An Effective Approach for Multi-label Classification with Missing Labels

1st Xin Zhang
University of South Carolina
Columbia, United States
xz8@email.sc.edu

2nd Rabab Abdelfattah
University of South Carolina
Columbia, United States
rabab@email.sc.edu

3rd Yuqi Song
University of South Carolina
Columbia, United States
yuqis@email.sc.edu

4th Xiaofeng Wang
University of South Carolina
Columbia, United States
wangxi@cec.sc.edu

Abstract—Compared with multi-class classification, multi-label classification that contains more than one class is more suitable in real life scenarios. Obtaining fully labeled high-quality datasets for multi-label classification problems, however, is very expensive, and sometimes even impossible, with respect to annotation efforts, especially when the label spaces are too large. This motivates the research on partial-label classification, where only a limited number of labels are annotated and the others are missing. To address this problem, we first propose a pseudo-label based approach to reduce the cost of annotation without bringing additional complexity to the existing classification networks. Then we quantitatively study the impact of missing labels on the performance of classifier. Furthermore, by designing a novel loss function, we are able to relax the requirement that each instance must contain at least one positive label, which is commonly used in existing approaches. Through comprehensive experiments on three large-scale multi-label image datasets, i.e., MS-COCO, NUS-WIDE, and Pascal VOC12, we show that our method can handle the imbalance between positive labels and negative labels, while still outperforming existing missing-label learning approaches in most cases, and in some cases even approaches with fully labeled datasets.

Index Terms—Deep learning, multi-label classification, missing label, pseudo label, label imbalance

I. INTRODUCTION

In deep learning, multi-class classification is a common problem where the goal is to classify a set of instances, each associated with a unique class label from a set of disjoint class labels. A generalized version of multi-class problem is multi-label classification [1], which allows the instances to be associated with more than one class. It is more a practical problem in real life because of the intrinsic multi-label property of the physical world [2]: automatic driving always needs to identify which objects are contained in the current scene, such as cars, traffic lights and pedestrians; CT scan can detect a variety of possible lesions; a movie can simultaneously belong to different categories, for instance. Ideally, multi-label classification is a form of supervised learning [3], which requires lots of accurate labels. In practice, however, annotating all labels for each training instance raises a great challenge in multi-label classification, which is time-consuming and even impractical especially in the presence of a large number of categories [4], [5]. Therefore, how to leverage the performance of multi-label classifier and the cost of collecting labels receives significant interests in recent years.

The main strategies can be roughly divided into two categories: (1) generating annotations automatically and (2) training with missing labels. The former uses the web as the supervisor to generate annotations [6], since the web is a large amount of imagery data with labeled information available on the web, such as social media hashtags and connections between web-pages and user feedback. However, these methods may introduce additional noises to the label space, which can degrade a classifier’s performance. For the latter, missing labels means that only a subset of all the labels can be observed and the rest remains unknown. It can be further divided into several representative settings: fully observed labels (FOL), partially observed labels (POL), which is the most common setting. Two variations of POL include: partially observed positive labels (PPL) and single positive label (SPL). It should be pointed out that POL setting is more common than PPL in real life. For example, in many execution records of industrial devices [7], the probability of each component’s failure is extremely low. Therefore, it is almost impossible to guarantee that each instance corresponds to one positive label, let alone in the setting of missing labels.

This paper focuses on multi-label classification with missing labels. Although there has been a lot of work done along this direction [5], there are still some critical issues to be addressed:

- To solve the multi-label classification with missing labels, many state-of-the-art (SOTA) methods [2] [4] rely on additional structures, such as GNN and label estimator, which further increase the complexity of networks. A natural question is whether this problem can be effectively solved without significantly increasing the network complexity.
- It is still not clear how the missing ratio of the labels affects the classification performance, which is of great importance for us to balance the performance of classifier and the annotation cost.
- Due to imbalance between positive and negative labels, most methods dealing with missing labels require that...
there is at least one positive label per instance, i.e., PPL, instead of POL, which is more common in real life.

With these observations, this paper investigates new approaches for multi-label classification with missing labels. The main contributions are summarized as follows:

- We propose a pseudo-label-based approach to predict all possible categories with missing labels, which can effectively balance the performance of classifiers and the cost of annotation. The network structure in our approach is the same as the classifier trained with full labels, which means that our approach will not increase the network complexity. The major difference lies in the novel design of loss functions and training schemes.

- We provide systematical and quantitative analysis of the impact of labels’ missing ratio on the classifier’s performance. In particular, we relax the strict requirement that the label space of each instance must contain at least one positive label, which is commonly seen in the related work [4], [5]. Therefore, our method is applicable to general POL settings, not only PPL.

