Fusion Label Enhancement for Multi-Label Learning

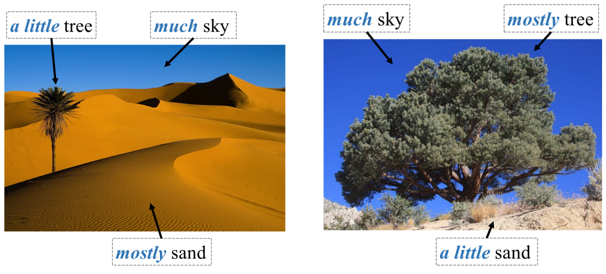
Xingyu Zhao^{1,2} , **Yuexuan An**^{1,2} , **Ning Xu**^{1,2} and **Xin Geng**^{1,2*}
^{1}School of Computer Science and Engineering, Southeast University, Nanjing 211189, China
^{2}Key Laboratory of Computer Network and Information Integration (Ministry of Education), Southeast University, Nanjing 211189, China
{xyzhao, yx_an, xning, xgeng}@seu.edu.cn

Abstract

Multi-label learning (MLL) refers to the problem of tagging a given instance with a set of relevant labels. In MLL, the implicit relative importance of different labels representing a single instance is generally different, which recently gained considerable attention and should be fully leveraged. Therefore, *label enhancement* (LE) has been widely applied in various MLL tasks as the ability to effectively mine the implicit relative importance information of different labels. However, due to the fact that the label enhancement process in previous LE-based MLL methods is decoupled from the training process on the predictive models, the objective of LE does not match the training process and finally affects the whole learning system. In this paper, we propose a novel approach named *Fusion Label Enhancement for Multi-label learning* (FLEM) to effectively integrate the LE process and the training process. Specifically, we design a matching and interaction mechanism which leverages a novel interaction label enhancement loss to avoid that the recovered label distribution does not match the need of the predictive model. In the meantime, we present a unified label distribution loss for establishing the corresponding relationship between the recovered label distribution and the training of the predictive model. With the proposed loss, the label distributions recovered from the LE process can be efficiently utilized for training the predictive model. Experimental results on multiple benchmark datasets validate the effectiveness of the proposed approach.

1 Introduction

Learning with ambiguity has become one of the hot topics among the machine learning communities [Geng, 2016; Gao *et al.*, 2017]. Multi-label learning (MLL) is a common choice to deal with the label ambiguity problem, because an instance can be annotated by multiple labels simultaneously in MLL [Zhang and Zhou, 2014; Liu *et al.*, 2021]. In MLL, each class label is treated as a logical indicator of whether the corresponding label is relevant or irrelevant to the instance, i.e., +1 represents relevant to the instance and -1 represents irrelevant to the instance. Such label represented by -1 or +1 is called *logical label*. During the past years, multi-label learning techniques have been widely applied to various fields such as document classification [Liu *et al.*, 2017], video concept detection [Lo *et al.*, 2011], image classification [Lanchantin *et al.*, 2021], fraud detection [Wang *et al.*, 2020] etc.



(a) IRI: *sand* $\succ$ *sky* $\succ$ *tree* (b) IRI: *tree* $\succ$ *sky* $\succ$ *sand*

Figure 1: Exemplar images which have been annotated with multiple labels *sand*, *sky* and *tree*. The implicit relative importance (IRI) information is marked in the images.

In real-world MLL problems, the implicit relative importance of each possible label is generally different. For example, as shown in Fig.1 which has two images annotated with three positive labels *sand*, *sky* and *tree*. For the relevant label variance in Fig.1(a), the importance of *sand* should be greater than that of *sky* and *tree*, because *sand* can describe the image more apparently. Meanwhile, in Fig.1(b), the importance of *tree* should be greater than that of other labels. Therefore, the implicit relative importance information of different labels provides more accurate information for MLL and should be fully mined and leveraged.

Recently a novel technology named *label distribution* which can model the implicit relative importance of different labels has been proposed [Geng, 2016]. Formally speaking, given an instance $\boldsymbol{x}$, label distribution assigns each $y \in \mathcal{Y}$ a real value $d_{\boldsymbol{x}}^{y}$ (label description degree), which indicates the importance of y to $\boldsymbol{x}$. Since label distribution extends the supervision from binary to distribution, it is more applicable for real-world scenarios. Unfortunately, many train-

*Corresponding Author

ing sets only contain simple logical labels rather than label distributions due to the difficulty of obtaining the label distributions directly. Therefore, the technology of *label enhancement* (LE) [Xu *et al.*, 2021a] has been proposed. The aim of LE is to recover the label distributions from the logical labels in the training set via leveraging the topological information of the feature space and the correlation among the labels. Recently LE has attracted increasing research attention due to its potential to address the label ambiguity problem in machine learning and success in many real-world multi-label applications [Xu *et al.*, 2020; Xu *et al.*, 2021b].

Most of the existing LE-based MLL methods have two stages: 1) the numerical labels are first reconstructed, and 2) the predictive model is trained by levering the reconstructed labels. In these approaches, LE is independent of the training of the predictive model. When solving specific tasks, LE may not meet the needs of the training process on the predictive model. Other LE methods are one-stage approaches. These methods implement LE and the training process on the predictive model iteratively. However, these methods lack control over the interaction and matching between LE and the training process, i.e., the label enhancement process is decoupled from the training process, which finally affects the whole learning system. Therefore, it is essential to effectively integrate the label enhancement process and the training process to avoid the mismatch between the two processes.

