Probabilistic Label Tree for Streaming Multi-Label Learning

Tong Wei, Jiang-Xin Shi, Yu-Feng Li*
National Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210023, China
{weit,shijx,liyf}@lamda.nju.edu.cn

ABSTRACT

Multi-label learning aims to predict a subset of relevant labels for each instance, which has many real-world applications. Most extant multi-label learning studies focus on a fixed size of label space. However, in many cases, the environment is open and changes gradually and new labels emerge, which is coined as streaming multi-label learning (SMLL). SMLL poses great challenges in twofolds: (1) the target output space expands dynamically; (2) new labels emerge frequently and can reach a significantly large number. Previous attempts on SMLL leverage label correlations between past and emerging labels to improve the performance, while they are inefficient when deal with large-scale problems. To cope with this challenge, in this paper, we present a new learning framework, i.e., the probabilistic streaming label tree (Pslt). In particular, each non-leaf node of the tree corresponding to a subset of labels, and a binary classifier is learned at each leaf node. Initially, Pslt is learned on partially observed labels, both tree structure and node classifiers are updated while new labels emerge. Using carefully designed updating mechanism, Pslt can seamlessly incorporate new labels by first passing them down from the root to leaf nodes and then update node classifiers accordingly. We provide theoretical bounds for the iteration complexity of tree update procedure and the estimation error on newly arrived labels. Experiments show that the proposed approach improves the performance in comparison with eleven baselines in terms of multiple evaluation metrics. The source code is available at https://gitee.com/pslt-kdd2021/pslt.

CCS CONCEPTS

• **Computing methodologies → Supervised learning**.

KEYWORDS

multi-label learning, streaming label learning

ACM Reference Format:
Tong Wei, Jiang-Xin Shi, Yu-Feng Li. 2021. Probabilistic Label Tree for Streaming Multi-Label Learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467226

1 INTRODUCTION

Multi-label learning (MLL) [7, 37] aims to annotate objects with the relevant labels from an extremely large number of candidate labels. MLL recently owns many real-world applications. For example, in webpage categorization [17], labels (categories) are collected in Wikipedia and one needs to annotate a new webpage with all relevant labels from such a big candidate set; in image annotation [4], one wishes to tag each individual picture with multiple labels from such a candidate label set; in recommendation systems [14], one hopes to make informative personalized recommendations from a candidate set of items (labels).

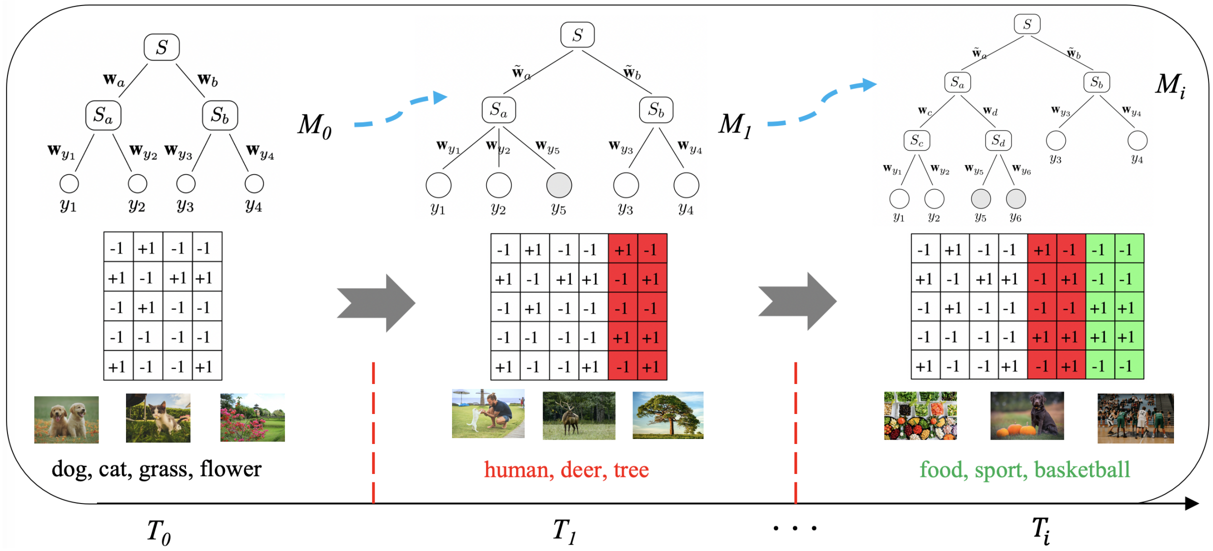
Most existing literature of multi-label learning focuses on a fixed set of labels [1, 13, 20, 26–29, 34, 36, 37]. That is, they assume that all the labels in the learning process are given at once. In the opposite, this paper investigates an important problem in data streams, i.e., multi-label classification under streaming emerging labels, since in many real-world applications, the environment is open and changes gradually. For instance, in webpage categorization, new labels (categories) are created for historic events in Wikipedia, such as the "2022 Winter Olympics"; in recommendation systems, many new labels (items) are becoming available to users. If multi-label models do not take new labels into account, it deteriorates the performance.

The main challenges of SMLL are in twofolds: (1) the target output space expands dynamically; (2) new labels emerge frequently and can reach a significantly large number. To cope with these problems, one of previous works SLL [33] assumes that each label can be represented as a linear combination of other labels and it learns a binary linear classifier for each new label incorporating the relationship with past labels. However, the linear representation across labels limits the SLL performance, and the training knowledge from classifiers of past labels is neglected. Recently, DSLL [25] proposes a novel DNN-based framework to effectively model the correlation between past labels and emerging new labels. DSLL has the ability to explore and utilize deep correlations between new labels and past labels without computationally intensive retraining. However, DSLL needs to train a separate model for newly arrived labels based on initial teacher model and has difficulties to deal with large-scale applications. Moreover, a unified model is desired in many cases, which evolves using emerging data stream.

