30] J. Whitehill, T.f. Wu, J. Bergsma, J. Movellan, P. Ruvolo, Whose vote should count more: optimal integration of labels from labelers of unknown expertise, Adv. Neural Inf. Process. Syst. 22 (2009) 2035–2043.
- [31] W. Yang, C. Li, L. Jiang, Learning from crowds with decision trees, Knowl. Inf. Syst. 64 (2022) 2123–2140.
- [32] H. Zhang, L. Jiang, W. Zhang, G.I. Webb, Dual-view learning from crowds, ACM Trans. Knowl. Discov. Data 19 (2025) 61:1–61:21.
- [33] J. Zhang, Knowledge learning with crowdsourcing: a brief review and systematic perspective, IEEE/CAA J. Autom. Sin. 9 (2022) 749–762.
- [34] J. Zhang, V.S. Sheng, B. Nicholson, X. Wu, CEKA: a tool for mining the wisdom of crowds, J. Mach. Learn. Res. 16 (2015) 2853–2858.
- [35] J. Zhang, V.S. Sheng, J. Wu, X. Wu, Multi-class ground truth inference in crowdsourcing with clustering, IEEE Trans. Knowl. Data Eng. 28 (2016) 1080–1085.
- [36] W. Zhang, L. Jiang, Z. Chen, C. Li, FNNWV: farthest-nearest neighbor-based weighted voting for class-imbalanced crowdsourcing, Sci. China Inf. Sci. 67 (2024).
- [37] W. Zhang, L. Jiang, C. Li, IWBVT: instance weighting-based bias-variance trade-off for crowdsourcing, in: A. Globersons, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J.M. Tomczak, C. Zhang (Eds.), Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024, 2024.
- [38] Y. Zhang, L. Jiang, C. Li, Attribute augmentation-based label integration for crowdsourcing, Front. Comput. Sci. 17 (2023) 175331.
- [39] Y. Zhang, L. Jiang, C. Li, Instance redistribution-based label integration for crowdsourcing, Inf. Sci. 674 (2024) 120702.