In light of the above observations, we propose *Fusion Label Enhancement for Multi-label learning* (FLEM), in which label enhancement and the training process on the predictive model are performed simultaneously and interactively. Specifically, we first design a matching and interaction mechanism which leverages a novel interaction label enhancement loss to avoid the mismatch between the label enhancement process and the training process. Meanwhile, we establish the relationship between LE and the training process by employing a novel unified label distribution loss. With the proposed loss functions, the label enhancement information can be can effectively utilized to guide the training process on the predictive model. By simultaneously performing LE and the training process, the problem of mismatch between the two processes can be well addressed, and the performance of the model can be effectively improved.

2 Related Work

Multi-label learning (MLL) has attracted increasing interest recently. Traditional MLL approaches can be roughly grouped into three types based on the thought of the order of label correlations: first-order, second-order and high-order [Zhang and Zhou, 2014]. The first-order approaches decompose MLL into a series of binary classification and neglect the fact that the information of one label may be helpful for the learning of another label [Boutell *et al.*, 2004]. The second-order approaches consider the correlations between pairs of class labels, but they only focus on the difference between the relevant label and irrelevant label [Fürnkranz *et al.*, 2008]. The high-order approaches consider the correlations among label subsets or all the class labels [Wang *et al.*, 2019]. But these approaches take the equal label importance assumption.

Some researchers try to assign each label different importance through class frequencies. [Wu *et al.*, 2020] proposes a distribution-balanced loss for multi-label tasks by re-balancing weights and mitigating the over suppression of negative labels. [Ridnik *et al.*, 2021] designs an asymmetric loss that uses different γ values to weight positive and negative samples in focal loss. [Huang *et al.*, 2021] combines negative-tolerant regularization (NTR) [Wu *et al.*, 2020] and class-balanced focal loss (CB) [Cui *et al.*, 2019] and presents a novel loss function named CB-NTR. Balanced softmax method [Zhang *et al.*, 2021] turns the multi-label learning loss into comparisons of relevant label scores and irrelevant label scores, and balances the importance of each label automatically. However, these methods are based on elaborately designed rules that can not dynamically adjust the importance of labels based on the given instances. Meanwhile, these approaches ignore the correlations between labels.

Label enhancement (LE) is a recent method proposed for mining the implicit relative importance among different labels and the correlation between them. Many novel label enhancement methods have been proposed in recent years [Xu *et al.*, 2021a; Xu *et al.*, 2020]. These methods can be broadly categorized into two directions: two-stage approaches and one-stage approaches. GLLE [Xu *et al.*, 2021a] and LEVI [Xu *et al.*, 2020] are representative algorithms of two-stage approaches. They assume that the label distribution space should share a similar topological structure with the feature space. The main concern of these approaches is they separate the label enhancement process from the model training process, which leads to label enhancement may not meet the needs of model training. LEMLL [Shao *et al.*, 2018] and VALEN [Xu *et al.*, 2021b] are representative algorithms of one-stage approaches. They perform label enhancement and the training process on the predictive model iteratively. However, these approaches lack control over the interaction and matching between label enhancement and model training, which may cause the mismatch between the model training and label enhancement.

In the next section, a novel LE approach for multi-label learning will be introduced. Different from existing label enhancement approaches, a novel matching and interaction mechanism is designed to perform the label enhancement process and the training process on the predictive model simultaneously and avoid the mismatch between them.

3 The FLEM Approach

First of all, the main notations used in this paper are listed as follows. The instance variable is denoted by $\boldsymbol{x}$, the particular i-th instance is denoted by $\boldsymbol{x}_i$, the label variable is denoted by y, the particular j-th label value is denoted by y_j, the logical label vector of $\boldsymbol{l}_i$ is denoted by $\boldsymbol{l}_i = (l_{\boldsymbol{x}_i}^{y_1}, l_{\boldsymbol{x}_i}^{y_2}, ..., l_{\boldsymbol{x}_i}^{y_c})^{\mathrm{T}}$, where c is the number of possible labels. The description degree of y to $\boldsymbol{x}$ is denoted by $d_{\boldsymbol{x}}^{y}$, and the label distribution of $\boldsymbol{x}_i$ is denoted by $\boldsymbol{d}_i = \left(d_{\boldsymbol{x}_i}^{y_1}, d_{\boldsymbol{x}_i}^{y_2}, ..., d_{\boldsymbol{x}_i}^{y_c}\right)^{\mathrm{T}}$.

3.1 Overview

In FLEM, the label enhancement process and the training process on the predictive model are performed simultaneously. Specifically, FLEM adopts a predictive model $f(\cdot)$ for performing the MLL task and a label distribution estimator $g(\cdot)$ for label enhancement. The predictive model $f(\cdot)$ and the label distribution estimator $g(\cdot)$ are trained simultaneously during the whole training stage. For training the label distribution estimator, FLEM adopts three strategies to guide it to generate label distributions effectively. For avoiding inconsistency between LE and the training process on the predictive model, FLEM employs a matching and interaction mechanism which leverages a novel interaction label enhancement loss from the perspective of instances. For training the predictive model, FLEM integrates a novel unified label distribution loss to build a corresponding relationship between label distribution and the training of the predictive model. The algorithmic description of FLEM is given in Algorithm 1.

3.2 Label Enhancement

The existing MLL methods simply divide the label set into relevant labels and irrelevant labels, which can not be extended to the form of label distribution. Therefore, we need to recover the label distributions from the given logical label vectors, so as to guide the training of the predictive model.