In this paper, we propose a tree-based streaming multi-label learning framework, i.e., the Probabilistic Streaming Label Tree (Pslt), which can seamlessly take advantage of emerging new labels. In Pslt, labels are organized in the tree with the hope of capturing hierarchical correlations among labels. Our algorithm works by first partitioning the label space, such that each label corresponds to a leaf of the tree. In particular, a clustering algorithm is employed to determine the partition of labels and Pslt learns a binary classifier for each child node for the purpose of inference. This procedure repeats recursively until the stop condition is meet. While dealing

*Tong Wei and Jiang-Xin Shi contributed equally to the paper

KDD '21, August 14–18, 2021, Virtual Event, Singapore
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467226



-1	+1	-1	-1
+1	-1	+1	+1
-1	+1	-1	-1
-1	-1	-1	-1
+1	-1	-1	+1

-1	+1	-1	-1	+1	-1
+1	-1	+1	+1	-1	+1
-1	+1	-1	-1	-1	-1
-1	-1	-1	-1	+1	+1
+1	-1	-1	+1	-1	+1

-1	+1	-1	-1	+1	-1	-1	-1
+1	-1	+1	+1	-1	+1	-1	-1
-1	+1	-1	-1	-1	-1	+1	+1
-1	-1	-1	-1	+1	+1	+1	+1
+1	-1	-1	+1	-1	+1	-1	-1

Figure 1: Illustration of the streaming multi-label learning setup. At each timestamp T_i, the label matrix is expanded by a batch of emerging new labels, where each row represents the label vector for a sample and +1s or −1s respectively denote labels are relevant or irrelevant to the sample. M_i denotes the model at time T_i which evolves from M_{i-1} such that its output label space expands accordingly. We use tree-based models for the purpose of demonstration.

with emerging labels, PsLT evolves itself by considering both tree structure and node classifiers. It first performs the label insertion procedure which passes new labels from the root to leaf nodes guided by label proximity. Noted that if the size of a leaf node meets the size limit, an additional partition is triggered. After label insertion, node classifiers are learned or modified using emerging labels. In the experiments, we observe that PsLT outperforms eleven baselines on several datasets and it is far more effective than the deep learning based method DSLL. We demonstrate the studied SMLL setup in Figure 1.

We summarize our main contributions as follows.

- We study a general problem setup, i.e., the streaming multi-label learning. A new framework based on the probabilistic label tree is proposed to effectively update the model by leveraging label correlations.
- Analyses on the iteration complexity of the model update procedure and estimation error on emerging labels are provided. Memory complexity of the model is also presented.
- The proposed approach achieves state-of-the-art performance in comparison with eleven baselines in terms of multiple performance metrics on three benchmark multi-label datasets.

The rest of the paper is organized as follows. Section 2 introduces the previous related work. Section 3 gives the problem and describes the proposed framework in detail. Section 4 presents the theoretical analysis. Section 5 presents the experimental results validating the model and providing several insights. We conclude this work in Section 6.

2 RELATED WORK

Tree-based Multi-label Learning. Most extant multi-label learning approaches focus on fixed label space. Tree-based methods greatly reduce inference time, which generally scales logarithmically in the number of labels. There are typically two types of trees including instance trees [18, 21] and label trees [3, 10, 32], depending whether instance or label is partitioned in tree nodes. These methods can readily scale up to problems with hundreds of thousands of labels. Recently, OPLTs [8, 9] is proposed to learn in the online manner. In OPLTs, new labels traverse down from the root to leaves by randomly choosing a child node or calculating score that maximizes balancedness of tree and fitness of data, in which label correlations are neglected.

Learning with Emerging Classes. One of previous work SLL [33] assumes that: (1) a label is represented as a linear combination of other labels, and (2) the relationship (linear combination) between labels can be inherited by classifiers of different labels. Constrained by these hypotheses, SLL trains a linear classifier for new labels with the relationship between past labels and new labels. However, the linear representation across labels limits the SLL performance, and the training knowledge from classifiers of past labels is neglected. To alleviate this issue, DSLL [25] propose a novel DNN-based framework to effectively model the emerging new labels by using leveraging knowledge of previous trained model to learn high-order representations. From the point of view that new classes are outliers, [38] and [15] use anomaly detection to detect new classes and update models with emerging data, this is different from the streaming multi-label learning. [39] further extends this problem to semi-supervised learning for the presence of unlabeled samples in emerging data. In this paper, we show that it is natural and efficient to incorporate emerging new labels in the probabilistic label tree framework, which takes advantage of label correlations in the meanwhile.

3 PROBABILISTIC STREAMING LABEL TREE

3.1 Problem Formulation

We first introduce the streaming multi-label learning setup and some notations. Suppose that an initial training data set is denoted by $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1); ...; (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$, where $\boldsymbol{x}_i \in \mathbb{R}^D$ is a real vector representing an input feature (instance) and $\boldsymbol{y}_i \in \{0, 1\}^L$ is the corresponding output label vector for the i-th example. The input feature matrix is denoted as $\mathbf{X} = [\boldsymbol{x}_1; ...; \boldsymbol{x}_N] \in \mathbb{R}^{D\times N}$ and the initial output label matrix is $Y = [\boldsymbol{y}_1; ...; \boldsymbol{y}_N] \in \{0, 1\}^{L\times N}$. Note that, $Y_{ij} = 1$ if the j-th label is relevant to sample $\boldsymbol{x}_i$ and $Y_{ij} = -1$ otherwise. By observing $\mathcal{D}$, multi-label learning aims to derive a proper model that generates the prediction for a test sample. Then new labels arrive batch by batch in a streaming fashion. For simplicity, we denote $\boldsymbol{y}_i^{new} = [y_i^{L+1}; ...; y_i^{L+m}]$ as the newly observed m labels for sample $\boldsymbol{x}_i$, where $m \geq 1$. The goal of streaming multi-label learning is to derive *a unified model* by taking care of the continually emerging new labels. We summarize definition of notations used in the rest of this paper in Table 1.

Table 1: Notations.

Notations	Definition
N	size of training set
D	feature dimension
L	size of label set
$\boldsymbol{X} \in \mathbb{R}^{N\times D}$	feature matrix, $\boldsymbol{X} = [\boldsymbol{x}_1; \dots; \boldsymbol{x}_N]$
$Y \in \{0, 1\}^{N\times L}$	label matrix, $Y = [\boldsymbol{y}_1; \dots; \boldsymbol{y}_N]$
$\boldsymbol{V}$	label representation matrix, $\boldsymbol{V} = [\boldsymbol{v}_1; \dots; \boldsymbol{v}_L]$
l_j	the j-th label
T	a probabilistic streaming label tree
K	branch factor for tree node
$\mathcal{V}_T$	the set of nodes of T
$\mathcal{L}_T$	the set of leaf nodes of T
r_T	the root of T
L_v	number of leaf nodes in the subtree rooted at v
Path(v)	set of nodes in the path from root to node v
$\boldsymbol{c}_v$	the set of labels in node v
$\boldsymbol{w} \in \mathbb{R}^D$	node classifier
m	number of emerging new labels

3.2 Tree Construction

In this subsection, we describe the procedure of building probabilistic streaming label tree. Our approach processes one node of the tree a time, starting with the root node. At each non-leaf node, it partitions the labels into a fixed number of disjoint subsets as child nodes and learns a binary classifier for each of the children for the purpose of inference. The procedure might repeats multiple times.

