ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230431

Label Aggregation with Self-Supervision Enhanced Graph Transformer

Jiacheng Liu[a,b], Feilong Tang[c;*] and Xiaofeng Hou[b]

[a]Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China
[b]Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
[c]School of Data Science and Engineering, East China Normal University, Shanghai, China

Abstract. Aggregating noisy labels produced by the crowd of workers to generate true labels is a challenging problem in crowdsourcing. The key behind label aggregation is to effectively utilize the hidden information (e.g., characteristics of workers and questions which are often missing) in the labeling process. Existing methods mainly generated aggregation models based on the complicated Bayesian model or some strong assumptions. Recently, deep learning-based methods attempt to automate label aggregation but need various labels. These all make them hard to deploy to real-world applications. In fact, abundant information in the process of crowdsourcing itself can be extremely helpful to aggregate the labels. In this paper, we propose ATHENA *(lAbel aggregaTion witH sElf-supervision eNhanced grAph transformer)* to aggregate labels by utilizing the self-supervision signals in crowdsourcing. Firstly, we propose a transformer-based graph neural network that can learn from the crowdsourcing topology and features. Then, we use self-supervision signals inherently included in the dataset to help to aggregate the labels. To be specific, we identify the answer-based self-supervision signal that can predict the answer of any user given to different tasks. In our evaluations, we compare the proposed ATHENA with the other 11 representative methods on 10 datasets. Our experimental results demonstrate that ATHENA is highly effective in aggregating labels and obtains much better performance than existing methods.

1 Introduction

Crowdsourcing is the practice of utilizing the wisdom of millions of human workers to produce more efficient and accurate machine learning methods. For example, the ImageNet dataset built with crowdsourcing [29] has significantly improved the accuracy of computer vision. Despite the advantages of crowdsourcing, the crowdsourcing answers are always noisy due to several reasons such as the difficulty of the task and the diverse backgrounds of the workers. Therefore, aggregating these noisy labels to generate the correct labels has become an important research topic that attracts many researchers [40, 41, 1, 27].

In the process of label aggregation, it is critical to accurately estimate the ability of various workers and the difficulty of different tasks. However, it is impossible to directly obtain this kind of ability or difficulty. Instead, we have to estimate them based on other hidden information in the labeling process. Taking the flower image labeling task as an example, the workers who have a hobby related to flowers are likely to have better abilities. To estimate the ability or difficulty, a large number of approaches based on different strategies have been proposed to exploit these hidden signals in the crowdsourcing process. Among these, most approaches simply model the process of label aggregation with strong assumptions which makes them sub-optimal. For example, some approaches use a probabilistic graphical model to emulate the generation process of the answers [38, 45]. These assumption-based approaches can reveal some hidden signals but may fail when the assumptions are inaccurate or incorrect. Recently motivated by the success of deep learning, there are a few approaches that try to use deep neural networks to automatically learn the hidden signals [40, 23]. Although these methods are theoretically applicable, in practice, it is often difficult to achieve good results due to the high complexity of the models. Furthermore, these approaches either require a substantial amount of data [23] or heavily depend on task-specific features [8, 40, 1]. All these approaches can not effectively reveal the hidden signals in crowdsourcing.

To devise an effective label aggregation method, we need to tackle two-fold challenges. The first one is *how to get rid of the strong assumptions*. Given a complicated label aggregation problem, an over-simplified aggregation model may lead to inaccuracy and make it hard to get the true answer. Recently, some methods try to alleviate the dependence of label aggregation on strong assumptions by generating aggregation models in an unsupervised manner. However, in practical scenarios, few labeled data are always available before distributing the crowdsourcing task to the crowds of workers (e.g., is a common practice to add the golden questions in the crowdsourcing process [4, 33]). Hence, a completely unsupervised approach might not be optimal. The second one is *how to aggregate the labels without enough amounts of labels or features*. Since label aggregation is mainly used for building a large labeled dataset, it may not initially have large amounts of labels. This will be especially challenging for some deep learning methods that highly depend on the labeled data. It is attractive to aggregate labels based on more available and intuitive features in crowdsourcing tasks like their typologies.

In this paper, we propose *lAbel aggregaTion witH sElf-supervision eNhanced grAph transformer* termed as ATHENA (as shown in Figure 1). Firstly, ATHENA addresses the above challenges by introducing a novel representation approach, which learns to present the attributes of workers and tasks by a graph transformer with minimal assumptions. The graph transformer learns the representations with an effective self-attention mechanism which helps to fuse different tasks' and workers' attributes. In this way, using the graph transformer can effectively solve the challenge of strong assumptions. Secondly, we demonstrate

