IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 13, 2019, accepted December 29, 2019, date of publication January 6, 2020, date of current version January 28, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2964484

Truth Inference With a Deep Clustering-Based Aggregation Model

LI'ANG YIN[1], YUNFEI LIU[1], WEINAN ZHANG[2], AND YONG YU[1]
[1]Department of Computer Science, Shanghai Jiao Tong University, Shanghai 200240, China
[2]John Hopcroft Center for Computer Science, Shanghai Jiao Tong University, Shanghai 200240, China

Corresponding author: Weinan Zhang (wnzhang@sjtu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61702327, Grant 61772333, and Grant 61632017, and in part by Shanghai Sailing Program under Grant 17YF1428200.

ABSTRACT Traditional truth inference algorithms take multiple source labels as input and infer true labels for objects. Besides source labels, object features have been introduced in inference algorithms to achieve superior performance. A typical algorithm such as learning from crowds learns a classification model with the guide of inferred true labels where true labels are inferred from source labels. However, the main shortcoming exists in current algorithms and limits their inference performance: label noise. Since source labels from real-world data are noisy, a classifier is likely to be misguided to learn an imprecise decision boundary. In this paper, we propose a deep clustering-based aggregation model (DCAM) to overcome the shortcoming. DCAM introduces clustering for object features to form fine-grained clusters, where objects in the same cluster are supposed to have similar labels. DCAM exploits a cluster label distribution to represent the labeling information of all objects in the corresponding cluster to overcome the problem of label noise. To implement the idea of clustering-based truth inference, DCAM integrates source label generation and deep clustering in a unified framework by utilizing maximum a posteriori (MAP) estimation. Therefore, the proposed model is a novel approach for truth inference with object features. Experimental results on eight real-world inference tasks show that DCAM has a significant improvement of inference accuracy over the state-of-the-art truth inference algorithms. We further discuss the effect of cluster numbers, the quality of clustering, and illustrate the learned embeddings to support the effectiveness of DCAM.

INDEX TERMS Crowdsourcing, truth inference, clustering methods, neural networks, unsupervised learning, machine learning.

I. INTRODUCTION

Truth inference aims at inferring true labels or objective opinions from different sources, which is also known as truth discovery or label aggregation for crowdsourcing tasks [25], [49]. A simple algorithm, majority voting infers the most voted label as the true label and is widely used in many voting scenarios [4]. As a key technology to solve the problems of massive instance labeling and information explosion, truth inference attracts increasing attention in machine learning.

Supervised machine learning tasks usually need a large number of labeled training instances to yield a desirable model, especially for deep learning models containing millions of parameters to optimize [21]. A common way to obtain labeled instances is from domain experts, but the cost of labeling is usually high and it is time-consuming to collect sufficient labels. An alternative is to distribute labeling tasks to online users where each user labels a certain part of the whole task. Crowdsourcing platforms such as Amazon Mechanical Turk [17] and CrowdFlower [9] provide promising approaches to labeling by the crowds. Crowdsourced labeling has the advantages of efficiency and low cost, but labels from ordinary online users are commonly less accurate than from experts. To reduce the noise, aggregating multiple labels is necessary for many crowdsourced labeling tasks [36]. On the other hand, information publication and propagation have become convenient and efficient today. Everyone can be regarded as an information source. When searching for information, we confront descriptions, discussions, and opinions from various sources. Even for the same object or event, the information from various sources may be different or conflicting with each other. Finding true or valuable information (truth) is one of the essential topics in the era of information explosion [25].

The associate editor coordinating the review of this manuscript and approving it for publication was Qiuhua Huang.

Truth inference is usually considered in the domain of unsupervised learning since labeling tasks do not provide ground truth [25]. Traditionally, an inference algorithm takes multiple labels from various users or sources as input and infers true labels for objects. During the past decade, researchers have developed sophisticated inference algorithms by modeling the relationships between true labels and labels provided by sources, where the capability of a source to provide correct labels is considered as the most important factor. In the literature, weighted majority voting [4], [24], trust propagation [14], [29], [46], and generative models [5], [30], [35], [36], [38], [41], [42] are extensively researched. Recently proposed label-aware autoencoders bridging label aggregation and neural networks, provide more flexible modeling approaches for truth inference [45].

When object features are available, we can improve the performance of an inference algorithm by introducing object features in the model to utilize the additional information from those features. Such an algorithm not only models the relationship between true labels and source labels but also models the relationship between true labels and object features [1], [3], [11], [30]–[33], [44]. Figure 1 gives a workflow of truth inference with object features. Learning from crowds is a typical framework to infer true labels by using object features [31]. Without ground-truth labels, the original paper considers a supervised learning problem to train a classifier with imperfect source labels. The proposed label inference model is also suitable for truth inference in an unsupervised manner. For example, a classifier is learned with the guide of inferred true labels where true labels are inferred from source labels [31].

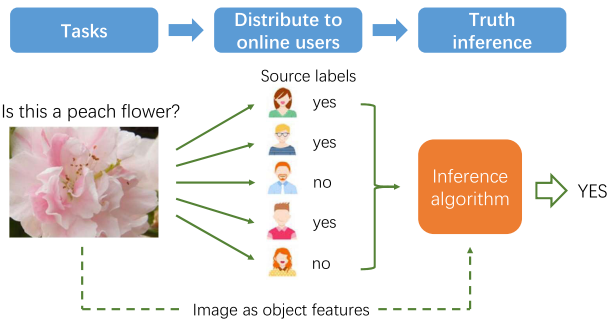


FIGURE 1. A workflow of truth inference by using object features. In the example, the image is used as object features, which is denoted by a dashed arrow. On the other hand, a traditional inference algorithm only uses source labels as input. In this paper, we focus our scope on inference algorithms.

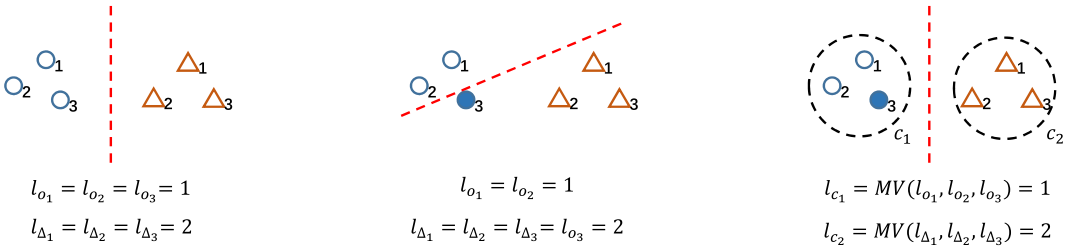
Though inference accuracy is improved comparing with algorithms without using object features, there exists a main shortcoming in currently proposed methods which limits their performance: label noise. Most algorithms adopt the learning-from-crowds framework and use inferred true labels to guide the training of the classifier [1], [11], [31], [33], [44]. The framework is inspired by supervised learning where a classifier is guided by ground-truth labels. However, source labels from real-world data are noisy, which may also lead to noisy inferred labels. As a result, a noisy label may misguide the classifier to learn an imprecise decision boundary. Figure 2 gives an intuitive example. The problem of label noise is severe with sparse source labels where an object receives labels from only a tiny number of sources. Those labels are insufficient to infer reliable true labels. From our point of view, object features provide additional information than source labels and are supposed to bring objects with similar features together to reduce the noise from source labels. Unfortunately, previous methods consider each object in isolation and do not provide explicit solutions to using object features to overcome label noise.

We propose a deep clustering-based aggregation model (DCAM) to make full use of object features and overcome the problem of label noise. DCAM introduces deep clustering for object features to form clusters. The number of clusters is usually set larger than label categories, to obtain fine-grained clusters which reflect underlying unambiguous patterns of objects. Similar objects in the same cluster are then supposed to have similar labels, therefore inferring a true label for a single object benefits from source labels of other objects in the same cluster. With enriched labels, a cluster may infer a more accurate label than using a single object. Thus label noise is substantially alleviated. Figure 2c gives an intuitive example to illustrate utilizing clustering to reduce label noise. In DCAM, we introduce cluster labels as a model parameter to represent the aggregated information of source labels of all objects in the corresponding clusters.

Specifically, DCAM develops a deep clustering method to obtain fine-grained clusters in the embedding feature space from object features, and a generative process to generate source labels by utilizing cluster labels. DCAM exploits maximum a posteriori (MAP) estimation to integrate deep clustering and label generation in a unified framework. In the framework, label generation is regarded as a likelihood of generating source labels from model parameters, and deep clustering is regarded as a prior for model parameters given object features. After model optimization, cluster labels are used to infer reliable true labels. To the best of our knowledge, DCAM is a pioneering algorithm to introduce clustering for truth inference and to bridge label inference and deep clustering in a unified framework. The performance of DCAM is compared with the state-of-the-art truth inference algorithms. The experimental results show a significant inference accuracy improvement of DCAM over compared algorithms. The effect of cluster numbers is further discussed and learned embeddings are illustrated to support the effectiveness of DCAM.