In FLEM, the label distribution estimator $g(\cdot)$ is used to recover the label distributions. For generating the recovered label distribution, $g(\cdot)$ needs to use the topological information in both feature space and label space, so as to better obtain the correlation between labels according to the feature information. Therefore, we adopt two modules g_F and g_L for feature embedding and label embedding, which transform feature information and label information into continuous embedding values respectively. For the instance $\boldsymbol{x}_i$, we define the predicted values for label enhancement as

$$\boldsymbol{d}_i^* = g_D\left(\mathcal{C}\left(g_F\left(\boldsymbol{x}_i\right), g_L\left(\boldsymbol{l}_i\right)\right)\right), \tag{1}$$

where g_D is the module of g for decoding, $\mathcal{C}(\cdot,\cdot)$ is the concatenation operation. Then the recovered label distribution vector $\boldsymbol{d}_i$ can be obtained by the softmax operation on $\boldsymbol{d}_i^*$.

Training Strategies

In order to enable the label distribution estimator to generate a reasonable estimate of label distribution, we propose the following three training strategies. Each of them can lead to effective optimization.

1) The Similarity Strategy. The first strategy trains $g(\cdot)$ by matching the similarities of instances in feature space and label space. Specifically, the similarity matrix A of a batch of instances can be obtained by:

$$A_{mn} = \mathcal{S}\left(\boldsymbol{x}_m, \boldsymbol{x}_n\right), \tag{2}$$

where $\mathcal{S}$ is cosine similarity, $\boldsymbol{x}_m$ and $\boldsymbol{x}_n$ are the m-th and n-th instances. Meanwhile, by estimating the label distribution vector of these instances, we can obtain the similarity matrix Z of the predicted values for label enhancement:

$$Z_{mn} = \mathcal{S}\left(\boldsymbol{d}_m^*, \boldsymbol{d}_n^*\right), \tag{3}$$

where $\boldsymbol{d}_m^*$ and $\boldsymbol{d}_n^*$ are the predicted values for label enhancement of the m-th and n-th instances. Then the distance between A and Z is minimized to train the label distribution estimator. In FLEM, we adopt the following loss:

$$\mathcal{L}_{LE} = \frac{1}{M^2}\sum_{m=1}^{M}\sum_{n=1}^{M}\left(A_{mn} - Z_{mn}\right)^2, \tag{4}$$

where M is the number of instances.

2) The Threshold Strategy. The second strategy is based on the criterion that the minimum description degree of relevant labels is not less than the maximum description degree of irrelevant labels. Therefore, for a batch of instances, the loss function is:

$$\mathcal{L}_{LE} = \frac{1}{M}\sum_{i=1}^{M}\left[\max\left(\max_{n\in\Omega_{neg}} d_{\mathbf{x}_i}^{y_n^*} - \min_{p\in\Omega_{pos}} d_{\mathbf{x}_i}^{y_p^*} + \epsilon, 0\right)\right], \tag{5}$$

where Ω_{pos} and Ω_{neg} are the sets of relevant and irrelevant labels respectively, $d_{\boldsymbol{x}_i}^{y_n^*}$ and $d_{\boldsymbol{x}_i}^{y_p^*}$ are predicted values for label enhancement and ϵ is the threshold.

3) The Label Distribution Strategy. The third strategy considers that the output of the predictive model is useful for training the label distribution estimator. Specifically, it optimizes the Kullback-Leibler (KL) divergence between the output of the predictive model and the label distribution estimator:

$$\mathcal{L}_{LE} = \frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{c}\mathcal{P}_i^{(j)}\log\frac{\mathcal{P}_i^{(j)}}{d_{\mathbf{x}_i}^{y_j}}, \tag{6}$$

where $\mathcal{P}$ is the j-th element of the output of the predictive model for instance i through the softmax operation.

Matching and Interaction Mechanism

Using label distribution to guide multi-label model training faces an urgent problem, i.e., how to ensure that the generated label distribution is the recovered label distribution required for predictive model training. The existing label enhancement approaches adopt two-stage or iterative methods, which leads to the separation of the training processes of the predictive model and the label distribution estimator. In fact, the label distribution estimator can use the information of the predictive model to guide its own training, so as to realize matching and interaction.

The most important problem is how the label distribution estimator interacts with the predictive model. An efficient way is to align their training processes to achieve training matching. Inspired by asymmetric focal loss [Ridnik *et al.*, 2021] which is a powerful approach to balance the importance of different instances, we design the interaction label enhancement loss for each training instance $\boldsymbol{x}_i$:

$$L_{IN} = -\frac{1}{c}\sum_{j=1}^{c}\begin{cases}\left(1-p_j\right)^{\gamma+}\log\left(p_j^*\right), & j\in\Omega_{pos}\\ \left(p_j\right)^{\gamma-}\log\left(1-p_j^*\right), & j\in\Omega_{neg}\end{cases}, \tag{7}$$

where p_j is the prediction of the predictive model through the sigmoid operation, p_j^* is the prediction of the label distribution estimator through the sigmoid operation, $\gamma+$ and $\gamma-$ are two hyper-parameters to control each part of the loss function.

