An Adaptive Dropout Deep Computation Model for Industrial IoT Big Data Learning With Crowdsourcing to Cloud Computing

Qingchen Zhang, Laurence Tianruo Yang, Zhikui Chen, Peng Li, and Fanyu Bu

***Abstract*—Deep computation, as an advanced machine learning model, has achieved the state-of-the-art performance for feature learning on big data in industrial Internet of Things (IoT). However, the current deep computation model usually suffers from overfitting due to the lack of public available labeled training samples, limiting its performance for big data feature learning. Motivated by the idea of active learning, an adaptive dropout deep computation model (ADDCM) with crowdsourcing to cloud is proposed for industrial IoT big data feature learning in this paper. First, a distribution function is designed to set the dropout rate for each hidden layer to prevent overfitting for the deep computation model. Furthermore, the outsourcing selection algorithm based on the maximum entropy is employed to choose appropriate samples from the training set to crowdsource on the cloud platform. Finally, an improved supervised learning from multiple experts scheme is presented to aggregate answers given by human workers and to update the parameters of the ADDCM simultaneously. Extensive experiments are conducted to evaluate the performance of the presented model by comparing with the dropout deep computation model and other state-of-the-art crowdsourcing algorithms. The results demonstrate that the proposed model can prevent overfitting effectively and aggregate the labeled samples to train the parameters of the deep computation model with crowdsouring for industrial IoT big data feature learning.**

***Index Terms*—Big data, cloud computing, deep computation, dropout, industrial Internet of Things.**

I. Introduction

RECENTLY, Internet of Things (IoT) have been widely applied in industrial manufacturing, resulting in industrial IoT [1]. In a typical industrial IoT system, a large quantity of data about industrial manufacturing, usually called industrial IoT big data, is first collected by sensing terminals, and then it is transmitted to the cloud data center via wireless sensor networks or the internet [2], [3]. Afterwards, the cloud data center can automatically control the process of industrial manufacture according to the collected industrial IoT big data. Moreover, if the cloud data center can analyze the collected data accurately before upcoming abnormal events, some actions can be taken in time to prevent the catastrophic destruction [4], [5]. Obviously, big data analytic and learning in the cloud center play an important role for industrial IoT to provide intelligent services such as smart city and intelligent transportation [6], [7].

As an important data learning technique, deep learning aims to automatically learn hierarchical features by stacking multiple traditional artificial neural networks [8]. Representative deep learning models include stacked autoencoders, deep belief networks, and deep convolutional neural networks which are built by autoencoders, restricted Boltzmann machines, and convolutional neural networks, respectively. In recent years, deep learning models have been widely used in image classification, video retrieval, and cloud resource prediction. To learn features for big data, a deep computation model (DCM) was developed by generalizing the stacking autoencoders to the tensor space [9]. Specially, the DCM has achieved better results than the conventional stacked autoencoder model and multimodal deep learning models for big data feature learning. Generally speaking, the DCM requires a large number of labeled samples to fine-tune the initial parameters for classification tasks. However, it is every expensive or tedious to acquire many labeled training samples in some cases. In fact, the lack of available labeled training samples is a major challenge for not only the DCM, but also other machine learning technologies such as deep learning and active learning. Furthermore, the DCM usually suffers from overfitting due to the lack of publicly available labeled training samples, lowering the accuracy for big data classification tasks.

In this paper, an adaptive dropout deep computation model (ADDCM) with crowdsourcing is presented for big data feature learning in industrial IoT. Dropout was proposed to prevent overfitting for deep learning models [10]. Specially, dropout is especially effective to learn the large-scale deep neural networks with a small number of training samples. However, dropout ignores the relationship between the activating rate and the layer opposition since it always sets the same activating rate with 0.5 for each hidden layer, limiting its effectiveness in avoiding overfitting for the DCM. To set an appropriate activating rate for each hidden layer, a distribution function with regard to the layer

Manuscript received September 23, 2017; revised December 13, 2017; accepted December 27, 2017. Date of publication January 9, 2018; date of current version April 3, 2019. Paper no. TII-17-2204. *(Corresponding author: Laurence T. Yang.)*

Q. Zhang and L. T. Yang are with the School of Electronic Engineering, University of Electronic Science and Technology of China, Chengdu 611731 China, and also with the Department of Computer Science, St. Francis Xavier University, Antigonish, NS B2G 2W5, Canada (e-mail: qzhang@stfx.ca; ltyang@gmail.com).

Z. Chen, P. Li, and F. Bu are with the School of Software Technology, Dalian University of Technology, Dalian 116620, China (e-mail: zkchen@dlut.edu.cn; lipeng2015@mail.dlut.edu.cn; bufanyu@imufe.edu.cn).

Digital Object Identifier 10.1109/TII.2018.2791424

position is defined, and furthermore an ADDCM based on the defined distribution function is presented to prevent overfitting.

Motivated by the active learning framework, the crowdsourcing technique is used to aggregate the labeled samples for training the parameters of the ADDCM in this paper. Crowdsourcing combines human intelligence and the computer power to solve the problem of the lack of available training samples, which is particularly potential for machine learning models without enough labeled training samples [11]. Therefore, crowdsourcing, with cloud computing, is offering a potential opportunity to improve the performance of the DCM. To obtain the labeled training samples for the DCM, some unlabeled samples are outsourced to the cloud platform, and then the labels are acquired by aggregating answers given by human workers in the cloud platform. There are two challenging issues to be addressed for implementing the presented scheme. The first issue is how to select appropriate unlabeled samples to outsource, which is crucial to improve the performance of the answer aggregation. To tackle this challenge, the maximum entropy theory is used in this paper. Specially, the unlabeled samples with the least certainty are chosen as the outsourcing objects. The second issue is how to obtain the reliable labels from human workers by answer aggregation. In this paper, an answer aggregation method based on supervised learning from multiple experts (SLME) is presented. SLME was initially proposed by Raykar *et al.* [12] for answer aggregation by introducing two well-known measures, i.e., sensitive and specificity, which are proved effective in many applications. However, SLME is initially designed for binary classification. Therefore, this strategy is extended for multilabeling to train the parameters of the presented ADDCM by introducing the confusion matrix in this paper. Finally, extensive experiments are conducted to compare the proposed model with the dropout deep computation model (DDCM) and other state-of-the-art crowdsourcing algorithms. Results demonstrate that the presented scheme can effectively obtain the labeled training samples and further prevent overfitting, proving the potential of the presented model for big data feature learning in industrial IoT.

