IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 2, 2020, accepted November 22, 2020, date of publication December 7, 2020,
date of current version December 16, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3042838

Imprecise Deep Forest for Partial Label Learning

JIE GAO, WEIPING LIN, KUNHONG LIU, QINGQI HONG, GUANGYI LIN, AND BEIZHAN WANG
School of Informatics, Xiamen University, Xiamen 361005, China

Corresponding author: Qingqi Hong (hongqq@xmu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772023 and Grant 61502402, in part by the Natural Science Foundation of Fujian Province of China under Grant 2020J01006, in part by the Fundamental Research Funds for the Central Universities under Grant 20720180073, and in part by the National Key Research and Development Program of China under Grant 2019QY1803.

ABSTRACT In partial label (PL) learning, each instance corresponds to a set of candidate labels, among which only one is valid. The objective of PL learning is to obtain a multi-class classifier from the training instances. Because the true label of a PL training instance is hidden in the candidate label set and inaccessible to the learning algorithm, the training process of the classifier is significantly challenging. This study proposes a novel deep learning method for PL learning based on the improved error-correcting output codes (ECOC) algorithm and deep forest (DF) framework. For the ECOC algorithm, we extract the prior knowledge of the candidate label sets from the PL training set to optimize the generation of its coding matrix, where different binary training sets can be derived from the PL training set based on the dichotomy corresponding to each column code. For the DF framework, this improved ECOC algorithm is embedded as a unit in its cascade structure; moreover, an imprecise evaluation method is designed to determine the growth of the cascade of the DF. The effectiveness of the proposed method is verified by conducting several experiments on artificial and real-world PL datasets.

INDEX TERMS Deep forest, error-correcting output codes, partial label learning.

I. INTRODUCTION

Partial label (PL) learning is a type of weakly supervised learning, where each PL training instance is associated with a set of candidate labels, among which only one is valid [1], [2]. We can explain the application scenario of PL learning using a simple example: In news images that contain several people, each face can be considered a PL instance, and a set of names extracted from the corresponding news content can be considered a set of candidate labels for each instance. In this case, the correspondence between each face and its true name is ambiguous. The objective of PL learning is to obtain a multi-class classifier from the PL training set to predict unseen instances [3].

From the perspective of data acquisition, fully labeled data are expensive even in today's era of big data. They are time-consuming to collect and even need to be annotated by domain experts. However, acquiring PL data is relatively easy because of imprecise annotations. PL data represent a good trade-off between fully labeled data and unlabeled data [4]. In the machine learning community, more attention has been paid on using PL data to supplement fully labeled data in existing learning frameworks. In recent years, PL learning technologies have been widely used in many real-world domains such as text mining [5], image classification [6], and ecological informatics [7].

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

Mathematically, we suppose that $X = \mathrm{R}^{\mathrm{d}}$ is the d-dimensional instance space, and $Y = \{y_q | 1 \leq q \leq \mathrm{Q}\}$ is the label space with Q class labels. Let $D = \{(\mathrm{x}_i, S_i) | 1 \leq i \leq \mathrm{m}\}$ be the training set consisting of m PL training instances, where $\mathrm{x}_i \in X$ is a d-dimensional feature vector, and $S_i \subseteq Y$ is the set of candidate labels associated with x_i. The PL learning task is to obtain a classifier f: $X \rightarrow Y$ from D. In PL learning, the true label t_i of x_i is hidden in its candidate label set (i.e., $t_i \in S_i$). Because the true label of a PL training instance is inaccessible to the learning algorithm, training the classifier is difficult. Moreover, false-positive labels occur alongside the true label within the candidate label set (i.e., $S_i t_i$) [8]. The greater the number of false-positive labels in the candidate label set, the greater the difficulty in finding the true label for the PL training instance, which further increases the training difficulty.

In most existing PL learning algorithms, commonly used supervised learning classification techniques are modified to fit PL data. For the maximum likelihood technique, the likelihood estimation of each PL training instance is defined over its candidate label set instead of the true label [9]. For the K-nearest neighbor technique, weighted voting is implemented on the union of the candidate label sets of the neighboring instances to predict unknown instances (PL-KNN [10] for short). For the maximum margin technique, the classification margin over each PL training instance is defined as the difference in the modeling outputs between the candidate and non-candidate labels (PL-SVM [4] for short).

Unlike the aforementioned methods, other methods modify the PL data to adapt to existing learning technologies. For the convex optimization technique, PL training instances perform feature mapping via an improved convex loss optimization, which transforms the PL learning problem into a binary classification problem in the mapping space (CLPL [3] for short). For the error-correcting output codes (ECOC) technique, a set of binary training sets are derived from the PL training set based on a randomly generated coding matrix (PL-ECOC [8] for short). It converts the PL learning problem into a combination of multiple binary classification problems. For the multi-output regression technique, Zhang *et al.* [11] proposed a feature-aware disambiguation strategy that utilizes the manifold structure of the feature space to generate normalized labeling confidences over candidate label sets, and then the predictive model is learned by performing regularized multi-output regression over the generated labeling confidences (PL-LEAF for short). Xu *et al.* [12] proposed a learning scheme from partial label instances via label enhancement [13]. The generalized label distributions were recovered using the topological information of the feature space, and then a predictive model was learned by fitting a regularized multi-output regression with the generalized label distributions (PL-LE for short). Both these approaches explore potentially useful information from the feature space to promote the disambiguation process of the partial labels. To the best of our knowledge, so far, no authors have applied deep learning techniques to solve PL learning tasks. Because of the outstanding performance of deep learning in various classification tasks, it is meaningful to explore the application of deep learning techniques in the field of PL learning.

In this article, we propose an imprecise deep forest for PL learning (PL-DF). Deep forest (DF) [14], [15] is a multi-layer cascade structure that uses a random forest (RF) [16] as a unit. It shows excellent performance in the classification task of supervised learning and can be considered an alternative to deep neural networks (DNNs) [17]. Our main idea is to endow each unit in the DF the ability to process PL instances so that the entire DF framework is suitable for PL learning. The main contributions of this study are as follows: (1) We propose an improved ECOC algorithm for PL learning, which uses the prior knowledge of the frequency counting of candidate label sets to optimize the generation of the ECOC coding matrix; (2) We perform a cascaded ECOC framework for PL learning by embedding the above ECOC algorithm as a unit into the DF. The original DF framework relies on a greedy training mechanism to satisfy the growth of the cascade in the context of supervised learning. However, this original mechanism does not adapt to the case of PL learning. To solve this problem, an imprecise evaluation algorithm is designed to determine the convergence of the cascade.

Although the proposed method is unrelated to forest-style or tree-style classifiers, we used the DF framework as the basis, so we named the proposed method as PL-DF. The rest of this article is organized as follows: Related studies are presented in Section II. The construction of the PL-DF is described in Section III. The experimental results and corresponding discussions are presented in Section IV. Finally, concluding remarks are provided in Section V.

II. BACKGROUND

A. OVERVIEW OF ERROR-CORRECTING OUTPUT CODES (ECOC)

The ECOC algorithm is a representative example of ensemble learning, which can solve multi-classification problems very well under the framework of supervised learning. It was originally proposed by Dietterich and Bakiri [18] in 1995, with a divide-and-conquer mechanism to decompose a multi-classification problem into a set of binary classification problems, and then obtain the final result by integrating the predicted results of all the binary classifications [19]. So far, many novel ECOC algorithms have been proposed based on different principles; however, there is still no universal rule for designing an optimal ECOC coding matrix [20], [21].