Algorithm 1 FLEM algorithm

```
Input: The MLL training set $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{l}_i)\}_{i=1}^n$, the number of epoch $T$, the number of iteration $I$, hyper-parameters $\alpha$ and $\beta$;
1: Initialize the parameters of the label distribution estimator $g(\cdot)$ and the predictive model $f(\cdot)$.
2: for $t = 1, ..., T$ do
3:     Shuffle training set $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{l}_i)\}_{i=1}^n$ into $I$ mini-batches;
4:     for $k = 1, ..., I$ do
5:         Obtain label distribution $\boldsymbol{d}_i$ for each example $\boldsymbol{x}_i$ by Eq. (1) and the softmax operation;
6:         Update $f$ and $g$ by forward computation and back-propagation by fusing Eq. (8) and Eq. (10);
7:     end for
8: end for
Output: The predictive model $f(\cdot)$.
```

The interaction loss matches the label distribution estimator with the training process of the predictive model from the perspective of instances. Specifically, it can distinguish "easy" instances and "hard" instances, and place a higher weight of loss on "hard-to-distinguish", which may have more inter-class information. With the knowledge from both models, the label distribution estimator can effectively generate label distributions that meet the needs of the training process on the predictive model. Then the whole training loss for the label distribution estimator can be given:

$$\mathcal{L}_{LDE} = \alpha\mathcal{L}_{LE} + (1-\alpha)\,\mathcal{L}_{IN}. \tag{8}$$

3.3 Classifier Training

Unified Label Distribution Loss

With the recovered label distribution vector $\boldsymbol{d}_i$, the implicit relative importance information can be can utilized to guide the training of the predictive model. Since the label distribution is introduced, we can optimize the predictive model from a global perspective. Specifically, we minimize the following unified label distribution loss to guide the training of $f(\cdot)$:

$$\mathcal{L}_{LD} = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{c} d_{\boldsymbol{x}_i}^{y_j}\log d_{\boldsymbol{x}_i}^{y_j}\left(e^{Z_i - s_i^{(j)}}\right)\right), \tag{9}$$

where n is the number of instances, $s_i^{(j)}$ is the output logit of the predictive model for the i-th instance and $Z_i = \sum_{k=1}^{c} s_i^{(k)}$. The unified label distribution loss $\mathcal{L}_{LD}$ establishes a corresponding relationship between the recovered label distribution vector and the output of the model. Compared with the losses of existing MLL approaches, it is more effective by using the distribution information of labels. In particular, we have the following theorem.

Theorem 1. *$\mathcal{L}_{LD}$ gives an upper bound for cross-entropy loss.*

Since logical labels are effectively extended to label distribution, our unified label distribution loss is a generalization form of cross-entropy loss. It also indicates that if accurate label distribution information can be obtained, it can effectively guide the training of the model.

Dataset	N_{train}	N_{val}	N_{test}	D	L	F
AAPD	53840	1000	1000	512	54	text
Reuters	5827	1943	3019	512	90	text
VOC07	2501	2510	4952	2048	20	image
VOC12	5516	200	5823	2048	20	image
COCO14	78081	4000	40137	2048	80	image
COCO17	113266	4000	4952	2048	80	image
CUB	5794	200	5794	2048	312	image
NUS	146000	4000	60260	2048	81	image

Table 1: Characteristics of multi-label datasets. N_{train}, N_{val}, N_{test}, D, L denote the number of training samples, validation samples, testing samples, feature dimensions and labels respectively. F denotes the domain of the dataset.

Training Process

As mentioned in Section 3.1, in FLEM, the predictive model $f(\cdot)$ and the label distribution estimator $g(\cdot)$ are trained simultaneously. For the training of the predictive model, we not only concern the distance between the recovered distribution and the expected label distribution, but also the sign consistency of them. It leads to the minimization of

$$\mathcal{L}_{CLS} = \beta\mathcal{L}_{LD} + (1-\beta)\,\mathcal{L}_{SC}, \tag{10}$$

where $\mathcal{L}_{SC}$ can be any loss function that can lead to sign consistency. For generality, we adopt the vanilla binary cross-entropy loss:

$$L_{SC} = -\frac{1}{c}\sum_{j=1}^{c}\begin{cases}\log(p_j), & j\in\Omega_{pos}\\ \log(1-p_j), & j\in\Omega_{neg}\end{cases}, \tag{11}$$

where p_j is the prediction of the predictive model through the sigmoid operation. The gradient-based optimization method can be used to optimize the model parameters. It should be noticed that when using Eq.(10) to update $f(\cdot)$, it will not update $g(\cdot)$, i.e. the stop-gradient operation is performed on each term about $g(\cdot)$ for Eq.(10). Meanwhile, using Eq.(8) to update $g(\cdot)$ will not update $f(\cdot)$ either.

In FLEM, f is set as linear classifier, g_F, g_L and g_D are set as three-layer-neural networks with a residual structure. In the prediction stage, given a new instance $\boldsymbol{x}^*$, the prediction of the model can be obtained by $f(\boldsymbol{x}^*)$.

4 Experiments

In this section, the efficiency and the performance of FLEM are evaluated in multiple MLL datasets. All methods are implemented by PyTorch. All the computations are performed on a GPU server with NVIDIA Tesla V100, Intel Xeon Gold 6240 CPU 2.60 GHz processor and 32 GB GPU memory. Our code is available at: https://github.com/ailearn-ml/FLEM.