II. RELATED WORK

The original purpose of truth inference is to extract diagnosis data from multiple doctors. In 1979, Dawid and Skene proposed a classical aggregation model that evaluates the credibility of doctors and is optimized by the expectation-maximization algorithm [8]. Recently, the topic of truth



(a) A noise-free case (b) A case with noisy labels (c) Utilize clustering to reduce label noise

FIGURE 2. An intuitive example to illustrate how noisy labels misguide the classifier to learn an imprecise decision boundary (b) and how clustering is utilized to reduce label noise (c). We consider binary labels and use circles to denote category 1 and triangles to denote category 2. As a simple example, we assume each object infers a true label by majority voting from source labels and $l_{o_1} = 1$ indicates the majority voting result is category 1 for object o_1. Object features are represented by the relative position of an object in the figure. We use a dashed line to represent the learned decision boundary. (a) A noise-free case. In the ideal case, source labels are noise-free and majority voting results are correct. Then the learned decision boundary with the guide of majority voting results is precise. (b) A case with noisy labels. Source labels of object o_3 are noisy and lead to a noisy majority voting label $l_{o_3} = 2$, where the correct true label should be 1. With the noisy label, a classifier may mistake o_3 as it is from category 2 and learn an imprecise decision boundary. (c) Clustering is utilized to reduce label noise. Close objects form clusters. In the simple example, we apply majority voting on inferred labels of objects in the same cluster and obtain a cluster label. Cluster labels (l_{c_1} and l_{c_2}) are more likely to be correct by considering all objects in the same cluster rather than a single object, that guides a model to learn a correct decision boundary.

inference is also researched as truth discovery or aggregating crowd wisdom, which is regarded as a promising approach to alleviate the noise from multiple sources [25], [46], [49]. The rapid development of online crowdsourcing platforms such as Amazon Mechanical Turk [17] and CrowdFlower [9] support the research and application of crowdsourced labeling. The research on distributing crowdsourcing tasks [23], [34] and aggregating labels [25], [39] is becoming popular.

Traditional truth inference algorithms focus on exploiting labels from online users or sources. Among them, weighted majority voting, trust propagation, and generative models attract the most attention and are extensively researched. As a direct extension from traditional majority voting, weighted majority voting estimates a weight for each source [4], [24]. This kind of method has advantages for understanding, illustration, and implementation. However, since the model is relatively simple, the inference accuracy is mediocre. Trust propagation algorithms explicitly assign trustworthiness for each source and reliability for each label. A typical algorithm iteratively propagates source trustworthiness and label reliability to reach convergence [46]. The underlying assumption is that labels from trustworthy sources are more reliable and sources with reliable labels are more trustworthy. The generative framework is utilized by more recent work for truth inference [5], [26], [30], [35], [38], [41], [42]. A probabilistic model is usually constructed to generate observed labels from the interaction of underlying (unknown) true labels and source credibility. Optimization methods such as maximum a posteriori (MAP) or Bayesian estimation are adopted for model inference. With the flexibility of probabilistic models, factors besides source credibility are introduced by various models, such as object difficulty [5], [42], latent space [41], and confusion matrix [26], [35].

Recently proposed label-aware autoencoders build a bridge between truth inference and neural networks [45]. Analogizing to variational autoencoders [20], label-aware autoencoders infer true labels by simultaneously learning a classifier and a reconstructor, which are parameterized by neural networks respectively. Introducing neural networks makes truth inference modeling more flexible. Other interesting work about truth inference algorithms includes max-margin majority voting [36], truth existence modeling [50], minimax conditional entropy [51], [52], rank aggregating [15], [28], crowd clustering [16], etc. Truth inference techniques are also exploited for multi-label inference in recent work [37], [48].

Besides labels from sources, features of objects are utilized to achieve higher inference accuracy. There are mainly two modeling frameworks: classifying and generative methods. The classifying framework is also known as learning from crowds, which is conventionally researched in the domain of supervised learning to train classifiers from imperfect source labels. Classifying methods generate source labels from a true label and the true label is obtained from the corresponding object features through a classifier [30], [31]. Yan et al. further model source credibility affected by features of the labeling object based on the classifying framework [44]. Besides linear classifiers, deep neural networks are also exploited to enhance the learning capacity of the classifier [1], [11], [33]. The generative framework assumes object features and source labels are both generated from a true label of the corresponding object [12], [13]. Atarashi et al. recently proposed a model for semi-supervised learning by using labels from crowds under the generative framework [3]. To make model inference easier, the proposed model utilizes the technology of variational autoencoders (VAE) [19]. Some algorithms design a specific structure for specific data types. For example, combining latent Dirichlet allocation (LDA) in the model for textual data [32] and utilizing convolutional neural networks for image data [1], [7]. Proposed DCAM is a general approach that does not assume a specific data type.

A recent work utilizes traditional clustering methods for truth inference [47]. The method succeeds on small datasets with well-formed clusters, but this may not be the case of large and sparse datasets where a few clusters corresponding to the label categories cannot separate the dataset well (e.g. our experimental datasets). DCAM utilizes fine-grained clustering to overcome the problem.

Deep clustering is a recently proposed research topic to combine deep learning and clustering. The idea is to exploit a deep model (usually a deep neural network) to learn latent embedding of object features, and to conduct clustering in the embedding feature space. Since the embedding feature space represents objects more clearly, deep clustering methods achieve superior performance than clustering in the original feature space [2]. Most deep clustering methods use an autoencoder to learn latent embedding and exploit a self-paced objective function for model optimization. Such an objective function pulls a latent embedding close to the corresponding cluster centroid and pushes the embedding away from other centroids [22], [43]. Other deep clustering methods adopt deep Gaussian mixture models and construct deep clustering as a generative process: First generate a cluster index from a specific distribution; Then generate underlying embedding from the cluster; Finally generate object features from the embedding [10], [18]. These models benefit from VAE by simplifying model inference. Though truth inference and deep clustering are respectively researched, to the best of our knowledge, there is no pioneering work combining both of them in a unified framework, which utilizes clusters to infer true labels.

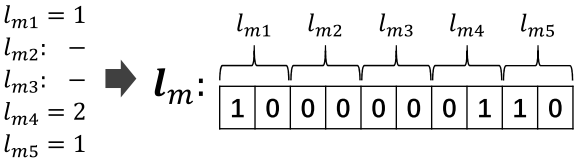


FIGURE 3. An example of constructing label vector $\boldsymbol{l}_m$ by collecting all labels received by object m. There are are 5 sources in total and labels are in binary categories. For object m, source 1, 4, and 5 contribute their labels respectively while source 2 and 3 do not. A label vector $\boldsymbol{l}_m$ is one-hot encoded for each source block. Zeros are filled for unlabeled source blocks.

III. PRELIMINARIES

A. PROBLEM DEFINITION

A label set $\{l_{mn}\}$ contains labels for M objects from N sources. l_{mn} denotes a label object m received from source n, $m \in \{1, \ldots, M\}$ and $n \in \{1, \ldots, N\}$. We consider categorical labels, which means $l_{mn} \in \{1, \ldots, K\}$, where K is the number of label categories. For the convenience of representation and easiness of use in neural networks, we follow label vectorization in label-aware autoencoders and arrange all labels belonging to object m in a label vector $\boldsymbol{l}_m$ [45]. $\boldsymbol{l}_m$ has N source blocks corresponding to N sources and thus contains all labeling information of the object (see Figure 3). Traditional truth inference aims to infer a true label $\tilde{t}_m$ for each object m given the label vector. In this paper, object feature $\boldsymbol{x}_m$ of each object is supposed to be available to make high-quality inference. $\boldsymbol{x}_m \in \mathbb{R}^I$ where I is the dimensionality of a feature vector.

B. LEARNING FROM CROWDS

We introduce (deep) learning from crowds [31], [33] for truth inference, which is used as a baseline in this paper. The introduction is also helpful to show the difference between the learning-from-crowds framework which directly generates a true label through a classifier and proposed DCAM which generates a true label from clusters. Though learning from crowds is originally proposed for training a classifier from imperfect source labels in supervised learning, the framework is quite suitable for truth inference with object features in an unsupervised manner.

A learning-from-crowds model aims to maximize the probability of generating source labels from a true label

$$p(\{\boldsymbol{l}_m\}; \Theta) = \prod_{m=1}^{M} \sum_{y_m=1}^{K} p(\boldsymbol{l}_m | y_m; \Theta) p(y_m; \Theta), \tag{1}$$

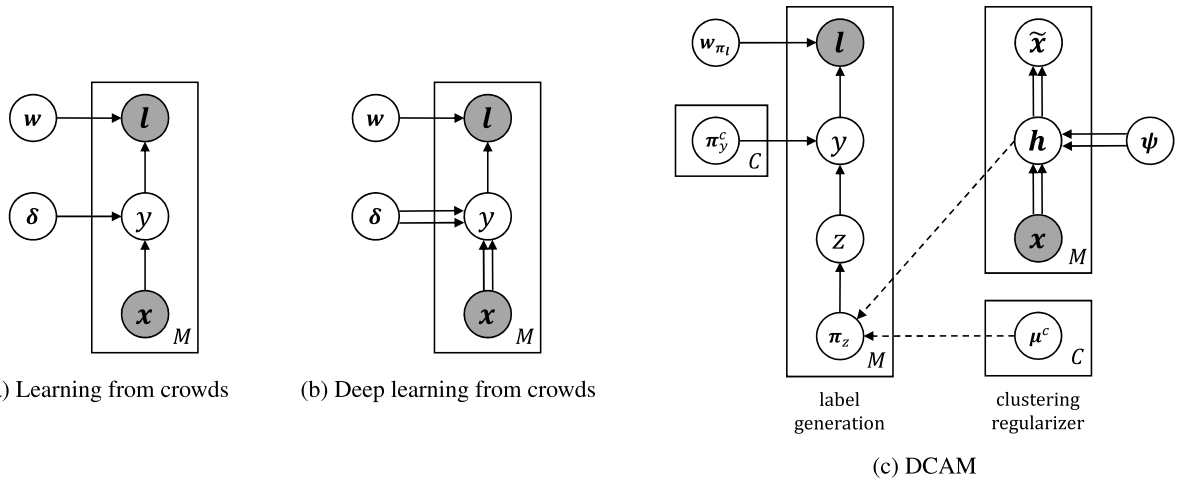
where y_m represents an inferred true label for object m. Model parameters are denoted by Θ. Each label vector $\boldsymbol{l}_m$ is assumed to be independent and identically distributed given a true label y_m and model parameters Θ. Since label vector $\boldsymbol{l}_m$ is a combination of N source blocks, we have a label l_{mn} for each source n generated from a categorical distribution independently.

$$\begin{aligned} p(\boldsymbol{l}_m | y_m; \Theta) &= \prod_{n=1}^{N} p(l_{mn} | y_m; \Theta) \\ p(l_{mn} | y_m; \Theta) &= \mathrm{Cat}\big(\boldsymbol{\pi}^n(y_m)\big). \end{aligned} \tag{2}$$

$\mathrm{Cat}(\cdot)$ denotes a categorical distribution. Distribution parameter $\boldsymbol{\pi}^n(y_m)$ for each source n is output by using y_m as input, via a neural network with weight matrix $\boldsymbol{w}^n$. y_m is one-hot encoded as the input in the neural network. The neural network output should be a valid distribution, therefore softmax is exploited as the activation function of the output layer. To make a simpler representation and convenient implementation, we join all N small neural networks with weight $\boldsymbol{w}^n$ into a big neural network parameterized by $\boldsymbol{w}$. The big neural network computes labels for all source blocks from y_m simultaneously and results in label vector $\boldsymbol{l}_m$. The neural network implementation is an extended weighted majority voting [45] which is comparable to the K-coin model in learning from crowds [31]. Authors of deep learning from crowds call this neural network a crowd layer [33]. By the treatment, learning from crowds and deep learning from crowds have the same representation of generating source labels from a true label.