- Comprehensive experiments verify that our approach can be effectively applied to missing-label classification. Specifically, our approach outperforms most existing missing-label learning approaches, and in some cases even approaches trained with fully labeled datasets. More importantly, our approach can adopt POL settings, which is incompatible with most existing methods.

II. RELATED WORK

A. Multi-label Learning with Missing Labels

Recently, numerous methods have been proposed for multi-label classification with missing labels. Herein, we briefly review the relevant studies.

Positive and Unlabeled Learning (PU-learning). PU-learning is an alternative solution [8], which studies the problem with a small number of positive examples and a large number of unlabeled examples for training. Most methods can be divided into the following three categories: two-step techniques [9], biased learning [10], and class prior incorporation [11]. All these methods require that the training data consists of positive and unlabeled examples [12]. In other words, they treat the negative labels as unlabeled, which discard the existing negatives and does not make full use of existing labels.

Pseudo Label. Pseudo-labeling was first proposed in [13]. The goal of pseudo-labeling is to generate pseudo-labels for unlabeled samples [14]. There are different methods to generate pseudo labels: the work in [13], [15] uses the predictions of a trained NN to assign pseudo labels. Neighborhood graphs are used in [16]. The approach in [17] updates pseudo labels through an optimization framework. It is worth mentioning that MixMatch-family semi-supervised learning methods [18] achieve SOTA on multi-class problem by utilizing pseudo labels and consistency regularization [19]. However, creation of negative pseudo-labels (i.e., labels which specify the absence of specific classes) is not supported by these methods, which therefore affects the performance of classifier by neglecting negative labels [15]. Instead, the work in [15] obtains the reference values of pseudo labels directly from the network predictions and then generates hard pseudo labels by setting confidence thresholds for positive and negative labels, respectively. Different from [15], we simplify this process by studying the proportion of positive and negative labels to generate pseudo labels.

B. Imbalance

A key characteristic of multi-label classification is the inherent positive-negative imbalance created when the overall number of labels is large [20]. Missing labels exacerbate the imbalance and plague recognizing positives [5]. Therefore, the work in [5] mandates that each instance in the training set must have at least one positive label, which means that they focus on PPL setting instead of “real” POL. Obviously, this assumption may not always hold in real life scenarios. To relax this assumption, a trivial solution is to treat the instances with only negative labels as unlabeled instances. In this case, however, it wastes the value of negative labels.

III. FORMULATION

Given a multi-label classifier with full labels, let \( \mathcal{X} = \mathbb{R}^M \) be the input attribute space of \( M \)-dimensional feature vectors and \( \mathcal{Y} = \{1,2,\ldots,L\} \) denote the set of \( L \) possible labels. An instance \( \mathbf{x} \in \mathcal{X} \) is associated with a subset of labels \( \mathbf{y} \in 2^L \), which can be represented as an \( L \)-vector \( \mathbf{y} = [y_1, y_2, \ldots, y_L] = \{0,1\}^L \) where \( y_j = 1 \) iff the \( j \)th label is relevant (otherwise \( y_j = 0 \)). Let \( \mathcal{D} = \{ (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \} \) is the training set of \( N \) samples. Given \( \mathcal{D} \), a multi-label classifier \( h : \mathcal{X} \rightarrow \mathcal{Y} \) learns to map the attribute input space to the label output space. We use \( \hat{y} \) to present the prediction of classifier \( h \), that is, \( \hat{y} = f(h(\mathbf{x})) \), where \( f(\cdot) \) stands for a function (commonly the sigmoid function as \( \sigma(s) = \frac{1}{1+e^{-s}} \)) that turns confidence outputs into a prediction. In this case, most of the existing work [1], [21]–[23] adopts the binary cross entropy (BCE) function as the loss function, which is formulated by

\[
\mathcal{L}(\hat{y}, y) = -\frac{1}{N} \sum_{i=1}^{L} [(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))].
\] (1)

For multi-label classification with missing-label, we use \( \mathbf{z} = [z_1, z_2, \ldots, z_L] = \{0,1,\varnothing\}^L \) to present the observed labels, where \( z_j = \varnothing \) means the corresponding label is missing. It is worth mentioning that the training set is \( \mathcal{D} = \{ (\mathbf{x}_1, z_1), \ldots, (\mathbf{x}_N, z_N) \} \), while validation and test set is \( \mathcal{D}_v = \mathcal{D}_t = \{ (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \} \) in this case. In other words, we still use full labels for validation and testing.
IV. PROPOSED METHOD

Without changing the basic structure of the classification network, we address multi-label classification with missing labels by introducing pseudo-label, designing new loss functions, and adjusting training schemes. The pipeline of our method is shown as Fig. 1.