Therefore, the contributions of this paper can be summarized in the following three aspects:

1) The traditional dropout technique ignores the relationship between the activating rate and the layer position, resulting in a low effectiveness for preventing overfitting in the DCM. To tackle this problem, a distribution function is designed to set the activating rate with regard to the layer position. Furthermore, the distribution function is employed to develop an ADDCM for big data feature learning in industrial IoT.
2) To select appropriate unlabeled samples to outsource, an outsourcing sample selection algorithm based on maximum entropy is proposed. In the proposed algorithm, the samples with least certainty are selected as the outsourcing objects.
3) The current SLMP algorithm can only adapt to binary classification for answer aggregation from crowdsourcing. To tackle this issue, this strategy is extended for multilabeling by introducing the confusion matrix. Furthermore, it is used to obtain the labels of outsourcing samples from human workers for fine-tuning the parameters of the ADDCM.

The paper is organized as follows. The related work is reviewed in Section II and the presented ADDCM is introduced in Section III. The outsourcing sample selection algorithm and the aggregation answer method are described in Sections IV and V, respectively. The experimental results are presented in Section VI and the paper is concluded in Section VII.

II. Related Work

The related work, including the standard DCM and the existing answer aggregation algorithms, are reviewed in this section. The standard DCM is illustrated first, followed by the existing answer aggregation algorithms.

A. Deep Computation Model

The standard DCM is established by stacking tensor autoencoders for big data feature learning [9]. The tensor autoencoder was proposed based on the tensor big data representation model where each sample is represented by a tensor. For example, an image in the RGB space can be represented by a three-order tensor $R^{I_h \times I_w \times I_c}$ with $I_h \times I_w, I_c$ denoting the resolution and the color channel, respectively. Specially, an image with the resolution of 1280×1024 in the RGB space can be represented by $R^{1280\times1024\times3}$. Furthermore, a video chip in the MPEG-4 format can be represented by a four-order tensor $R^{I_h \times I_w \times I_c \times I_n}$, where I_n denotes the number of the frames. Thus, a video chip with 150 frames, each of which is the same as the above image, in the MPEG-4 format can be represented by $R^{1280\times1024\times3\times150}$. Given the input sample X represented by a K-order tensor, the tensor autoencoder projects it to the hidden layer H represented by a J-order tensor using a $K+1$-order weight tensor $W^{(1)} \in R^{a\times I_1\times I_2\times\cdots\times I_K}$ with a subtensors and a K-order bias tensor $b^{(1)}$:

$$H_{j_1\ldots j_L} = f\left(\sum_{i_1\cdots i_K}^{I_1\cdots I_K} W^{(1)}_{\alpha i_1\cdots i_K} X_{i_1\cdots i_K} + b^{(1)}_{j_1\cdots j_L}\right) \quad (1)$$

where $\alpha = j_L + \sum_{t=1}^{L-1}(j_t - 1)\prod_{s=t+1}^{L} J_s$, $a = \prod_{l=1}^{L} J_L$ and f is the activation function. Specially, the Sigmoid function, $f(x) = 1/(1+e^{-x})$, is used as the activator since it is a differentiable activation function whose gradient can be obtained for training. Furthermore, Sigmoid can yield the output with an un-normalized probability, simplifying the classification tasks.

Afterwards, the tensor autoencoder projects the hidden layer H to the output layer by an $L+1$-order tensor $W^{(2)} \in R^{b\times J_1\times J_2\times\cdots\times J_L}$ with b subtensors and an L-order bias tensor $b^{(2)}$:

$$Y_{i_1\ldots i_K} = f\left(\sum_{j_1\cdots j_L}^{J_1\cdots J_L} W^{(2)}_{\beta j_1\cdots j_L} H_{j_1\cdots j_L} + b^{(1)}_{i_1\ldots i_K}\right) \quad (2)$$

where $\beta = i_K + \sum_{t=1}^{K-1}(i_t - 1)\prod_{s=t+1}^{K} I_s$ and $b = \prod_{k=1}^{K} I_k$.

For the training sample X, the tensor autoencoder defines the reconstruction error function with regard to the parameter set

$\theta = \{W^{(1)}, b^{(1)}; W^{(2)}, b^{(2)}\}$ as

$$J(\theta; X) = (Y - X)^T G(Y - X) \tag{3}$$

where G denotes the metric matrix [13].

To train the parameters of the tensor autoencoder model, a high-order back-propagation algorithm was implemented in the tensor space. Furthermore, a DCM was established to learn hierarchical features for big data by stacking tensor autoencoders.

To improve the training efficiency for the parameters, two improved DCMs, i.e., a Tucker DCM and a CP DCM, were presented by using the tensor decomposition algorithms to compress the parameters significantly [13], [14].

B. Answer Aggregation Techniques in Crowdsourcing

Answer aggregation is an important challenge in data-driven crowdsourcing. It aims to aggregate the answers given by human workers for the hidden ground truth [12]. The human workers usually have different levels of background and expertise, resulting in the contradiction and uncertainty of the given answers. Moreover, some malicious workers will give incorrect answers sometimes. Therefore, answer aggregation is generally an extremely challenging issue. In the past few years, some algorithms have been introduced for the answer aggregation, which can be roughly categorized by two groups: noniterative methods and iterative methods.