An effective coding matrix would be beneficial for the ECOC algorithm. Common coding strategies fall into two categories [22]: (1) Data-independent category, including One-VS-All (OVA), One-VS-One (OVO) [23], and ordinal ECOC [18], where the coding matrices are predefined based on a simple partition of the label space, and without considering the relationship between the data features and data distribution; (2) Data-dependent category, such as D-ECOC [24] and ECOC-ONE [25], where the coding matrixes are generated based on the data distribution; ECOC-MDC [19] uses data complexity [26] to guide the generation of the coding matrix. In contrast, the data-dependent ECOC algorithms use the inherent information of the data to guide the generation of the coding matrices, which are intuitively better suited to handle difficult multi-classification tasks. Unfortunately, all the above ECOC algorithms are proposed under the framework of supervised learning and cannot be directly applied to PL learning.

B. PL-ECOC ALGORITHM

Recently, the PL-ECOC algorithm [8] has been specially designed to deal with the multi-classification problem of PL instances under the framework of weakly supervised learning. The coding matrix of PL-ECOC is randomly generated, where each column of the coding matrix corresponds to a dichotomy of the training instances, dividing the label space into a positive group and a negative group. A binary training set is obtained, where each PL training instance is used as a positive instance or a negative instance only if its candidate label set entirely falls into a positive group or a negative group. The PL-ECOC algorithm can be divided into two phases: encoding and decoding. In the encoding phase, PL-ECOC randomly generates a coding matrix M:

$$M \in \{+1, -1\}^{QxL}, \quad 1 \leq q \leq Q,\ 1 \leq l \leq L, \tag{1}$$

where Q is the number of labels in the label space, and L is the number of base classifiers in PL-ECOC. The q-th row of the coding matrix can be represented as $M(q,:)$, corresponding to L-bits codeword for the class label $\mathrm{y_q}$. The l-th column of the coding matrix corresponds to a Q-bits column code:

$$M\,(:, l) = v = [v_1, v_2, v_3, \ldots, v_{\mathrm{Q}}]^{\mathrm{T}} \in \{+1, -1\}^{\mathrm{Q}} \tag{2}$$

Eqs. (3) and (4) dichotomize the label space Y into a positive group (Y_v^+) and a negative group (Y_v^-):

$$Y_v^+ = \{y_q | v_q = +1, 1 \leq q \leq Q\}, \tag{3}$$
$$Y_v^- = \{y_q | v_q = -1, 1 \leq q \leq Q\}. \tag{4}$$

Considering the training of a base classifier, the training set $D = \{(\mathrm{x}_i, S_i) | 1 \leq i \leq \mathrm{m}\}$ can be transformed into a binary training set B_v, and the transformation process should satisfy the following conditions:

$$B_v = \{B_v^+ \cup B_v^-\}, \tag{5}$$
$$B_v^+ = \{(x_i, +1) | S_i \subseteq Y_v^+, 1 \leq i \leq \mathrm{m}\}, \tag{6}$$
$$B_v^- = \{(x_i, -1) | S_i \subseteq Y_v^-, 1 \leq i \leq \mathrm{m}\}. \tag{7}$$

From Eqs. (5) to (7), any PL training instance x_i is used as a positive instance or a negative instance only if its candidate label set S_i entirely falls into Y_v^+ or Y_v^-. Otherwise, x_i will be discarded, i.e., x_i will not participate in the training of the corresponding base classifier. To avoid the case where the binary training set B_v has few instances, a constraint condition (τ) is set in PL-ECOC to control the minimum size of B_v. In other words, B_v is used to train a binary base classifier (h_l) only if $\tau |B_v| >$.

In the decoding phase, given an unknown instance x^*, the output of PL-ECOC is an L-bits codeword obtained by concatenating the prediction results of the L base classifiers:

$$h\left(x^*\right) = \left[h_1\left(x^*\right), h_2\left(x^*\right), h_3\left(x^*\right), \ldots, h_L\left(x^*\right)\right]. \tag{8}$$

The label whose codeword $M\ (q, :)$ is closest to $h\ (x^*)$ will be selected as the final prediction on x^*:

$$f\left(x^*\right) = \arg\min_{y_q (1 \leq q \leq Q)} dist(h\left(x^*\right), M\ (q, :)), \tag{9}$$

where $dist(*, *)$ represents a distance metric such as the Hamming distance [18], Euclidean distance [27], or loss-based distance [28]. Numerical experiments [8] have indicated that the PL-ECOC either outperforms or is comparable with many well-known PL learning approaches. However, the main drawback of PL-ECOC is that the randomly generated coding matrix may destabilize the algorithm.

C. DEEP FOREST

Recently, the deep forest (DF) [14] has been proposed as an alternative to DNNs. It deals with classification tasks under the supervised learning framework. The main part of the DF method is a multi-layer cascade structure, which is similar to that of DNNs. Each layer in the DF is the ensemble of a set of random forests (RFs). Each RF can be considered a unit of the DF, just like a neuron in DNNs. The DF concatenates the original input vector and the output of the previous layer as the input at the next layer. Representation learning in the DF relies on layer-by-layer data processing and in-model data transformation. Moreover, the number of layers can be adaptively determined through a greedy training mechanism.

Compared with DNNs that require significant effort for tuning the hyper-parameters and large-scale training data, DF has better characteristics including simple training, few hyper-parameters, and reasonable performance on medium and small-sized datasets. Zhou and Feng [15] showed that the performance of the DF method is highly competitive with those of DNNs. Utkin *et al.* [29] believed that the DF is not only an excellent classification algorithm in the deep learning field but also a novel ensemble learning framework. The DF can be applied in diverse application scenarios by endowing the corresponding function to its units [30]–[32].

III. PROPOSED METHOD

A. EXTRACTING PRIOR KNOWLEDGE

Let $X = \{\mathrm{x}_i | 1 \leq i \leq \mathrm{m}\}$ denote an instance space; $S = \{\mathrm{S}_i | 1 \leq i \leq \mathrm{m}\}$ denotes all candidate label sets; $Y = \{y_1, y_2, \ldots, y_Q\}$ denotes the label space, where $\mathrm{S}_i \subseteq Y$. In Algorithm 1, we aim to extract the prior knowledge from S. First, we remove the repeating items from S to ensure that all the remaining items are unique. We then sort the remaining items in descending order in terms of their occurrence frequency.

We explain Algorithm 1 with a simple example. Suppose the label space Y = {1,2,3,4,5} and all candidate label sets S = {(1,2,3,4), (1,2,3), (1,2,3), (3,4), (3,5)}, the frequency counting of each unique term is:

$$\begin{aligned}
\text{The occurrence frequency of (1,2,3)} &= 3,\\
\text{The occurrence frequency of (3,4)} &= 2,\\
\text{The occurrence frequency of (1,2,3,4)} &= 1,\\
\text{The occurrence frequency of (3,5)} &= 1.
\end{aligned} \tag{10}$$

S' = {(1, 2, 3), (3, 4), (1, 2, 3, 4), (3, 5)} is the set of unique items, sorted in descending order in terms of the corresponding frequency counting. In other words, S' is the prior knowledge regarding the distribution of the candidate label sets from the PL training data D.