Label Representation. During label partitioning, representations for labels are necessary, based on which label proximities are calculated. Ideally, labels with similar semantic meaning are desired to be grouped in same clusters. Since label semantic representations are usually unaccessible in many real-world applications. To this end, we provide a simple and effective walk around. Specifically, label representations are constructed by averaging all the training examples for which it is relevant to the corresponding labels. Formally, the label representation matrix $\boldsymbol{V} = [\boldsymbol{v}_1; \dots; \boldsymbol{v}_L]$ is given as $\boldsymbol{V} = \boldsymbol{Y}^\top \boldsymbol{X}$. Using the mean vector of positive instances as label representation has been used in many research fields, e.g., few-shot learning [22], which has demonstrated adorable performance.

Label Partitioning. The aim of label partitioning is to split $\mathcal{S}$ into disjoint subsets by taking label similarity into account. This is achieved by k-means clustering, which also presents many choices such as number of clusters and degree of balancedness among the clusters. The same process is repeated on each of the newly-created K child nodes in an iterative manner. Let $\boldsymbol{c}_k$ denotes the set of labels of the k-th cluster, the objective function of label partitioning can be formulated as

$$\min_{\boldsymbol{c}_1,\dots,\boldsymbol{c}_K} \sum_{k=1}^{K} \sum_{i\in \boldsymbol{c}_k} dist\left(\boldsymbol{v}_i, \boldsymbol{\mu}_k\right) \tag{1}$$

where $dist(\cdot,\cdot)$ represents a distance function and $\boldsymbol{\mu}_k$ denotes the center of the k-th cluster. The distance function is defined in terms of the cosine similarity as $dist\left(\boldsymbol{v}_i, \boldsymbol{\mu}_k\right) = 1 - \frac{\boldsymbol{v}_i^\top \boldsymbol{\mu}_k}{||\boldsymbol{v}_i||\cdot||\boldsymbol{\mu}_k||}$. Problem (1) is NP-hard and an approximate solution can be found using the standard K-means algorithm [12].

Learning Node Classifiers. Once the label space is partitioned into a tree structure, we learn a K-way One-vs-All classifier [6, 34] at each node of the tree. These classifiers are learned independently by considering only the training samples that have at least one positive label in the current node labels. We distinguish the leaf nodes and non-leaf nodes in the following way: (i) for non-leaf nodes, the classifier learns K linear classifiers separately, each maps to one of the K children. During prediction, the output of each classifier determines whether the test point should traverse down the corresponding child. (ii) for leaf nodes, the classifier learns to predict the actual labels on the node. The One-vs-All procedure is shown as *ONE-VS-ALL* in Algorithm 1. Formally, at each node, we consider the following optimization problem for learning linear classifier for each of the K child:

$$\min_{\boldsymbol{w}} f\left(\boldsymbol{y}, \boldsymbol{w}\right) := \mathcal{R}\left(\boldsymbol{w}\right) + C \sum_{i=1}^{N} \ell\left(y_i, \boldsymbol{w}^\top \boldsymbol{x}_i\right), \tag{2}$$

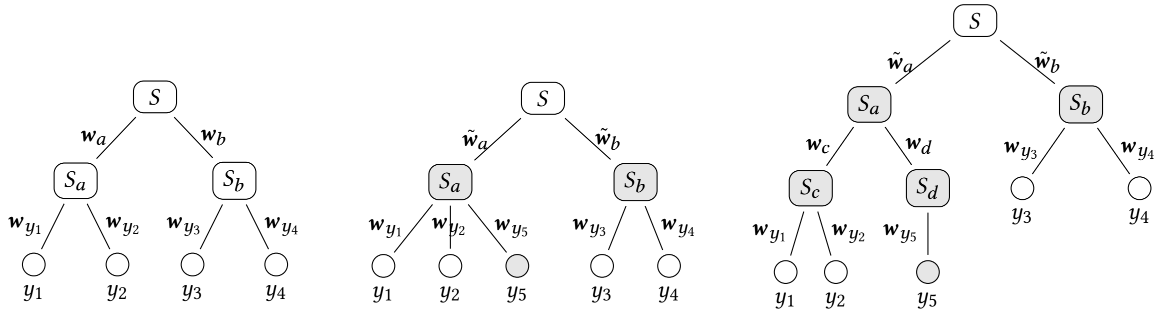
where $\{\boldsymbol{x}_i, y_i\}_{i=1}^N$ are constructed training data for a particular child node. For non-leaf nodes, $y_i = +1$ iff at least one relevant label of $\boldsymbol{x}_i$ is in the child node, otherwise -1. For leaf nodes, y_i denotes the true label of $\boldsymbol{x}_i$ with respect to label l_j. Concretely, if squared hinge loss, i.e., $\ell(y, \hat{y}) = \max\left(0, 1 - y \cdot \hat{y}\right)^2$, and L_2-regularization, i.e., $\mathcal{R}(\boldsymbol{w}) := ||\boldsymbol{w}||_2^2$, are used, the objective of linear SVM becomes,

$$\min_{\boldsymbol{w}} f\left(\boldsymbol{y}, \boldsymbol{w}\right) := ||\boldsymbol{w}||_2^2 + C \sum_{i=1}^{N} \max\left(0, 1 - y_i \boldsymbol{w}^\top \boldsymbol{x}_i\right)^2. \tag{3}$$

The above problem can be readily solved using off-the-shelf LIBLINEAR [5] which generally scales linearly with the number of training samples.

3.3 Inference

With the tree structure and node classifiers learned in the above manner, testing points are passed down from the root to leaf nodes.



(a) Tree T_{t-1} after $t-1$ iterations.

(b) Variant 1: A leaf node for emerging label y_5 is added as a child of an internal node S_a.

(c) Variant 2: A re-partition of internal node S_a is conducted after inserting emerging label y_5.