When object features are available, $p(y_m; \Theta)$ becomes a prior distribution which directly generates y_m from a classifier given object feature $\boldsymbol{x}_m$ as input

$$p(y_m; \Theta) = p(y_m | \boldsymbol{x}_m; \boldsymbol{\delta}). \tag{3}$$



(a) Learning from crowds (b) Deep learning from crowds (c) DCAM

FIGURE 4. Plate notation for learning from crowds, deep learning from crowds, and DCAM. Shaded circles denote observed data (label vectors and object features). A single solid arrow denotes a generative distribution; Paired solid arrows denote a distribution or function parameterized by a multiple-layer neural network. (a) A learning-from-crowds model directly generates a true label through a linear classifier. (b) A deep-learning-from-crowds model generates a true label through a deep neural network. (c) DCAM generates an inferred label from clusters. The framework contains a probabilistic model for label generation and a deep neural network for the clustering regularizer. A single dashed arrow denotes a function to compute $\boldsymbol{\pi}_z$ from $\boldsymbol{h}$ and $\boldsymbol{\mu}^c$. In the figure, some prior distributions on model parameters are omitted for simplicity.

$\boldsymbol{\delta}$ denotes the parameter of the classifier. A learning-from-crowds model exploits a linear classifier, which can be constructed by a fully connected neural network with only input and an output layer. The activation function of the output layer is softmax to make the output a valid distribution. A deep-learning-from-crowds model replaces the linear classifier with a deep classifier which is constructed by a multiple-layer neural network to enrich the learning capacity [33]. Model parameters of both learning from crowds and deep learning from crowds are optimized by using Expectation-Maximizing methods to maximize Eqn. (1). Figure 4a and 4b illustrate the structures of those models.

IV. DEEP CLUSTERING-BASED AGGREGATION MODEL

A. MAP ESTIMATION AND LABEL GENERATION

We propose a deep clustering-based aggregation model (DCAM) which utilizes object features to improve inference quality for truth inference tasks. Given label set $\{\boldsymbol{l}_m\}$ and object feature set $\{\boldsymbol{x}_m\}$, DCAM supposes source labels are generated from model parameters Θ while model parameters depend on the object feature set. Specifically, DCAM optimizes model parameters Θ via maximum a posteriori (MAP) estimation. The MAP estimation is further written into a likelihood term and a prior term.

$$\begin{aligned}\Theta^* &= \arg\max_{\Theta} p(\Theta|\{\boldsymbol{l}_m\}, \{\boldsymbol{x}_m\}) \\ &= \arg\max_{\Theta} p(\{\boldsymbol{l}_m\}|\Theta)p(\Theta|\{\boldsymbol{x}_m\}) \\ &= \arg\max_{\Theta} \sum_{m=1}^{M} \log p(\boldsymbol{l}_m|\Theta) + \log p(\Theta|\{\boldsymbol{x}_m\}). \end{aligned} \quad (4)$$

The equation assumes each object has a label vector independent with other objects given model parameters. The first term in the right-hand is the log-likelihood of generating source labels from model parameters, and the second term is the log-prior on model parameters. In DCAM, we treat the log-prior term as a regularizer to constrain model parameters to allow flexible model construction.

DCAM introduces clusters for true label inference. For the label generation part, DCAM supposes source labels are generated from cluster labels which are associated with corresponding clusters. The probability of belonging to a specific cluster is computed by measuring distances between the latent embedding of features of the object and all cluster centroids. For the regularizer part, DCAM develops a deep clustering method to produce and constrain latent embeddings and cluster centroids based on the object feature set. In this manner, DCAM bridges label inference and deep clustering in a unified framework, as a novel approach for truth inference tasks with available object features. Figure 4c illustrates the framework of DCAM. Note that DCAM usually exploits many more *fine-grained* clusters than label categories. Benefiting from a sufficient number of clusters, the assumption that objects in the same clusters have similar labels is easy to satisfy so that DCAM achieves a desirable performance.

In this subsection, we focus on the label generation part. A generative process is constructed to generate source labels from clusters. Suppose there are C clusters. For each object, we draw a cluster index $z \in \{1, \ldots, C\}$ first. Then we draw an inferred label y for the object from the chosen cluster z. Finally, we draw label vector $\boldsymbol{l}$ from y. Note that here we use a brief notation that omits the subscript m for a specific object. Formally, the generative process is described as follows.

For each object:

1. Draw cluster index z with

$$p(z) = \text{Cat}(\boldsymbol{\pi}_z). \quad (5)$$

$\text{Cat}(\boldsymbol{\pi}_z)$ is a categorical distribution which uses $\boldsymbol{\pi}_z$ as the probability of each cluster being chosen.[1] $\boldsymbol{\pi}_z$ is a vector of C dimension where the c-th element is computed from a normal distribution with the distance between latent embedding $\boldsymbol{h}$ and cluster centroid $\boldsymbol{\mu}^c$. The result is then normalized by computing the distributions between the latent embedding and all cluster centroids.

$$\boldsymbol{\pi}_z|_c = \frac{e^{-||\boldsymbol{h}-\boldsymbol{\mu}^c||^2/2}}{\sum_{c'=1}^{C} e^{-||\boldsymbol{h}-\boldsymbol{\mu}^{c'}||^2/2}}. \tag{6}$$

We use $\boldsymbol{v}|_c$ to denote the c-th element in vector $\boldsymbol{v}$. A normal distribution assigns a large probability for close clusters and a tiny probability for distant clusters. In practice, the distribution is effective to distinguish several close clusters from many distant clusters when the number of clusters in DCAM is usually more than 100. Latent embedding $\boldsymbol{h}$ and cluster centroids $\{\boldsymbol{\mu}^c\}$ are produced and constrained from a deep clustering regularizer which is described in the next subsection.

2. Draw inferred label y given cluster index z with

$$p(y|z=c) = \text{Cat}(\boldsymbol{\pi}_y^c). \tag{7}$$

$y \in \{1, \ldots, K\}$ and $\boldsymbol{\pi}_y^c$ has dimensionality of K where K is the number of label categories. Each cluster c has its own cluster label distribution $\boldsymbol{\pi}_y^c$. The distribution is regarded as a model parameter to optimize.

3. Draw label vector $\boldsymbol{l}$ from y with

$$\begin{aligned} p(\boldsymbol{l}|y) &= \prod_{n=1}^{N} p(l_n|y) \\ p(l_n|y) &= \text{Cat}\big(\boldsymbol{\pi}_l^n(y)\big). \end{aligned} \tag{8}$$

We use l_n to denote the label received from source n of the object. Here we utilize the same neural network representation as in the description for Eqn. (2), where each small neural network for source n is denoted as $\boldsymbol{w}_{\pi_l}^n$, and the joint big neural network is denoted as $\boldsymbol{w}_{\pi_l}$.

By exploiting the generative process, we have the joint distribution over $\boldsymbol{l}$, y, and z.

$$p(\boldsymbol{l}, y, z) = p(\boldsymbol{l}|y)p(y|z)p(z). \tag{9}$$

By using the definitions of corresponding distributions (5), (7), and (8), the likelihood of generating $\boldsymbol{l}$ from model parameters in Eqn. (4) is derived as

$$\begin{aligned} p(\boldsymbol{l}) &= \sum_{y=1}^{K}\sum_{z=1}^{C} p(\boldsymbol{l}|y)p(y|z)p(z) \\ &= \sum_{k=1}^{K}\sum_{c=1}^{C}\left(\prod_{n=1}^{N}\prod_{k'=1}^{K}\left(\boldsymbol{\pi}_l^n(k)|_{k'}\right)^{\mathbf{1}(l_n=k')}\right)\boldsymbol{\pi}_y^c|_k\,\boldsymbol{\pi}_z|_c, \end{aligned} \tag{10}$$

where $\mathbf{1}(l_n = k')$ is an indicator. It is 1 if the expression $(l_n = k')$ is true, otherwise $\mathbf{1}(l_n = k') = 0$.

[1] Subscript z in $\boldsymbol{\pi}_z$ indicates the probability is used to generate variable z. It is only a name which does not change with the variable.

B. REGULARIZER AND DEEP CLUSTERING

We treat the log-prior term in Eqn. (4) of DCAM as a regularizer, which may not strictly follow traditional probabilistic distributions but offers a more flexible approach to constraining model parameters. From Eqn. (6), (7), and (8), we have model parameters in DCAM as

$$\Theta = \big\{\boldsymbol{\psi}, \{\boldsymbol{\mu}^c\}, \{\boldsymbol{\pi}_y^c\}, \boldsymbol{w}_{\pi_l}\big\}, \tag{11}$$

where $c \in \{1, \ldots, C\}$. $\boldsymbol{\psi}$ is the model parameter of a mapping function $f_{\boldsymbol{\psi}}(\cdot)$ to compute embedding $\boldsymbol{h}$ from input feature vector $\boldsymbol{x}$

$$\boldsymbol{h} = f_{\boldsymbol{\psi}}(\boldsymbol{x}). \tag{12}$$

Generally, the mapping function can be chosen from various models. In this work, we use a multiple-layer fully connected neural network as f and $\boldsymbol{\psi}$ denotes model parameters in the neural network.