B. OPTIMIZING THE COLUMN OF CODING MATRIX

An example is shown to explain the optimization of the coding matrix using the prior knowledge of the candidate label sets (See Figure 1). Given that one column of the coding matrix of the ECOC is $M(:, l) = \{+1, +1, -1, -1, -1\}^{\mathrm{T}}$,

Algorithm 1 Extracting the Prior Knowledge From the PL Training Set

```
Input:
S = {S_i | 1 ≤ i ≤ m}: all candidate label sets in the PL training
    set, m is the number of training instances.
Output:
S': all candidate label sets in S' are unique and descending
sorted.
Algorithm process:
S' = [ ]
Frequency_list = [ ]
FOR each $S_i \in S$ DO
  frequency = 0
  FOR each $S_j \in S$ DO
    IF $S_i \subseteq S_j$ THEN
      frequency ++
    END IF
  END FOR
  IF S'.count ($S_i$) = 0 THEN
    S'.append($S_i$)
    Frequency_list.append(frequency)
  END IF
END FOR
Descending sort S' based on the corresponding frequency
RETURN S'
```

Algorithm 2 Optimizing a Column of the Coding Matrix

```
Input:
$v$ : randomly generated column code, $v = [v_1, v_2, \ldots, v_Q]^T$.
$Y$ : label space, $Y = \{Y^+ \cup Y^-\}$.
  // refer to Eq. (2)
$Y^+$ : positive group.    //refer to Eq. (3)
$Y^-$ : negative group.    //refer to Eq. (4)
$S'$ : set of non-repetitive candidate label sets in descending
order.
  //refer to the output of Algorithm 1
Output:
The optimized column code.
Algorithm process:
Case1: FOR each $S_t \in S'$ DO
           IF $Y^+ \subseteq S_t$ THEN
               $Y^+ = S_t$
               $Y^- = YY^+$
               $S'.remove(S_t)$
               BREAK Loop
           END IF
       END FOR
       $Y^+$ and $Y^-$ are used for adjusting $v$;
Case2: If $S_t$ is not found to optimize $Y^+$, then $Y^-$ is optimized
similarly;
Case3: If neither $Y^+$ nor $Y^-$ is optimized, we keep $v$
unchanged.
```

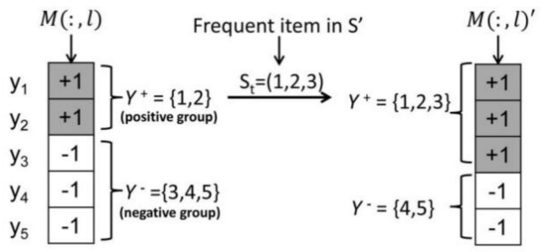


FIGURE 1. Example explaining the optimization of the column code.

the positive and negative groups are $Y^+ = (1, 2)$ and $Y^- = (3, 4, 5)$, respectively. We first optimize the positive group, and it searches for the frequent items containing Y^+ in S' (see Eq. (10)). There are (1,2,3) and (1,2,3,4). Because the frequency of (1,2,3) is higher, it is selected to optimize Y^+.

In the optimization process, the eligible frequent term in S' is used to replace Y^+, i.e., $Y^+ = (1, 2, 3)$ and $Y^- = Y \backslash Y^+ = (4, 5)$. Finally, the adjusted column code becomes $M(:, l)' = \{+1, +1, +1, -1, -1\}^T$, and the used frequent item will be removed from S'. The purpose of optimization is to encourage more PL instances to participate in the training of the base classifier. Moreover, it reduces the class imbalance rate of the training set of the base classifier. The details of the optimization are fully shown in the experimental section. In the above example, if an eligible frequent term in S' is not found to optimize Y^+, we perform a similar operation on Y^-. If Y+ and Y- are not optimized, the column code remains unchanged. The pseudo-code for optimizing the column code is given in Algorithm 2.

C. CONSTRUCTING UNIT ALGORITHM BASED ON ECOC

As explained in Section II, the coding matrix of PL-ECOC is randomly generated without considering the prior knowledge of the data distribution, and this may destabilize the algorithm. Moreover, the dichotomy of each column code excludes a part of the training instances, and although PL-ECOC restrains the minimum size of the training set for each base classifier, it may still lead to insufficient training of the base classifier. In this study, we propose an improved ECOC algorithm that uses the prior knowledge of the candidate label sets to optimize the randomly generated coding matrix, which encourages more PL instances to participate in the training process of the base classifiers. This improves performance and reduces instability. The proposed ECOC algorithm (see Algorithm 3) mainly includes the following steps:

Step 1: The prior knowledge of the candidate label sets is extracted (see Algorithm 1).

Step 2: After randomly generating a column code, we used the extracted information to optimize the positive group or the negative group of the column code (see Algorithm 2), to ensure that a maximum number of PL training instances can participate in the training of the base classifier.

Step 3: We transform the PL training instances into a binary training set based on the optimized column code.

Step 4: We check whether the size of the binary training set is eligible or not. If eligible, the binary training set is used to train the base classifier; otherwise, we will discard the current column code and regenerate one.

Step 5: Repeat step 2 to step 4 until all the column codes are generated; in other words, all the base classifiers are trained.

Step 6: Given the unknown instance x*, we concatenate the predicted results of all the base classifiers as the codeword of x*. The class whose codeword is closest to the codeword of x* is the prediction label.

Because the proposed ECOC algorithm acts as a unit in the DF framework, we call it ECOC-UNIT for short. Moreover, to match ECOC-UNIT with the DF framework, we need to transform the output of ECOC-UNIT in the form of a class probability vector [15]. Using Eq. (9), we can obtain a distance vector $d = [d_1, d_2, d_3, \ldots, d_Q]$, where $1 < q < Q$, and d_q indicates the distance between the codeword of x* and the codeword of the q-th label. The corresponding class probability vector can be expressed as:

$$p = [p_1, p_2, p_3, \ldots, p_Q], \quad s.t. \ \sum\nolimits_{q=1}^{Q} p_q = 1, \tag{11}$$

$$p_q = \left(1 - \frac{d_q}{\sum_{q=1}^{Q} d_q}\right) / (\mathrm{Q} - 1). \tag{12}$$

In Eq. (11), the sum of all elements of the class probability vector is equal to 1. In Eq. (12), each element p_q represents the classification probability of the ECOC-UNIT on the unknown instance x*, where $\left(1 - d_q \middle/ \sum_{q=1}^{Q} d_q\right)$ indicates that, the lower the distance, the greater the classification probability, and (Q-1) is the normalization factor.

D. IMPRECISE DEEP FOREST

In this section, we propose an imprecise DF model for the classification task of PL instances. For simplicity, the term unit refers to ECOC-UNIT. Before introducing this model, we define some notations and indices. Suppose there are R layers in a trained PL-DF (see Fig. 2), where each layer has K units, K is a user-defined parameter that depends on the available computational resource. The components of PL-DF can be represented as follows:

- L_r is the r-th layer, which is the ensemble of multiple units, where r ∈ [1, R].
- F_r represents the cascade of multiple layers, starting from the first layer up to the r-th layer.
- $E_{k,r}$ denotes the k-th unit in the r-th layer, where $k \in [1, K]$.

It is assumed that an unknown instance x* (input feature vector) and three classes should be predicted. Eq. (11) indicates that each unit generates a three-dimensional class probability vector. For K units in a layer (K = 4), a twelve-dimensional (3 × 4) class vector is concatenated with x* as the input at the next layer. The correlation between the layer and the cascade can be mathematically expressed as follows:

$$F_r(x^*) = \begin{cases} L_1(x^*) & r = 1 \\ L_r([x^*, F_{r-1}(x^*)]) & r > 1 \end{cases} \tag{13}$$

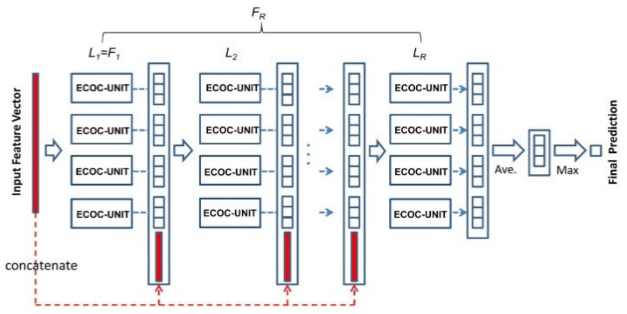


FIGURE 2. Cascade structure of the imprecise deep forest.

where $[x^*, F_{r-1}(x^*)]$ indicates the concatenation of the raw feature vector and the output of the previous cascade. In the last layer of PL-DF, the sum of the corresponding elements of all the class vectors is averaged, where the element with the maximum value corresponds to the predicted label. More specifically, the output of each unit in the last layer (the R-th layer) can be expressed as:

$$E_{k,R}(x^*) = \left[p_1^{k,R}, p_2^{k,R}, \ldots, p_Q^{k,R}\right], \quad 1 \le q \le Q, 1 \le k \le K, \tag{14}$$

where Q is the number of labels, and K is the number of units in a layer. The overall output of the R-th layer (or the R-th cascade) is the concatenation of the outputs of all units in this layer. Finally, the prediction of the entire PL-DF model is expressed as:

$$PL - DF(x^*) = \arg max_{1 \le q \le Q} \sum\nolimits_{k=1}^{K} p_q^{k,R}. \tag{15}$$