The most representative noniterative method is majority voting (MV) which aggregates the answer of each sample independently [15]. Given a sample that is annotated by many human workers, the MV algorithm counts the number of each label. The true label is recognized as one with the biggest number of votes. This algorithm is simple and straightforward, but it does not take into account the different background and expertise of workers. Honeypot (HP) [16] and expert label injected crowd estimation [17] are another two noniterative algorithms for answer aggregation in data-driven crowdsourcing. They filter the untrustworthy workers by merging some trapping samples whose true labels are already known in a preprocessing step. They are more effective than MV, however, they still have several drawbacks in many applications. For example, the trapping samples are not always available and appropriate trapping samples are difficult to select in many cases.

The most typical algorithm of the iterative category is expectation maximization (EM) [18]. EM iteratively calculates the probability of each sample toward every label by an expectation step and a maximization step. The expectation step estimates the sample probability depending on the current estimation of the workers expertise while the maximization step re-estimates the expertise level of each worker according to the current sample probability. SLME is another algorithm of this type [12]. SLME also utilizes EM to estimate the sample probability and the expertise level of each worker. Different from the EM technique, SLME characterizes the expertise level of each worker by two representative measures, i.e., sensitivity and specificity, instead of the confusion matrix. Other methods of this type include GLAD [19], ITER [20], and M-X [21].

TABLE I
SUMMARIZATION OF ANSWER AGGREGATION ALGORITHMS

Category	Best	Second best	Worst
computation	MV	HP	EM
accuracy	EM	SLME	HP
robustness to spammers	SLME	EM	ITER
adaptivity to multilabeling	EM	MV	HP

Table I presents the summarization of the representative answer aggregation algorithms [22].

Although SLME achieves the best accuracy and works robustly against spammers, it is only suitable for binary classification. In this paper, we extend the SLME algorithm for multilabeling applications.

III. Adaptive Dropout Deep Computation Model

In this part, an adaptive distribution function is presented to set the activating rate with a probability ρ. Among the commonly used distribution functions, the normal distribution can reveal the law of nature effectively, and it is in accordance with the law of human cognition of things. However, the normal distribution is a monotonically increasing function, so it cannot be directly used to set the activating rate of each hidden layer with regard to the layer position. Therefore, a distribution model ρ with regard to the layer position l is defined by modifying the normal distribution function as follows:

$$\rho = \begin{cases} 1 - \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{l} \exp(-\frac{(l-\frac{n}{2})^2}{2\sigma^2}) dl & n = 2k \\ 1 - \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{l} \exp(-\frac{(l-\frac{n+1}{2})^2}{2\sigma^2}) dl & n = 2k+1 \end{cases} \tag{4}$$

where n represents the number of layers in the ADDCM, l denotes the layer position, and σ is used to control the decay velocity.

Obviously, the defined distribution function has the following three properties:

1) Monotonically decreasing.
2) The activating rate of the middle hidden layer equals 0.5.
3) The activating rate is always in $(0, 1)$ for $l = 1, 2, \ldots, n$.

Since the latter part of the distribution function ρ is a normal distribution function with regard to l, the values of the activating rate in each hidden layer, which are set by the proposed distribution function ρ, will be subjected to the approximated normal distribution from top to bottom in the DCM. An ADDCM can be constructed by applying the distribution function to the standard DCM. Furthermore, the parameters of the ADDCM can be trained by combining the high-order back-propagation algorithm and the proposed distribution function, outlined in Algorithm 1.

In Algorithm 1, the forward-pass is performed to compute the activating values of the hidden layer and the output layer combining with the presented adaptive distribution function on lines 3–9. Afterwards, the back-propagation is performed to compute the gradients of the reconstruction error function with regard to the parameters on lines 10–23. Finally, the gradient descent algorithm is performed to update the parameters on lines 25–26.

Algorithm 1: Adaptive Dropout High-order Back-propagation Algorithm.

Input: $\{(X^{(i)}, Y^{(i)})\}_{i=1}^{M}$, η, $threshold$
Output: $\theta = \{W^{(1)}, b^{(1)}; W^{(2)}, b^{(2)}\}$

$\rho = f(l)$;
for $example = 1, 2, ..., M$ **do**
  for $j_l = 1, \ldots, J_l (l = 1, \ldots, L)$ **do**
    Use Eq.(1) to compute $H_{j_1 \ldots j_L}$;
  Use Eq.(4) to compute ρ;
  $mark\{j_1 \ldots, j_L\} = rand(size(H^{(2)}_{j_1 \ldots j_L}) > \rho)$;
  $H^{(2)}_{j_1 \ldots j_L} = H^{(2)}_{j_1 \ldots j_L} .* mark\{j_1 \ldots, j_L\}$;
  for $i_k = 1, \ldots, I_k (k = 1, \ldots, K)$ **do**
    Use Eq.(2) to compute $Y_{i_1 \ldots i_K}$;
  if $J_{TAE}(\theta) > threshold$ **then**
    for $i = 1, ..., I_1 \times I_2 \times \cdots \times I_K$ **do**
      Compute $\sigma_i^{(3)}$ [13];
    for $j_l = 1, \ldots, J_l (l = 1, \ldots, L)$ **do**
      Compute $\sigma^{(2)}_{j_1 j_2 \ldots j_L}$ [13];
    $\sigma^{(2)}_{j_1 \ldots j_L} = \sigma^{(2)}_{j_1 \ldots j_L} .* [ones(size(\sigma^{(2)}_{j_1 \ldots j_L}, 1), 1) mark\{j_1 \ldots, j_L\}]$;
    for $i_k = 1, \ldots, I_k (k = 1, \ldots, K)$ **do**
      $\Delta b^{(2)}_{i_1 \ldots i_K} = \Delta b^{(2)}_{i_1 \ldots i_K} + \sigma^{(3)}_{i_1 \ldots i_K}$;
      for $j_l = 1, \ldots, J_l (l = 1, \ldots, L)$ **do**
        $\Delta w^{(2)}_{i_1 \ldots i_K j_1 \ldots j_L} = \Delta w^{(2)}_{i_1 \ldots i_K j_1 \ldots j_L} + H^{(2)}_{j_1 \ldots j_L} \cdot \sigma^{(3)}_{i_1 \ldots i_K}$;
    for $j_l = 1, \ldots, J_l (l = 1, \ldots, L)$ **do**
      $b^{(1)}_{j_1 \ldots j_L} = \Delta b^{(1)}_{j_1 \ldots j_L} + \sigma^{(2)}_{j_1 \ldots j_L}$;
      for $i_n = 1, \ldots, I_n (n = 1, \ldots, N)$ **do**
        $\Delta w^{(1)}_{j_1 \ldots j_L i_1 \ldots i_K} = \Delta w^{(1)}_{j_1 \ldots j_L i_1 \ldots i_K} + x_{i_1 \ldots i_K} \cdot \sigma^{(2)}_{j_1 \ldots j_L}$;
    //Update the parameters
    $W = W - \eta \times (\frac{1}{M} \Delta W)$;
    $b = b - \eta \times (\frac{1}{M} \Delta b)$;