We develop a deep clustering regularizer to constrain $\boldsymbol{\psi}$ and $\{\boldsymbol{\mu}^c\}$. In other words, we utilize deep clustering to cluster similar objects in the embedding feature space, instead of directly clustering objects in the input object feature space. A latent embedding $\boldsymbol{h}$ thus has the same dimensionality as cluster centroids, and a hyperparameter J denotes the dimensionality of the embedding feature space. A clustering regularizer aims to learn effective embedding by balancing two forces. One force is to pull the embedding vector of an object close to the cluster centroid which the object belongs to and to push the embedding vector away from other cluster centroids. The other force is to maintain essential information of input object features in the corresponding embedding vectors to avoid trivial solutions. We exploit a clustering loss and a reconstruction loss to represent these two forces respectively. Formally, the proposed deep clustering regularizer has the following objective function

$$L_{cr}(\boldsymbol{\psi}, \{\boldsymbol{\mu}^c\}; \{\boldsymbol{x}_m\}) = L_c(\boldsymbol{\psi}, \{\boldsymbol{\mu}^c\}; \{\boldsymbol{x}_m\}) + L_r(\boldsymbol{\psi}; \{\boldsymbol{x}_m\}), \tag{13}$$

where L_c denotes the clustering loss and L_r denotes the reconstruction loss.

The clustering loss is constructed via a self-paced learning manner, to encourage the current clustering distribution $\boldsymbol{q}$ to approach a self-paced target distribution $\boldsymbol{s}$, which is implemented by the cross entropy between these two distributions

$$\begin{aligned} L_c(\boldsymbol{\psi}, \{\boldsymbol{\mu}^c\}; \{\boldsymbol{x}_m\}) &= L_c(\{\boldsymbol{\mu}^c\}, \{\boldsymbol{h}_m\}) \\ &= -\sum_{m=1}^{M}\sum_{c=1}^{C} \boldsymbol{s}_m|_c \log \boldsymbol{q}_m|_c\,, \end{aligned} \tag{14}$$

where $\boldsymbol{h}_m = f_{\boldsymbol{\psi}}(\boldsymbol{x}_m)$. Current clustering distribution $\boldsymbol{q}$ is a vector of C dimension, which is obtained from the Euclidean distance between $\boldsymbol{h}_m$ and $\boldsymbol{\mu}^c$ via a t-distribution. The c-th element in the vector is

$$\boldsymbol{q}_m|_c = \frac{(1 + ||\boldsymbol{h}_m - \boldsymbol{\mu}^c||^2/\nu)^{-\frac{\nu+1}{2}}}{\sum_{c'=1}^{C}(1 + ||\boldsymbol{h}_m - \boldsymbol{\mu}^{c'}||^2/\nu)^{-\frac{\nu+1}{2}}}, \tag{15}$$

where ν is the number of degrees of freedom in t-distribution. In this paper, we set $\nu = 1$ to encourage the change of clusters

during model learning. A self-paced target distribution $\boldsymbol{s}$ is a sharper version of $\boldsymbol{q}$

$$\boldsymbol{s}_m|_c = \frac{(\boldsymbol{q}_m|_c)^\alpha}{\sum_{c'=1}^{C}(\boldsymbol{q}_m|_{c'})^\alpha}. \tag{16}$$

In this paper, we set $\alpha \to +\infty$ to reduce noise when the number of clusters is usually more than 100 in DCAM. That results in a smoothed one-hot vector

$$\boldsymbol{s}_m|_c = \begin{cases} 1-(C-1)\epsilon, & \text{if } \boldsymbol{q}_m|_c \text{ is the largest in } \boldsymbol{q}_m, \\ \epsilon, & \text{otherwise.} \end{cases} \tag{17}$$

ϵ is a small value to avoid zeros in $\boldsymbol{s}$, since zeros may lead to overfitting by learning the distance between $\boldsymbol{h}$ and $\boldsymbol{\mu}^c$ to be infinite. By exploiting the self-paced target $\boldsymbol{s}_m$, the cross entropy in Eqn. (14) pulls the embedding vector $\boldsymbol{h}_m$ close to the cluster centroid $\boldsymbol{\mu}^c$ since $\boldsymbol{s}_m|_c > \boldsymbol{q}_m|_c$, if $\boldsymbol{q}_m|_c$ is the largest element in the distribution vector, and pushes the embedding vector away from other cluster centroids since $\boldsymbol{s}_m|_{c'} < \boldsymbol{q}_m|_{c'}$, $c' \neq c$. The introduction of smoothed one-hot target distribution is the main difference from other deep clustering methods [2], which is effective to distinguish one cluster from other clusters for an object when the number of clusters is large.

A reconstruction loss measures the similarity between input object feature vectors and reconstructed vectors, which is defined by the logarithm probability of generating input vector $\boldsymbol{x}_m$ from reconstructed vector $\tilde{\boldsymbol{x}}_m$

$$L_r(\boldsymbol{\psi}; \{\boldsymbol{x}_m\}) = -\sum_{m=1}^{M} \log p(\boldsymbol{x}_m|\tilde{\boldsymbol{x}}_m), \tag{18}$$

where a reconstructed vector

$$\tilde{\boldsymbol{x}} = g_{\tilde{\boldsymbol{\psi}}}(\boldsymbol{h}) = g_{\tilde{\boldsymbol{\psi}}}\big(f_{\boldsymbol{\psi}}(\boldsymbol{x})\big). \tag{19}$$

Here we omit the subscript m for simplicity. $g_{\tilde{\boldsymbol{\psi}}}(\cdot)$ is a mapping function to compute $\tilde{\boldsymbol{x}}$ from $\boldsymbol{h}$, which usually has a mirror structure as $f_{\boldsymbol{\psi}}(\cdot)$. We use two multiple-layer neural networks to model $f_{\boldsymbol{\psi}}(\cdot)$ and $g_{\tilde{\boldsymbol{\psi}}}(\cdot)$ respectively, which results in an autoencoder to reconstruct $\tilde{\boldsymbol{x}}$ from $\boldsymbol{x}$, through embedding $\boldsymbol{h}$.

If $\boldsymbol{x}$ is assumed from a multivariate Bernoulli distribution

$$p(\boldsymbol{x}|\tilde{\boldsymbol{x}}) = \text{Ber}(\tilde{\boldsymbol{x}}), \tag{20}$$

then

$$\log p(\boldsymbol{x}|\tilde{\boldsymbol{x}}) = \sum_{i=1}^{I} \big(\boldsymbol{x}|_i \log \tilde{\boldsymbol{x}}|_i + (1-\boldsymbol{x}|_i)\log(1-\tilde{\boldsymbol{x}}|_i)\big), \tag{21}$$

where I is the dimensionality of object feature vector $\boldsymbol{x}$. When using a multivariate Bernoulli distribution, we exploit sigmoid as the activation function for the reconstruction layer to ensure each element in a reconstructed vector $\tilde{\boldsymbol{x}}|_i \in \{0, 1\}$, $i \in \{1, \ldots, I\}$. If $\boldsymbol{x}$ is assumed from a multivariate normal distribution, the result is equivalent to the minus square distance between $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$. When using a multivariate normal distribution, there is no need to exploit a specific activation function.

Besides the deep clustering regularizer in Eqn. (13) constrains model parameters $\boldsymbol{\psi}$ and $\{\boldsymbol{\mu}^c\}$, we also assign prior distributions for model parameters $\boldsymbol{\psi}$, $\{\boldsymbol{\mu}^c\}$, $\{\boldsymbol{\pi}_y^c\}$, and $\boldsymbol{w}_{\pi_l}$ respectively to prevent them from overfitting. Each element in the model parameters is supposed to follow a normal distribution $\mathcal{N}(0, 1)$. And the corresponding log-prior is

$$\log p(\Theta) = -\frac{1}{2}\Big(||\boldsymbol{\psi}||^2 + ||\boldsymbol{w}_{\pi_l}||^2 + \sum_{c=1}^{C}(||\boldsymbol{\pi}_y^c||^2 + ||\boldsymbol{\mu}^c||^2)\Big). \tag{22}$$

We rearrange all weight elements as a vector, when calculating the Euclidean norm of weights in a neural network.

C. MODEL TRANING AND TRUE LABEL INFERENCE

An overall objective function of DCAM is derived by substituting Eqn. (10), (13), and (22) into the MAP estimation (4)

$$\Theta^* = \arg\max_{\Theta} \sum_{m=1}^{M} \log p(\boldsymbol{l}_m|\Theta) - L_{cr}(\boldsymbol{\psi}, \{\boldsymbol{\mu}^c\}; \{\boldsymbol{x}_m\}) + \log p(\Theta). \tag{23}$$

Since we introduce a self-paced learning manner in the deep clustering regularizer, in each training epoch, we first calculate the self-paced learning target $\boldsymbol{s}$ based on current model parameters, then update all model parameters given $\boldsymbol{s}$ via gradient ascent. Algorithm 1 illustrates a brief training process of DCAM.