4.1 Datasets and Preprocessing

To evaluate the proposed approach, we conduct experiments on several real-world datasets, including AAPD [Yang *et al.*, 2018], Reuters [Debole and Sebastiani, 2005], VOC07, VOC12 [Everingham *et al.*, 2015], COCO14, COCO17 [Lin *et al.*, 2014], CUB [Wah *et al.*, 2011] and NUS [Chua *et al.*,

Algorithm	AAPD	Reuters	VOC07	VOC12	COCO14	COCO17	CUB	NUS	Avg. Rank
ML-KNN [Zhang and Zhou, 2007]	0.0448(11)	0.0138(12)	0.0281(11)	0.0323(11)	0.0363(11)	0.0369(11)	0.1011(11)	0.0232(11)	11.1
RAKEL [Tsoumakas *et al.*, 2011]	0.0861(12)	0.0094(11)	0.0586(12)	0.0801(12)	0.1205(12)	0.1071(12)	0.2244(12)	0.0869(12)	11.9
FL [Lin *et al.*, 2020]	0.0282(7)	0.0042(5)	0.0249(7)	0.0235(6)	0.0196(6)	0.0218(8)	0.0867(8)	0.0145(5)	6.5
CB [Cui *et al.*, 2019]	0.0294(10)	0.0043(6)	0.0249(7)	0.0240(9)	0.0195(5)	0.0221(10)	0.0867(8)	0.0147(7)	7.8
R-FL [Wu *et al.*, 2020]	0.0281(5)	0.0043(6)	0.0254(9)	0.0239(8)	0.0197(8)	0.0214(6)	0.0865(7)	0.0147(7)	7.0
DB [Wu *et al.*, 2020]	0.0286(8)	0.0043(6)	0.0255(10)	0.0256(10)	0.0199(9)	0.0214(6)	0.0862(6)	0.0145(5)	7.5
BS [Zhang *et al.*, 2021]	0.0288(9)	0.0041(3)	0.0244(5)	0.0238(7)	0.0200(10)	0.0219(9)	0.0870(10)	0.0143(4)	7.1
LEVI [Xu *et al.*, 2020]	0.0281(5)	0.0045(9)	0.0244(5)	0.0231(4)	0.0196(6)	0.0209(2)	0.0852(4)	0.0156(10)	5.6
VALEN [Xu *et al.*, 2021b]	0.0276(3)	0.0045(9)	0.0242(3)	0.0229(3)	0.0194(4)	0.0209(2)	0.0857(5)	0.0153(9)	4.8
FLEM-S (Ours)	0.0276(3)	**0.0040(1)**	0.0239(2)	**0.0226(1)**	**0.0191(1)**	**0.0207(1)**	0.0850(3)	0.0138(2)	1.8
FLEM-T (Ours)	**0.0271(1)**	**0.0040(1)**	0.0243(4)	0.0228(2)	0.0192(2)	0.0209(2)	0.0849(2)	0.0139(3)	2.1
FLEM-D (Ours)	**0.0271(1)**	0.0041(3)	**0.0237(1)**	0.0231(4)	0.0192(2)	0.0209(2)	**0.0848(1)**	**0.0137(1)**	1.9

Table 2: Performance of each comparing algorithm on *Hamming loss* (↓).

Algorithm	AAPD	Reuters	VOC07	VOC12	COCO14	COCO17	CUB	NUS	Avg. Rank
ML-KNN [Zhang and Zhou, 2007]	0.0677(11)	0.0324(12)	0.0531(11)	0.0612(11)	0.0809(11)	0.0730(11)	0.1721(11)	0.0370(11)	11.1
RAKEL [Tsoumakas *et al.*, 2011]	0.0742(12)	0.0232(11)	0.0598(12)	0.0807(12)	0.1177(12)	0.1071(12)	0.2395(12)	0.0806(12)	11.9
FL [Lin *et al.*, 2020]	0.0555(7)	0.0113(8)	0.0228(9)	0.0160(8)	0.0264(7)	0.0331(9)	0.1099(10)	0.0185(6)	8.0
CB [Cui *et al.*, 2019]	0.0598(9)	0.0111(6)	0.0222(8)	0.0156(7)	0.0265(8)	0.0319(8)	0.1092(9)	0.0185(6)	7.6
R-FL [Wu *et al.*, 2020]	0.0589(8)	0.0123(10)	0.0242(10)	0.0165(10)	0.0265(8)	0.0318(7)	0.1088(8)	0.0184(5)	8.3
DB [Wu *et al.*, 2020]	0.0620(10)	0.0099(5)	0.0216(7)	0.0161(9)	0.0276(10)	0.0331(9)	0.1084(7)	0.0187(8)	8.1
BS [Zhang *et al.*, 2021]	0.0541(6)	0.0095(4)	0.0201(5)	0.0155(6)	0.0249(6)	0.0299(6)	0.1077(6)	0.0160(4)	5.4
LEVI [Xu *et al.*, 2020]	0.0486(3)	0.0118(9)	0.0205(6)	0.0146(4)	0.0237(2)	0.0298(5)	0.1057(3)	0.0210(9)	5.1
VALEN [Xu *et al.*, 2021b]	0.0489(4)	0.0111(6)	0.0199(3)	0.0146(4)	0.0245(5)	0.0297(4)	0.1076(5)	0.0210(9)	5.0
FLEM-S (Ours)	0.0522(5)	0.0092(2)	0.0194(2)	0.0140(3)	0.0239(4)	0.0286(2)	0.1058(4)	0.0149(3)	3.1
FLEM-T (Ours)	0.0483(2)	0.0094(3)	0.0199(3)	**0.0138(1)**	0.0238(3)	0.0286(2)	0.1048(2)	0.0148(2)	2.3
FLEM-D (Ours)	**0.0432(1)**	**0.0091(1)**	**0.0189(1)**	0.0139(2)	**0.0231(1)**	**0.0279(1)**	**0.1039(1)**	**0.0139(1)**	1.1

Table 3: Performance of each comparing algorithm on *Ranking loss* (↓).