[*] Corresponding Author. Email: fltang@dase.ecnu.edu.cn

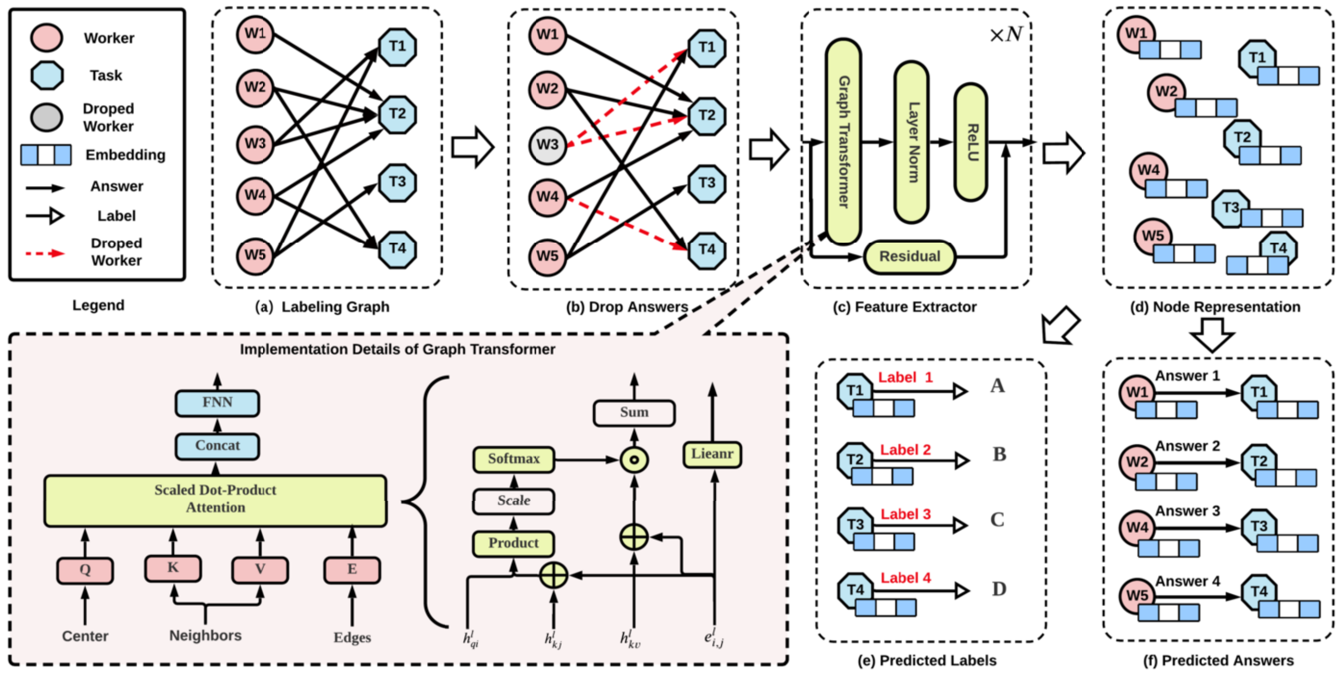


Figure 1: Overview of ATHENA. (a) Constructing a labeling graph to present the crowdsourcing task. (b) Dropping some answers to prevent overfitting. (c) Implementing a transformer-based graph neural network to infer the features of workers and questions. (d) Using predicted answers to train ATHENA for accurate label aggregation.

that conventional GNN-based aggregation models established in a semi-supervised way cannot excel at realistic label aggregation with little initial labeled data (e.g., less than 10). In ATHENA, we alleviate the dependence of label aggregation on large amounts of labeled data by constructing effective self-supervision signals. They mainly exploit the inner structure of data to make the graph transformer can work with minimal initiate labels. Overall, our proposal lies in the fact that the process of crowdsourcing itself contains much information that helps model the tasks and the answers. Once we set the learning objectives properly, we can get supervision from the crowdsourcing task itself. The contributions of the paper are as follows,

- To the best of our knowledge, we are the first to apply self-supervision to aggregate labels in crowdsourcing tasks with graph transformers.
- We develop a new transformer-based graph neural network model that can fuse tasks, workers, and answers in the label aggregation process.
- We design a novel algorithm that uses several effective self-supervised tasks to abstract the important signals but hidden in noisy labels.
- To validate our proposed method, we compare it with other 11 existing label aggregation methods based on 10 real-world datasets. The results show that ATHENA performs best on different crowdsourcing problems.

2 Graph Transformer-based Label Aggregation Model

In this section, we begin by presenting the problem formulation (section 2.1), followed by an introduction to the feature representation (section 2.2) and the graph transformer model (section 2.3).

2.1 Problem Formulation

Without loss of generality, we consider optimizing label aggregation in classification tasks. This process contains two sets of entities, namely, the task set $\mathbf{T} = \{t_1, t_2, \ldots, t_n\}$ including n tasks and the worker set $\mathbf{W} = \{w_1, w_2, \ldots, w_m\}$ of m workers. We assume that the classification task has p options, compromising an option set called $\mathbf{O} = \{o_1, o_2, \ldots, o_p\}$. In this paper, we consider a realistic classification scenario where a worker only answers a partial set of tasks. We denote the set of workers answering the task t_j as $\mathbf{W}^{t_j}$. We also depict the set of tasks answered by the worker w_i as $\mathbf{T}^{w_i}$. We leverage $\mathbf{A} = \{a_{t_j}^{w_i}\}$ to represent all the collected answers in the crowdsourcing task. The $a_{t_j}^{w_i}$ means the answer of task t_j provided by worker w_i, which belongs to the option set $\mathbf{O}$.

To incorporate all the information into the process of label aggregation, we present the classification task as a *labeling graph* as shown in Figure 1(a). We denote this graph as $\mathcal{G} = \{\mathbf{V}, \mathbf{\Psi}, \mathbf{X}, \mathbf{E}\}$, where $\mathbf{V}$ is the node set that contains both of the worker nodes and task nodes, $\mathbf{\Psi}$ is an edge set, $\mathbf{X}$ is a node attribute set and $\mathbf{E}$ is an edge attribute set. By representing the information as a *labeling graph*, not only are the collected answers depicted, but all interactions between the workers and questions are effectively captured.

Based on the labeling graph, we define label aggregation as shown in Definition 1. In the initial process of label aggregation, there are always a few labeled data available [4]. We term these initially labeled data as golden answers. Thus, the label aggregation with a few golden answers can be defined as,

Definition 1 (Label aggregation with a few golden answers)**.** *In a label aggregation problem, given its labeling graph $\mathcal{G}$ and a small set of labeled tasks $\mathbf{T}^*$ with their true answers, we aim to aggregate the true label of each task in $\mathbf{T}$.*

2.2 Feature Representations

While the structure of $\mathcal{G}$ is intuitive, extracting the hidden information from the graph to estimate the abilities of workers and the difficulties of tasks can be a challenging task. There has been limited work on learning the hidden information provided by $\mathcal{G}$ [41]. However, we address an even more challenging scenario where the feature representation of workers or tasks is unknown. Inspired by previous works [8], we use the noisy ability and the noisy requirement as the features for workers and answers respectively. Specifically, we first use the majority voting algorithm to get a reliable answer for each task as $\tilde{a}_{t_i}$. Then, for each worker w_i, its feature F^{w_i} is a n dimensional vector, which is defined as,

$$F^{w_i} = [\mathbf{1}(a^{w_i}_{t_1} = \tilde{a}_{t_1}), \cdots, \mathbf{1}(a^{w_i}_{t_n} = \tilde{a}_{t_n}))] \tag{1}$$

where $\mathbf{1}(a^{w_i}_{t_j} = \tilde{a}_{t_j})$ is a indicator function. If the answer of worker w_i to task t_j is the same as the majority voting result, namely $a^{w_i}_{t_j} = \tilde{a}_{t_j}$, the value of $\mathbf{1}()$ is 1 otherwise 0. If the worker does not answer the task, we will fill it with -1. We encode the task's feature in a similar way. Differently, each dimension of the task feature vector represents if a worker gives the correct answer to the task. Finally, the edge feature is set to the one-hot encoding of the answers.