Algorithm 1 DCAM

Input:
$\{\boldsymbol{l}_m\}$, set of source labels
$\{\boldsymbol{x}_m\}$, set of object features
C, number of clusters
J, ν, hyperparameters

Output:
$\Theta = \{\boldsymbol{\psi}, \{\boldsymbol{\mu}^c\}, \{\boldsymbol{\pi}_y^c\}, \boldsymbol{w}_{\pi_l}\}$, model parameters
$\{\tilde{t}_m\}$, set of inferred true labels

1: Initialize embedding $\{\boldsymbol{h}_m\}$ by constructing an autoencoder to minimize Eqn. (18)
2: Initialize cluster centroids $\{\boldsymbol{\mu}^c\}$ by K-means on $\{\boldsymbol{h}_m\}$
3: **while** not convergent **do**
4: Calculate self-paced learning target $\{\boldsymbol{s}_m\}$ given $\{\boldsymbol{h}_m\}$ and $\{\boldsymbol{\mu}^c\}$ via Eqn. (17)
5: Update Θ based on current $\{\boldsymbol{s}_m\}$ via Eqn. (23)
6: **end while**
7: **for** $m = 1$ to M **do**
8: Infer $\tilde{t}_m$ as the category of maximum probability in $p(y_m|\boldsymbol{l}_m; \Theta)$ via Eqn. (24)
9: **end for**

After model training and model parameters are optimized, a true label of an object is inferred by using the poster probability given $\boldsymbol{l}$.

$$p(y|\boldsymbol{l}; \Theta) = \frac{\sum_z p(\boldsymbol{l}, y, z; \Theta)}{\sum_{y',z'} p(\boldsymbol{l}, y', z'; \Theta)} = \frac{\sum_{z=1}^{C} p(\boldsymbol{l}|y)p(y|z)p(z)}{\sum_{y'=1}^{K}\sum_{z'=1}^{C} p(\boldsymbol{l}|y')p(y'|z')p(z')}. \tag{24}$$

An inferred true label of the object is then chosen by the category which achieves the maximum probability. From the equation, we can explain the effectiveness of clustering-based truth inference. Besides generating source labels through $p(\boldsymbol{l}|y)$, a true label is inferred by collecting cluster label distributions $p(y|z)$ with the probability $p(z)$ of belonging to the corresponding clusters. A cluster label distribution contains labeling information of all objects belonging to the cluster, therefore it is effective to alleviate the problem of label sparsity and label noise.

D. COMPUTATIONAL COMPLEXITY

We briefly analyze the computational complexity of DCAM. DCAM is mainly constructed and implemented with neural networks. By using back-propagation, the computational complexity of training a neural network is proportional to the number of objects, the size of the network, and training epochs. Denote the number of objects as M, the maximum training epochs as T. The size of a neural network can be represented by the number of weights in the network. For the label generation part, we have $\mathcal{O}(NK^2)$ for $p(\boldsymbol{l}|y)$, $\mathcal{O}(CK)$ for $p(y|z)$, $\mathcal{O}(CJ)$ for $p(z)$, and $\mathcal{O}(CK)$ for $p(\boldsymbol{l})$, according to Eqn. (8), (7), (5), (10) respectively. Here N is the number of sources; K is the number of label categories; C is the number of clusters; And J is the dimensionality of the embedding feature space. Then the computational cost for the label generation part is $\mathcal{O}(NK^2+CK+CJ)$ for each object in one training epoch. The computational complexity of the clustering regularizer has two parts: clustering and reconstruction. The cost of clustering is $\mathcal{O}(CJ)$ according to Eqn. (14). The cost of reconstruction depends on the scale of the reconstruction multiple-layer neural network. Here we suppose the scale of the network is $\mathcal{O}(IJL)$, which maps an input feature vector of dimension I to a latent embedding vector of dimension J, through L layers. Putting the cost of label generation and cluster regularizer together, we have the overall computational complexity of DCAM as $\mathcal{O}\big(TM(NK^2 + CK + CJ + IJL)\big)$.

Considering the number of label categories has an upper bound for most tasks (e.g. $K \leq 10$), the computing time of DCAM is therefore linear with the increasing of the number of objects, sources, or clusters. Computational complexity analysis theoretically shows that DCAM is practical for large inference tasks.

V. EXPERIMENTS

A. INFERENCE TASKS

We conduct truth inference experiments on eight real-world multiple-labeling tasks with available object features. Table 1 summaries task statistics.

Reuters-21578 contains a **document** categorization task. 1,786 documents with labels from online users are from 8 categories. 38 users contribute labels giving an average of approximate 3 answers per document [32]. We apply latent Dirichlet allocation to bag-of-words feature vectors to obtain 200 topics as object features [6].

TABLE 1. Task statistics.

Task	#objects	#sources	#labels	density	source acc.*
document	1,786	38	5,410	7.97%	59.59%
bill	6,033	498	30,165	1.00%	75.94%
head	6,033	479	30,165	1.04%	82.02%
shape	6,033	507	30,165	0.99%	80.42%
forehead	6,033	464	30,165	1.08%	82.77%
throat	6,033	446	30,165	1.12%	85.88%
underpart	6,033	479	30,165	1.04%	92.41%
breast	6,033	458	30,165	1.09%	72.95%

* Average labeling accuracy of sources.

CUB-200-2010 dataset contains several tasks to label binary local characteristics for 6,033 bird images [40]. Seven labeling tasks are used in the experiment, namely **bill** (bill shape is all-purpose or not), **head** (head pattern is plain or not), **shape** (shape is perching-like or not), **forehead** (forehead is black or not), **throat** (throat is black or not), **underpart** (underpart is yellow or not), and **breast** (breast pattern is solid or not). For each task, about 500 users contribute labels and each image receives 5 labels. These tasks are challenging since source labels are sparse and often disagree with each other. We collect ground truth from whatbird.com and other bird websites for evaluation. Extracting high-level features from image pixels through a deep convolutional neural network is difficult on the dataset since the number of images is not large enough to train useful high-level features from complex image contents which often contain several other objects besides a bird. Alternatively, we use 287 local attributes as object features of an image. Local attributes are collected from online users.

B. IMPLEMENTATION DETAILS

We train DCAM on inference tasks by using Algorithm 1. A common approach is used to search for hyperparameters. We first split the dataset into a training set and a validation set. The model is trained on the training set and we observe the value of the objective function on the validation set (i.e. calculate the MAP estimation on the validation set). Hyperparameters are chosen which achieves the largest value on the validation set. We then use the hyperparameters to train the model on the whole dataset. In the experiment, hyperparameters are set as $J = 40$ and $\nu = 1$. The inference accuracy of DCAM is mainly affected by the number of clusters. We will illustrate the effect via experiments. The autoencoder constructed in deep clustering has a multiple-layer structure, with node numbers $\{I, I/2, 100, J, 100, I/2, I\}$ from the bottom up. I is the number of nodes in the input layer, and J is the number of nodes in the embedding layer. The reconstruction loss is chosen according to the distribution of input object features. We adopt a multivariate Bernoulli distribution as in Eqn. (21) for all tasks. We use gradient ascent to train DCAM. The learning rate is set at 0.001 for the underpart task, and 0.0001 for the other tasks. We implement DCAM

TABLE 2. Inference accuracy comparison.

Algorithm	document	bill	head	shape	forehead	throat	underpart	breast
MV	0.7128	0.8168	0.8760	0.8754	0.8651	0.8934	0.9385	0.7608
TF	0.7223	0.8170	0.8760	0.8797	0.8651	0.8934	0.9385	0.7608
DS	0.6937	0.8221	0.8584	0.8777	0.8540	0.8772	0.9415	0.7499
DARE	0.7245	0.8145	0.8652	0.9105	0.8619	0.8913	0.9430	0.7630
LAA	0.7251	0.8175	0.8762	0.8996	0.8651	0.8936	0.9385	0.7610
MomResp†	0.8416	0.8248	0.8810	0.9113	0.8719	0.8964	0.9466	0.7656
LC†	0.8632	0.8263	0.8881	0.9090	0.8662	0.8971	0.9456	0.7661
DLC†	0.8763	0.8381	0.8777	0.9108	0.8674	0.8936	0.9417	0.7684
ML†	-	0.8410	0.8873	0.9211	0.8704	0.8959	0.9468	0.7633
DCAM†	**0.9052**	**0.8726**	**0.9173**	**0.9218**	**0.9057**	**0.9214**	**0.9528**	**0.8017**

† Algorithms utilizing object features.

by Tensorflow which supports automatic differentiation and GPU acceleration.[2]

We train DCAM on one GTX TITAN X GPU. The training process takes 500 epochs for all tasks. The document categorization task takes 50 seconds to complete, with an average of 0.1s per epoch. The other tasks in the CUB-200-2010 dataset take about 40 seconds respectively, with an average of 0.08s per epoch. The running time empirically shows DCAM is practical for truth inference tasks.

C. INFERENCE ACCURACY COMPARISON

We compare inference accuracy among representative truth inference algorithms. Inference accuracy is the ratio of the number of objects with correct inferred true labels to the overall number of objects. Compared algorithms are:

- MV: Majority voting.
- TF: TruthFinder is a typical trust propagation algorithm by propagating source trustworthiness and label reliability when learning a model [46].
- DS: Dawid & Skene's model is a classical truth inference algorithm which uses a confusion matrix to model source credibility [8].
- DARE: A generative algorithm models both source credibility and object difficulty [5].
- LAA: Label-aware autoencoders exploit the framework of VAE and utilize a neural network to model source credibility [45].
- MomResp: A Bayesian model combines a feature generation component with source label generation [12].
- LC: A learning-from-crowds model generates source labels from a true label and exploits a linear classifier to generate the true label from input object features [31].
- DLC: A deep-learning-from-crowds model replaces the linear classifier in LC with a deep neural network to enhance learning capacity [33].
- ML: A multiple-labelers algorithm models source credibility affected by features of the labeling object based on LC [44].
- DCAM: The proposed model in this paper integrates label inference and deep clustering in a unified framework and introduces clusters for truth inference.

Among compared algorithms, MV, TF, DS, DARE, and LAA only use source labels as input. Besides source labels, MomResp, LC, DLC, ML, and DCAM utilize object features for truth inference. We exploit a K-coin model to model source credibility [31] for algorithms utilizing object features where K is the number of label categories, to fairly compare with DLC and DCAM. The capacity of the neural network used in DLC and DCAM to generate source labels from a true label is equivalent to a K-coin model. Note that we propose DCAM as a general algorithm for truth inference tasks and do not specify data types. Some algorithms specifying data types are not directly compared [1], [7], [32]. As a pioneering approach introducing clustering for truth inference, in this paper, we use a general fully-connected network structure for the clustering part in DCAM to illustrate the effectiveness of clustering rather than using specific structures to deal with data type.[3] We consider data specific structures as further work. Since DCAM has randomness in the clustering part, we run the algorithm 20 times and report the average result. Table 2 illustrates inference accuracy for all compared algorithms.