Let $I \in \max\{I_1, I_2, \ldots, I_N\}$ and $J \in \max\{J_1, J_2, \ldots, J_M\}$. From the steps of Algorithm 1, the computational complexity of the adaptive dropout high-order back-propagation algorithm is dominated by the computation of the partial derivatives of the reconstruction error function with regard to the parameters. Therefore, the computational complexity can be approximated as $O(\rho k I^N J^M)$ where k denotes the number of iterations and ρ is the activating rate.

IV. Outsourcing Sample Selection

The uncertainty sampling is an effective algorithm to select the appropriate samples to outsource for binary classification. Uncertainty sampling always chooses the most ambivalent or least certain unlabeled samples to outsource. In detail, given a binary classification probabilistic model, such as $p(z_i = 1|x_i) = (1 + exp(-\alpha^T x_i - \beta))^{-1}$, uncertainty sampling will select the samples with the probability $\mathrm{p(z = 1|x)} \approx 1/2$. Therefore, the uncertainty sampling algorithm can be viewed as the following optimization problem:

$$\hat{x} := \min_x \left(\frac{1}{2} - p(z|x) \right)^2. \tag{5}$$

Generally speaking, uncertainty sampling is very effective to select useful samples to outsource for binary classification probabilistic models. However, it cannot be applied to multi-classification models directly. To tackle this issue, the maximum entropy is used to select outsourcing samples.

An initial ADDCM can be obtained after the parameters are pretrained by Algorithm 1. The output of the initial model, $\Pr(z_i|x_i, \theta)$, denotes the probability of the sample x_i toward the class z_i. Since the true class of x_i is unknown, the predicted label z_i is not useful to the purpose. However, $\Pr(z_i|x_i, \theta)$ can be used to measure the confidence of the prediction. Furthermore, $\Pr(z_i|x_i, \theta)$ can be used to compute the entropy of each sample. Specially, the information entropy with all class label probabilities of x_i can be calculated by:

$$x_i^{\text{Entropy}} = -\sum_{j=1}^{J} \Pr(z_i|x_i, \theta) \log \Pr(z_i|x_i, \theta). \tag{6}$$

Since the entropy can measure the uncertainty of random variables effectively according to the maximum information entropy theory, the samples with the maximum entropy can be selected as the outsourcing objects.

V. Aggregation Answer Based on EM

A. Probabilistic Model

Given N samples, $X = \{x_1, x_2, \ldots, x_N\}$ is selected from the training set according to the outsourcing sample selection algorithm proposed in the above section, each sample will be labeled by at most T annotators. Let $y_i^{(t)}$ represent the label for the ith sample given by the annotator t. Assuming that the number of classes is J, let z_i be the unknown true label for the ith sample. A confusion matrix $\{\pi_{ij}^{(t)}\}$ is defined in this paper, where $\pi_{ij}^{(t)}$ denotes the probability that the annotator t assigns class j to the sample with the true class i. The aggregation answer aims to estimate the parameters $\{w, \pi_{ij}^{(t)}\}$, where w denotes the weights of the ADDCM, and to produce an estimate for the ground-truth $Z = \{z_1, z_2, \ldots, z_N\}$.

Assuming that the annotators are independent for the training set $X = \{x_1, x_2, \ldots, x_N\}$, and the annotation provided by the annotator t only depends on the unknown truth label z_i, the likelihood function of the parameters $\theta = \{w, \pi_{ij}^{(t)}\}$ for the observation $Y = \{x_i; y_i^{(1)}, y_i^{(2)}, \ldots, y_i^{(T)}\}_{i=1}^{N}$ can be defined as

$$\Pr(Y|\theta) = \prod_{i=1}^{N} \Pr\left(y_i^{(1)}, \ldots, y_i^{(T)}|x_i; \theta\right). \tag{7}$$

By including the unknown true label $Z = \{z_1, \ldots, z_N\}$, the likelihood function can be decomposed as

$$\Pr(Y|\theta) = \prod_{i=1}^{N} \sum_{z_i} \Pr\left(y_i^{(1)}, \ldots, y_i^{(T)}|z_i;\theta\right) \cdot \Pr\left(z_i|x_i;\theta\right). \tag{8}$$

Given the unknown ground-truth label z_i with the assumption that the annotators make their decisions independently, the following equation can be obtained:

$$\Pr\left(y_i^{(1)}, \ldots, y_i^{(T)}|z_i;\theta\right) = \prod_{t=1}^{T}\prod_{j=1}^{J}\left(\pi_{ij}^{(t)}\right)^{\sigma\left(y_i^{(t)},j\right)}. \tag{9}$$

$\Pr(z_i|x_i;\theta)$ denotes the output of the ADDCM. For ease of exposition in the following part, $\Pr(z_i|x_i;\theta)$ is denoted as p_i.