2.3 Graph Transformer

Based on the labeling graph $\mathcal{G}$, ATHENA uses a graph neural network-based approach to learn the representation of workers and tasks as shown in Figure 1(c). We choose the transformer-based GNN to aggregate the label for the following reasons:

- Firstly, the graph provides an informative and intuitive representation for the problem of label aggregation. A labeling graph can present comprehensive information.
- Secondly, one of the most difficult challenges in label aggregation is that in most cases a worker only answers a small fraction of questions. This poses a difficulty for traditional deep learning approaches, which may not perform well on extremely sparse data.
- Thirdly, it is well recognized that the transformer-based GNN is the most effective approach to model this graph learning problem [31]. Furthermore, this approach is highly interpretable, allowing for a better understanding of the resulting models and their predictions.

Inspired by the idea of Transformer [35, 31], we propose a novel transformer-based graph neural network to learn the representations of the constructed labeling graph. For simplicity, we only illustrate the learning process of the tasks' representation in this part. We omit the learning process of the workers' feature, which is carried out in a similar manner.

In our transformer-based graph neural network, we present worker's features in layer l with $H^l_w = \{h^l_{w_1}, h^l_{w_2}, \cdots, h^l_{w_m}\}$ and tasks' features in layer l with $H^l_t = \{h^l_{t_1}, h^l_{t_2}, \cdots, h^l_{t_n}\}$. We first encode the features of the central node and its neighbors with different weight parameters (W^l_k, W^l_v, W^l_q) and bias parameters (b^l_k, b^l_v, b^l_q) and map them to the same dimension d. The query vector, value vectors, and key vector of the transformer are respectively calculated with the following equations.

$$\begin{aligned}\mathbf{q}^l_{t_i} &= \mathbf{W}^l_q \mathbf{h}^l_{t_i} + \mathbf{b}^l_q \\ \mathbf{v}^l_{w_j} &= \mathbf{W}^l_v \mathbf{h}^l_{w_j} + \mathbf{b}^l_v \\ \mathbf{k}^l_{w_j} &= \mathbf{W}^l_k \mathbf{h}^l_{w_j} + \mathbf{b}^l_k\end{aligned} \tag{2}$$

The edge features in the labeling graph contain much meaningful information about the answers. To incorporate these, we also encode them to the same dimension as the tasks'.

$$\mathbf{e}^l_{ij} = \mathbf{W}^l_e e^l_{ij} + \mathbf{b}^l_e \,, \tag{3}$$

where $\mathbf{W}^l_e$ and $\mathbf{b}^l_e$ are the weight and bias parameters at layer l. Then, we calculate the self-attention weight from worker node w_j to task node t_i with the exponential scale dot-product function. The self-attention weight $\mathbf{s}^l_{ij}$ is formulated as,

$$\mathbf{s}^l_{ij} = \exp\left(\frac{\mathbf{q}^{l\top}_{t_i}(\mathbf{k}^l_{w_j} + \mathbf{e}^l_{ij})}{\sqrt{d}}\right) \tag{4}$$

where d is the dimension of the hidden layer. Based on these, the final attention is calculated as

$$\alpha^l_{i,j} = \frac{s^l_{ij}}{\sum_{j \in \mathcal{N}(t_i)} s^l_{ik}} \tag{5}$$

where $\mathcal{N}(t_i)$ is the set of worker nodes that answered task node t_i. After getting the attention, we can aggregate the feature of task node q_i based on the attention value. In addition, we also include the edge feature e_{ij} by adding it to the value vector. Thus, we compute the feature of task i with,

$$\hat{\mathbf{h}}^{(l)}_i = \sum_{j \in \mathcal{N}(i)} \alpha^{(l)}_{ij} \left(\mathbf{v}^{(l)}_j + \mathbf{e}_{ij}\right) \tag{6}$$

To help stabilize the result, we adopt the multi-head attention mechanism that trains C attention layers simultaneously and eventually concatenates them as the output. In this way, the feature of the task i is ultimately computed with

$$\hat{\mathbf{h}}^{(l)}_i = \|_{c=1}^{C} \left[\sum_{j \in \mathcal{N}(i)} \alpha^{(l)}_{c,ij} \left(\mathbf{v}^{(l)}_{c,j} + \mathbf{e}_{c,ij}\right)\right] \tag{7}$$

where $\|$ is the concatenation operation for different head attentions. Then, the output is passed to a fully connected network.

To prevent the over-smoothing problem, we also add a residual connection as,

$$\mathbf{h}^{l+1}_i = \mathrm{Norm}(\hat{\mathbf{h}}^l_i + \mathbf{W}^l_r \mathbf{h}^l_i + \mathbf{b}^l_r) \tag{8}$$

where $\mathbf{W}^l_r$ is the weight matrix in layer l, $\mathbf{b}^l_r$ is the bias term, and Norm is a normalization layer [3]. It performs the following computations.

$$\mathrm{Norm}(\mathbf{x}) = \frac{\mathbf{x} - \mathrm{E}[\mathbf{x}]}{\sqrt{\mathrm{Var}[\mathbf{x}] + \epsilon}} \odot \gamma + \beta \tag{9}$$

where the mean E[**x**] and standard-deviation Var[**x**] are calculated across all nodes and all node channels.

Based on the output of the graph transformer, we finally leverage fully connected layers to map these representations to predict the correct labels of the crowdsourcing tasks.

$$\hat{a_{t_j}} = \sigma\left(\mathbf{M}^t_L \cdots \mathrm{ReLU}\left(\mathbf{M}^t_1 h_{t_j} + \mathbf{b}^t_1\right) + \mathbf{b}^t_L\right) \,, \tag{10}$$

where σ is the softmax function, M^t_* is the weight matrix, and b^t_* is the bias term.

Finally, we leverage the cross-entropy loss function to measure the label computation loss between the predicted tasks' labels and the true labels as follows,

$$\mathcal{L}_q = -\sum_{t\in\mathbf{T}^*}\sum_{m=1}^{|O|} a_{t_j,m}\log \hat{a}_{t_j,m} \tag{11}$$

where $a_{t_j,m}$ is a binary indicator (i.e., 0 or 1) of the class label m for the probability of task node t_j.