From Table 2, we can first observe that algorithms utilizing object features usually achieve superior performance than traditional truth inference algorithms using only source labels. Since object features bring additional information, this result seems obvious and encourages researchers to model object features for truth inference if object features are available. TF, DS, DARE, and LAA show only a slight or no advantage compared with MV, which indicates that it is difficult to learn accurate source credibility from sparse and conflicting source labels. In algorithms modeling object features, MomResp and LC have comparable performances on most tasks, which indicates the generative method (e.g. MomResp) and the classifying method (e.g. LC) are both feasible modeling frameworks to utilize object features and source labels. DLC achieves higher inference accuracy than LC on some tasks, which

[2]Demo code is available at https://github.com/coverdark/dcam_demo_code

[3]For fair comparison on all tasks, we use a similar fully-connected network structure for DLC instead of CNN in the original paper.

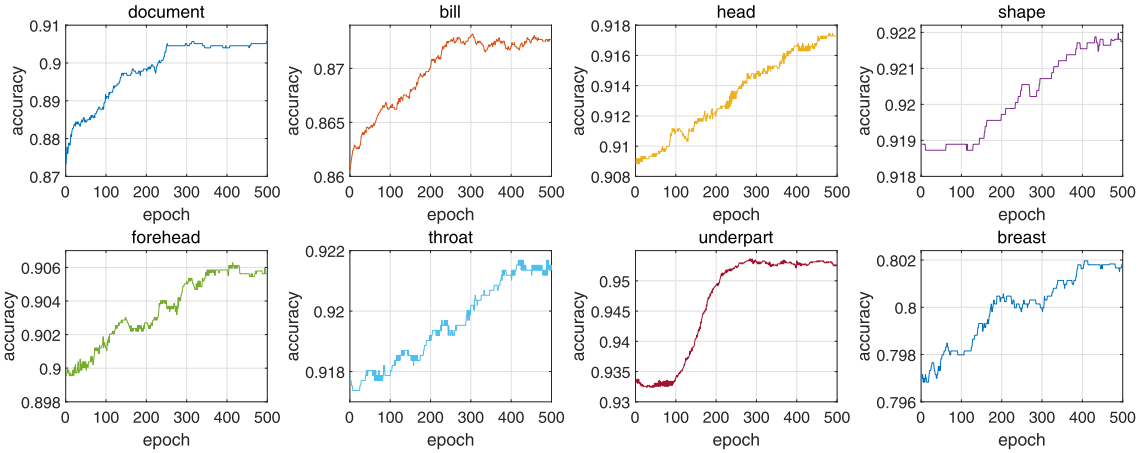


FIGURE 5. Learning curves of DCAM on truth inference tasks.

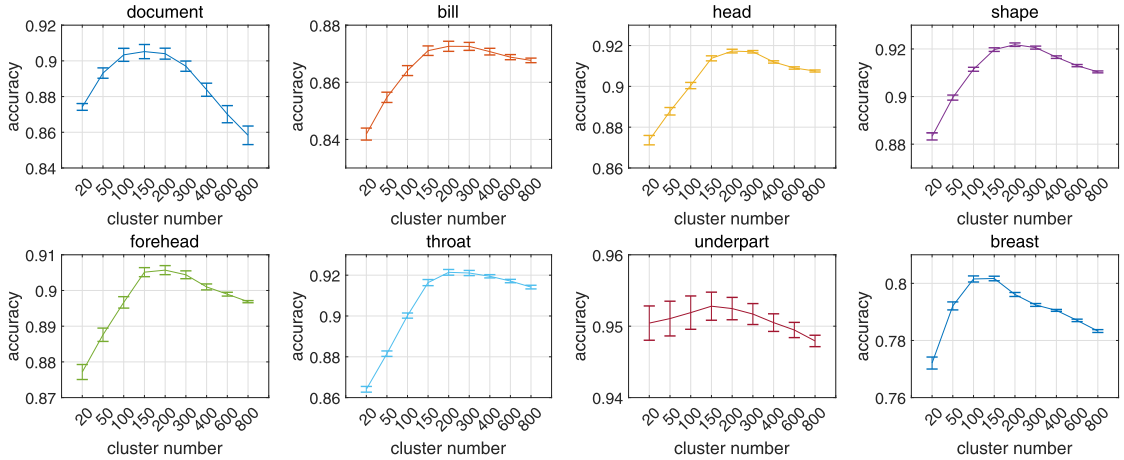


FIGURE 6. Inference accuracy varies with different numbers of clusters.

shows the effectiveness of exploiting deep neural networks. However, on the other tasks, DLC does not show superiority because of label noise. A deep classifier enhances learning capability but may increase the risk of being affected by label noise in unsupervised learning. ML has relatively high inference accuracy on CUB-200-2010 tasks but fails on the document task. It is because modeling source credibility affected by object features may not work when features are noisy. Among compared algorithms, DCAM achieves the highest inference accuracy. The result is significant by conducting a t-test between DCAM and the other algorithm achieving the highest accuracy (see Table 3), with a null hypothesis of DCAM and the other algorithm having the same accuracy. p-values on all tasks are smaller than 0.005. This result shows DCAM makes full use of object features and is more effective than the state-of-the-art truth inference algorithms.

Figure 5 illustrates the corresponding learning curves of DCAM on all tasks. The algorithm shows stable performance after the model converges.

TABLE 3. Significance test.

Task	Algorithm	accuracy	DCAM accuracy	p-value
document	DLC	0.8763	**0.9052** $\pm$ 0.0033	1.1×10^{-19}
bill	ML	0.8410	**0.8726** $\pm$ 0.0019	9.6×10^{-25}
head	LC	0.8881	**0.9173** $\pm$ 0.0008	1.4×10^{-31}
shape	ML	0.9211	**0.9218** $\pm$ 0.0008	1.1×10^{-3}
forehead	MonResp	0.8719	**0.9057** $\pm$ 0.0013	1.1×10^{-28}
throat	LC	0.8971	**0.9214** $\pm$ 0.0022	1.9×10^{-21}
underpart	ML	0.9468	**0.9528** $\pm$ 0.0013	1.4×10^{-14}
breast	DLC	0.7684	**0.8017** $\pm$ 0.0005	1.5×10^{-36}

D. EFFECT OF CLUSTER NUMBER

The number of clusters represents the core idea of the clustering-based aggregation model, which is the key hyper-parameter affecting the inference accuracy of DCAM. Figure 6 illustrates the inference accuracy along with different numbers of clusters on truth inference tasks. The inference accuracy increases when the cluster number increases from a small value and usually achieves the highest when the number

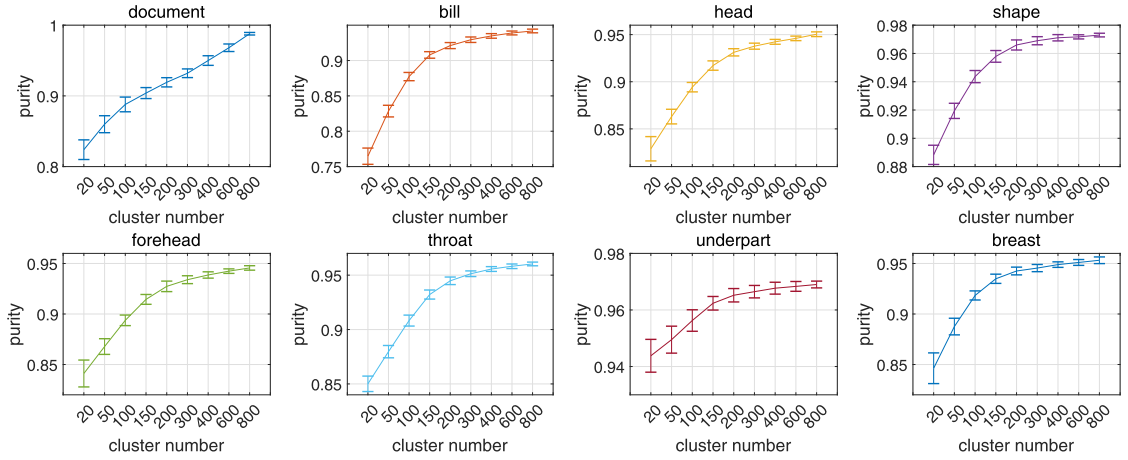
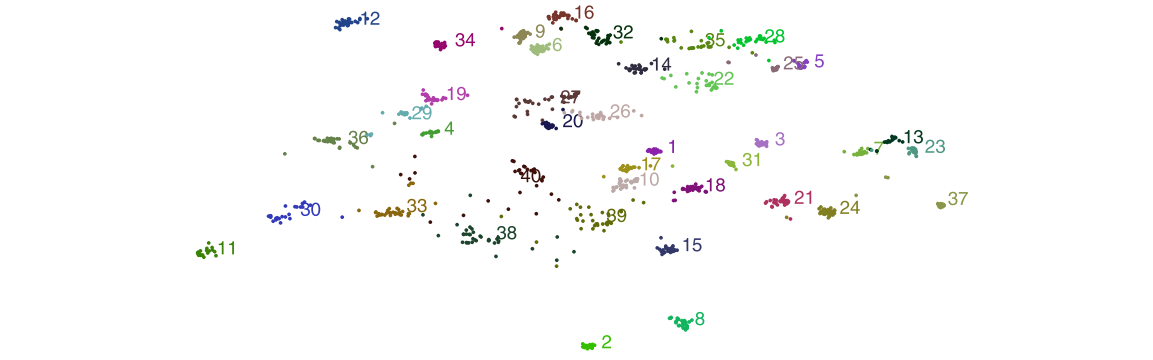


FIGURE 7. Cluster purity varies with different numbers of clusters.