The training process of the PL-DF is similar to that of the DF. The original DF is proposed under supervised learning, which divides the dataset into a training set, a validation set, and a testing set. The training set is used to generate each layer of the cascade, the validation set is used to evaluate whether the cascade should grow, and the testing set is used to estimate the final performance of the model. In the training process of each layer, to reduce the risk of overfitting, each unit generates a class probability vector for each training instance using cross-validation, which is then concatenated with the original feature vector to be inputted to the next layer of the cascade. In the training stage, the layer-by-layer processing of the instances is similar to that in the prediction stage, except that the generation process of the class probability vector is slightly different. After expanding a new layer, the performance of the entire cascade will be estimated on the validation set, and the training procedure will terminate if there is no significant performance gain. More specifically, the DF terminates the training only if the performance of F_r is lower than that of F_{r-1}, then retains F_{r-1} as the final model. Thus, the layer number (R) of the DF is adaptively generated through a greedy training mechanism. More details of the training process can be found elsewhere [14], [15].

In PL learning, the PL instances are used to generate the algorithm model, and the fully labeled instances

Algorithm 3 Improved ECOC Algorithm for Processing the Partial Label Data

```
Input:
$D$: partial label training set, $\{(x_i, S_i) \,|1 \le i \le m\}$, where $x_i \in X$, $S_i \subseteq Y$, $Y = \{y_q|1 \le q \le Q\}$.
$L$: number of base classifiers of ECOC.
$M$: coding matrix of ECOC. //Eq. (1)
$M\,(:,l)$ : one column of the coding matrix $M$.
$h$: base classifier.
$\tau$ : minimum size of the training set for each base classifier.
$x^*$ : unknown instance.
Output:
$y^*$ : predicted label for $x^*$, $y^* \in Y$.
Algorithm process:
Extract the prior knowledge from the PL training set D; //Refers to Algorithm 1
$l = 0$;
WHILE $l \ne L$ DO
    Randomly generate a Q-bits column code $v = [v_1, v_2, \ldots, v_Q]^T \in \{+1, -1\}^Q$; //Eq. (2)
    Optimize column code $v$ via the prior knowledge;    //Refers to Algorithm 2
    Dichotomize the label space into positive and negative groups;    //Eqs. (3)-(4)
    Transform the PL training set $D$ into a binary training set $B_v$;    //Eqs. (5)-(7)
    IF $\tau\,|B_v| \ge$ THEN
        $l = l + 1$;
        Set the $l$-th column of the coding matrix M to $v$, $M\,(:,l) = v$;
        Train the $l$-th base classifier based on $B_v$;
    END IF
END WHILE
Obtain a codeword for $x^*$ by concatenating the outputs of the $L$ base classifiers; //Eq. (8)
RETURN $y^* = f(x^*)$. //Eq. (9)
```

(single-label instances) are used to verify the final performance of the model. In principle, the fully labeled data are not used in the training process. In the PL-DF, the dataset is divided into three parts, where the training and validation sets are PL data, and the testing set is fully labeled data. Each part contains 60%, 20%, and 20%, respectively, of the original dataset instances. Whenever the PL-DF extends a new layer, although the training of each unit can be done using PL data (see Algorithm 3), we still need to evaluate the performance of the current cascade to determine whether the training process terminates. However, at this time, each instance in the validation set corresponds to a set of candidate labels; therefore, the validation set cannot accurately evaluate the performance gain of the cascade. To solve this problem, we propose an imprecise evaluation method (see Algorithm 4) that uses PL data as the validation set to imprecisely calculate the performance of each cascade, to realize the application of the DF framework in PL learning.

Two simple examples are used to explain Algorithm 4. Suppose x' is a PL instance in the validation set, the associated candidate label set of x' is S'=(1,2), and three classes should be predicted. The prediction of the current cas- cade on x' is a class probability vector, i.e., $\mathrm{Fr}(x') = [\mathrm{p}_1, \mathrm{p}_2, \mathrm{p}_3]$. We select the top |S'| labels with the highest probability from the vector. Suppose $\mathrm{p}_3 > \mathrm{p}_2 > \mathrm{p}_1$, two labels with the highest probability in the prediction result of $\mathrm{F_r}$ are (3, 2); thus, the coverage rate of the prediction results for the candidate label set is:

$$\frac{|\,(1,2) \cap (3,2)\,|}{|S'|} = \frac{1}{2} = 0.5. \tag{16}$$

Suppose $\mathrm{p}_2 > \mathrm{p}_1 > \mathrm{p}_3$, two labels with the highest probability in the prediction results of $\mathrm{F_r}$ are (2, 1); thus, the coverage rate of the prediction result for the candidate label set is:

$$\frac{|\,(1,2) \cap (2,1)\,|}{|S'|} = \frac{2}{2} = 1. \tag{17}$$

For the above examples, we introduce an adjustable parameter ω to present the threshold of the coverage rate, $\omega \in [0, 1]$. When the calculated coverage rate is greater than the threshold, we consider that this instance is correctly classified by the current cascade, and vice versa. Thus, we can use the PL instances as the validation set to imprecisely evaluate the performance of each cascade. We discuss the effect of this parameter on the algorithm performance in the experimental section.

IV. EXPERIMENTS AND DISCUSSIONS

A. DATASETS AND COMPARISON ALGORITHMS

In the experimental section, we use two types of datasets: controlled UCI datasets and real-world datasets. Tables 1 and 2 list the details of the datasets used. First, we follow the control protocol [3], [9], [33] commonly used in PL learning to

Algorithm 4 An Imprecise Evaluation Method for the Cascade

```
Input:
x': PL instance in the validation set.
S': candidate label set of x'.
F: cascade that needs to be evaluated.
p: prediction of the cascade, it is a class probability vector,
F(x') = p = [p_1,p_2,...,p_Q], where Q is the number of labels.
$\omega$ : adjustable threshold, $\omega \in [0, 1]$.
Output:
CP: coverage rate (CP) of the prediction result for the candidate label set;
and determine if the instance x' is correctly classified by the cascade F.
Algorithm process
p' = descending_argsort(p)
temp=[ ]
FOR i in range (|S'|) DO
    temp.append(p'[i])
END FOR
$CP = \frac{|temp \cap S'|}{|S'|}$
IF CP≥ $\omega$ THEN
    The instance x' is correctly classified by the cascade F;
ELSE
    The instance x' is misclassified by the cascade F;
```

TABLE 1. Controlled UCI datasets.

Datasets	Instances	Features	Classes	Configurations
Glass	214	9	6	
Ecoli	336	7	8	$\alpha = 2$,
Deter	358	23	6	
Vehicle	864	18	4	$\beta \in [0.1, 0.2, \dots, 0.7]$
Abalone	4177	7	27	
Usps	9298	256	10	

TABLE 2. Real-world partial label datasets.

Datasets	Instances	Features	Classes	Avg. Labels
Lost	1122	108	16	2.23
MSRCv2	1758	48	23	3.16
Bird Song	4998	38	13	2.18
Soccer Player	17472	279	171	2.09
Yahoo! News	22991	163	219	1.91

transform multi-class UCI datasets into artificial PL datasets using two controlling parameters α and β, where α controls the number of false-positive labels in the candidate label sets (i.e., $|S_i| = \alpha + 1$), and β controls the proportion of PL instances. As listed in Table 1, we set $\alpha = 2$, and $\beta \in [0.1, 0.2, \dots, 0.7]$, i.e., for each UCI dataset, a total of seven control parameter configurations are used to generate the artificial PL training sets. Suppose $\alpha = 2$ and $\beta = 0.7$, the proportion of PL instances in the artificial dataset accounts for 70%, and the candidate label set of each PL instance is composed of the true label and two false-positive labels randomly selected from the label space, i.e., $|S_i| = 1 + 2 = 3$.