Figure 2: Two variants of tree extension for an emerging new label y_5. Nodes in gray indicate the modification of their node classifiers. We use w and $\tilde{w}$ to denote the previous and modified classifier weights, respectively.

```
Algorithm 1 TSLL.TRAIN($X, Y, \mathcal{L}, V, K, d_{max}$)
n = new NODE
$n.\mathcal{L} = [L]$
$n_T.v = \frac{1}{|\mathcal{L}|}\sum_{i \in \mathcal{L}} V_i$
$c_1, \dots, c_K$ = K-MEANS($\mathcal{L}, V, K$)
for $i = 1, \dots, K$ do
    $child.\mathcal{L} = c_i$
    $child.v = \frac{1}{|c_i|}\sum_{j \in c_i} V_j$
    $n.Ch = \emptyset$
    if $d_{max} > 0$ then
        n.isLeaf = FALSE
        child = TRAIN($c_i, V, K, d_{max} - 1$)
        $n.Ch = n.Ch \cup \{child\}$
    else
        n.isLeaf = TRUE
        $n.W$ = ONE-VS-ALL($X, Y, n.\mathcal{L}$)
    end if
end for
return r
```

Typically, at each node, the testing point x is evaluated by K binary classifiers corresponding to K child nodes. Let $z_v = \mathbb{I}\left(\sum_{j \in c_v} y_j \geq 1\right)$ denotes at least one relevant label of x is in the node labels c_v, provided that the true label y of x is observed. Then, it holds based on the chain rule that for any node $v \in \mathcal{V}_T$:

$$\eta_v(x) = \mathbb{P}(z_v = 1 \mid x) = \prod_{v' \in \mathrm{Path}(v)} \eta\left(x, v'\right), \tag{4}$$

where $\mathrm{Path}(v)$ denotes the path from root to node v and $\eta(x, v) = \mathbb{P}\left(z_v = 1 \mid z_{\mathrm{pa}(v)} = 1, x\right)$ for non-root nodes, where $\mathrm{pa}(v)$ denotes the parent node of v, and $\eta(x, v) = \mathbb{P}(z_v = 1 \mid x)$ for the root. Without loss of generality, we have $\eta(x, r_T) = \mathbb{P}\left(z_{r_T} = 1 \mid x\right) = 1$, that is, x is relevant to at least one label in the root labels. To calculate Equation (4), $\mathbb{P}\left(z_v = 1 \mid z_{\mathrm{pa}(v)} = 1, x\right)$ is estimated by node classifiers of v. Notice that for leaf nodes we get the conditional probabilities of labels, that is, $\eta_{l_j}(x) = \mathbb{P}\left(y_j = 1 \mid x\right), \forall l_j \in \mathcal{L}_T$ and $j \in [L]$.

3.4 Tree Update

In this subsection, we describe the way of updating pre-trained tree, which involves two parts, i.e., label insertion and node classifier modification.

Label Insertion. While inserting a new label y^{new}, we first calculate its representation from observed positive samples, which is hereby denoted as v^{new}. Then y^{new} is traversed downwards start from root of T. The passing direction is decided by the similarity between y^{new} and child nodes. In specific, the most proximate child node j is obtained by

$$j = \underset{i \in [K]}{\arg\max} \frac{v_i^\top v^{new}}{||v^{new}|| \cdot ||v_i||}. \tag{5}$$

We use cosine similarity as the proximity measure which is in consistent with the label partitioning procedure while building the tree. The label insertion procedure is done recursively until a leaf node is reached. To take case of newly arrived labels, we update node labels by inserting the new label to nodes which it passes by and node representations are updated accordingly. To do this on-the-fly, sum of representations of labels is maintained and the summation is updated by each newly arrived label. Note that it is not desired to have too many labels in a single leaf node because the time complexity of training a K-way One-Vs-All classifier is linear in the number of labels. To this end, the leaf node is further partitioned when the node label size meets a threshold. We illustrate two tree variants in Figure 2.

Node Classifier Modification. With emerging new labels inserted in proximate tree nodes, it is desired to update the node classifiers using collected data related to new labels. However, training classifiers for each node from scratch is computational prohibitive. We update node classifiers using previous trained ones as initialization. In particular, for node v, we use each of its K classifier w for initialization as $\tilde{w} = w$ and update $\tilde{w}$ using newly arrived samples via gradient-based optimization methods. After that, $\boldsymbol{w}$ is substituted by $\tilde{\boldsymbol{w}}$ for future use. For the newly inserted label, a binary classifier is desired for the purposed of inference.

Algorithm 2 TSLL.PREDICT($r_T, \boldsymbol{x}$)

```
$Q = \emptyset$
$Q.add((r_T, 1))$
$\hat{\boldsymbol{y}} = \mathbf{0}$
while $Q \neq \emptyset$ do
    $(v, \hat{\eta}_v(\boldsymbol{x})) = Q.pop()$
    if $v.isLeaf$ then
        $\hat{y}_v = \hat{\eta}_v(\boldsymbol{x})$
    else
        for $child \in v.Ch$ do
            $\hat{\eta}_{child}(\boldsymbol{x}) = g(v.\boldsymbol{W}_{child}^{\top}\boldsymbol{x})$
            $Q.add((child, \hat{\eta}_v(\boldsymbol{x}) \times \hat{\eta}_{child}(\boldsymbol{x})))$
        end for
    end if
end while
return $\hat{\boldsymbol{y}}$
```

Algorithm 3 TSLL.UPDATE($n, y^{new}, \boldsymbol{v}^{new}, \mathcal{A}$)

```
if $n.isLeaf$ then
    $n.\mathcal{L} = n.\mathcal{L} \cup \{y^{new}\}$
    $\boldsymbol{w}^{new} = trainBinaryClassifier(y^{new})$
    $n.\boldsymbol{W} = n.\boldsymbol{W} \cup \{\boldsymbol{w}^{new}\}$
else
    $cosim = 0$
    $nextCh = NONE$
    for $child \in n.Ch$ do
        $cosim' = \frac{child.\boldsymbol{v}^{\top}\boldsymbol{v}^{new}}{||child.\boldsymbol{v}|| \cdot ||\boldsymbol{v}^{new}||}$
        if $cosim' \geq cosim$ then
            $cosim = cosim'$
            $nextCh = child$
        end if
    end for
    for $\boldsymbol{w} \in n.\boldsymbol{W}$ do
        Initialize $\tilde{\boldsymbol{w}} = n.\boldsymbol{w}$
        Solve subproblem by $\mathcal{A}$ using $y^{new}$ and $\tilde{\boldsymbol{w}}$
        Set $n.\boldsymbol{w} = \tilde{\boldsymbol{w}}$
    end for
    UPDATE($nextCh, y^{new}, \boldsymbol{v}^{new}, \mathcal{A}$)
end if
```

4 THEORETICAL ANALYSIS

In this section, we first show that the node classifier modification procedure can be done effectively with an iteration complexity bound. Then, we establish an estimation error bound, which guarantees the learning performance of the tree. Finally, we analyze the total time and memory complexities of the proposed approach.