3 Training with Self-Supervision Signals

The transformer-based graph neural network proposed in this work can demonstrate strong performance when trained on a large amount of labeled data. However, in the context of label aggregation, obtaining a sufficient number of labeled examples is often infeasible. To address this challenge, we propose a novel approach that leverages self-supervised learning techniques, informed by recent progress in the field [46, 32]. By incorporating a well-designed self-supervision signal, our method allows for effective training of the transformer-based graph neural network using a limited set of labeled examples, thus mitigating the need for a large amount of labeled data.

3.1 Predicting Worker's Answer

Instead of collecting the initial labels of tasks before the label aggregation process, we compute a predicted answer based on the workers' answers to specific questions, which is easier to obtain. Specifically, the predicted answer of a worker to a task is inferred with the representations of the worker and the task as shown in Figure 1 (f). Ideally, these representations reflect the characteristics of the worker and the task. For the task, the feature reflects its requirements on domain knowledge, and for the worker, the feature represents his ability to handle this task. Thus, to alleviate the dependence on initial labels, we design a method that can predict the worker's answer to a given task based on the learned representations.

Specifically, given worker w_i's representation h_{w_i} and task t_j's representation h_{t_j}, we train a model to predict the answer $a_{t_j}^{w_i}$. This forms a new classification problem that derives large amounts of labels directly from the labeling process. To build this answer prediction model, we first concatenate the representations of the worker and task. Then, we use a fully connected network to predict the answer. The predicted answer is computed as follows,

$$\hat{a}_{t_j}^{w_i} = \sigma\left(\mathbf{M}_L^a \cdots \mathrm{ReLU}\left(\mathbf{M}_1^a\langle h_{w_i}, h_{t_j}\rangle + \mathbf{b}_1^a\right) + \mathbf{b}_L^a\right) , \tag{12}$$

where $\sigma()$ is the softmax function, $\langle h_{w_i}, h_{t_j}\rangle$ means the concatenation of h_{w_i} as well as h_{t_j}, $\mathbf{M}_*^a$ is the weight matrices and b_*^a is the bias term.

Afterward, we can use the predicted answers to train our transformer-based graph neural network. In the training process, we use the cross entropy loss function to iteratively update the parameters in the neural network. The answer prediction loss is formulated as.

$$\mathcal{L}_a = -\sum_{a\in\mathbf{A}}\sum_{m=1}^{|O|} a_{t_j,m}^{w_i}\log \hat{a}_{t_j,m}^{w_i} \tag{13}$$

where $a_{t_j,m}^{w_i}$ is a binary indicator of class label m for the probability that the worker w_i answers the question t_j.

3.2 Adaptively Drop Answers

Given that the structure of the graph in label aggregation tasks is primarily determined by how the crowd workers are assigned to tasks, it may not necessarily reflect the workers' actual abilities. To address this issue, we propose a method for dropping answers which can help alleviate the influence of the assignment process.

We observe that if multiple workers do not provide a response for a particular task, it is unlikely to significantly impact the true label for that task. Leveraging this insight, we introduce an adaptive dropping mechanism to remove unnecessary answers and prevent overfitting. Specifically, we utilize answer dropping techniques for two reasons:

- In the label aggregation problem, the graph structure is highly influenced by task assignments [34]. But we do not want this assignment bias to influence the final results. So we propose dropping answers that can change the structure of the graph to alleviate this bias.
- Meanwhile, this proposal also has some sort of connection with the regularisation technique in the graph learning community like DropEdge [28]. Therefore, this can also prevent our model from overfitting.

Instead of randomly dropping answers in prior work [28], we selectively drop some answers according to their relative importance to produce the task label. In a label aggregation process, important answers follow the principle that the majority is more informative. Therefore, we compute the probability of dropping an answer as,

$$P(a_{t_n}^{w_i}) = \frac{1-\mathbf{1}(a_{t_n}^{w_i} = \tilde{a}_{t_n})}{\sum_j(1-\mathbf{1}(a_{t_n}^{w_j} = \tilde{a}_{t_n}))} \tag{14}$$

where $\tilde{a}_{t_n}$ is the answer of t_n generated by majority voting.

3.3 Training Algorithm

The overall loss in the training process of ATHENA is determined by both the label prediction loss and the answer prediction loss, namely the $\mathcal{L}_q$ and $\mathcal{L}_a$. To efficiently utilize different training signals, we adjust the weight of these two types of loss functions to compute the overall loss. Thus, the overall loss function is formulated as follows,

$$\mathcal{L} = \frac{ep}{EP}\mathcal{L}_q + (1-\frac{ep}{EP})\mathcal{L}_a \tag{15}$$

where ep is the current time epoch index and EP is the overall number of the time epoch. Based on the overall loss, we train ATHENA according to the algorithm shown in Algorithm 1. In this algorithm,

- We first construct a labeling graph $\mathcal{G}$ to represent the classification problem using equation (1) (line 1).
- Then, we drop some of the answers according to their predefined probability as shown in Section 3.2, Meanwhile, we learn the representations of the workers and tasks with a graph transformer (lines 3-4).

Algorithm 1 The Training Algorithm for ATHENA

Input: The labeling graph $\mathcal{G}$, the number of attention heads K, the learning rate η.

```
 1: Calculating the features of $\mathcal{G}$ according to Eq. (1)
 2: for epoch $\leq$ maxEpoch do
 3:     Dropping some answers, i.e., the edges of $\mathcal{G}$.
 4:     Learning tasks' and workers' representations.
 5:     Calculating the label prediction by Eq. (10).
 6:     Predicting the answers by Eq. (12).
 7:     Calculating the loss with Eq. (15).
 8:     Backpropagation and updating parameters.
 9: end for
10: return The labels of all tasks.
```

- After that, we get the tasks' labels and the predicted answers according to Section 3.1 (lines 5-6).
- Finally, we update the parameters of ATHENA based on the overall loss (lines 7-10).

Table 1: Statistics of the datasets used in our experiments.

Dataset	Worker	Task	Answer	Option
SENTI	1,960	98,980	569,274	5
FACT	57	42,624	214,915	3
TREC	762	19,033	88,385	2
WEB	177	2,665	15,567	5
ZC_all	78	2,040	20,125	2
ZC_in	25	2,040	10,495	2
ZC_us	74	2,040	11,155	2
DOG	109	807	8,070	4
MS	44	700	2,945	10
CF	461	300	1,720	5

4 Experiments and Evaluations

This section presents an overview of the datasets and baselines used in our experiments, followed by a detailed presentation of the performance results and analysis of ATHENA.