Table 2 lists the real-world PL datasets obtained from different domains used in the experiment, such as Lost [3], Soccer player [6], Yahoo! News [34] from automatic face naming; MSRCv2 [9] from object classification; Bird Song [7] from sound classification. For automatic face naming, a face appearing in the image or video is considered a PL instance, and the names extracted from the corresponding caption and subtitle are used as the candidate labels. For object classification, image segmentation represents a PL instance, whereas objects appearing within the same image are considered candidate labels. For sound classification, every 10 s of birdcall are taken as a PL instance, where bird species that jointly sing are considered a set of candidate labels [8]. Table 2 also provides the average sizes of the candidate label sets in each dataset.

To verify the performance of PL-DF, we select six classical algorithms for comparison: PL-ECOC [8], PL-KNN [10], PL-SVM [4], CLPL [3], PL-LEAF [11], and PL-LE [12]. The default settings in these algorithms are based on the values reported in the corresponding articles. The parameters of PL-DF are divided into unit-related and cascade-related. First, each unit is the ensemble of multiple base classifiers, the number of base classifiers is set to $L = [10^* \log_2(Q)]$, the type of base classifiers is SVM with the radial basis function (RBF) kernel [35], and the minimum size of the training set of each base classifier is set to $\tau = 0.1^*|D|$. In addition, PL-DF is the cascade of multiple layers, where four units are placed in each layer, the number of layers is determined adaptively, and the parameter ω is set to 0.5 to evaluate the performance of each cascade imprecisely. We analyze the effects of these parameters on the model performance in the following sections.

B. EXPERIMENTAL RESULTS

As listed in Table 1, each UCI dataset can be converted to seven artificial datasets with different PL instance proportions based on seven configurations. Each algorithm performs ten five-fold cross-validations on each dataset to obtain the average prediction accuracy. With the increase in β, we plotted the performance change curves of PL-DF and other compared algorithms on all the artificial PL datasets (see Fig. 3). When the red curve is superior to the other curves in space, we consider that PL-DF outperforms the compared algorithms (win). When the red curve overlaps with the other curves in space to a large extent, we consider that the performance of PL-DF is similar to those of the corresponding algorithms (tie). Otherwise, PL-DF is said to be inferior to the algorithms in terms of the performance (loss).

As shown in Fig. 3, we use the rank statistics (win/tie/loss) of the Wilcoxon signed-rank test [36] to compare the classification performance between PL-DF and the other six algorithms (see Table 3). The "Subtotal" column in Table 3 lists the pairwise comparisons between PL-DF and the other algorithms. In most cases, PL-DF exhibits the best performance, whereas in a few cases, its performance is similar to those of the compared algorithms.

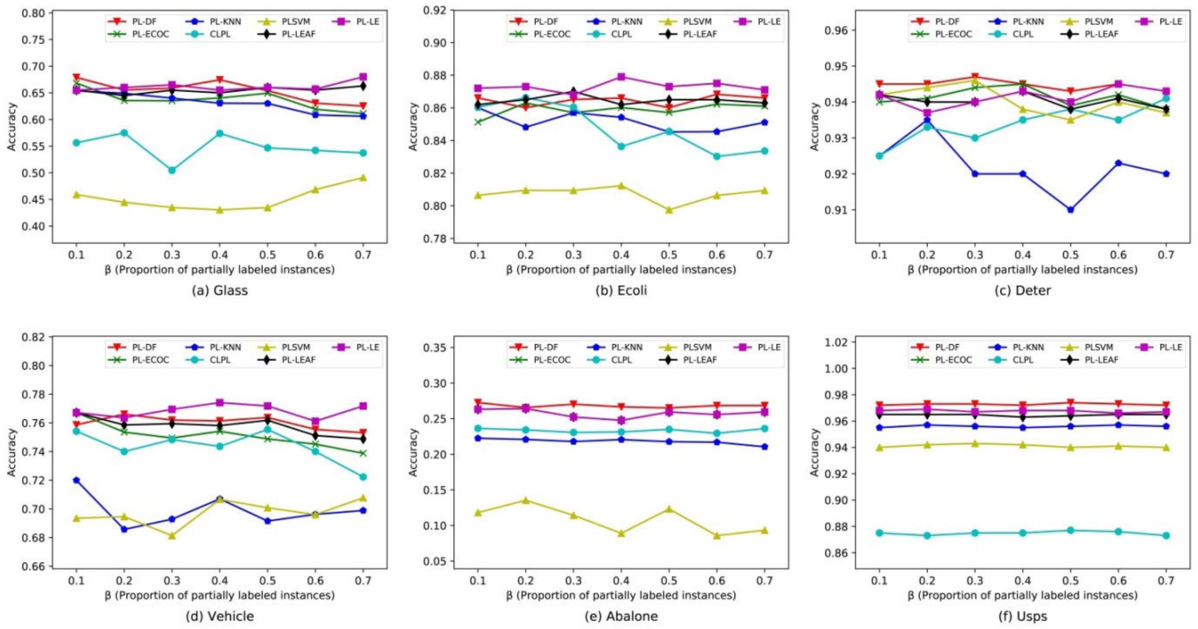


FIGURE 3. Accuracy change trends of all algorithms with increasing β (proportion of partially labeled instances), with $\alpha = 2$ (two false-positive labels).

TABLE 3. Win/Tie/Loss statistics on the classification performance of PL-DF against the compared algorithms on controlled UCI datasets.

	Datasets							
PL-DF against	Glass	Ecoli	Deter	Vehicle	Abalone	Usps	Subtotal	Total
PL-LE	tie	loss	win	loss	win	win	3/1/2	
PL-LEAF	tie	tie	win	win	win	win	4/2/0	
PL-ECOC	tie	tie	win	win	win	win	4/2/0	28/6/2
PL-KNN	tie	win	win	win	win	win	5/1/0	
PL-SVM	win	win	win	win	win	win	6/0/0	
CLPL	win	win	win	win	win	win	6/0/0	

TABLE 4. Average performance rank of all algorithms on real-world partial label datasets.

Datasets	PL-DF	PL-LE	PL-LEAF	PL-ECOC	PL-KNN	PL-SVM	CLPL
Lost	0.675±0.022(4)	0.773±0.043(1)	0.664±0.020(5)	0.653±0.021(6)	0.424±0.033(7)	0.729±0.038(3)	0.742±0.035(2)
MSRCv2	0.499±0.025(1.5)	0.499±0.037(1.5)	0.459±0.013(4.5)	0.459±0.039(4.5)	0.447±0.035(6)	0.475±0.020(3)	0.415±0.039(7)
Bird Song	0.756±0.019(1)	0.730±0.013(2)	0.706±0.012(4)	0.719±0.018(3)	0.650±0.021(6)	0.676±0.035(5)	0.632±0.021(7)
Soccer Player	0.541±0.015(1)	0.536±0.020(2)	0.515±0.004(4)	0.528±0.013(3)	0.497±0.012(5)	0.485±0.012(6)	0.367±0.010(7)
Yahoo! News	0.659±0.008(1)	0.653±0.006(2)	0.597±0.004(5)	0.639±0.009(3)	0.457±0.004(7)	0.610±0.010(4)	0.463±0.008(6)
Average Rank	(1.7)	(1.7)	(4.5)	(3.9)	(6.2)	(4.2)	(5.8)

We report the average accuracy and standard deviation of all the algorithms on the real-world PL datasets (see Table 4). We use the average rank of the Friedman test [37] to verify the performance difference between the algorithms, where the values in brackets represent the performance rank of the algorithms on the same dataset. More specifically, the best-performing algorithm is assigned rank 1, the second-best is assigned rank 2, and so on. In the case of similarities (e.g., for the MSRCv2 dataset in Table 4), the mean ranks are assigned (e.g., $(1+2)/2 = 1.5$). The last row in Table 4 lists the average rank of the classification performance of each algorithm on all datasets. The lower the average rank, the better the average classification performance of the corresponding algorithm on all the datasets. Among the seven algorithms, the average ranks of PL-DF and PL-LE are much higher than those of the other compared algorithms, where the performance of the PL-LE is particularly close to that of the PL-DF on both the artificial and real-world PL datasets.