A. Pseudo-labels

For all missing-label settings, the label of any instance in the training set can be divided into two categories: observed and unobserved labels. [24] directly set the unobserved labels as negative. However, performance drops because a lot of ground-truth positive labels are initialized as negatives [25]. Therefore, we decide to introduce soft pseudo labels for the unobserved part and take pseudo labels as the target value of unobserved labels in calculation of the loss. Another benefit of introducing pseudo labels is that by properly designing updating methods, it will not add complexity to the existing classifier network. In contrast, ROLE [4] adds another NN as the label estimator and takes the predictions of this NN as the target. [2] adds a GNN on top of a CNN. Obviously, these methods bring additional complexity to the network.

In our approach, the classifier actually has two goals in the training process:

- For observed labels, we expect the predicted value to be pushed to its ground-true value.
- For unobserved labels, we expect the predicted value to be as close as possible to the value of its corresponding pseudo label.

All the design steps in our method revolve these two points. For (1), we just need to do as traditional supervised learning does. For (2), there are two issues in front of us, that is, how to generate pseudo-labels and how to update them. In the following discussion, we denote pseudo labels as \( \tilde{y} \) and the pseudo-labeling value corresponding to the \( i \)th instance’s \( j \)th class as \( \tilde{y}_{ij} \), which can be any real number between \([0, 1]\). By using the \( \text{max} \) function in (2), it means we have an initial guess that each instance has at least one positive label in average, either observed or unobserved (of course, such an initial guess is not necessarily true).

The initial method relies on a fact that even though most of the labels are missing, the statistics of the observed labels can still be used as a reference for the overall distribution of this dataset. Therefore, if the number of an instance’s observed positive labels, \( P_p \), is less than the average \( \max(P_p/m, 1) \), we assign the difference, divided by \( T_u \), as the pseudo labels of those unobserved labels on this instance. Otherwise, if the number of the observed positive labels exceeds the average, we set the pseudo labels to 0. Note that this setting is only for initialization. The pseudo labels will be updated during the training progresses.

It is worth mentioning that the previous approaches often initialize the pseudo labels as 0.5 or 0. Compared with the 0.5 setting, they do not take full advantage of the statistic information of the dataset. As to the 0 setting, it may aggravate the label imbalance.

**Update.** We update the value of pseudo labels after each epoch of updating the network. A common way is to simply use the network predictions as the pseudo labels. To check the trajectory of the pseudo labels of this method, we randomly select 10 training instances and record the fluctuations of their predictions and pseudo labels’ value in SPL settings. Due to space limitation, we only show the fluctuations of one category of one instance in Fig. 2. Obviously, the predictions fluctuate greatly in this common update method (as shown in the left plot in Fig. 2). Note that even in the later stage of training, the fluctuation still exists, which makes it hard to converge. This is clearly not what we expect in training.

Therefore, we use a stack to record the predictions of our classifier for the past \( n \) epochs and take the average of these historical predictions as the value for the associated pseudo labels, where \( n \) is a hyperparameter. Following this method, we record the trajectories of the predictions and the pseudo labels on the same training instances under SPL setting during training (the right plot in Fig. 2). It is easy to find that this running-average method smooths the fluctuations of predictions and accelerate the convergence. We formulate this updating method as follows:

\[
\tilde{y}_{ij} = \frac{1}{n} \sum_{k=0}^{n-1} \tilde{y}_{ij}^{m-k}
\]

where \( m \) stands for the current epoch index and \( \tilde{y}_{ij}^{m-k} \) stands for the predicted value of the \( j \)th category in the \( i \)th instance at the \( m \)th epoch.

**Disturbance Injection.** In experiments, we find that the predictions of the classifier for some unobserved categories remain around 0.5. As mentioned before, we believe that it is always better for the classifier to give an answer instead of being ambiguities. Following this concept, we devise a detector to detect such vague decisions. By utilizing the
Fig. 1. Pipeline of our method. We feed an image to the classifier network and obtain the predictions. Then different strategies are utilized to deal with observed and unobserved labels. For the unobserved part, we introduce pseudo labels by leveraging the prior knowledge of the dataset and design a novel loss function $L_{\text{Unobserved}}$. For the observed part, we treat it as a full-label classification problem. The total loss is defined by adding the losses of these two parts with designed weights. The network used in our method is the most common classification network, which means that there is no additional complexity in the network structure.

Fig. 2. Fluctuations in predicted values and pseudo labels during training. The GT value for this specific label is 1. The left and right plot shows the trajectories of the predicted and pseudo labels under traditional updating method (set the pseudo label equal to the predicted value) and our running-average updating method, respectively. It is easy to see that our method accelerates convergence and smooths updating trajectories.

Algorithm 1 summarizes the updating process of pseudo labels. It should be pointed that the detection will not be performed in the first $D_s$ epochs and the last $D_e$ epochs of training, since the predicted labels may fall into $[0.5-d, 0.5+d]$ simply due to non-convergence at the beginning of training and training may stop before reaching convergence in the last several epochs. Here $D_s$ and $D_e$ are also hyperparameters.