1 Hooded Warbler	11 Anna Hummingbird	21 Pied Kingfisher	31 American Goldfinch
2 Painted Bunting	12 Orchard Oriole	22 Elegant Tern	32 Black footed Albatross
3 Yellow headed Blackbird	13 Summer Tanager	23 Vermilion Flycatcher	33 Carolina Wren
4 Spotted Catbird	14 Long tailed Jaeger	24 Downy Woodpecker	34 Hooded Merganser
5 White Pelican	15 Bay breasted Warbler	25 Ivory Gull	35 Heermann Gull
6 Western Grebe	16 Gadwall	26 Brewer Blackbird	36 Brown Thrasher
7 Pine Grosbeak	17 Common Yellowthroat	27 Groove billed Ani	37 Red headed Woodpecker
8 Blue Grosbeak	18 Prothonotary Warbler	28 Western Gull	38 Yellow billed Cuckoo
9 Pacific Loon	19 Least Auklet	29 Geococcyx	39 White eyed Vireo
10 Yellow breasted Chat	20 Red winged Blackbird	30 White crowned Sparrow	40 Palm Warbler

FIGURE 8. The clustering result of DCAM on the bill task. Cluster number is set as $C = 200$. 40 bird categories are illustrated with different colors in the figure and the corresponding bird categories are shown in the box below.

ranges from 100 to 300. The accuracy then decreases when the cluster number is too large from an appropriate value. We explain the effect of the cluster number as follows: When the cluster number is too small, each cluster contains too many objects even they are not very similar. That confuses clusters and results in inaccurate cluster labels. When the cluster number is too large, a cluster contains too few objects, which is insufficient to infer a reliable cluster label. In the extreme case when the number of clusters equals the number of objects, each object is regarded as a cluster and the advantage of DCAM of bringing similar objects together to enrich the information of cluster labels is eliminated.

E. QUALITY OF CLUSTERING

To further illustrate the quality of clustering, we calculate the purity of learned clusters with different cluster numbers in Figure 7. Purity measures the consistency of clusters, i.e. whether objects in the same cluster have the same gold label or not. After DCAM is learned, we assign each object to cluster c with the maximum probability $\boldsymbol{\pi}_{z|c}$ in Eqn. (6). Then we collect gold labels in each cluster and choose the most frequent label as the gold label of the cluster. Purity is calculated by the ratio of the total number of objects agree with their cluster labels to the number of all objects.[4] Note that in DCAM there are many more clusters than label categories, that make it easier to form fine-grained clusters to group similar objects in a short distance in the embedding feature space. Therefore purity is more likely to achieve a high value when the number of clusters increases. In the extreme case,

[4]Gold labels are only used for illustration and evaluation.

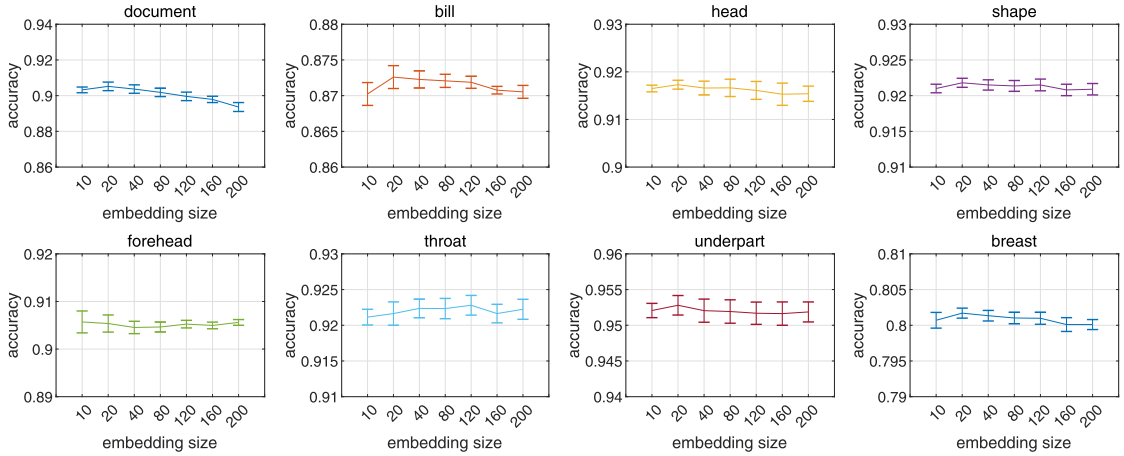


FIGURE 9. DCAM is robust with different settings of embedding size on experiment tasks.

purity reaches 1 if the number of clusters equals the number of objects.

Purity intuitively illustrates the quality of clustering. A low purity indicates inferior clustering quality and inference accuracy that objects in one cluster are not similar or have the same label. This case usually happens when the number of clusters is small. When the number of clusters increases from a small value, purity increases rapidly to achieve a desirable value, and then the increase becomes mild when the number of clusters is sufficient. More clusters may hurt the overall inference accuracy since a cluster does not have enough objects (and source labels) to support reliable results, even the purity score is higher.

F. A CASE STUDY OF EMBEDDINGS AND CLUSTERS

From Figure 6, we can observe that DCAM achieves the highest inference accuracy when the number of clusters is 200 on most tasks from the CUB-200-2010 dataset. Interestingly, the CUB-200-2010 dataset contains bird images from 200 real-world bird categories. By comparing clusters learned from DCAM and real-world bird categories, we can show the effectiveness of learned clusters. To see this, we show the clustering result on the bill task in Figure 8 when the cluster number is set at 200. For a clear view, we exploit t-SNE [27] to map the learned embeddings into 2-dimension, and plot 40 bird categories with one color for each category. From the figure, we can observe most points in a cluster have the same color, which shows the corresponding objects or images are from the same real-world category. The observation intuitively indicates the learned embeddings from DCAM form fine-grained clusters which roughly agree with real-world bird categories. Images of the same bird category usually share the same characteristics, the corresponding cluster label is therefore relatively accurate and supports the superior performance of DCAM.

G. EFFECT OF EMBEDDING SIZE

Embedding size is one of the hyperparameters to define the structure of deep clustering in DCAM. In the experiment, we find DCAM is robust with different settings of embedding size. We set the dimensionality of latent embedding $\boldsymbol{h}$ ranging from 10 to 200 for DCAM and illustrate the inference accuracy on inference tasks in Figure 9. The results are stable with different embedding sizes when $J \geq 20$ for most tasks. This result indicates embedding size $J = 20$ is sufficient for storing useful information of input object features for experiment tasks. A large embedding size may bring noise or slightly decrease accuracy on the document task, but the decreasing is not observed on the other tasks through the experiment.

VI. CONCLUSION AND FUTURE WORK

We point out the main problem that limits the performance of truth inference algorithms utilizing object features: label noise. We propose a deep clustering-based aggregation model (DCAM) to overcome the problem. DCAM exploits clustering to form fine-grained clusters to help true label inference. With fine-grained clusters, objects in the same cluster are supposed to have similar labels and they all contribute to the corresponding cluster label distribution to overcome the problem of label noise. DCAM integrates source label generation and deep clustering in a unified framework, which contributes a novel approach for truth inference with object features. Experimental results show that DCAM has a significant improvement of inference accuracy comparing with the state-of-the-art inference algorithms on real-world truth inference tasks. The effect of different numbers of clusters in DCAM is discussed to further support the idea of clustering-based truth inference. The experiment also shows the robustness of DCAM with different embedding sizes.

In this paper, we propose DCAM as a general approach for different data types. As future work, we will extend DCAM to specific data types to figure out whether combining clustering-based inference model with specific structures (e.g. CNN for image data) will further improve the inference accuracy. Another extension is to fit DCAM in the supervised learning setting as used in learning from crowds. It needs

a modification of the true label inference since there are usually only object features available on the test dataset.

REFERENCES

[1] S. Albarqouni, C. Baur, F. Achilles, V. Belagiannis, S. Demirci, and N. Navab, "AggNet: Deep learning from crowds for mitosis detection in breast cancer histology images," *IEEE Trans. Med. Imag.*, vol. 35, no. 5, pp. 1313–1321, May 2016.

[2] E. Aljalbout, V. Golkov, Y. Siddiqui, M. Strobel, and D. Cremers, "Clustering with deep learning: Taxonomy and new methods," 2018, *arXiv:1801.07648*. [Online]. Available: https://arxiv.org/abs/1801.07648

[3] K. Atarashi, S. Oyama, and M. Kurihara, "Semi-supervised learning from crowds using deep generative models," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018.

[4] B. I. Aydin, Y. S. Yilmaz, Y. Li, Q. Li, J. Gao, and M. Demirbas, "Crowdsourcing for multiple-choice question answering," in *Proc. AAAI*, 2014, pp. 2946–2953.

[5] Y. Bachrach, T. Graepel, T. Minka, and J. Guiver, "How to grade a test without knowing the answers—A Bayesian graphical model for adaptive crowdsourcing and aptitude testing," in *Proc. 29th Int. Conf. Mach. Learn. (ICML)*, 2012, pp. 1183–1190.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.

[7] P. Cao, Y. Xu, Y. Kong, and Y. Wang, "Max-MIG: An information theoretic approach for joint learning from crowds," 2019, *arXiv:1905.13436*. [Online]. Available: https://arxiv.org/abs/1905.13436

[8] A. P. Dawid and A. M. Skene, "Maximum likelihood estimation of observer error-rates using the EM algorithm," *Appl. Statist.*, vol. 28, no. 1, pp. 20–28, 1979.

[9] J. De Winter, M. Kyriakidis, D. Dodou, and R. Happee, "Using crowdflower to study the relationship between self-reported violations and traffic accidents," *Procedia Manuf.*, vol. 3, pp. 2518–2525, Jan. 2015.