2009]. For AAPD and Reuters, we use the feature extracted by pre-trained XML-CNN [Liu *et al.*, 2017]. For the other datasets, ResNet-50 pre-trained on ImageNet is used to extract the feature of instances. Following common practices [Liu *et al.*, 2017; Lanchantin *et al.*, 2021], these datasets are split into training set, validation set and testing set. Statistics of these real-world datasets are given in Table 1.

4.2 Comparing Algorithms and Evaluation Metrics

We compare our methods with several traditional and state-of-the-art MLL and LE technologies. The compared methods include: 1) ML-KNN [Zhang and Zhou, 2007]: a classical approach, which adopts kNN into MLL. 2) *Random k-labelsets* (RAKEL) [Tsoumakas *et al.*, 2011]: a high-order approach which transforms MLL into an ensemble of multi-class learning problems. 3) *Focal loss* (FL) [Lin *et al.*, 2020]: a simple but widely used strategy for classification. 4) *Class-balanced loss* (CB) [Cui *et al.*, 2019]: a class-wise re-weighting guided by the effective number of each class $E_n = (1-\beta^n)/(1-\beta)$. 5) *Rebalanced focal loss* (R-FL) [Wu *et al.*, 2020]: a combination of re-balanced weighting and focal loss. 6) *Distribution balance loss* (DB) [Wu *et al.*, 2020]: a recently proposed distribution-balanced loss for MLL. 7) *Balanced softmax method* (BS) [Zhang *et al.*, 2021]: a balanced loss function for MLL. 8) *Label enhancement via variational inference* (LEVI) [Xu *et al.*, 2020]: an advanced technology for LE-based MLL, which has been proved to be effectively applied to MLL. 9) *VAriational Label ENhancement* (VALEN) [Xu *et al.*, 2021b]: a recent LE technology, whose training strategy for the predictive model is set to the same strategy as LEVI in our experiments. Among them, ML-KNN and RAKEL are representative traditional MLL algorithms, LEVI and VALEN are advanced LE-based MLL technologies, and the others are recent methods proposed for MLL. For all of these state-of-the-art algorithms, we train linear classifiers for classification tasks. The optimization process spans over 30 epochs using the AMSGrad variant [Reddi *et al.*, 2018] of AdamW [Loshchilov and Hutter, 2017] with a weight decay of 0.0001. The learning rate is set to 0.001 for all algorithms. For FLEM, hyperparameters α and β are are selected by grid search from the set $\{0.0001, 0.001, 0.01, 0.1\}$.

Four widely-used MLL evaluation metrics are selected in this experiment, i.e., *Hamming loss*, *Ranking loss*, *Coverage* and *Average precision*. For each algorithm, we select the model that obtains the optimal *Hamming loss* on the validation set for testing.

4.3 Performance Comparison

Tables 2, 3, 4 and 5 tabulate the experimental results of different algorithms on the eight MLL datasets evaluated by four evaluation metrics, and the best performance on each dataset is highlighted by boldface. In Tables 2, 3, 4 and 5, "FLEM-S", "FLEM-T" and "FLEM-D" refer to the FLEM algorithm using the three strategies proposed in section 3.2 respectively. For each evaluation metric, "↓" indicates the smaller the better while "↑" indicates the larger the better. The ranks are given in the parentheses right after the performance values. The average rank of each algorithm over all the datasets is also calculated and given in the last row of each table.

From Tables 2, 3, 4 and 5, it can be observed that FLEM is superior to existing MLL algorithms in almost all cases.

Algorithm	AAPD	Reuters	VOC07	VOC12	COCO14	COCO17	CUB	NUS	Avg. Rank
ML-KNN [Zhang and Zhou, 2007]	0.1554(12)	0.0471(12)	0.0857(12)	0.0977(11)	0.1798(12)	0.1703(12)	0.5943(11)	0.0793(11)	11.6
RAKEL [Tsoumakas *et al.*, 2011]	0.1337(8)	0.0337(11)	0.0848(11)	0.1055(12)	0.1792(11)	0.1660(11)	0.6889(12)	0.1070(12)	11.0
FL [Lin *et al.*, 2020]	0.1273(7)	0.0213(8)	0.0514(9)	0.0445(8)	0.0899(7)	0.1052(9)	0.5303(10)	0.0457(7)	8.1
CB [Cui *et al.*, 2019]	0.1340(9)	0.0200(6)	0.0507(7)	0.0441(7)	0.0901(8)	0.1035(7)	0.5199(8)	0.0451(6)	7.3
R-FL [Wu *et al.*, 2020]	0.1357(10)	0.0230(10)	0.0537(10)	0.0457(10)	0.0902(9)	0.1051(8)	0.5152(5)	0.0448(5)	8.4
DB [Wu *et al.*, 2020]	0.1406(11)	0.0194(5)	0.0513(8)	0.0451(9)	0.0939(10)	0.1080(10)	0.5246(9)	0.0460(8)	8.8
BS [Zhang *et al.*, 2021]	0.1239(6)	0.0180(3)	0.0478(4)	0.0439(6)	0.0861(5)	0.0980(4)	**0.5016(1)**	0.0402(4)	4.1
LEVI [Xu *et al.*, 2020]	0.1193(4)	0.0218(9)	0.0483(6)	0.0428(4)	0.0841(3)	0.1000(5)	0.5175(6)	0.0489(9)	5.8
VALEN [Xu *et al.*, 2021b]	0.1180(3)	0.0207(7)	0.0478(4)	0.0429(5)	0.0864(6)	0.1002(6)	0.5190(7)	0.0495(10)	6.0
FLEM-S (Ours)	0.1217(5)	0.0182(4)	0.0474(2)	0.0420(2)	0.0841(3)	0.0965(3)	0.5072(3)	0.0381(2)	3.0
FLEM-T (Ours)	0.1160(2)	**0.0173(1)**	0.0476(3)	**0.0417(1)**	0.0839(2)	0.0962(2)	0.5092(4)	0.0383(3)	2.3
FLEM-D (Ours)	**0.1065(1)**	0.0177(2)	**0.0464(1)**	0.0420(2)	**0.0826(1)**	**0.0948(1)**	0.5060(2)	**0.0368(1)**	1.4

Table 4: Performance of each comparing algorithm on *Coverage* (↓).