Algorithm 1 Update pseudo()

Input: predicted value $\hat{y}_{ij}$ for $j$th class of $i$th instance at epoch $m$, stack $S_{ij}$, start epoch $D_s$, end epoch $D_e$

1: if $D_s < m < D_e$ then
2: if $S_{ij}$ is FULL then
3: $S_{ij}.pop()$
4: end if
5: $S_{ij}.push(\hat{y}_{ij})$
6: if detector($S_{ij}$) is TRUE then
7: update pseudo labels $\tilde{y}_{ij}$ based on (4)
8: else
9: $\tilde{y}_{ij} \leftarrow \text{avg}(S_{ij})$ based on (3)
10: end if
11: end if
Output: $\tilde{y}_{ij}$, $S_{ij}$

B. Design of Loss function

At this point, we have solutions to meet the two goals mentioned in Subsection IV-A. Next, we relate these two goals together by designing an appropriate loss function.

historical prediction stack $S_{pse} = \{\hat{y}_{ij}^{m-k}\}_{k=0}^{n-1}$, we determine whether these predictions are all in interval $[0.5-d, 0.5+d]$, where $d$ is a hyperparameter. If so, we update the pseudo labels using (4), instead of updating pseudo labels in (3), which adds random disturbances to these kind of unobserved categories to push the pseudo labels away from 0.5:

$$\tilde{y}_{ij} = \begin{cases} \hat{y}_{ij} - \text{random}(0, \hat{y}_{ij}), & \hat{y}_{ij} < 0.5 \\ \hat{y}_{ij} + \text{random}(0, 1 - \hat{y}_{ij}), & \text{otherwise} \end{cases}$$

where $\text{random}(a, b)$ stands for a function which generates a random real number between $a$ and $b$. 
Intuitively, we can obtain the loss function by adding the losses of the observed part and unobserved part:

\[
L = L_{\text{Observed}} + L_{\text{Unobserved}}
\] (5)

where \(L_{\text{Observed}}\) is obtained by modifying the number of classes \(L\) in (1) to the number of observed labels \(O_i\) in the \(i\)th instance:

\[
L_{\text{Observed}} = -\frac{1}{O_i} \sum_{j=1}^{O_i} [y_j \log(\hat{y}_j) + (1 - y_j) \log(1 - \hat{y}_j)].
\] (6)

Before passing the pseudo labels into the loss function, we define a threshold function on the pseudo labels, \(f(\hat{y}_i, t)\), where \(t\) is a hyperparameter. In order to simplify the description, this threshold is ignored below. For \(L_{\text{Unobserved}}\), since the pseudo labels may contain errors, we can generally regard them as the GT values with noises. Therefore, we follow the idea in [26] to design \(L_{\text{SBCE}}\) as follows, which has been proven to be robust to noises:

\[
L_{\text{Unobserved}} = \alpha L_{\text{SBCE}}(\hat{y}, \tilde{y}) + \beta L(\tilde{y}, \hat{y}).
\] (7)

where \(\alpha\) and \(\beta\) stand for two decoupled hyperparameters.

**Imbalance between Positives and Negatives.** Label imbalance significantly affects the generalization performance of the multi-label predictive model. The classifier can be easily reduced to the trivial “always predict positive/negative” solution. To alleviate imbalance, we add confidence scores into \(L_{\text{Unobserved}}\) to balance the contributions of positive and negatives during training:

\[
L_{\text{CFS}}(x, y, C_1, C_2, L) = -\frac{1}{T} \sum_{i=1}^{L} [C_1 y_i \log(y_i) + C_2 (1 - y_i) \log(1 - y_i)],
\] (8)

where \(C_1 = \frac{N}{T}\) and \(C_2 = \frac{P}{T}\) are the weights, \(N\) and \(P\) stand for the total number of the observed negative and positive labels, respectively, and \(T = N + P\) stands for the total number of observed labels in the training set. Obviously, \(C_1 + C_2 = 1\). Accordingly, the loss function for unobserved part can be defined as follows:

\[
L_{\text{Unobserved}} = \alpha L_{\text{CFS}}(\hat{y}, \tilde{y}, C_1, C_2, U) + \beta L_{\text{SBCE}}(\hat{y}, \tilde{y}, C_1, C_2, U),
\] (9)

where \(U\) stands for the number of the unobserved labels on the current instance. It should be pointed out that the focal loss in [27] also designs scores for positive and negative labels, which are fine-tuned hyperparameters. Compared with [27], our method is more intuitive by using the scale information of observed labels in the training set.