Therefore, the estimation of the parameters $\theta = \{w, \pi_{ij}^{(t)}\}$ can be obtained by maximizing the likelihood function:

$$\hat{\theta} := argmax_\theta\{\Pr(Y|\theta)\}. \tag{10}$$

B. Maximum Likelihood Estimation

Maximizing the likelihood function will become the following problem after taking the logarithm and considering the unknown label $Z = \{z_1, z_2, \ldots, z_N\}$:

$$\hat{\theta} := \sum_{i=1}^{N} \ln \sum_{z_i} \Pr(y_i^{(1)}, \ldots, y_i^{(T)}|z_i;\theta) \cdot \Pr(z_i|x_i;\theta). \tag{11}$$

In this paper, this problem is solved by using the EM algorithm as follows:

1) Initialization: In this paper, the MV is employed to initialize the aggregated labels μ_i depending on the crowdsourcing matrix $Y = \{x_i; y_i^{(1)}, y_i^{(2)}, \ldots, y_i^{(T)}\}_{i=1}^{N}$. Afterwards, μ_i is used to initialize the confusion matrix $\{\pi_{ij}^{(t)}\}$. Finally, the parameters w are initialized by the unsupervised pretraining in the ADDCM.

2) E-Step: According to the crowdsourcing matrix Y and the estimation of the parameters $\theta = \{w, \pi_{ij}^{(t)}\}$, the conditional expectation is computed via:

$$E\{\ln \Pr(Y, Z|\theta)\} = \sum_{i=1}^{N}\sum_{z_i} u_i \ln a_i p_i \tag{12}$$

where $a_i = \Pr(y_i^{(1)}, y_i^{(2)}, \ldots, y_i^{(T)}|z_i;\theta)$, $p_i = \Pr(z_i|x_i;\theta)$, and the expectation is with regard to $\Pr(Y, Z|\theta)$.

The E-step aims to calculate μ_i by Bayes theorem as follows:

$$\mu_i = \frac{p_i \cdot \prod_{t=1}^{T}\prod_{j=1}^{J}\left(\pi_{ij}^{(t)}\right)^{\sigma\left(y_i^{(t)},j\right)}}{\sum_{z_i} p_i \cdot \prod_{t=1}^{T}\prod_{j=1}^{J}\left(\pi_{ij}^{(t)}\right)^{\sigma\left(y_i^{(t)},j\right)}}. \tag{13}$$

3) M-Step: Depending on the current estimated values of the aggregated labels μ_i, the parameters $\pi_{ij}^{(t)}$ are calculated by equating the derivative of $E\{\ln \Pr(Y, Z|\theta)\}$ with regard to $\pi_{ij}^{(t)}$ to 0 as follows:

$$\pi_{ij}^{(t)} = \frac{\sum_{i=1}^{N} \mu_i \cdot \sigma(y_i^{(t)}, j)}{\sum_{i=1}^{N} \mu_i}. \tag{14}$$

TABLE II
TOP-1 ERROR RATES ON CUAVE

Model/Layers	2	3	4	5
DCM	21.2%	19.6%	26.6%	31.3%
DDCM	18.5%	14.2%	12.9%	18.1%
ADDCM	16.4%	12.5%	10.7%	16.5%

Then the parameters w can be updated by the ADDCM.

From the procedure of the answer aggregation based on EM, the E-Step and the M-Step are calculated by the forward-pass and the back-propagation, respectively, in each iteration. Therefore, given the labels as the supervised samples from the crowdsourcing, the parameters of the ADDCM can be refined by iterating E-step and M-step till convergence.

VI. EXPERIMENT

In the experiments, the ADDCM and the ADDCM with crowdsourcing to cloud (ADDCM-DDC) are compared with the standard DCM and the DDCM on two representative data sets, i.e., CUAVE and SNAE2 [23], [24]. Specially, CUAVE is a bimodal dataset consisting of 1800 utterances of digits 0 to 9 said by 36 individuals for five times, which is widely used to evaluate the performance of deep learning models. SNAE2 consists of four groups of video clips, i.e., sport, new, advertisement, and entertainment, each with two labels. Top-1 and top-2 error rates are used to evaluate the classification performance of the presented model since top-N is the most widely used statistics metric for validating deep learning models [25]. Top-1 is used to check if the top class (the one having the highest probability) is the same as the target label, while top-N is used to check if the target label is one of the top N predictions (the N ones with the highest probabilities). Since there are only a few labels in the two experimental datasets, top-1 and top-2 are selected to evaluate the performance of the presented model. Furthermore, each model is run for five times, each running with distinct initial parameters, to evaluate the robustness of the presented model. The average top-1 and top-2 error rates are reported in the following parts. The experiments are conducted using MATLAB R2014b on the laptop with 3.2 GHz Core i7 CPU, 8 GB memory, and 1T driver.

A. Performance Evaluation of ADDCM

In this section, ADDCM is compared with DDCM with a constant activating rate of 0.5 and DCM on CUAVE and SNAE2. To test the effect of the number of hidden layers, several deep architectures are also established, each architecture with different number of hidden layers. For each architecture, the unsupervised pretraining is first performed to obtain the initial parameters, and then the supervised fine-tuning using the training set is performed to obtain the final parameters. Finally, different parameters are used to classify the testing set. The results are presented in Tables II–V.