4.1 Analysis on the Iteration Complexity of Node Classifier Modification

With a little bit of notation abuse, Let $\boldsymbol{y}$ be meta-label vector used to train $\boldsymbol{w}$ and $\tilde{\boldsymbol{y}}$ be the modified label vector after inserting new labels. We further denote the discrepancy between $\boldsymbol{y}$ and $\tilde{\boldsymbol{y}}$ as $\ell_H(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \sum_{i=1}^{N} \mathbb{I}(\boldsymbol{y}_i \neq \tilde{\boldsymbol{y}}_i)$, and derive an upper bound for the classifier update procedure for tree node as follows.

Lemma 1. *Assume $\ell(\cdot)$ in Problem (2) is α-Lipschitz and the optimal solution before and after incorporating emerging labels be $\boldsymbol{w}^*$ and $\tilde{\boldsymbol{w}}^*$ respectively, which satisfy $||\boldsymbol{w}^*||_2 \leq B$ and $||\tilde{\boldsymbol{w}}^*||_2 \leq B$. Then*

$$f\left(\tilde{\boldsymbol{y}}, \boldsymbol{w}^*\right) - f\left(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{w}}^*\right) \leq 4C\alpha B \sum_{i=1}^{N} \mathbb{I}(\tilde{y}_i = 1) \wedge \mathbb{I}(y_i = -1)$$

To go a step further based on Lemma 1, we develop the following theorem to show that we could significantly reduce runtime upper bound with the initialization.

Theorem 1. *Assume $\ell(\cdot)$ is α-Lipschitz and we have a solver $\mathcal{A}$ with sublinear convergence rate that can solve each subproblem k with ϵ precision in $T_k = O\left(\frac{f\left(\tilde{\boldsymbol{y}}_k, \boldsymbol{w}_k^*\right) - f\left(\tilde{\boldsymbol{y}}_k, \tilde{\boldsymbol{w}}_k^*\right)}{\epsilon^p}\right)$ iterations. We can bound the total number of iterations of the node classifier modification procedure by $T_{total} = \sum_{k=1}^{K} T_k = O\left(\frac{\sum_{k=1}^{K} \ell_H(\tilde{\boldsymbol{y}}_k, \boldsymbol{y}_k)}{\epsilon^p}\right) = O\left(\frac{\Delta_{\boldsymbol{y}_k \tilde{\boldsymbol{y}}_k}}{\epsilon^p}\right)$, where $\Delta_{\boldsymbol{y}\tilde{\boldsymbol{y}}} = \sum_{i=1}^{N} \mathbb{I}(\tilde{y}_i = 1) \wedge \mathbb{I}(y_i = -1)$. However, under a naive zeros initialization with $\tilde{\boldsymbol{w}}_k^* = \mathbf{0}$, the upper bound is $T_{total} = \sum_{k=1}^{K} T_k = O\left(\frac{NK}{\epsilon^p}\right)$.*

4.2 L_1 Estimation Error Bound

In addition to the effectiveness of node classifier modification, we are interested in multi-label classifiers that estimate conditional probabilities of labels, $\eta_j = \mathbb{P}(y_j = 1 \mid \boldsymbol{x}), j \in [m]$, as accurately as possible, that is, with possibly small L_1-estimation error, $\left|\eta_j(\boldsymbol{x}) - \hat{\eta}_j(\boldsymbol{x})\right|$. According to [8], we show that the weighted expected L_1 estimation error averaged over all labels can be bounded by a weighted sum of expected L_1 error of node classifiers divided by the number of labels.

Theorem 2 (Estimation Error Bound [8]). *For any tree T, distribution $\mathbb{P}(\boldsymbol{x}, \boldsymbol{y})$, a set of emerging new label $\{\boldsymbol{x}_i, \boldsymbol{y}_i^{new}\}_{i=1}^{m}$, the expected L_1 estimation error bound holds:*

$$\frac{1}{m} \sum_{j=1}^{m} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}(\boldsymbol{x})} \left[\left|\eta_j(\boldsymbol{x}) - \hat{\eta}_j(\boldsymbol{x})\right|\right]$$
$$\leq \frac{1}{m} \sum_{v \in \mathcal{V}} |\mathcal{L}_v| \, \mathbb{P}(z_{\mathrm{pa}(v)} = 1) \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}(\boldsymbol{x} | z_{\mathrm{pa}(v)} = 1)} \left[\left|\eta\left(\boldsymbol{x}, v'\right) - \hat{\eta}\left(\boldsymbol{x}, v'\right)\right|\right]$$

where for the root node $\mathbb{P}(z_{pa(r_T)} = 1) = 1$.

4.3 Time and Memory Complexity Analysis

In this subsection, we provide bounds for the time and memory complexities of Pslt.

Lemma 2. *Under the case specified in Theorem 1, for an emerging new label set $\{\boldsymbol{y}_i^{new}\}_{i=1}^{m}$ and a k-ary tree T, the time complexity of the tree update is bounded by $O\left(m\left(dK\hat{D} + dK\frac{\Delta_{\boldsymbol{y}\tilde{\boldsymbol{y}}}}{\epsilon^p} + N\hat{D}\right) + |\mathcal{L}_T|\tau t K\hat{D}\right)$.*

Table 2: Statistics of datasets.

Dataset	Domain	#Train	#Test	#Feature	#Label
MirFlickr	image	10,417	2,083	1,000	38
Delicious	web	12,920	3,185	500	983
EURlex	text	15,479	3,869	5,000	3,993
Wiki10	text	14,146	6,616	101,938	30,938

Proof. The tree update time complexity of a new label $\boldsymbol{y}_i^{new}$ is the sum of the complexities of the label insertion, existing node classifier fine-tuning, and the training of its classifier. On label insertion, $\boldsymbol{y}_i^{new}$ traverses from the root to a leaf node with at most d steps where d is the depth of the tree T. At each step, cosine similarity between the representation of $\boldsymbol{y}_i^{new}$ and tree node is calculated with time complexity of $O(dK\hat{D})$ where $\hat{D}$ is the number of non-zero features of node representations on average. The second part involves fine-tuning node classifier along the path. It is obvious to see that at most dK node classifiers need to be updated. As stated in Theorem 1, a good estimation of best parameters can be obtained in $\Delta_{\boldsymbol{y}\tilde{\boldsymbol{y}}} = \sum_{i=1}^{N} \mathbb{I}(\tilde{y}_i = 1) \wedge \mathbb{I}(y_i = -1)$ iterations, where $\Delta_{\boldsymbol{y}\tilde{\boldsymbol{y}}}$ denote the averaged discrepancy between $\boldsymbol{y}$ and $\tilde{\boldsymbol{y}}$ in the path. Hence the time complexity reaches $O(dK\frac{\Delta_{\boldsymbol{y}\tilde{\boldsymbol{y}}}}{\epsilon^p})$. Additionally, to train a new classifier for $\boldsymbol{y}_i^{new}$, $O(N\hat{D})$ is afford for a linear classifier. Lastly, we consider the time complexity when leaf node partition is triggered during label insertion. In the worst case, all leaf nodes with size $|\mathcal{L}_T|$ meet the set node labels threshold τ, which leads to time complexity $O\left(|\mathcal{L}_T|\tau t K\hat{D}\right)$. Overall, the total time complexity of tree update procedure gives $O\left(m\left(dK\hat{D} + dK\frac{\Delta_{\boldsymbol{y}\tilde{\boldsymbol{y}}}}{\epsilon^p} + N\hat{D}\right) + |\mathcal{L}_T|\tau t K\hat{D}\right)$. □