**The Weighted Loss Function.** To improve the loss, we follow the concept from curriculum learning [28]: start learning from the axioms that have been carefully tested, and then apply the axioms to real problems in life. Consequently, we expect our model to follow the same pattern during the training process, i.e., it should first focus on the observed part, which takes the GT labels as target, and then gradually shifts its attention to the unobserved part. Therefore, we design the time-varying confidence scores for \(L_{\text{Observed}}\) and \(L_{\text{Unobserved}}\) to reflect this dynamic by utilizing the index of the current epoch \(e\):

\[
L = \frac{T_e - e}{T_e} L_{\text{Observed}} + \frac{e}{T_e} L_{\text{Unobserved}}.
\] (10)

where \(T_e\) is the total number of epochs. We will use this loss function in (10) in our experiments.

V. Experiments

We test the effectiveness of our approach separately on different labeling settings (POL, PPL, and SPL) and various datasets, and compare it with several representative baseline methods. Then we provide the ablation study that evaluates the contribution of each component described in Section IV.

A. Datasets

We conduct comprehensive experiments on three large-scale multi-label image datasets: COCO [29], NUS-WIDE [30], and Pascal VOC [31]. Each instance in these three datasets is fully annotated with clean labels that can be used as the GT in performance evaluation.

To test our approach on various missing-label settings, we need to corrupt label spaces of these three datasets by discarding a part of labels. The processing method we use is similar to [4] which generate labels for SPL setting. The difference in our case is that besides SPL setting, we also generate other settings by randomly selecting positive/negative labels at specified proportions. If a label (0 or 1) is discarded, we use ∅ to denote it. Note that we only do this operation once so that the labels on each instance of the training set can keep consistent during multiple training processes.

For COCO and Pascal VOC, we adopt the official training/testing splitting methods. In COCO, the official training set consists of 82081 images, where each one is associated with 80 possible categories, and there are 40,137 images in the official testing set. The official training and testing sets of Pascal VOC contain 5,717 and 5,823 images respectively, and each image has 20 different categories. For NUS-WIDE, we download it from Flickr, mix the official training and testing sets together, and then follow the splitting method of [20]. Finally, we collect NUS-wide with 116445 and 50720 images as training set and test set and each image has 81 labels.

After determining the training and testing sets, to be consistent with [4], we choose 20% images from the training set as the validation set. Then we train end-to-end fine-tuned deep CNNs across these three datasets in different settings.

As for the data augmentation and pre-processing, we first resize the original input image of all these three datasets to the shape of 448 × 448. And then, horizontal flip is applied with a probability of 0.5. In the end, by using the standard Imagenet statistics, we normalize the input image.
TABLE I
THE mAP RESULTS IN POL SETTINGS ON PASCAL VOC. IN EACH COLUMN WE BOLD THE BEST PERFORMING METHOD AND UNDERLINE THE SECOND-BEST EXCEPT BCE UNDER FOL SETTING.

| Format | POL_02 | POL_04 | POL_06 | POL_08 |
|--------|--------|--------|--------|--------|
| Ours   | 81.5   | 86.5   | 88.0   | 89.2   |
| AN [20] | 78.2   | 84.6   | 86.7   | 88.0   |
| Focal [27] | 77.7   | 82.1   | 85.9   | 87.3   |
| AN [33] | 63.1   | 69.4   | 72.8   | 76.5   |
| WAN [24] | 70.6   | 78.9   | 80.0   | 83.4   |
| ROLE [4] | 69.2   | 77.0   | 81.4   | 82.9   |

B. Metrics
There are different standard metrics to evaluate the performance of multi-label classifiers [1], [23], such as mean average precision (mAP), precision-at-k, and recall-at-k, to name a few. In our case, we adopt mAP as the primary evaluation metric.

C. Baselines
In order to demonstrate the effectiveness of our method and evaluate the impact of the missing ratio, we first choose BCE and BCE-LS with full labels as strong baselines. These two methods are the most commonly used methods in multi-label problem. The former takes BCE (1) as the loss function and the latter uses label smoothing BCE [32], which is proposed to reduce overfitting and has been shown to be effective in mitigating the negative impacts of label noises.

Besides taking BCE and BCE-LS under FOL setting as strong baselines, we select some representative approaches for comparison.

In PPL setting, we choose AN [33], WAN [24], and ROLE [4] and directly use the official experimental settings and code. AN assumes that unobserved labels are always negative, which is perhaps the most common method for PPL settings. WAN down-weights terms in the loss related to negative labels by introducing a weight parameter. ROLE considers regularized online estimation of unobserved labels.

For POL, as stated before, most existing methods cannot be applied in this setting directly. Therefore, we have to make some modifications to the original methods. Specifically, besides AN, WAN and ROLE, the approaches for comparison in this setting include Focal [27], and ASL [20]. For AN, WAN and ROLE, due to the severe imbalance in missing-label settings, Cole et al. [4] require that each instance must have one positive labels in the code implementation. We directly remove the restriction in the official code for comparison. Note that although Focal and ASL are designed to solve the imbalance issue, the authors did not consider them in missing-label settings. Therefore, in order to make a comparison with our method, we follow [24], [34], [35] and regard the labels for unobserved part as negative when implementing these two approaches.