4.1 *Experimental Methodologies*

4.1.1 *Datasets*

For our experiments, we selected 10 real-world datasets that have been previously used in related works [14, 36, 45, 44, 27]. These datasets, as shown in Table 1, vary in size from 300 to over 90,000 records, and cover a diverse range of classification crowdsourcing problems. Specifically, they include both binary and multi-class classification tasks, and cover a variety of classification domains such as sentiment analysis and entity recognition. For our proposed ATHENA, we randomly selected 9 tasks and provided them with golden labels. These datasets were selected to provide a comprehensive evaluation of our approach in various settings and to demonstrate its effectiveness across a diverse range of crowdsourced classification problems.

4.1.2 *Baselines*

To conduct a comprehensive evaluation of our algorithm, we compare ATHENA with eight other mainstream classic methods.

- *MV (Majority Voting)* is the most conventional method which selects the most voted answer as the true label.
- *DS (Dawid & Skene)* is a label aggregation method based on the EM approach, which only considers the model of the individual worker [5].
- *GLAD (Generative model of Labels, Abilities, and Difficulties)* is a representative label aggregation method that constructs a probabilistic model of task and worker to infer the true label [38].
- *PM (Participant-Mine voting)* leverages a weighted aggregation approach to estimate workers' ability in order to obtain the right answers [2].
- *CATD (Confidence-Aware Truth Discovery)* simply treats the workers differently and adopts a weighted majority voting model to estimate the confidence interval of worker credibility [16].
- *BWA (Bayesian Weighted Average)* depends on the Bayesian model to adjudicate highly redundant annotations [18].
- *IBCC (Independent Bayesian Classifier Combination)* focuses on exploiting worker correlation for label aggregation [15].
- *EBCC (Enhanced Bayesian Classifier Combination)* is enhanced with an independent Bayesian classifier combination based on *IBCC* [19].

Additionally, we compare it with three recently proposed deep learning-based methods.

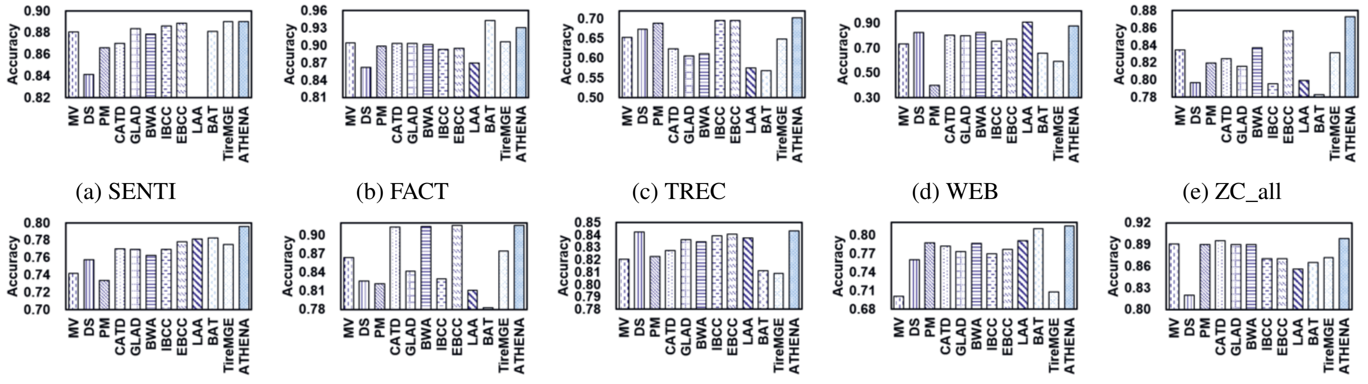
- *LAA (Label-Aware Autoencoders)* is a representative method that integrates a classifier and a reconstructor into a unified model to infer labels in an unsupervised manner [40].
- *TiReMGE (Reliability-driven Multiview Graph Embedding framework for Truth inference)* is a method that can infer the truth with a multi-view graph embedding framework [39].
- *BAT (Bipartite Attention-driven Truth)* is a method that represents the labeling process as a bipartite graph and uses the designed graph attention neural network to infer the truth [23]. Since this is a semi-supervised learning algorithm that uses lots of labels, we compare it using the same amounts of labels as ATHENA.

In addition to these existing baselines, we would like to highlight that some, such as DS and GLAD, can be extended to a label-enhanced version that can be viewed as a semi-supervised variant of the original algorithm. Therefore, we also implement five label-enhanced algorithms: *EDS* (enhanced DS), *EGLAD* (enhanced GLAD), *ECATD* (enhanced CATD), *EPM* (enhanced PM), and *ELAA* (enhanced LAA).

Specifically, as all of these label-enhanced algorithms are based on a variant of the EM algorithm, we use the same number of labeled questions in the E-step. Specifically, we use the correct label instead of the label estimated by the model to infer the parameters. These algorithms are also trained using the same initial labels as those used in ATHENA. In order to assess the effectiveness of these initial labels in our proposal, we compare ATHENA to the five label-enhanced variants of the compared baselines. To ensure a fair comparison, we use the implementation and parameters of these baselines as reported in their respective papers. Additionally, we run all of these baselines on a server equipped with an Intel Xeon Gold 6148 CPU and 128GB of memory. Due to the large size of some datasets, such as SENTI, the LAA algorithm encounters a memory error. Therefore, we set the value as null for these datasets.

4.2 *Performance Analysis*

In this part, we first compare our proposed ATHENA with the other 11 methods. As we can see in Figure 2, ATHENA can get better results in these datasets with extremely few labels, while existing methods only perform well in some cases. Among the traditional methods, EBCC only achieves the second-best performance on four datasets, while DS achieves the best on two datasets. Although PM achieves good results on many datasets, it only achieves a score of 0.3985 on the WEB dataset, which is much worse than the average. This is because the inherent modeling approach of PM is not well-suited for the WEB dataset. In terms of more recent deep learning-based methods, we found that LAA and TireMGE achieve good performance on only one or two datasets. As for BAT, while it achieves good results with a large number of labels [23], it performs poorly when trained with only nine labels. Besides, from the perspective of algorithms, we find that different datasets present different characteristics. For example, SENTI and FACT are similar in that even naive majority voting can get good results. For them, modeling the individual component may not help much even degrade the performance.



(a) SENTI (b) FACT (c) TREC (d) WEB (e) ZC_all

(f) ZC_in (g) ZC_us (h) DOG (i) MS (j) CF

Figure 2: The accuracy of different algorithms on 10 real-world datasets.