Algorithm	AAPD	Reuters	VOC07	VOC12	COCO14	COCO17	CUB	NUS	Avg. Rank
ML-KNN [Zhang and Zhou, 2007]	0.4450(11)	0.3307(12)	0.6949(11)	0.6472(11)	0.3520(11)	0.3844(11)	0.1273(12)	0.2850(11)	11.3
RAKEL [Tsoumakas *et al.*, 2011]	0.3519(12)	0.3314(11)	0.5606(12)	0.4590(12)	0.2076(12)	0.2534(12)	0.1765(11)	0.1525(12)	11.8
FL [Lin *et al.*, 2020]	0.4994(5)	0.3800(9)	0.8341(9)	0.8656(8)	0.6994(7)	0.6724(8)	0.2903(8)	0.4927(7)	7.6
CB [Cui *et al.*, 2019]	0.4839(8)	0.3804(8)	0.8370(8)	0.8654(9)	0.6996(6)	0.6715(9)	0.2905(7)	0.4916(8)	7.9
R-FL [Wu *et al.*, 2020]	0.4816(9)	0.3684(10)	0.8327(10)	0.8650(10)	0.6981(8)	0.6741(7)	0.2902(9)	0.4845(10)	9.1
DB [Wu *et al.*, 2020]	0.4683(10)	0.3837(7)	0.8449(5)	0.8658(7)	0.6955(9)	0.6744(6)	0.2942(4)	0.4872(9)	7.1
BS [Zhang *et al.*, 2021]	0.4918(7)	0.4021(3)	0.8439(6)	0.8679(6)	0.6870(10)	0.6667(10)	0.2849(10)	0.4940(6)	7.3
LEVI [Xu *et al.*, 2020]	0.5018(3)	0.3936(4)	0.8421(7)	0.8699(4)	0.7021(4)	0.6837(4)	0.2935(6)	0.5063(5)	4.6
VALEN [Xu *et al.*, 2021b]	0.4971(6)	0.3842(6)	0.8453(3)	**0.8702(1)**	0.7026(3)	0.6836(5)	0.2941(5)	0.5075(4)	4.1
FLEM-S (Ours)	0.5006(4)	**0.4024(1)**	0.8458(2)	0.8700(3)	0.7027(2)	0.6852(2)	0.2951(3)	0.5109(3)	2.5
FLEM-T (Ours)	**0.5053(1)**	**0.4024(1)**	0.8450(4)	**0.8702(1)**	**0.7028(1)**	0.6841(3)	0.2953(2)	0.5113(2)	1.9
FLEM-D (Ours)	0.5040(2)	0.3910(5)	**0.8478(1)**	0.8697(5)	0.7020(5)	**0.6855(1)**	**0.2961(1)**	**0.5143(1)**	2.6

Table 5: Performance of each comparing algorithm on *Average precision* (macro-averaged) (↑).

Algorithm	Hamming loss ↓	Ranking loss ↓	Coverage ↓	Average precision ↑
FLEM-I	0.0242	0.0210	0.0492	0.8441
FLEM-II	0.0241	0.0198	0.0472	0.8444
Ours	**0.0237**	**0.0189**	**0.0464**	**0.8478**

Table 6: Performance of FLEM in different cases on VOC07.

This result indicates that the label enhancement method in FLEM is effective and can help the model significantly improve classification performance. Meanwhile, compared with the advanced LE methods, FLEM has achieved better results in almost all metrics. This observation demonstrates that the matching and interaction mechanism proposed in this paper is effective and can well match the label enhancement with the model training to improve the performance of the model.

When looking at the average ranks over all the eight real-world datasets, FLEM achieves rather competitive performance over other algorithms. All three methods proposed in this paper can exceed existing methods. When compared with these state-of-the-art algorithms, FLEM achieves 1st in 96.9% cases. Thus, FLEM possesses rather superior performance over the state-of-the-art algorithms across all the evaluation measures.

4.4 Ablation Study

We further conduct an ablation study to verify the effectiveness of the unified label distribution loss and the interaction label enhancement loss. Based on the *Label Distribution* strategy proposed in section 3.2, we evaluate three cases of our method: 1) FLEM-I: without the unified label distribution loss, 2) FLEM-II: without the interaction label enhancement loss and 3) Ours: the proposed FLEM. Table 6 tabulates the experimental results of different versions on the VOC07 dataset, and the best performance on each comparison is highlighted by boldface. From Table 6, we can find that both the interaction label enhancement loss and the unified label distribution loss are beneficial to improve the performance, which proves the validity of the proposed matching and interaction mechanism and the unified label distribution loss.