Proposition 1. *The memory complexity of a K-ary tree T is bounded by* $OO\left((K+1)\left(|\mathcal{V}_T| - |\mathcal{L}_T|\right)\hat{D} + |\mathcal{L}_T|\hat{D}\right)$ *where* $|\mathcal{V}_T|$ *is the total number of nodes and* $|\mathcal{L}_T|$ *is the number of leaves in* T.

Proof. Each *internal node* of T stores the mean vector of its associated label representations and K classifiers corresponding to K child nodes. Moreover, each *leaf node* of T stores the classifiers of corresponding labels. Firstly, let $|\mathcal{V}_T| - |\mathcal{L}_T|$ be the number of internal nodes of T. Consequently, it takes $O\left(\left(|\mathcal{V}_T| - |\mathcal{L}_T|\right)\hat{D}\right)$ to store the mean representations and $O\left(K\left(|\mathcal{V}_T| - |\mathcal{L}_T|\right)\hat{D}\right)$ to store node classifiers. In addition, as the number of leafs equals $|\mathcal{L}_T|$, it takes $O\left(|\mathcal{L}_T|\hat{D}\right)$ to store label classifiers on average. In total, the space complexity of T is given as $O\left((K+1)\left(|\mathcal{V}_T| - |\mathcal{L}_T|\right)\hat{D} + |\mathcal{L}_T|\hat{D}\right)$. □

5 EXPERIMENTS

In this section, we conduct extensive experiments to evaluate the performance of our proposed approach (Pslt) and compare with eleven baselines using various multi-label learning metrics.

Datasets. We perform experiments on four multi-label datasets which are publicly available from the XML repository. Detailed statistics are summarized in Table 2, where $\bar{L}$ denotes average labels per sample and $\hat{D}$ denotes average non-zero features per sample.[1]

Baselines. In our experiments, we compare the proposed methods with ten well-established or state-of-the-art multi-label learning algorithms:[2]

- Binary Relevance (BR) [34] is a set of m independent logistic regression classifiers.
- Classifier chains (CC) [19] transforms the multi-label learning problem into a chain of binary classification problems to incorporate label dependencies.
- RAKEL [24] considers a set of k labels in a multi-label training set as a new single-label classification task.
- ML-kNN [36] identifies each unseen instance's k nearest neighbors in the training set and utilizes the maximum a posteriori principle to determine the label set.
- SLEEC [2] is a well-known embedding-based method, which learns the label embedding by preserving the pairwise distances between a few nearest label neighbors.
- SML [11] is the state-of-the-art embedding-based method, which further improves the convergence rate of SLEEC.
- SLL [33] is a streaming label learning method, which assumes linear relationships between past labels and new labels.
- Deep-ML [35] introduces a shallow neural network as the multi-label model trained to minimize a ranking loss. Here, we deepen the network to boost performance.
- DNN-BCE [16] makes use of ReLU, AdaGrad, Dropout, and other deep learning techniques to train a DNN-based model.
- C2AE [31] is the state-of-the-art deep learning method for multi-label classification, which integrates the DNN architectures of canonical correlation analysis and auto-encoder to learn the deep latent space.
- DSLL [25] is the state-of-the-art deep learning method for streaming multi-label classification, which integrates the knowledge distillation technique.

Evaluation Metrics. The predictive accuracy of multi-label learning can be evaluated in different ways. Bipartition-based metrics assess a model's ability to predict sets of labels, whereas ranking-based metrics evaluate MLL methods, which instead produce rankings of labels. Therefore, our proposed method was evaluated in terms of both bipartition-based metrics, i.e., macro-F1, micros-F1, and instance-F1, and ranking-based metrics, i.e., Precision@k, coverage, ranking loss, average precision (AP), macro-AUC, and micro-AUC [30]. The details of these metrics can be referred in Appendix 6.

6 EVALUATION METRICS

Settings. To handle newly arrived labels, we investigated the performance of models with different batch sizes following previous settings [33]. In particular, for each dataset, we selected randomly 50% (e.g., 483 on Delicious) labels as past labels and constructed a classifier for the past labels. Then, the remaining labels were treated as new labels emerging in the mini-batch manner. For comparison,

[1] Datasets are available at the Extreme Classification Repository and https://github.com/chihkuanyeh/C2AE

[2] The codes of baseline methods are provided by the authors or scikit-multilearn [23]

Table 3: Ranking performance of each comparison method for learning new labels with different batch size (denoted by #label) by treating 50% labels as past labels. ↓(↑) means the smaller (larger) the value, the better the performance.