D. Implement Details
Network Structure. We use an end-to-end network for all experiments: a ResNet-50 [36], pre-trained on ImageNet [37], as the backbone and a fully connected layer, which is the same as the multi-label classifier under FOL setting. Our approach does not add any extra structure to the network.

Hyperparameters. We train our classifier for 10 epochs, that is, $T_e = 10$. For the learning rate and batch size, we use a hyperparameter search method and select the hyperparameters with the best mAP on the validation set, where the learning rate is in $[1e-3, 1e-4, 1e-5, 1e-6]$ and batch size is in $[8, 16]$. We assign the threshold $t$ to be 0.7 and $d$ to be 0.2. Recall that $d$ is used to decide whether to introduce disturbances in pseudo labels. The historical stack size $n$ is 3, considering the memory consumption. $D_s$ and $D_c$ are set to be 3 and 7, respectively, which are determined after several trials. Finally, we set $\alpha = 0.95$ and $\beta = 0.05$.

E. Analysis
POL. Alleviating the imbalance between positives and negatives is one highlight of our approach. In POL, the impact of the imbalance may be exacerbated since the observable labels are randomly selected. There is equal chance for positive and negative labels to be selected, which means that it is even possible that an instance has multiple positive labels, while another one has no positive labels. This is the reason why most related methods are incompatible with this setting. We opt to use the modified version of AN, WAN, ROLE, Focal and ASL as baseline methods for comparison, and the detailed modification process is described in Sec. V-C.

We conduct experiments on Pascal VOC in POL_02, POL_04, POL_06, and POL_08 settings. The experimental results are shown in Table I. As excepted, since AN does not take imbalance into consideration, it performs the worst among all other methods in our experiments. Moreover,
TABLE II

THE MAP RESULTS IN PPL SETTINGS ON COCO, PASCAL VOC, AND NUSWIDE DATASETS. PPL_04, PPL_06, AND PPL_08 REPRESENT RANDOMLY RETAINING 40%, 60% AND 80% OF POSITIVE LABELS FOR EACH INSTANCE, RESPECTIVELY, AND DISCARDING ALL NEGATIVE LABELS. SPL IS EQUIVALENT TO PPL_SINGLE, WHICH IS AN EXTREME VARIATION OF PPL. IN EACH COLUMN WE HOLD THE BEST PERFORMING METHOD AND UNDERLINE THE SECOND-BEST EXCEPT BCE AND BCE-LS THAT IS PERFORMED UNDER FOL SETTING.

| Dataset   | SPL | PPL_04 | PPL_06 | PPL_08 | SPL | PPL_04 | PPL_06 | PPL_08 | SPL | PPL_04 | PPL_06 | PPL_08 |
|-----------|-----|--------|--------|--------|-----|--------|--------|--------|-----|--------|--------|--------|
| COCO      |     |        |        |        |     |        |        |        |     |        |        |        |
| BCE (FOL) | 75.8|        |        |        | 88.9|        |        |        | 52.6|        |        |        |
| BCE-LS (FOL) | 76.8| 90.0  |        |        | 53.5|        |        |        |     |        |        |        |
| AN [33]   | 63.9| 66.4  | 67.0   | 69.2   | 84.7| 86.8   | 87.3   | 88.0   | 40.0| 45.1   | 48.3   | 50.3   |
| WAN [24]  | 64.3| 69.1  | 71.3   | 74.2   | 85.9| 87.5   | 87.9   | 88.5   | 43.7| 46.9   | 48.5   | 50.6   |
| ROLE [4]  | 65.9| 73.1  | 75.4   | 77.1   | 87.0| 88.9   | 90.0   | 90.3   | 43.3| 48.0   | 50.4   | 52.0   |
| Ours      | 67.8| 74.4  | 75.9   | 77.8   | 86.6| 88.9   | 90.8   | 91.1   | 46.0| 49.6   | 51.9   | 53.3   |

ROLE, which won the second place in PPL setting in our experiments, can not cope with POL setting well. Relying on the ingenious design of the loss function, our method has a significant improvements compared with other baseline-methods. In particular, we notice that the fewer labels are available, the greater the performance of our approach can be over other baselines. Obviously, the experimental results show that our approach can effectively deal with PPL settings, which is the most common settings in real world.

To further quantify the impact of the missing ratio of labels on the classification performance, we run our approach in POL_005, POL_01, and the results are summarized in Fig. 3. The performance of our algorithm improves as the number of observed labels increases. Note that in POL_06, we use only 60% of labels and the achieved score is only 1.1% lower than the strong baseline. In POL_08, the performance even exceeds the strong baseline with only 80% of labels. Considering that the cost of annotation is expensive, we can find a balance between 60% and 80% of labels to trade off accuracy of the classifier and the annotation cost.