From Tables II–V, the classification accuracy produced by DCM becomes gradually lower in terms of top-1 and top-2 when

TABLE III
TOP-2 ERROR RATES ON CUAVE

Model/Layers	2	3	4	5
DCM	13.3%	10.4%	14.6%	20.3%
DDCM	14.4%	9.4%	9.2%	13.6%
ADDCM	10.5%	7.6%	7.5%	9.7%

TABLE IV
TOP-1 ERROR RATES ON SNAE2

Model/Layers	2	3	4	5
DCM	33.6%	27.5%	29.3%	35.1%
DDCM	22.4%	19.2%	18.7%	26.4%
ADDCM	19.9%	16.3%	15.2%	19.4%

TABLE V
TOP-2 ERROR RATES ON SNAE2

Model/Layers	2	3	4	5
DCM	22.5%	17.6%	26.3%	32.2%
DDCM	18.6%	16%	15.5%	19.7%
ADDCM	17%	13.4%	12.9%	16.6%

TABLE VI
TOP-1 WITH DIFFERENT σ ON CUAVE

Layers/σ	3	5	10	15	20
2	38.1%	21.2%	16.9%	16.4%	16.3%
3	31.2%	19.6%	13.1%	12.5%	12.5%
4	20.2%	16.5%	10.8%	10.7%	10.6%
5	26.4%	19.6%	18.3%	16.5%	16.2%

TABLE VII
TOP-2 WITH DIFFERENT σ ON CUAVE

Layers/σ	3	5	10	15	20
2	24.6%	16.2%	11.1%	10.5%	10.3%
3	19.8%	14.4%	7.9%	7.6%	7.5%
4	19.1%	13.2%	7.5%	7.5%	7.5%
5	22.5%	15.7%	10.3%	9.7%	9.6%

the number of hidden layers is more than 3 due to the overfitting. However, the classification accuracy produced by ADDCM and DDCM still increases slightly when they have four hidden layers. Such observations demonstrate that both ADDCM and DDCM can prevent the overfitting more effectively than the standard DCM. Furthermore, ADDCM yields smaller top-1 and top-2 values than DDCM. For example, when they have four hidden layers, ADDCM yields the top-1 and top-2 values of 15.2% and 12.9%, while DDCM yields 18.7% and 15.5%. Therefore, the presented model achieves higher classification accuracy than DDCM in terms of top-1 and top-2 on CUAVE and SNAE2.

Furthermore, the influence of the parameter σ that is used to control the decay velocity of the activating rate ρ with regard to the layer l is verified. Tables VI–IX present the average

TABLE VIII
TOP-1 WITH DIFFERENT σ ON SNAE2

Layers/σ	3	5	10	15	20
2	51%	41.1%	20.2%	19.9%	19.9%
3	42.1%	31.6%	17.3%	16.3%	16.1%
4	39.8%	27.2%	15.8%	15.2%	15.2%
5	49.5%	28.7%	20.1%	19.4%	19.2%

TABLE IX
TOP-2 WITH DIFFERENT σ ON SNAE2

Layers/σ	3	5	10	15	20
2	46.4%	36%	18.4%	17%	16.8%
3	39.1%	27.2%	15.2%	13.4%	13.4%
4	36.6%	22.1%	13.7%	12.9%	12.8%
5	44.5%	26.1%	17.1%	16.6%	16.5%

TABLE X
TOP-1 ERROR RATES ON CUAVE

Model/Layers	2	3	4	5
ADDCM-UN	20.8%	17.4%	16.2%	20.2%
ADDCM-SU	16.4%	12.5%	10.7%	16.5%
ADDCM-DDC	17.2%	13.1%	11.5%	17.4%

classification results of the presented model with different σ in terms of top-1 and top-2 on CUAVE and SNAE2.

From Tables VI–IX, the classification accuracy is very low when $\sigma < 10$ since most of connections in the higher layers are disconnected. When $\sigma \geq 10$, the classification accuracy changes significantly with σ varying. Specially, top-2 keeps the same value of 7.5% when $\sigma \geq 10$ on CUAVE. Such results demonstrate that some of the connections in the fully connected layers are redundant.

B. Performance Evaluation of ADDCM Based on Data-Drive Crowdsourcing (ADDCM-DDC)

Now, the performance of the ADDCM-DDC is evaluated on CUAVE and SNAE2. The unsupervised pre-training on ADDCM is performed using the training set to get the initial parameters, denoted by ADDCM-UN. Afterwards, the proposed outsourcing sample selection algorithm is used to choose 50% samples from the training set to be labeled by the annotators and then the ADDCM-DDC model is used to fine-tune the parameters, denoted by ADDCM-DDC. Moreover, the supervised fine-tuning on ADDCM is performed using the ground-truth labels in the training set to get final parameters, denoted by ADDCM-SU. Finally, ADDCM-UN, ADDCM-DDC, and ADDCM-SU are used to classify CUAVE and SNAE2, respectively. The results are reported in Tables X–XIII.

From Tables X–XIII, the worst results are generally produced by ADDCM-UN since the parameters are not fine-tuned. ADDCM-SU achieves the smallest top-1 and top-2 values since the ground-truth labeled data are used to fine-tune the parameters. Such two observations indicate that the labeled samples are very important to the classification accuracy of

TABLE XI
TOP-2 ERROR RATES ON CUAVE

Model/Layers	2	3	4	5
ADDCM-UN	14.4%	12.2%	11.5%	13%
ADDCM-SU	10.5%	7.6%	7.5%	9.7%
ADDCM-DDC	12%	9.6%	9%	11.6%

TABLE XII
TOP-1 ERROR RATES ON SNAE2

Model/Layers	2	3	4	5
ADDCM-UN	26.1%	21.4%	19.5%	23.2%
ADDCM-SU	19.9%	16.3%	15.2%	19.4%
ADDCM-DDC	18.8%	15.9%	15.4%	11.6%