5 Conclusions

This paper proposes a novel label enhancement method for multi-label learning. By utilizing a novel matching and interaction mechanism, the mismatch between label enhancement and classifier training is effectively avoided. In the meantime, a novel unified label distribution loss is proposed for guiding MLL models to learn with the recovered label distributions. Extensive experimental results demonstrate that FLEM can bring significant performance improvements over the existing MLL methods. In future work, we will further explore the theory of FLEM, promote the versatility of FLEM and design more effective methods.

Acknowledgments

This research was supported by the National Key Research & Development Plan of China (No. 2018AAA0100104, No. 2018AAA0100100), the National Science Foundation of China (62125602, 62076063), China Postdoctoral Science Foundation (2021M700023), Jiangsu Province Science Foundation for Youths (BK20210220).

References

[Boutell *et al.*, 2004] Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 2004.

[Chua *et al.*, 2009] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. NUS-WIDE: a real-world web image database from national university of singapore. In *CIVR*, 2009.

[Cui *et al.*, 2019] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge J. Belongie. Class-balanced loss based on effective number of samples. In *CVPR*, 2019.

[Debole and Sebastiani, 2005] Franca Debole and Fabrizio Sebastiani. An analysis of the relative hardness of reuters-21578 subsets. *JASIST*, 2005.

[Everingham *et al.*, 2015] Mark Everingham, Seyed Mohammadali Eslami, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 2015.

[Fürnkranz *et al.*, 2008] Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. Multilabel classification via calibrated label ranking. *Machine Learning*, 2008.

[Gao *et al.*, 2017] Bin-Bin Gao, Chao Xing, Chen-Wei Xie, Jianxin Wu, and Xin Geng. Deep label distribution learning with label ambiguity. *TIP*, 2017.

[Geng, 2016] Xin Geng. Label distribution learning. *TKDE*, 2016.

[Huang *et al.*, 2021] Yi Huang, Buse Giledereli, Abdullatif Köksal, Arzucan Özgür, and Elif Ozkirimli. Balancing methods for multi-label text classification with long-tailed class distribution. In *EMNLP*, 2021.

[Lanchantin *et al.*, 2021] Jack Lanchantin, Tianlu Wang, Vicente Ordonez, and Yanjun Qi. General multi-label image classification with transformers. In *CVPR*, 2021.

[Lin *et al.*, 2014] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and Charles Lawrence Zitnick. Microsoft COCO: common objects in context. In *ECCV*, 2014.

[Lin *et al.*, 2020] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *TPAMI*, 2020.

[Liu *et al.*, 2017] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. Deep learning for extreme multi-label text classification. In *SIGIR*, 2017.

[Liu *et al.*, 2021] Weiwei Liu, Haobo Wang, Xiaobo Shen, and Ivor W. Tsang. The emerging trends of multi-label learning. *TPAMI*, 2021.

[Lo *et al.*, 2011] Hung-Yi Lo, Ju-Chiang Wang, Hsin-Min Wang, and Shou-De Lin. Cost-sensitive multi-label learning for audio tag annotation and retrieval. *TMM*, 2011.

[Loshchilov and Hutter, 2017] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, 2017.

[Reddi *et al.*, 2018] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *ICLR*, 2018.

[Ridnik *et al.*, 2021] Tal Ridnik, Emanuel Ben Baruch, Nadav Zamir, Asaf Noy, Itamar Friedman, Matan Protter, and Lihi Zelnik-Manor. Asymmetric loss for multi-label classification. In *ICCV*, 2021.

[Shao *et al.*, 2018] Ruifeng Shao, Ning Xu, and Xin Geng. Multi-label learning with label enhancement. In *ICDM*, 2018.

[Tsoumakas *et al.*, 2011] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis P. Vlahavas. Random k-labelsets for multilabel classification. *TKDE*, 2011.

[Wah *et al.*, 2011] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. *California Institute of Technology*, 2011.

[Wang *et al.*, 2019] Haobo Wang, Weiwei Liu, Yang Zhao, Chen Zhang, Tianlei Hu, and Gang Chen. Discriminative and correlative partial multi-label learning. In *IJCAI*, 2019.

[Wang *et al.*, 2020] Haobo Wang, Zhao Li, Jiaming Huang, Pengrui Hui, Weiwei Liu, Tianlei Hu, and Gang Chen. Collaboration based multi-label propagation for fraud detection. In *IJCAI*, 2020.

[Wu *et al.*, 2020] Tong Wu, Qingqiu Huang, Ziwei Liu, Yu Wang, and Dahua Lin. Distribution-balanced loss for multi-label classification in long-tailed datasets. In *ECCV*, 2020.

[Xu *et al.*, 2020] Ning Xu, Jun Shu, Yun-Peng Liu, and Xin Geng. Variational label enhancement. In *ICML*, 2020.

[Xu *et al.*, 2021a] Ning Xu, Yun-Peng Liu, and Xin Geng. Label enhancement for label distribution learning. *TKDE*, 2021.

[Xu *et al.*, 2021b] Ning Xu, Congyu Qiao, Xin Geng, and Min-Ling Zhang. Instance-dependent partial label learning. In *NeurIPS*, 2021.

[Yang *et al.*, 2018] Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. SGM: sequence generation model for multi-label classification. In *COLING*, 2018.

[Zhang and Zhou, 2007] Min-Ling Zhang and Zhi-Hua Zhou. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, 2007.

[Zhang and Zhou, 2014] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *TKDE*, 2014.

[Zhang *et al.*, 2021] Ningyu Zhang, Xiang Chen, Xin Xie, Shumin Deng, Chuanqi Tan, Mosha Chen, Fei Huang, Luo Si, and Huajun Chen. Document-level relation extraction as semantic segmentation. In *IJCAI*, 2021.