PPL & SPL. In order to demonstrate the effectiveness of our approach in PPL settings, we consider different labeling settings (PPL_04, PPL_06, PPL_08 and SPL) on three datasets. For fair comparison, we directly execute the official code of AN, WAN and ROLE without any modification as baseline-methods. Meantime, we perform BCE and BCE-LS on these three fully-labeled datasets, i.e., under FOL setting, which are used as strong baselines for our approach. Table II summarizes the experimental results. For COCO dataset and NUS-WIDE dataset, our algorithm achieves the highest score in all settings. For Pascal VOC, our approach achieves the best mAP score in PPL_04, PPL_06 and PPL_08 settings and the second highest, but comparable, score in SPL setting. These results fully demonstrate that our approach can be effectively applied to PPL settings.

Compared with the strong baselines under FOL setting, the results show that for COCO, our approach can achieve comparable performance to BCE and BCE-LS in PPL_06 and PPL_08, respectively. For Pascal VOC, the results by our approach in PPL_06 and PPL_08 are even able to outperform these two FOL baselines. It is surprising to find that our approach in PPL_04 can even achieve a score comparable to strong baselines, which means we save 26.7% of the cost in labeling positives and 100% cost in labeling negatives, according to the statistics of the labels used in our experiments. As for NUS-WIDE, our method also can reach the strong baselines, even outperforms BCE(FOL) in PPL_08 setting. However, the other baseline-methods are not able to exceed the strong baselines.

F. Ablation study

We present an ablation study to measure the contributions of different components in our approach. We run experiments on Pascal VOC in PPL_06 and POL_06. The results are shown in Table III.

TABLE III

THE MAP RESULTS IN ABLATION STUDY UNDER PPL_06 AND POL_06 SETTINGS ON PASCAL VOC.

|                     | PPL_06 | POL_06 |
|---------------------|--------|--------|
| Ours Approach       | 90.8   | 88.0   |
| Without Update      | 90.1   | 86.6   |
| Without Disturbances| 88.2   | 86.0   |
| Without Imbalance Design | 91.1 | 76.9   |
| Without Weighted Loss | 90.9 | 79.4   |

For PPL_06 setting, both our proposed running-average updating method and the introduction of disturbances improve the performance of the approach. However, the novel loss function, including imbalance and weighted design, reduces the performance, because these two factors are designed for more general settings, not specifically for PPL. This can be found in POL_06 setting, where the effectiveness of our design is demonstrated: the imbalance and dynamically weighted design can significantly enhance the performance of the classifier. The former improves the result by more than 11.1%, and the latter improves by 8.6%. In contrast, the new updating method and disturbance have relatively limited, but still positive, improvements in POL setting (more than 1% increase).

VI. CONCLUSION

This paper explores the problem of multi-label classification with missing labels. We propose a novel pseudo-
label-based approach to cope with missing-label problem without increasing the complexity of the existing classification networks. By leveraging prior knowledge of the dataset, our approach relaxes the assumption that each instance in the training set must have at least one positive label, which is often required in existing approaches. We demonstrate the effectiveness of our approach by performing extensive experiments on three large-scale datasets. It is shown that our method can effectively reduce the annotation cost without significant degradation in classification performance.