C. PARAMETER ANALYSIS OF PL-DF

Before beginning this section, we review the parameters involved in the proposed model (see Table 5). From the overall structure of the proposed model, we see that PL-DF is the cascade of R layer, each layer of PL-DF is the ensemble of K units, and each unit is the ensemble of L base classifiers. We first analyze the unit-related parameters, namely L, τ, and h, and then analyze the cascade-related parameters, namely K, ω, and R.

TABLE 5. Review of the parameters of the proposed model.

Symbol	Description
L:	Number of base classifiers in a unit.
:	Minimum number of PL instances for training each base classifier
h:	Type of base classifier in the unit.
K:	Number of units in each layer of PL-DF.
ω:	Threshold combined with the coverage rate to imprecisely
R:	Layer number of PL-DF.

1) PARAMETER ANALYSIS OF THE UNIT

For convenience, we used three real-world PL datasets (Lost, MSRCv2, and Bird Song) for all the parameter sensitivity experiments. In ECOC-UNIT, L not only represents the length of the coding matrix rows but also represents the number of base classifiers. Some ECOC-based studies indicate that $L = [t^{*} \log_2(Q)]$ is close to the optimal [38], [39], where Q is the number of class labels, and t is an adjustable parameter with the step length $\log_2(Q)$. We suppose $t = \{1, 2, 3, \ldots, 15\}$, and observe the variation in the performance of ECOC-UNIT with increasing L. As shown in Fig. 4, the performance curve of ECOC-UNIT exhibits an upward trend when t is in the range of 1–9, and it tends to be stable after $t = 10$. According to the principle of Occam's Razor [40], entities should not be multiplied unnecessarily; therefore, we set $L = [10^{*} \log_2(Q)]$ as the number of base classifiers in the ECOC-UNIT.

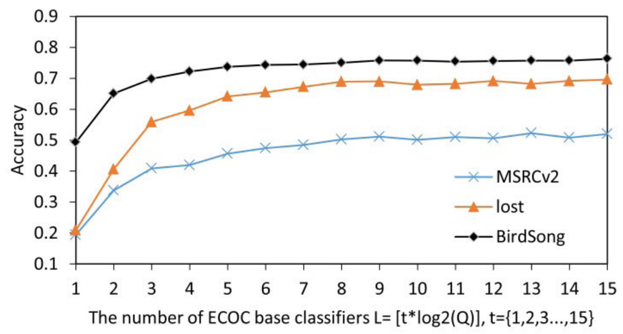


FIGURE 4. Accuracy change trend of ECOC-UNIT with increasing base classifiers (L).

Moreover, the size of τ is set in proportion to the number of PL training instances (see Table 6), $\tau = \varphi * |D|$, $\varphi = \{0.1, 0.2, 0.3, 0.4\}$. From Table 6, we find that when τ increases, the unit performance decreases. An increase in τ indicates a compulsive increase in the minimum number of PL instances used to train a single base classifier. In this case, there may be only a few ECOC column codes that meet the requirements, so the ECOC algorithm repeatedly generates similar column codes. In a good ensemble system, the base classifiers should be as accurate and diverse as possible, so that the combination of these base classifiers can compensate for the prediction error of a single base classifier and improve the performance of the ensemble [41], [42]. When many ECOC column codes are similar, the overall ensemble will lack diversity, resulting in performance degradation. Therefore, we simply set $\tau = 0.1 * |D|$ in ECOC-UNIT.

TABLE 6. Variation in the performance of ECOC-UNIT with an increase in $\tau(\varphi)$.

$\tau = \varphi * \lvert D \rvert$	Lost	MSRCv2	Bird Song
$\varphi = 0.1$	0.661±0.031	0.475±0.028	0.742±0.015
$\varphi = 0.2$	0.645±0.034	0.455±0.026	0.724±0.016
$\varphi = 0.3$	0.641±0.037	0.442±0.035	0.722±0.013
$\varphi = 0.4$	0.638±0.028	0.445±0.037	0.708±0.012

Both ECOC-UNIT and PL-ECOC are modified based on the ECOC framework. The difference between ECOC-UNIT and PL-ECOC is that the former uses the prior knowledge of the label distribution to optimize the generation of the coding matrix, whereas the latter uses a randomly generated coding matrix. In Table 7, we not only list the influence of the choice of the base classifier on the model performance but also list the performance difference between the two models when the same base classifier is used. From Table 7, we find that the performances of the two models are the best when the SVM is selected as the base classifier; when the same base classifier is used, ECOC-UNIT outperforms PL-ECOC.

Moreover, we use a supplementary experiment to fully demonstrate the impact of the coding matrix optimization on the model training process. First, we use a fixed random seed to generate the coding matrix of the PL-ECOC. Taking the Bird Song dataset as an example, we find that its class number is 13, and the column number corresponding to the coding matrix is 38 ($L = 10^{*} \log_2 13 = 38$). We then use the same coding matrix in ECOC-UNIT, and the optimization process had an impact on the six columns in the coding matrix. In other words, the proposed optimization algorithm improves the six base classifiers. Fig. 5 shows the difference between the training instances of these six base classifiers before and after the optimization. Each number on the X axis represents a base classifier, and each base classifier corresponds to two columns, which respectively represent the number of training instances before and after the optimization. The green and blue parts on each column represent the corresponding number of positive and negative instances in a training set. Table 8 lists the statistical data shown in Figure 5. The size of the training set for the first five base classifiers increases after the optimization; the percentage increase ranges from 8.48% to 27.25%; the training instances of the last base classifier is slightly reduced by 5.46%.

Another interesting phenomenon is that the class imbalance rate of the positive and negative instances of each classifier significantly decreases after the optimization. Many studies [43], [44] have shown that the increase in the training instances and the decrease in the class imbalance rate are the key factors for improving the performance of the classifier.

2) PARAMETER ANALYSIS OF THE CASCADE

In terms of the cascade, we focus on three parameters (K, ω, and R). First, from Table 9, we find that the performance of PL-DF gradually improves with the increase in K. Each layer of PL-DF is the ensemble of K units; PL-DF concatenates

TABLE 7. Effects of different base classifiers on model performance under the frameworks of PL-ECOC and ECOC-UNIT. Symbol "●/○" indicates that the performance of ECOC-UNIT is superior/ inferior to PL-ECOC.