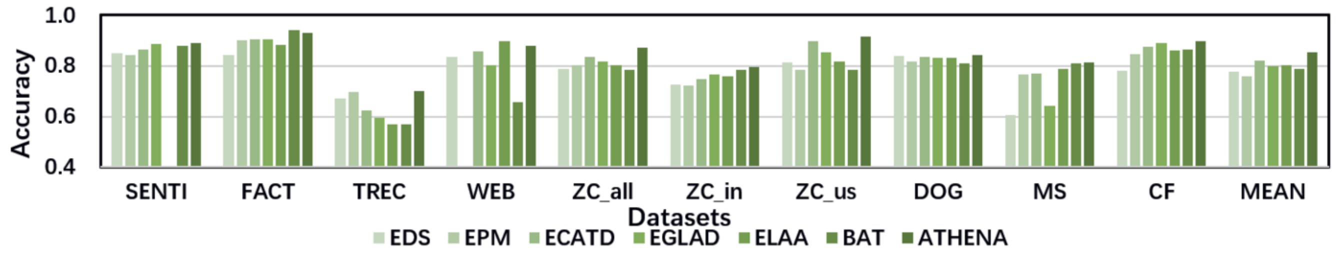


Figure 3: The comparison of different enhanced label aggregation approaches.

We also observe that the simplest model, MV, outperforms some of the more complex models, such as GLAD, on several datasets (e.g., TREC, ZC_us, etc.). This suggests that a complex modeling approach can be prone to inaccuracy or even incorrectness. The significant results of our model are because our proposed method can effectively model the interaction of workers and tasks with minimal assumptions as well as learn from the hidden signals in the labeling process. Also, our proposed ATHENA can forge a connection between two areas that have once evolved mostly independently: label aggregation and self-supervised learning. This connection allows us to leverage self-supervised learning methods to significantly improve the SOTA on label aggregation.

4.3 *Comparison with Enhanced Baselines*

We then compare ATHENA with the enhanced baselines. The results are shown in Figure 3. As we can see, the label-enhanced algorithms bring little improvement over their original baselines. In particular, although the EGLAD algorithm performs better than GLAD on most datasets, the difference is quite small. The biggest improvement is achieved in the ZC_us dataset which is only 0.01. Some other algorithms like DS do not benefit from the added labels as well instead that they have worse performance on some datasets (e.g., MS dataset). The same trend is observed for the deep learning-based methods (ELAA and BAT). This is because the added labels are too few in number to offer valuable information for these algorithms. Simply incorporating these labels into the iterative process does not improve the performance of these algorithms and may even harm them.

In addition, it is obvious that the proposed ATHENA ensures much higher accuracy than the other label-enhanced algorithms when using the same amount of training data. This is because the proposed self-training method can effectively extract information from the crowdsourcing process itself.

4.4 *Label Efficiency Analysis*

To gain a better understanding of the impact of correct labels, we conduct experiments with varying numbers of golden labels. Specifically, instead of using 9 labels for each class as in the previous experiments, we use 1, 3, 5, and 7 labels for each class. The results are shown in Figure 4, where we present the accuracy on different datasets using a bar plot and use a triangle to denote the mean accuracy for different label sizes.

As shown in the figure, the mean accuracy of the model improves with an increase in the number of labeled data. This suggests that more golden labels are beneficial for the learning process. We also observe that different datasets exhibit varying characteristics. For instance, some datasets such as TREC can generate good representations even with only one golden label for each class, whereas the WEB dataset requires a larger number of additional labels during training. Furthermore, we find that ATHENA can achieve good performance even with a small number of labels. In our experiment, it achieves an accuracy of 0.7787 by using only three labels per class, which is better than some algorithms such as PM.

4.5 *Ablation Study*

In this part, we analyze the effectiveness of the core components in the proposed ATHENA. Here we compare ATHENA with the following three variants of itself.

- **FNN**. This variant uses the fully connected neural network instead of the transformer-based neural network as depicted in section 2.3.
- **Answer**. This variant removes the self-supervision process of predicting workers' answers based on the representations of the workers and tasks as shown in Section 3.1.
- **Drop**. This variant randomly drops answers provided by the workers selectively throwing out some answers as shown in section 3.3.

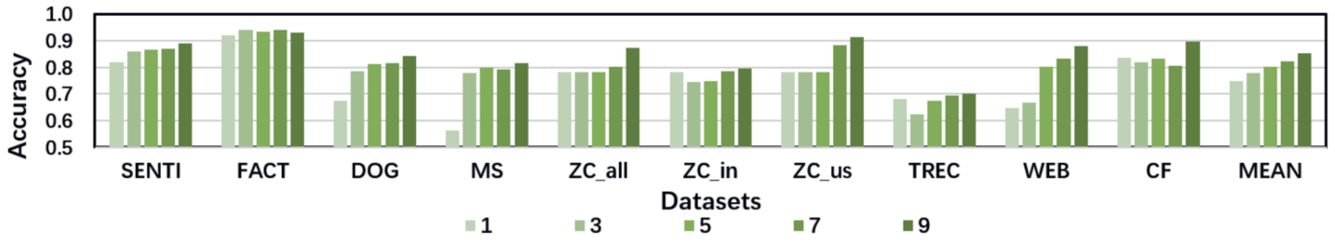


Figure 4: The accuracy of ATHENA with different label sizes under various datasets.

Table 2: The comparison of ATHENA and its variants.

Dataset	ATHENA	FNN	Answer	Drop
SENTI	**0.8901**	0.3068	0.5728	0.8702
FACT	**0.9308**	0.9217	0.9107	0.9217
TREC	0.7023	0.5605	0.5450	**0.7067**
WEB	**0.8788**	0.2596	0.3347	0.7630
ZC_all	0.8729	0.7829	0.7211	**0.8863**
ZC_in	**0.7957**	0.7339	0.7359	0.7873
ZC_us	**0.9154**	0.7829	0.7315	0.9100
DOG	**0.8431**	0.3048	0.4319	0.8314
MS	**0.8148**	0.1033	0.3098	0.7754
CF	**0.8983**	0.3059	0.8471	0.8706

The results are presented in Table 2. We observe that these well-designed components contribute to the improved performance of the model in predicting task labels. Specifically, the graph transformer is one of the most critical components in ATHENA. Without it, the model's performance decreases significantly. Additionally, answer prediction is also crucial for label aggregation. The absence of answer prediction results in extremely poor performance on some datasets (e.g., SENTI, MS) due to the lack of supervision signal in these datasets. These components are important and contribute to the high performance of ATHENA. Moreover, the dropped answers play a crucial role in preventing overfitting, even though there may be rare instances where their performance decreases due to the imprecise estimation of answer importance.