TABLE XIII
TOP-2 ERROR RATES ON SNAE2

Model/Layers	2	3	4	5
ADDCM-UN	21.2%	17.4%	17.1%	20%
ADDCM-SU	17%	13.4%	12.9%	16.6%
ADDCM-DDC	17.8%	15%	14.4%	20.2%

TABLE XIV
TOP-1 ERROR RATES ON CUAVE

Model	2	3	4	5
ADDCM-DDC	17.2%	13.1%	11.5%	17.4%
ADDCM-RS	19.1%	16.5%	17.8%	18.4%
ADDCM-EM	18.6%	15.7%	14.7%	17.6%
ADDCM-MV	18.2%	14.6%	15.2%	17.8%
ADDCM-HP	21.4%	16.4%	17.2%	18.5%

DCMs. Finally, ADDCM-DDC obtains the slightly lower classification accuracy than ADDCM-SU while it yields the significantly higher top-1 and top-2 values than ADDCM-UN. For example, ADDCM-DDC yields top-1 value of 15.4, while ADDCM-SU and ADDCM-UN produce 15.2 and 19.5, respectively, when they have four hidden layers on SNAE2. Such results demonstrate that the proposed scheme is effective to train the parameters by the DDC.

In order to further validate the performance of the presented model, the random selection strategy is substituted for the proposed outsourcing samples section algorithm, denoted by ADDCM-RS. Furthermore, the proposed answer aggregation scheme is replaced with three representative techniques, i.e., MV, HP, and EM, denoted by ADDCM-MV, ADDCM-HP, and ADDCM-EM, respectively. In this experiment, the samples with the least entropy are selected as the trapping questions of the HP technique. The results are presented in Tables XIV–XVII.

Tables XIV–XVII show that ADDCM-DDC produces the higher classification accuracy in terms of top-1 and top-2 than ADDCM-RS on CUAVE and SNAE2, validating the effectiveness of the proposed outsourcing samples selection algorithm. Futhermore, ADDCM-DDC outperforms ADDCM-EM, ADDCM-MV, and ADDCM-HP since ADDCM-DDC produces

TABLE XV
TOP-2 ERROR RATES ON CUAVE

Model	2	3	4	5
ADDCM-DDC	12%	9.6%	9%	11.6%
ADDCM-RS	14.8%	12%	0.13.8%	12.6%
ADDCM-EM	13.1%	9.8%	9.8%	12.1%
ADDCM-MV	13.4%	9.8%	10.2%	12.4%
ADDCM-HP	14.3%	12%	12.8%	12.9%

TABLE XVI
TOP-1 ERROR RATES ON SNAE2

Model	2	3	4	5
ADDCM-DDC	18.8%	15.9%	15.4%	21.6%
ADDCM-RS	22.4%	19.2%	18.5%	24.6%
ADDCM-EM	21.6%	17.6%	17.8%	23%
ADDCM-MV	19.8%	18.1%	16.6%	22.4%
ADDCM-HP	22.2%	19%	17.2%	23.2%

TABLE XVII
TOP-2 ERROR RATES ON SNAE2

Model	2	3	4	5
ADDCM-DDC	17.8%	15%	14.4%	20.2%
ADDCM-RS	20.2%	17.4%	17%	22%
ADDCM-EM	18.5%	15.9%	15.7%	21.6%
ADDCM-MV	18.6%	16.8%	15%	21.8%
ADDCM-HP	20.4%	17.3%	16.1%	22%

smaller top-1 and top-2 values than the other three models. For example, when they have four hidden layers, ADDCM-DDC produces the top-2 value of 9%, while ADDCM-EM, ADDCM-EM, and ADDCM-HP produce 9.8%, 1.2%, and 12.8%, respectively, on CUAVE. Such results demonstrate that the presented answer aggregation scheme performs better than other answer aggregation techniques.

VII. CONCLUSION

In this paper, an ADDCM with crowdsourcing to the cloud computing is presented for big data feature learning in industrial IoT. One advantage of the presented model is to design an adaptive distribution function to set the dropout rate of each hidden layer for preventing overfitting. Another advantage is the combination of the crowdsourcing technique and the parameter training, which can address the problem of the lack of available training samples for training parameters in the DCM. Experimental results demonstrated that the proposed model could avoid overfitting effectively and provide the labeled samples for training the parameters of the DCM, which has potential for big data feature learning in industrial IoT.

Generally, the classification results of deep learning models are affected significantly by the initialization in most cases. However, the optimal techniques for initialization were not considered in the presented scheme. To evaluate the robustness of the presented model, each model was trained for five times in experiments and the average classification accuracy was used to validated the effectiveness of the presented model. In future

work, some optimal techniques for initialization will be investigated to apply for the presented model, which is expected to further improve the performance of the presented model.

REFERENCES

[1] L. D. Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Trans. Ind. Informat.*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014.

[2] L. Yi, X. Deng, M. Wang, D. Ding, and Y. Wang, "Localized confident information coverage hole detection in internet of things for radioactive pollution monitoring," *IEEE Access*, vol. 5, pp. 18665–18674, 2017.

[3] M. Z. A. Bhuiyan, G. Wang, J. Wu, and J. Cao, "Dependable structural health monitoring using wireless sensor networks," *IEEE Trans. Dependable Secure Syst.*, vol. 14, no. 4, pp. 363–376, Jul./Aug. 2017.

[4] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "PPHOPCM: Privacy-preserving high-order possibilistic c-means algorithm for heterogeneous data fuzzy clustering," *IEEE Trans. Big Data*, 2017, doi: 10.1109/TBDATA.2017.2701816.

[5] L. Zhao, Z. Chen, Y. Hu, G. Min, and Z. Jiang, "Distributed feature selection for efficient economic big data analysis," *IEEE Trans. Big Data*, 2016, doi: 10.1109/TBDATA.2016.2601934.