Algorithms	Base classifier	Lost	MSRCv2	Bird Song
ECOC-UNIT	SVM (with RBF kernel)	0.661±0.031	0.475±0.028	0.742±0.015
	KNN	0.457±0.025	0.458±0.021	0.699±0.019
	Bayes	0.461±0.026	0.309±0.036	0.421±0.015
	Random Forest	0.465±0.034	0.438±0.021	0.728±0.014
PL-ECOC	SVM (with RBF kernel)	0.653±0.021●	0.459±0.039●	0.719±0.018●
	KNN	0.418±0.037●	0.426±0.027●	0.676±0.022●
	Bayes	0.422±0.038●	0.273±0.031●	0.443±0.021○
	Random Forest	0.427±0.024●	0.446±0.035○	0.703±0.015●

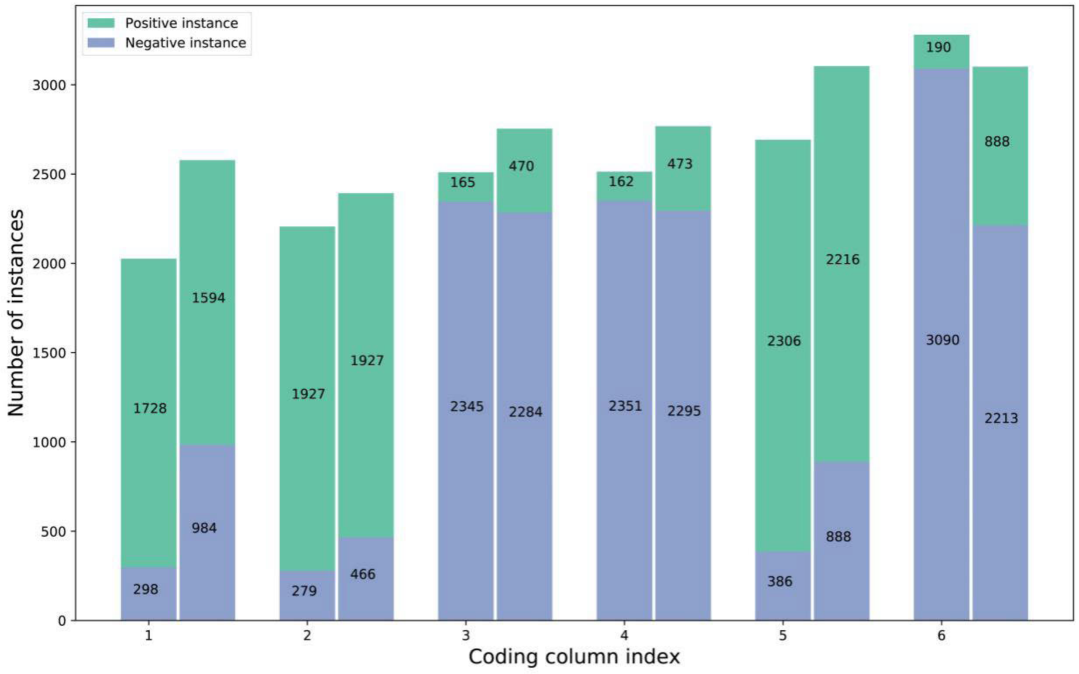


FIGURE 5. Impact of the optimization of coding matrix on the training set of base classifier.

TABLE 8. Impact of the optimization of coding matrix on the training set of base classifier (data statistics based on fig. 5).

Coding column index	1	2	3	4	5	6
Number of instances before optimization	2026	2206	2610	2513	2692	3280
Number of instances after optimization	2578	2393	2754	2768	3104	3101
The percentage increase in quantity	27.25%	8.48%	9.72%	10.15%	15.30%	-5.46%
Class imbalanced rate before optimization	5.79	6.90	14.21	14.51	5.97	16.26
Class imbalanced rate after optimization	1.62	4.13	4.85	4.85	2.49	2.49

the output of K units with a raw feature vector as the input of the next layer (see Fig. 2). On the one hand, this is the re-representation of the raw feature vector. On the other hand, it implies that the classification results of the previous layer guide the classification process of the next layer. When the dimension of the raw feature vector is very high, we can consider increasing the number of units in each layer to prevent the useful guidance information (the output of K units) of the previous layer from being overwhelmed within the high-dimensional feature vector. In the proposed PL-DF model, we place four units in each layer, and increasing the value of K depends on the available computing resources.

Second, Fig. 6 shows that the performance of the PL-DF remains stable with the increase in ω. The combination of ω and coverage rate (see Algorithm 4) allows PL-DF to use PL instances as a validation set, which imprecisely evaluates the performance of each cascade in the training stage of the model. This parameter only standardizes the performance

TABLE 9. Variation in the performance of PL-DF as K increases.

Number of units	Lost	MSRCv2	Bird Song
K = 1	0.614±0.033	0.443±0.029	0.734±0.016
K = 2	0.645±0.030	0.469±0.025	0.748±0.019
K = 3	0.657±0.041	0.482±0.031	0.751±0.018
K = 4	0.675±0.022	0.498±0.025	0.756±0.019

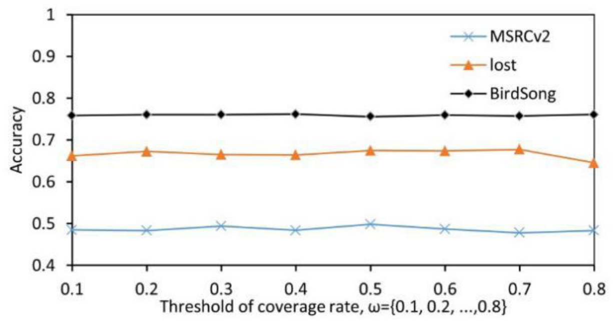


FIGURE 6. Accuracy change trend of PL-DF with increasing ω.

evaluation of each cascade, to determine the convergence of model training. It does not affect the final performance of PL-DF; we simply set this parameter to 0.5.

Third, theoretically, the deeper the cascade (layer), the better the performance of the model. In other words, the re-representation of an unknown instance is more discriminative after being processed layer-by-layer; it can be correctly classified by the output layer of the model more easily. However, the cascade number R is adaptively generated based on a greedy training mechanism with different datasets. Although R is an important parameter, it is not analyzed here.

V. CONCLUSION

In this study, we analyzed the problem of PL learning by developing a novel deep learning method based on the ECOC-UNIT and DF framework. From the unit level of DF, we give the unit the ability to process PL instances by modifying the conventional ECOC algorithm; from the cascade level, we use an imprecise evaluation method to determine the convergence of the cascade of the DF. The effectiveness of the proposed method was verified by conducting several experiments on artificial and real-world PL datasets.

The proposed method has a potential shortcoming in that not all the PL instances participated in the training of the base classifiers of the ECOC. An interesting research direction would be to develop a method to more effectively utilize the excluded PL training instances in the ECOC-based unit [1], [8]. In addition, more prior knowledge of the PL training instances can be used to design the new unit for the DF framework, such as data spatial information [45], data complexity [19], [26], and data splitting performance [46]. In the future, it is also important to explore other deep learning techniques to fit the PL learning problem.

REFERENCES

[1] X. Wu and M.-L. Zhang, "Towards enabling binary decomposition for partial label learning," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 2868–2874.

[2] F. Yu and M.-L. Zhang, "Maximum margin partial label learning," *Mach. Learn.*, vol. 106, no. 4, pp. 573–593, Apr. 2017.

[3] T. Cour, B. Sapp, and B. Taskar, "Learning from partial labels," *J. Mach. Learn. Res.*, vol. 12, pp. 1501–1536, Apr. 2011.

[4] N. Nguyen and R. Caruana, "Classification with partial labels," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2008, pp. 551–559.

[5] L. Jie and F. Orabona, "Learning from candidate labeling sets," in *Proc. 23rd Int. Conf. Neural Inf. Process. Syst.*, vol. 2, 2010, pp. 1504–1512.

[6] Z. Zeng, S. Xiao, K. Jia, T.-H. Chan, S. Gao, D. Xu, and Y. Ma, "Learning by associating ambiguously labeled images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 708–715.

[7] F. Briggs, X. Z. Fern, and R. Raich, "Rank-loss support instance machines for MIML instance annotation," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2012, pp. 534–542.

[8] M.-L. Zhang, F. Yu, and C.-Z. Tang, "Disambiguation-free partial label learning," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 10, pp. 2155–2167, Oct. 2017.

[9] L.-P. Liu and T. G. Dietterich, "A conditional multinomial mixture model for superset label learning," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, vol. 1, 2012, pp. 548–556.

[10] E. Hüllermeier and J. Beringer, "Learning from ambiguously labeled examples," in *Proc. 6th Int. Symp. Intell. Data Anal.*, 2005, pp. 168–179.

[11] M.-L. Zhang, B.-B. Zhou, and X.-Y. Liu, "Partial label learning via feature-aware disambiguation," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Francisco, CA, USA, Aug. 2016, pp. 1335–1344.