Datasets	#label	micro-AUC ↑											
		BR	CC	RAKEL	ML-*k*NN	SLEEC	SML	SLL	Deep-ML	DNN-BCE	C2AE	DSLL	PSLT
MirFlickr	3	0.6589	0.5049	0.6572	0.6296	0.5476	0.6123	0.7291	0.7448	0.7592	0.8051	0.8122	**0.9070**
	6	0.7100	0.5045	0.7187	0.6624	0.6400	0.6959	0.8388	0.8476	0.8606	0.8628	0.8713	**0.8909**
	9	0.7311	0.5071	0.7170	0.6754	0.7001	0.7331	0.8671	0.8692	0.8778	0.8890	0.8972	**0.9174**
	12	0.7151	0.5057	0.7223	0.6692	0.6862	0.7116	0.8630	0.8600	0.8763	0.8807	0.8886	**0.8974**
	15	0.7157	0.5051	0.7253	0.6611	0.6805	0.7015	0.8624	0.8584	0.8787	0.8749	0.8873	**0.8900**
Delicious	100	0.7133	0.5658	0.6382	0.5707	0.7813	0.8274	0.7981	0.8776	0.8615	0.8545	0.8820	**0.8889**
	200	0.7080	0.5606	0.6400	0.5692	0.7815	0.8258	0.7956	0.8626	0.8677	0.8321	0.8888	**0.8965**
	300	0.7177	0.5719	0.6493	0.5925	0.7941	0.8387	0.8068	0.8792	0.8655	0.8777	**0.9037**	0.9014
	400	0.7159	0.5705	0.6515	0.5916	0.7964	0.8141	0.8059	0.8736	0.8909	0.8656	**0.9048**	0.8968
	500	0.7144	0.5679	0.6445	0.5868	0.7926	0.8124	0.8030	0.8762	0.8697	0.8657	**0.9011**	0.8934
EURlex	200	0.6936	0.7308	0.7015	0.6255	0.8591	0.8216	0.8813	0.8130	0.8201	0.8561	0.8884	**0.9731**
	400	0.6738	0.7294	0.6821	0.6228	0.8481	0.8611	0.8905	0.8257	0.8423	0.8633	0.8928	**0.9760**
	600	0.6769	0.7321	0.6743	0.6256	0.8547	0.8735	0.8995	0.8218	0.8472	0.8414	0.9001	**0.9758**
	800	0.6825	0.7349	0.6575	0.6283	0.8548	0.8775	0.9016	0.8289	0.8491	0.8637	0.9034	**0.9769**
	1000	0.6733	0.7361	0.6708	0.6258	0.8406	0.8691	0.9115	0.8246	0.8431	0.8456	0.9106	**0.9773**
Wiki10	1k	0.6293	0.6535	0.6403	0.5694	0.8092	0.8113	0.8345	0.6978	0.7830	0.8274	0.8406	**0.9321**
	2k	0.5928	0.6244	0.6029	0.5510	0.7505	0.7557	0.7876	0.6545	0.6968	0.7937	0.8013	**0.9132**
	3k	0.5703	0.6078	0.5815	0.5405	0.7567	0.7192	0.7417	0.6505	0.7197	0.7939	0.8023	**0.9072**
	4k	0.5567	0.6008	0.5687	0.5364	0.7198	0.7105	0.7366	0.6556	0.7307	0.7944	0.7996	**0.8989**
	5k	0.5502	0.5984	0.5651	0.5345	0.7087	0.7194	0.7350	0.6463	0.7228	0.7824	0.7865	**0.8954**

Datasets	#label	Ranking loss ↓											
		BR	CC	RAKEL	ML-*k*NN	SLEEC	SML	SLL	Deep-ML	DNN-BCE	C2AE	DSLL	PSLT
MirFlickr	3	0.3032	0.5336	0.2842	0.3157	0.2393	0.2253	0.1179	0.1051	0.1025	0.0881	0.0809	**0.0228**
	6	0.2868	0.5786	0.2735	0.3337	0.2488	0.2189	0.0586	0.0571	0.0565	0.0569	**0.0562**	0.0655
	9	0.3366	0.6854	0.3194	0.3981	0.2186	0.2386	0.0672	0.0663	0.0609	0.0580	0.0570	**0.0405**
	12	0.4011	0.7816	0.3784	0.4690	0.2692	0.2528	0.0877	0.0865	0.0857	0.0772	**0.0750**	**0.0626**
	15	0.4209	0.7966	0.3873	0.4929	0.2857	0.2710	0.0899	0.0909	0.0837	0.0858	**0.0812**	0.0847
Delicious	100	0.4135	0.6551	0.4550	0.6521	0.2333	0.2235	0.1438	0.0916	0.0890	0.0929	**0.0830**	0.0923
	200	0.5115	0.8147	0.5603	0.7988	0.2712	0.2523	0.1771	0.1168	0.1217	0.1224	0.1140	**0.1013**
	300	0.5248	0.8303	0.5710	0.7918	0.2759	0.2594	0.1758	0.1093	0.1071	0.1073	0.1031	**0.1003**
	400	0.5311	0.8360	0.5711	0.7978	0.2661	0.2608	0.1778	0.1102	0.1079	**0.1050**	0.1082	0.1058
	500	0.5383	0.8435	0.5841	0.8096	0.2784	0.2679	0.1822	0.1109	0.1154	0.1094	**0.1067**	0.1096
EURlex	200	0.1473	0.1281	0.1403	0.1787	0.0654	0.0743	0.0198	0.0879	0.0337	0.0205	0.0138	**0.0062**
	400	0.2493	0.2048	0.2386	0.2880	0.1101	0.0901	0.0299	0.1062	0.0380	0.0274	0.0226	**0.0099**
	600	0.3468	0.2898	0.3494	0.4041	0.1552	0.1325	0.0404	0.1848	0.0562	0.0531	0.0313	**0.0134**
	800	0.4072	0.3400	0.4384	0.4780	0.1871	0.1618	0.0483	0.1530	0.0609	0.0495	0.0330	**0.0158**
	1000	0.4861	0.3960	0.4939	0.5619	0.2375	0.2085	0.0585	0.2326	0.0754	0.0721	0.0370	**0.0171**
Wiki10	1k	0.4746	0.4375	0.4213	0.5520	0.0918	0.1082	0.0867	0.2259	0.1011	0.0483	0.0421	**0.0395**
	2k	0.6205	0.5642	0.5705	0.6818	0.2385	0.2052	0.1259	0.3373	0.2225	0.0648	**0.0606**	0.0614
	3k	0.7324	0.6577	0.6823	0.7766	0.3306	0.2573	0.1897	0.3727	0.1911	0.0715	**0.0642**	0.0746
	4k	0.8167	0.7246	0.7569	0.8424	0.4037	0.3029	0.2123	0.3899	0.1831	0.0751	**0.0687**	0.0901
	5k	0.8538	0.7495	0.7941	0.8698	0.4313	0.4368	0.2234	0.4525	0.1972	0.0833	**0.0816**	0.1011

MLL methods can be extended to handle these new labels through independent modeling.

6.1 Results with Emerging New Labels

In this part, we evaluate PSLT and comparison methods on newly arrived labels with different batch sizes. For each method, it either learns a classifier on new labels separately or by incorporating information from previously learned model. After that, performance on new labels of each model is evaluated using test data.

Ranking performance evaluation. First, we evaluate PSLT and comparison methods with respect to the ranking-based metrics. Due to the limited space, we report results for two measures, i.e., micro-AUC and ranking loss, in Table 3. In addition, Precision@*k*, AP and, coverage are reported in Table C. Regarding micro-AUC, it can be seen that PSLT achieves the best performance in 17 out of 20 cases. In the other three cases, PSLT achieves the second best results with a minor gap between DSLL. For ranking loss, PSLT outperforms the others in 11 cases. Note that conventional multi-label learning methods fail to leverage the knowledge from past labels and the learned classifiers. Although DSLL is able to learn high-order representations using DNNs and takes advantage of the relationship between past labels and new labels, the hidden structure between labels is not well-explored. In the opposite, PSLT not only leverages the correlations among labels but also explores the latent label structure. The results justify the superiority of our method.