REFERENCES

[1] G. Tsoumakas and I. Katakis, “Multi-label classification: An overview,” International Journal of Data Warehousing and Mining (IJDWM), vol. 3, no. 3, pp. 1–13, 2007.
[2] T. Durand, N. Mehrasa, and G. Mori, “Learning a deep convnet for multi-label classification with partial labels,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 647–657.
[3] T. Hastie, R. Tibshirani, and J. Friedman, “Overview of supervised learning,” in The elements of statistical learning. Springer, 2009, pp. 9–41.
[4] E. Cole, O. Mac Aodha, T. Lorrieul, P. Perona, D. Morris, and N. Jojic, “Multi-label learning from single positive labels,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 933–942.
[5] Y. Zhang, Y. Cheng, X. Huang, F. Wen, R. Feng, Y. Li, and Y. Guo, “Simple and robust loss design for multi-label learning with missing labels,” arXiv preprint arXiv:2102.07368, 2021.
[6] M. A. Hedderich and D. Klakow, “Training a neural network in a low-resource setting on automatically annotated noisy data,” arXiv preprint arXiv:1807.00745, 2018.
[7] W. A. Smith and R. B. Randall, “Rolling element bearing diagnostics using the case western reserve university data: A benchmark study,” Mechanical systems and signal processing, vol. 64, pp. 100–131, 2015.
[8] X. Li and B. Liu, “Learning to classify texts using positive and unlabeled data,” in IJCAI, vol. 3, no. 2003. Citeseer, 2003, pp. 587–592.
[9] B. Liu, W. S. Lee, P. S. Yu, and X. Li, “Partially supervised classification of text documents,” in ICML, vol. 2, no. 485. Sydney, NSW, 2002, pp. 387–394.
[10] S. Sellamanickam, P. Garg, and S. K. Selvaraj, “A pairwise ranking based approach to learning with positive and unlabeled examples,” in Proceedings of the 20th ACM international conference on Information and knowledge management, 2011, pp. 663–672.
[11] L. Jiang, H. Zhang, and Z. Cai, “A novel bayes model: Hidden naive bayes,” IEEE Transactions on knowledge and data engineering, vol. 21, no. 10, pp. 1361–1371, 2008.
[12] J. Bekker and J. Davis, “Learning from positive and unlabeled data: A survey,” Machine Learning, vol. 109, no. 4, pp. 719–760, 2020.
[13] D.-H. Lee et al., “Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks,” in Workshop on challenges in representation learning, ICMIL, vol. 3, no. 2, 2013, p. 896.
[14] W. Shi, Y. Gong, C. Ding, Z. M. Tao, and N. Zheng, “Transductive semi-supervised deep learning using man-max features,” in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 293–315.
[15] M. N. Rizve, K. Duarte, Y. S. Rawat, and M. Shah, “In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning,” arXiv preprint arXiv:2101.06529, 2021.
[16] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, “Label propagation for deep semi-supervised learning,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 5070–5079.
[17] G.-H. Wang and J. Wu, “Repetitive reproduction deep decipher for semi-supervised learning,” in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 04, 2020, pp. 6170–6177.
[18] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliva, and C. A. Raffel, “Mixmatch: A holistic approach to semi-supervised learning,” Advances in Neural Information Processing Systems, vol. 32, 2019.
[19] H. Zhang, Z. Zhang, A. Odena, and H. Lee, “Consistency regularization for generative adversarial networks,” arXiv preprint arXiv:1910.12027, 2019.
[20] T. Ridnik, E. Ben-Baruch, N. Zamir, A. Noy, I. Friedman, M. Potter, and L. Zelnik-Maran, “Asymmetric loss for multi-label classification,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 82–91.
[21] J. Nam, J. Kim, E. Loza Mencía, I. Gurvych, and J. FURNKRAM, “Large-scale multi-label text classification—revisiting neural networks,” in Joint european conference on machine learning and knowledge discovery in databases. Springer, 2014, pp. 437–452.
[22] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu, “Cnn-rnn: A unified framework for multi-label image classification,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2285–2294.
[23] M. S. Sorower, “A literature survey on algorithms for multi-label learning,” Oregon State University, Corvallis, vol. 18, pp. 1–25, 2010.
[24] O. Mac Aodha, E. Cole, and P. Perona, “Presence-only geographical priors for fine-grained image classification,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 9596–9606.
[25] J. Joulin, L. v. d. Maaten, A. Jabri, and N. Vasilevski, “Learning visual features from large weakly supervised data,” in European Conference on Computer Vision. Springer, 2016, pp. 67–84.
[26] Y. Wang, X. Ma, Z. Chen, Y. Luo, J. Yi, and J. Bailey, “Symmetric cross entropy for robust learning with noisy labels,” in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 322–330.
[27] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” in Proceedings of the IEEE international conference on computer vision, 2017, pp. 2980–2988.
[28] C. Szegedy, V. Vanhoucke, R. Volpi, R. Vidal, and V. Murino, “Curriculum dropout,” in IEEE International Conference on Computer Vision (ICCV), 2017, pp. 3544–3552.
[29] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in European conference on computer vision. Springer, 2014, pp. 740–755.
[30] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, “Nus-wide: a real-world web image database from national university of singapore,” in Proceedings of the ACM international conference on image and video retrieval, 2009, pp. 1–9.
[31] M. Everingham, S. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes challenge: A retrospective,” International journal of computer vision, vol. 111, no. 1, pp. 98–136, 2015.
[32] G. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2818–2826.
[33] K. Kundu and J. Tighe, “Exploiting weakly supervised visual patterns to learn from partial annotations,” Advances in Neural Information Processing Systems, vol. 33, pp. 561–572, 2020.
[34] S. S. Bucak, R. Jin, and A. K. Jain, “Multi-label learning with incomplete class assignments,” in CVPR 2011. IEEE, 2011, pp. 2801–2808.
[35] Y.-Y. Sun, Y. Zhang, and Z.-H. Zhou, “Multi-label learning with weak label,” in Twenty-fourth AAAI conference on artificial intelligence, 2010.
[36] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1492–1500.
[37] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, “Imagenet: A large-scale hierarchical image database,” in 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009, pp. 248–255.