[10] N. Dilokthanakul, P. A. M. Mediano, M. Garnelo, M. C. H. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan, "Deep unsupervised clustering with Gaussian mixture variational autoencoders," 2016, *arXiv:1611.02648*. [Online]. Available: https://arxiv.org/abs/1611.02648

[11] K. G. Dizaji and H. Huang, "Sentiment analysis via deep hybrid textual-crowd learning model," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018.

[12] P. Felt, R. Haertel, K. E. Ringger, and D. K. Seppi, "Momresp: A Bayesian model for multi-annotator document labeling," in *Proc. LREC*, 2014, pp. 3704–3711.

[13] P. Felt, K. Black, E. Ringger, K. Seppi, and R. Haertel, "Early gains matter: A case for preferring generative over discriminative crowdsourcing models," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2015, pp. 882–891.

[14] A. Galland, S. Abiteboul, A. Marian, and P. Senellart, "Corroborating information from disagreeing views," in *Proc. 3rd ACM Int. Conf. Web Search Data Mining (WSDM)*, 2010, pp. 131–140.

[15] D. F. Gleich and L.-H. Lim, "Rank aggregation via nuclear norm minimization," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2011, pp. 60–68.

[16] R. G. Gomes, P. Welinder, A. Krause, and P. Perona, "Crowdclustering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 558–566.

[17] P. G. Ipeirotis, "Analyzing the Amazon Mechanical Turk marketplace," *XRDS*, vol. 17, no. 2, p. 16, Dec. 2010.

[18] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational deep embedding: An unsupervised and generative approach to clustering," 2016, *arXiv:1611.05148*. [Online]. Available: https://arxiv.org/abs/1611.05148

[19] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3581–3589.

[20] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*. [Online]. Available: https://arxiv.org/abs/1312.6114

[21] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[22] F. Li, H. Qiao, and B. Zhang, "Discriminatively boosted image clustering with fully convolutional auto-encoders," *Pattern Recognit.*, vol. 83, pp. 161–173, Nov. 2018.

[23] H. Li, B. Zhao, and A. Fuxman, "The wisdom of minority: Discovering and targeting the right group of workers for crowdsourcing," in *Proc. 23rd Int. Conf. World Wide Web (WWW)*, 2014, pp. 165–176.

[24] Q. Li, Y. Li, J. Gao, L. Su, B. Zhao, M. Demirbas, W. Fan, and J. Han, "A confidence-aware approach for truth discovery on long-tail data," *Proc. VLDB Endowment*, vol. 8, no. 4, pp. 425–436, Dec. 2014.

[25] Y. Li, J. Gao, C. Meng, Q. Li, L. Su, B. Zhao, W. Fan, and J. Han, "A survey on truth discovery," *ACM SIGKDD Explor. Newslett.*, vol. 17, no. 2, pp. 1–16, Feb. 2016.

[26] Q. Liu, J. Peng, and A. T. Ihler, "Variational inference for crowdsourcing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 692–700.

[27] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[28] P. Metrikov, V. Pavlu, and J. A. Aslam, "Aggregation of crowdsourced ordinal assessments and integration with learning to rank: A latent trait model," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2015, pp. 1391–1400.

[29] J. Pasternack and D. Roth, "Knowing what to believe (when you already know something)," in *Proc. 23rd Int. Conf. Comput. Linguistics*, 2010, pp. 877–885.

[30] G.-J. Qi, C. C. Aggarwal, J. Han, and T. Huang, "Mining collective intelligence in diverse groups," in *Proc. 22nd Int. Conf. World Wide Web (WWW)*, 2013, pp. 1041–1052.

[31] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy, "Learning from crowds," *J. Mach. Learn. Res.*, vol. 11, pp. 1297–1322, Apr. 2010.

[32] F. Rodrigues, M. Lourenco, B. Ribeiro, and F. C. Pereira, "Learning supervised topic models for classification and regression from crowds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2409–2422, Dec. 2017.

[33] F. Rodrigues and F. C. Pereira, "Deep learning from crowds," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018.

[34] N. B. Shah and D. Zhou, "Double or nothing: Multiplicative incentive mechanisms for crowdsourcing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1–9.

[35] E. Simpson, S. Roberts, I. Psorakis, and A. Smith, "Dynamic Bayesian combination of multiple imperfect classifiers," in *Decision Making Imperfection*. Berlin, Germany: Springer, 2013, pp. 1–35.

[36] T. Tian, J. Zhu, and Y. Qiaoben, "Max-margin majority voting for learning from crowds," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1621–1629.

[37] J. Tu, G. Yu, C. Domeniconi, J. Wang, G. Xiao, and M. Guo, "Multi-label answer aggregation based on joint matrix factorization," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2018, pp. 517–526.

[38] M. Venanzi, J. Guiver, G. Kazai, P. Kohli, and M. Shokouhi, "Community-based Bayesian aggregation models for crowdsourcing," in *Proc. 23rd Int. Conf. World Wide Web (WWW)*, 2014, pp. 155–164.

[39] D. A. Waguih and L. Berti-Equille, "Truth discovery algorithms: An experimental evaluation," 2014, *arXiv:1409.6428*. [Online]. Available: https://arxiv.org/abs/1409.6428

[40] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, "Caltech-UCSD birds 200," California Inst. Technol., Pasadena, CA, USA, Tech. Rep. CNS-TR-2010-001, 2010. [Online]. Available: http://www.vision.caltech.edu/visipedia/CUB-200.html

[41] P. Welinder, S. Branson, P. Perona, and S. J. Belongie, "The multidimensional wisdom of crowds," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 2424–2432.

[42] J. Whitehill, T.-F. Wu, J. Bergsma, J. R. Movellan, and P. L. Ruvolo, "Whose vote should count more: Optimal integration of labels from labelers of unknown expertise," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 2035–2043.

[43] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 478–487.

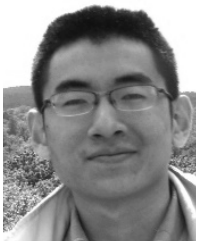
[44] Y. Yan, R. Rosales, G. Fung, R. Subramanian, and J. Dy, "Learning from multiple annotators with varying expertise," *Mach. Learn.*, vol. 95, no. 3, pp. 291–327, Jun. 2014.

[45] L. Yin, J. Han, W. Zhang, and Y. Yu, "Aggregating crowd wisdoms with label-aware autoencoders," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, AAAI Press, Aug. 2017, pp. 1325–1331.

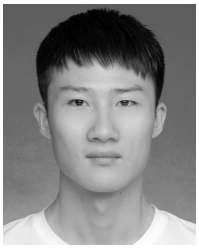
[46] X. Yin, J. Han, and P. Yu, "Truth discovery with multiple conflicting information providers on the Web," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 6, pp. 796–808, Jun. 2008.

[47] J. Zhang, V. S. Sheng, and J. Wu, "Crowdsourced label aggregation using bilayer collaborative clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 10, pp. 3172–3185, Oct. 2019.

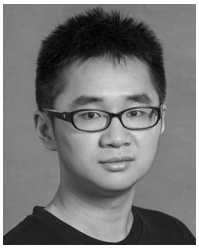
[48] J. Zhang and X. Wu, "Multi-label inference for crowdsourcing," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2018, pp. 2738–2747.
[49] Y. Zheng, G. Li, Y. Li, C. Shan, and R. Cheng, "Truth inference in crowdsourcing: Is the problem solved?" *Proc. VLDB Endowment*, vol. 10, no. 5, pp. 541–552, 2017.
[50] S. Zhi, B. Zhao, W. Tong, J. Gao, D. Yu, H. Ji, and J. Han, "Modeling truth existence in truth discovery," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2015, pp. 1543–1552.
[51] D. Zhou, S. Basu, Y. Mao, and J. C. Platt, "Learning from the wisdom of crowds by minimax entropy," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 2195–2203.
[52] D. Zhou, Q. Liu, J. C. Platt, and C. Meek, "Aggregating ordinal labels from crowds by minimax conditional entropy," in *Proc. ICML*, 2014, pp. 262–270.



LI'ANG YIN received the bachelor's degree in computer science and technology from Zhejiang University (Chu Kochen Honors College) and the master's degree in computer application technology from Shanghai Jiao Tong University, where he is currently pursuing the Ph.D. degree in computer science and technology. He is the Leader of the Crowdsourcing Group, Apex Data & Knowledge Management Lab. His current research domains include label aggregation, unsupervised learning, and neural networks. He is also interested in data mining, recommender systems, and social networks. He has published several research articles in WWW, IJCAI, ICDM, and RecSys.



YUNFEI LIU received the bachelor's degree from the Department of Computer Science, Shanghai Jiao Tong University (IEEE honored Class), where he is currently pursuing the master's degree. He is also a member of the Machine Learning Group, Apex Data & Knowledge Management Lab. His research interests include data mining, machine learning, and crowdsourcing.



WEINAN ZHANG received the B.Eng. degree from ACM Class of Shanghai Jiao Tong University, in 2011, and the Ph.D. degree from University College London, in 2016. He is currently an Assistant Professor with the John Hopcroft Center for Computer Science, Shanghai Jiao Tong University. He has published more than 70 research articles and has been serving as PC/SPC for conferences & journals including KDD, TKDE, SIGIR, ICML, ICLR, AAAI, WWW, WSDM, ICDM, JMLR, and IPM. His research interests include machine learning and big data mining, particularly, deep learning, and reinforcement learning techniques for real-world data mining scenarios, such as computational advertising, recommender systems, text mining, web search, and knowledge graphs.

YONG YU received the master's degree from the CS Department, East China Normal University. He is currently a Professor with the Department of Computer Science, Shanghai Jiao Tong University, and also the Director of the Apex Data & Knowledge Management Lab. As the principal investigator, he took charge of several National Natural Science Foundation of China (NSFC) and China National High Tech 863 Program projects. He has published more than 200 articles and served as a PC members for several conferences including WWW, RecSys, and a dozen other related conferences (e.g., NIPS, ICML, SIGIR, and ISWC) in these fields. His research interests include web search, semantic search, data mining, and machine learning.