[6] Q. Zhang, L. T. Yang, Z. Chen, P. Li, and M. J. Deen, "Privacy-preserving double-projection deep computation model with crowdsourcing on cloud for big data feature learning," *IEEE Internet Things J.*, 2017, doi: 10.1109/JIOT.2017.2732735.

[7] H. Li, K. Ota, M. Dong, A. Vasilakos, and K. Nagano, "Multimedia processing pricing strategy in GPU-accelerated cloud computing," *IEEE Trans. Cloud Comput.*, 2017, doi: 10.1109/TCC.2017.2672554.

[8] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A survey on deep learning for big data," *Inf. Fusion*, vol. 42, pp. 146–157, 2018.

[9] Q. Zhang, L. T. Yang, and Z. Chen, "Deep computation model for unsupervised feature learning on big data," *IEEE Trans. Services Comput.*, vol. 9, no. 1, pp. 161–171, Jan./Feb. 2016.

[10] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and U. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," arXiv preprint arXiv:1207.0580, 2012. [Online]. Available: https://arxiv.org/abs/1207.0580

[11] H. Garcia-Molina, M. Joglekar, A. Marcus, A. Parameswaran, and V. Verroios, "Challenges in data crowdsourcing," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 4, pp. 901–911, Apr. 2016.

[12] V. C. Raykar *et al.*, "Learning from crowds," *J. Mach. Learn. Res.*, vol. 11, pp. 1297–1322, 2010.

[13] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "An improved deep computation model based on canonical polyadic decomposition," *IEEE Trans. Syst., Man, Cybern.: Syst.*, 2017, doi: 10.1109/TSMC.2017.2701797.

[14] Q. Zhang, M. Lin, L. T. Yang, Z. Chen, and P. Li, "Energy-efficient scheduling for real-time systems based on deep Q-learning model," *IEEE Trans. Sustain. Comput.*, 2017, doi: 10.1109/TSUSC.2017.2743704.

[15] L. I. Kuncheva, "Limits on the majority vote accuracy in classifier fusion," *Pattern Anal. Appl.*, vol. 6, no. 1, pp. 22–31, 2013.

[16] K. Lee, J. Caverlee, and S. Webb, "The social honeypot project: Protecting online communities from spammers," in *Proc. Int. Conf. World Wide Web*, 2010, pp. 1139–1140.

[17] F. K. Khattak and A. Salleb-Aouissi, "Quality control of crowd labeling through expert evaluation" in *Proc. NIPS Workshop Comput. Soc. Sci. Wisdom Crowds*, 2011, pp. 1–5.

[18] Y. Yan, R. Rosales, G. Fung, and J. G. Dy, "Active Learning from Crowds," in *Proc. Int. Conf. Mach. Learn.*, 2011, pp. 1161–1168.

[19] J. Whitehill, T. 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.

[20] D. R. Karger, S. Oh, and D. Shah, "Iterative learning for reliable crowdsourcing systems," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 1953–1961.

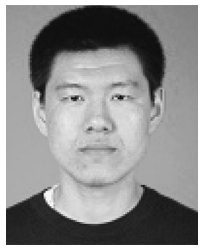
[21] D. Dang, Y. Liu, X. Zhang, and S. Huang, "A crowdsourcing worker quality evaluation algorithm on mapreduce for gig data applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 7, pp. 1879–1888, Jul. 2016.

[22] N. Quoc, V. Hung, N. T. Tam, L. N. Tran, and K. Aberer, "An evaluation of aggregation techniques in crowdsourcing," in *Proc. Int. Conf. Web Inf. Syst. Eng.*, 2013, pp. 1–15.

[23] W. M. Fisher, G. R. Doddington, and K. M. Goudie-Marshall, "The Darpa speech recognition research database: Specifications and status," in *Proc. DARPA Workshop Speech Recog.*, 1986, pp. 93–99.

[24] E. Patterson, S. Gurbuz, Z. Tufekci, and J. Gowdy, "Cuave: A new audio-visual database for multimodal human-computer interface research," in *Proc. 2002 IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2002, pp. 2017–2020.

[25] A. Krizhevsky, L. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.



Qingchen Zhang received the B.Eng. degree in computer science from Southwest University, Chongqing, China, in 2009, the Ph.D. degree in software engineering from Dalian University of Technology, Dalian, China, in 2015.

He is currently a Postdoc Fellow with St. Francis Xavier University, Antigonish, NS, Canada. His research interests include big data and deep learning.

Laurence Tianruo Yang received the B.Eng. degree in computer science from Tsinghua University, Beijing, China, in 1992, and the Ph.D. degree in computer science from the University of Victoria, Victoria, BC, Canada, in 2006.

He is currently a Professor with the University of Electronic Science and Technology of China, Chengdu, China, and St. Francis Xavier University, Antigonish, NS, Canada. His research has been supported by the National Sciences and Engineering Research Council, and the Canada Foundation for Innovation. His research interests include parallel and distributed computing, embedded and ubiquitous/pervasive computing, big data, and cyber-physical-social systems.

Zhikui Chen received the B.Eng. degree in mathematics from Chongqing Normal University, Chongqing, China, in 1990, and the Ph.D. degree in solid mechanics from Chongqing University, Chongqing, China, in 1998.

He is currently a Professor with Dalian University of Technology, Dalian, China. His research interests include internet of things and big data processing.

Peng Li received the B.Eng. degree in electronic and information engineering from Dezhou University, Dezhou, China, in 2012. He is currently working toward the Ph.D. degree with Dalian University of Technology, Dalian, China.

His research interests include deep learning and big data.

Fanyu Bu received the Bachelor's degree in computer science from Inner Mongolia Agricultural University, Hohhot, China, in 2003, and the Master's degree in computer application from Inner Mongolia University, Hohhot, China, in 2009.

His research interests include cyber-physical-social systems and big data.