[12] N. Xu, J. Lv, and X. Geng, "Partial label learning via label enhancement," in *Proc. 33rd AAAI Conf. Artif. Intell.*, Honolulu, HI, USA, 2019, pp. 5557–5564.

[13] X. Geng, "Label distribution learning," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 7, pp. 1734–1748, Jul. 2016.

[14] Z.-H. Zhou and J. Feng, "Deep forest: Towards an alternative to deep neural networks," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 3553–3559.

[15] Z.-H. Zhou and J. Feng, "Deep forest," *Nat. Sci. Rev.*, vol. 6, no. 1, pp. 74–86, 2019.

[16] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[17] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, Oct. 2019.

[18] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *J. Artif. Intell. Res.*, vol. 2, pp. 263–286, Jan. 1995.

[19] M. Sun, K. Liu, Q. Wu, Q. Hong, B. Wang, and H. Zhang, "A novel ECOC algorithm for multiclass microarray data classification based on data complexity analysis," *Pattern Recognit.*, vol. 90, pp. 346–362, Jun. 2019.

[20] M. Á. Bautista, S. Escalera, X. Baró, and O. Pujol, "On the design of an ECOC-compliant genetic algorithm," *Pattern Recognit.*, vol. 47, no. 2, pp. 865–884, Feb. 2014.

[21] K.-J. Feng, S.-T. Liong, and K.-H. Liu, "The design of variable-length coding matrix for improving error correcting output codes," *Inf. Sci.*, vol. 534, pp. 192–217, Sep. 2020.

[22] X.-L. Zhang, "Heuristic ternary error-correcting output codes via weight optimization and layered clustering-based approach," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 289–301, Feb. 2015.

[23] L. Zhou, Q. Wang, and H. Fujita, "One versus one multi-class classification fusion using optimizing decision directed acyclic graph for predicting listing status of companies," *Inf. Fusion*, vol. 36, pp. 80–89, Jul. 2017.

[24] O. Pujol, P. Radeva, and J. Vitria, "Discriminant ECOC: A heuristic method for application dependent design of error correcting output codes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 6, pp. 1007–1012, Jun. 2006.

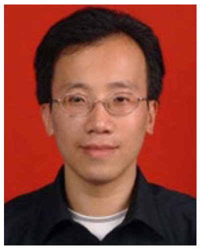
[25] P. Radeva, O. Pujol, and S. Escalera, "ECOC-ONE: A novel coding and decoding strategy," in *Proc. 18th Int. Conf. Pattern Recognit. (ICPR)*, vol. 3, Aug. 2006, pp. 578–581.

[26] J.-R. Cano, "Analysis of data complexity measures for classification," *Expert Syst. Appl.*, vol. 40, no. 12, pp. 4820–4831, Sep. 2013.

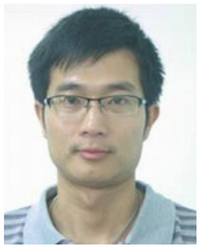
[27] T. E. Schouten and E. L. van den Broek, "Fast exact Euclidean distance (FEED): A new class of adaptable distance transforms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 11, pp. 2159–2172, Nov. 2014.
[28] S. Escalera, O. Pujol, and P. Radeva, "On the decoding process in ternary error-correcting output codes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 120–134, Jan. 2010.
[29] L. V. Utkin, A. A. Meldo, and A. V. Konstantinov, "Deep forest as a framework for a new class of machine-learning models," *Nat. Sci. Rev.*, vol. 6, no. 2, pp. 186–187, Mar. 2019.
[30] L. V. Utkin, M. S. Kovalev, and A. A. Meldo, "A deep forest classifier with weights of class probability distribution subsets," *Knowl.-Based Syst.*, vol. 173, pp. 15–27, Jun. 2019.
[31] L. V. Utkin, "An imprecise deep forest for classification," *Expert Syst. Appl.*, vol. 141, Mar. 2020, Art. no. 112978.
[32] L. V. Utkin and M. A. Ryabinin, "A siamese deep forest," *Knowl.-Based Syst.*, vol. 139, pp. 13–22, Jan. 2018.
[33] Y.-C. Chen, V. M. Patel, R. Chellappa, and P. J. Phillips, "Ambiguously labeled learning using dictionaries," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 12, pp. 2076–2088, Dec. 2014.
[34] M. Guillaumin, J. Verbeek, and C. Schmid, "Multiple instance metric learning from automatically labeled bags of faces," in *Proc. 11th Eur. Conf. Comput. Vis.*, vol. 6311, 2010, pp. 634–647.
[35] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, Apr. 2011.
[36] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bull.*, vol. 1, no. 6, pp. 80–83, 1945.
[37] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.
[38] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: A unifying approach for margin classifiers," *J. Mach. Learn. Res.*, vol. 1, pp. 113–141, Sep. 2001.
[39] O. Pujol, S. Escalera, and P. Radeva, "An incremental node embedding technique for error correcting output codes," *Pattern Recognit.*, vol. 41, no. 2, pp. 713–725, Feb. 2008.
[40] B. K. Natarajan, "Occam's Razor for functions," in *Proc. 6th Annu. Conf. Comput. Learn. Theory (COLT)*, 1993, pp. 370–376.
[41] H. Guo, H. Liu, R. Li, C. Wu, Y. Guo, and M. Xu, "Margin & diversity based ordering ensemble pruning," *Neurocomputing*, vol. 275, pp. 237–246, Jan. 2018.
[42] Q. Dai, R. Ye, and Z. Liu, "Considering diversity and accuracy simultaneously for ensemble pruning," *Appl. Soft Comput.*, vol. 58, pp. 75–91, Sep. 2017.
[43] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *J. Big Data*, vol. 6, no. 1, p. 27, Dec. 2019.
[44] J. Gao, K. Liu, B. Wang, D. Wang, and Q. Hong, "An improved deep forest for alleviating the data imbalance problem," *Soft Comput.*, early access, Aug. 2020.
[45] Y. Zhou and H. Gu, "Geometric mean metric learning for partial label data," *Neurocomputing*, vol. 275, pp. 394–402, Jan. 2018.
[46] J. Yan, Z. Zhang, L. Xie, and Z. Zhu, "A unified framework for decision tree on continuous attributes," *IEEE Access*, vol. 7, pp. 11924–11933, 2019.

JIE GAO received the B.Sc. and M.Sc. degrees from RMIT University, Australia, in 2008 and 2010, respectively. He is currently pursuing the Ph.D. degree in computer science with the School of Informatics, Xiamen University, China. His current research interests include data mining and machine learning.

WEIPING LIN received the bachelor's degree in management from the Dalian University of Technology, China, in 2019. He is currently pursuing the M.S. degree in software engineering with the School of Informatics, Xiamen University. His current research interests include data mining and machine learning.



KUNHONG LIU received the B.Sc. and M.Sc. degrees from Fujian Normal University in 1999 and 2004, respectively, and the Ph.D. degree from the University of Science and Technology of China. He is currently a Professor with the School of Informatics, Xiamen University. His research interests include machine learning with a focus on ensemble learning and evolutionary algorithm.



QINGQI HONG received the Ph.D. degree in computer science from the University of Hull, U.K. He is currently an Associate Professor with the School of Informatics, Xiamen University, China. His current research interests include medical imaging processing, 3-D visualization, 3-D modeling, computer-aided diagnosis and surgery, deep learning, and GPU computing.

GUANGYI LIN received the B.Sc. degree in computer science from Shandong University in 2011 and the M.Sc. degree from Xiamen University, China, in 2015, where he is currently pursuing the Ph.D. degree with the School of Informatics. His current research interests include data mining, machine learning, computer vision, and deep learning techniques.

BEIZHAN WANG received the Ph.D. degree in computer science from Northwestern Polytechnical University, China. He is currently a Professor with the School of Informatics, Xiamen University, China. His current research interests include data warehouse, data mining, and software architecture.