5 Related Work

5.1 Label Aggregation

Label Aggregation [6, 40, 41, 1, 27, 22, 24] is the process of inferring true answers from massive noisy and biased labels. Most label aggregation approaches use the modeling of the worker's ability, e.g., Gianluca et al. leverage a digital between 0 and 1 to represent the probability that a worker can correctly answer a question [6]. Dawid et al. use a confusion matrix to indicate the different abilities of workers [5]. There are also approaches to modeling the questions, the researchers make an assumption that different questions have different difficulties, and the worker will give wrong answers to a with higher probability [25, 38]. E.g., Jacob Whitehill et al. [38] model the difficulty of an image using parameter $\frac{1}{\beta}$ where β is a positive number. If $\frac{1}{\beta} = \infty$ the image labeling task is very difficult, even the most experienced worker can do no better than a random select label. On the other hand, $\frac{1}{\beta} = 0$ means the image is quite intuitive.

Recently, few studies investigate some intelligent methods such as the deep learning-based method for constructing more complicated models [23, 8, 40, 41, 1, 27], but these methods need too much training data which is impractical in real crowdsourcing applications [23]. They cannot effectively use the plentiful information of crowdsourcing compared to ours.

5.2 Self-supervised Learning

Self-supervised learning has emerged as a new technique to get supervision signals from the data itself [30, 37, 13]. It builds pseudo labels by constructing some pretext tasks. Self-supervised learning has been proven to be effective in different domains [12, 11]. In the computer vision domain, there are different pretext tasks is build to learn high-level image representations. Doersch et al. proposed a visual representation learning method based on context prediction, in this approach, random patches are generated from images, and they train a CNN to predict the relative location of these patches [7]. This work has motivated lots of work to try different pretext tasks to help the model to learn a meaningful representation. E.g., Gidaris, et al. propose to rotate the image randomly and predict the rotation degree [9]. Noroozi et al. use a jigsaw puzzle game as the pretext task and train a model to place the shuffled patches in the right place [26]. Zhang et al. propose to train a model to color a gray-scale image [43].

Recently, graph neural networks become an important approach to modeling the graph [21, 17, 10, 20] and there are very few works considered self-supervised learning in the graph domain. Sun et al. first utilized the clustering assignments of node embeddings as guidance to update the graph neural networks [32]. Zhu et al. propose to use Randomly Removing Links (RRL) and Randomly Covering Features (RCF) to help to learn a good representation [46], You et al. first comprehensively analyze the effectiveness of self-supervision in GCN. Their results indicate with a well-designed pretext task, self-supervision can help to learn a more generalized model [42].

6 Conclusion

Crowdsourcing is best applied to large and complex machine learning problems by leveraging a large distributed workforce. In this paper, we aim to exploit self-supervision signals in the crowdsourcing process to improve the efficiency and accuracy of label aggregation. To this end, we proposed ATHENA to effectively aggregate the labels with a self-supervision procedure. ATHENA is designed based on the graph transformer neural network. In ATHENA, a novel graph transformer can effectively infer the representations of workers and tasks on the labeling graph. On the basis of the inferred representations, the proposed algorithm can promote aggregating labels with limited features. The key novelty of ATHENA is its transformer-based feature inference technique may be applicable to other graph-structured machine learning applications.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grants 61832013 and 62172275; in part by the Science and Technology Commission of Shanghai Municipality (STCSM) AI Project under Grant 19511120300; and in part by the Huawei Technologies Company, Ltd., under Grant TC20210806016.

References

[1] Kyohei Atarashi, Satoshi Oyama, and Masahito Kurihara, 'Semi-supervised learning from crowds using deep generative models', in *AAAI*, (2018).

[2] Bahadir Ismail Aydin, Yavuz Selim Yilmaz, Yaliang Li, Qi Li, Jing Gao, and Murat Demirbas, 'Crowdsourcing for multiple-choice question answering', in *AAAI*, (2014).

[3] Jimmy Ba, J. Kiros, and Geoffrey E. Hinton, 'Layer normalization', *ArXiv*, (2016).

[4] Florian Daniel, Pavel Kucherbaev, Cinzia Cappiello, Boualem Benatallah, and Mohammad Allahbakhsh, 'Quality control in crowdsourcing: A survey of quality attributes, assessment techniques, and assurance actions', *CSUR*, (2018).

[5] Alexander Philip Dawid and Allan M Skene, 'Maximum likelihood estimation of observer error-rates using the em algorithm', *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, (1979).

[6] Gianluca Demartini, Djellel Eddine Difallah, and Philippe Cudré-Mauroux, 'Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking', in *WWW*, (2012).

[7] C. Doersch, A. Gupta, and Alexei A. Efros, 'Unsupervised visual representation learning by context prediction', *ICCV*, (2015).

[8] Alex Gaunt, Diana Borsa, and Yoram Bachrach, 'Training deep neural nets to aggregate crowdsourced responses', in *UAI*, (2016).

[9] Spyros Gidaris, Praveer Singh, and Nikos Komodakis, 'Unsupervised representation learning by predicting image rotations', *ICLR*, (2018).

[10] Xiaofeng Hou, Chao Li, Jiacheng Liu, Lu Zhang, Shaolei Ren, Jingwen Leng, Quan Chen, and Minyi Guo, 'Alphar: Learning-powered resource management for irregular, dynamic microservice graph', *IPDPS*, (2021).

[11] Xiaofeng Hou, Jiacheng Liu, Xuehan Tang, Chao Li, Jia Chen, Luhong Liang, Kwang-Ting Cheng, and Minyi Guo, 'Architecting efficient multi-modal aiot systems', *ISCA*, (2023).

[12] Xiaofeng Hou, Jiacheng Liu, Xuehan Tang, Chao Li, Kwang-Ting Cheng, Li Li, and Minyi Guo, 'Mmexit: Enabling fast and efficient multi-modal dnn inference with adaptive network exits', in *Euro-par*, (2023).

[13] W. Jin, Tyler Derr, Haochen Liu, Yiqi Wang, Suhang Wang, Zhiwei Liu, and Jiliang Tang, 'Self-supervised learning on graphs: Deep insights and new direction', *ArXiv*, (2020).

[14] Tatiana Josephy, Matt Lease, Praveen Paritosh, Markus Krause, Mihai Georgescu, Michael Tjalve, and Daniela Braga, 'Workshops held at the first aaai conference on human computation and crowdsourcing: A report', *AI Magazine*, (2014).

[15] Hyun-Chul Kim and Zoubin Ghahramani, 'Bayesian classifier combination', in *Artificial Intelligence and Statistics*. PMLR, (2012).

[16] Qi Li, Yaliang Li, Jing Gao, Lu Su, Bo Zhao, Murat Demirbas, Wei Fan, and Jiawei Han, 'A confidence-aware approach for truth discovery on long-tail data', in *PVLDB*, (2014).

[17] Xu Li, Feilong Tang, Jiacheng Liu, Laurence Tianruo Yang, Luoyi Fu, and Long Chen, 'Auto: Adaptive congestion control based on multi-objective reinforcement learning for the satellite-ground integrated network', in *USENIX Annual Technical Conference*, (2021).

[18] Yuan Li, Benjamin I. P. Rubinstein, and Trevor Cohn, 'Truth inference at scale: A bayesian model for adjudicating highly redundant crowd annotations', in *WWW*, (2019).

[19] Yuan Li, Benjamin Rubinstein, and Trevor Cohn, 'Exploiting worker correlation for label aggregation in crowdsourcing', in *ICML*, (2019).

[20] Jiacheng Liu, Xiaofeng Hou, and Feilong Tang, 'Fine-grained machine teaching with attention modeling', in *AAAI*, (2020).

[21] Jiacheng Liu, Feilong Tang, Long Chen, Xu Li, Jiadi Yu, Yanmin Zhu, Yichuan Yu, and Yanqin Yang, 'Eagle: Heterogeneous gnn-based network performance analysis', *IWQoS*, (2023).

[22] Jiacheng Liu, Feilong Tang, Long Chen, and Yanmin Zhu, 'Exploiting predicted answer in label aggregation to make better use of the crowd wisdom', *Information Sciences*, (2021).

[23] Jiacheng Liu, Feilong Tang, and Jie hao Huang, 'Truth inference with bipartite attention graph neural network from a comprehensive view', *ICME*, (2021).

[24] Jiacheng Liu, Feilong Tang, Yanmin Zhu, Jiadi Yu, Long Chen, and Ming Gao, 'Infer: Distilling knowledge from human-generated rules with uncertainty for stins', *Information Sciences*, (2023).

[25] Fenglong Ma, Yaliang Li, Qi Li, Minghui Qiu, Jing Gao, Shi Zhi, Lu Su, Bo Zhao, Heng Ji, and Jiawei Han, 'Faitcrowd: Fine grained truth discovery for crowdsourced data aggregation', in *KDD*, (2015).

[26] M. Noroozi and P. Favaro, 'Unsupervised learning of visual representations by solving jigsaw puzzles', in *ECCV*, (2016).

[27] Filipe Rodrigues and Francisco C Pereira, 'Deep learning from crowds', in *AAAI*, (2018).

[28] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang, 'Dropedge: Towards deep graph convolutional networks on node classification', *ICLR*, (2020).

[29] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei, 'ImageNet Large Scale Visual Recognition Challenge', *IJCV*, (2015).

[30] Madeline Chantry Schiappa, Yogesh Singh Rawat, and Mubarak Shah, 'Self-supervised learning for videos: A survey', *CSUR*, (2022).

[31] Yunsheng Shi, Z. Huang, Wenjin Wang, Hui Zhong, Shikun Feng, and Yu Sun, 'Masked label prediction: Unified massage passing model for semi-supervised classification', *IJCAI*, (2021).

[32] Ke Sun, Zhanxing Zhu, and Zhouchen Lin, 'Multi-stage self-supervised learning for graph convolutional networks', *AAAI*, (2020).

[33] Feilong Tang, 'Bidirectional active learning with gold-instance-based human training.', in *IJCAI*, (2019).

[34] Feilong Tang, 'Optimal complex task assignment in service crowdsourcing.', in *IJCAI*, pp. 1563–1569, (2020).

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, L. Kaiser, and Illia Polosukhin, 'Attention is all you need', in *NeurIPS*, (2017).

[36] Matteo Venanzi, Oliver Parson, Alex Rogers, and Nick Jennings, 'The activecrowdtoolkit: An open-source tool for benchmarking active learning algorithms for crowdsourcing research', in *AAAI*, (2015).

[37] X. Wang and A. Gupta, 'Unsupervised learning of visual representations using videos', *ICCV*, 2794–2802, (2015).

[38] Jacob Whitehill, Paul Ruvolo, Tingfan Wu, Jacob Bergsma, and Javier R. Movellan, 'Whose vote should count more: Optimal integration of labels from labelers of unknown expertise', in *NeurIPS*, (2009).

[39] Hanlu Wu, Tengfei Ma, Lingfei Wu, Fangli Xu, and Shouling Ji, 'Exploiting heterogeneous graph neural networks with latent worker/task correlation information for label aggregation in crowdsourcing', *TKDD*, **16**, 1 – 18, (2020).

[40] Li'ang Yin, Jianhua Han, Weinan Zhang, and Yong Yu, 'Aggregating crowd wisdoms with label-aware autoencoders', in *IJCAI*, (2017).

[41] Li'ang Yin, Y. Liu, W. Zhang, and Yong Yu, 'Aggregating crowd wisdom with side information via a clustering-based label-aware autoencoder', in *IJCAI*, (2020).

[42] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen, 'When does self-supervision help graph convolutional networks?', in *ICML*, (2020).

[43] Richard Zhang, Phillip Isola, and Alexei A. Efros, 'Colorful image colorization', in *ECCV*, (2016).

[44] Yuchen Zhang, Xi Chen, Dengyong Zhou, and Michael I Jordan, 'Spectral methods meet em: A provably optimal algorithm for crowdsourcing', in *NeurIPS*, (2014).

[45] Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng, 'Truth Inference in Crowdsourcing: Is the Problem Solved?', in *PVLDB*, (2017).

[46] Qikui Zhu, B. Du, and P. Yan, 'Self-supervised training of graph convolutional networks', *ArXiv*, (2020).