Classification performance evaluation. Second, we compare the performance of each method evaluated by bipartition-based metrics, where the prediction of each label belongs to $\{0, 1\}$ by thresholding rather than a continuous values. Figure 3 demonstrates the results with respect to instance-F1, micro-F1, and macro-F1 with different batch size, respectively. Results for several other bipartition-based metrics, e.g., hamming loss, macro-F1, and instance-F1, can

Table 4: Comparison of modeling new labels with different batch sizes by considering 50% labels as past labels. #label denotes the number of new labels. ↓(↑) means the smaller (larger) the value is, the performance will be the better.

Datasets	#label	Coverage ↓											
		BR	CC	RAKEL	ML-*k*NN	SLEEC	SML	SLL	Deep-ML	DNN-BCE	C2AE	DSLL	PSLT
MirFlickr	3	0.4530	0.5721	0.4468	0.4570	0.4117	0.3913	0.3568	0.3435	0.3442	0.3334	0.3236	**0.2181**
	6	0.3983	0.5815	0.3916	0.4249	0.3602	0.3472	0.2242	0.2247	0.2216	0.2213	0.2132	**0.2060**
	9	0.4611	0.6909	0.4552	0.5212	0.3651	0.3256	0.2321	0.2342	0.2286	0.2267	0.2153	**0.1391**
	12	0.5654	0.7882	0.5639	0.6372	0.4714	0.4246	0.2836	0.2859	0.2751	0.2732	0.2601	**0.1659**
	15	0.5817	0.8029	0.5733	0.6593	0.4954	0.4592	0.2862	0.2916	0.2809	0.2781	0.2663	**0.2139**
Delicious	100	0.4587	0.6937	0.5420	0.6992	0.3084	0.2515	0.2208	0.1510	0.1454	0.1499	**0.1302**	0.1588
	200	0.6147	0.8873	0.7399	0.8854	0.4399	0.3693	0.3408	0.2461	0.2281	0.2383	**0.2210**	0.2388
	300	0.6937	0.9518	0.8500	0.9486	0.5460	0.4366	0.4529	0.3167	0.2854	0.2934	**0.2512**	0.2910
	400	0.738	0.9679	0.8871	0.9673	0.6044	0.5074	0.5147	0.3669	0.3277	0.3266	**0.3245**	0.3394
	500	0.7716	0.9782	0.9232	0.9776	0.6626	0.4664	0.5708	0.3868	0.4150	0.3590	**0.3189**	0.3735
EURlex	200	0.1517	0.1356	0.1461	0.1823	0.0729	0.0813	0.0223	0.0932	0.0374	0.0234	0.0160	**0.0063**
	400	0.2588	0.2211	0.2529	0.2978	0.1250	0.1023	0.0360	0.1176	0.0444	0.0325	0.0270	**0.0101**
	600	0.3697	0.3242	0.3793	0.4280	0.1867	0.1439	0.0534	0.2127	0.0709	0.0669	0.0401	**0.0144**
	800	0.4457	0.3950	0.4839	0.5200	0.2317	0.1817	0.0686	0.1874	0.0812	0.0670	0.0461	**0.0174**
	1000	0.5408	0.4801	0.5679	0.6228	0.3102	0.2190	0.0988	0.3000	0.1076	0.1053	0.0948	**0.0211**
Wiki10	1k	0.5116	0.4810	0.4577	0.5757	0.3551	0.2942	0.1222	0.2812	0.1413	0.0713	0.0635	**0.0579**
	2k	0.6751	0.6366	0.6251	0.7168	0.5469	0.3524	0.1954	0.4428	0.3217	0.1107	0.1037	**0.1016**
	3k	0.7963	0.7573	0.7461	0.8188	0.6819	0.4835	0.3184	0.5405	0.3322	0.1441	**0.1317**	0.1435
	4k	0.8838	0.8477	0.8339	0.8922	0.7937	0.5126	0.3933	0.6152	0.3726	0.1795	**0.1674**	0.1986
	5k	0.9187	0.8868	0.8798	0.9226	0.8463	0.6424	0.4423	0.7134	0.4236	**0.2083**	0.2125	0.2450

Datasets	#label	Precision@*k* ($k = 1$) ↑											
		BR	CC	RAKEL	ML-*k*NN	SLEEC	SML	SLL	Deep-ML	DNN-BCE	C2AE	DSLL	PSLT
MirFlickr	3	0.3120	0.1723	0.3361	0.3269	0.3255	0.3574	0.4383	0.4580	0.4508	0.4642	**0.4729**	0.4445
	6	0.1906	0.0134	0.2189	0.2813	0.2967	0.3353	0.4402	0.4508	0.4575	0.4558	0.4614	**0.4920**
	9	0.3908	0.3341	0.3937	0.4249	0.3658	0.3589	0.5031	0.5122	0.5305	0.5386	**0.5511**	0.4387
	12	0.2780	0.1263	0.3068	0.3980	0.3845	0.4053	0.5842	0.5924	0.5972	0.6028	**0.6035**	0.4891
	15	0.2708	0.1440	0.2765	0.3826	0.4369	0.3946	0.5785	0.5655	0.6001	0.6015	**0.6020**	0.5650
Delicious	100	0.0776	0.1783	0.1852	0.1805	0.2914	0.2962	0.2967	0.3052	0.2659	0.3091	0.3099	**0.3786**
	200	0.0553	0.1991	0.2251	0.2232	0.3694	0.3697	0.3704	0.3706	0.3651	0.3623	0.3712	**0.5067**
	300	0.0333	0.2791	0.2474	0.3108	0.4672	0.4798	0.4893	0.4386	0.4650	0.4898	0.4904	**0.5676**
	400	0.0936	0.3463	0.2424	0.3425	0.5102	0.5382	0.5338	0.5268	0.5394	0.5130	0.5501	**0.5799**
	500	0.0424	0.3312	0.2041	0.3372	0.5171	0.5216	0.5504	0.5140	0.5024	0.5278	0.5498	**0.6018**
EURlex	200	0.0501	0.1124	0.1042	0.0636	0.1612	0.1601	0.1548	0.1357	0.1571	0.1641	0.1696	**0.1743**
	400	0.0649	0.1755	0.1548	0.1031	0.2357	0.2474	0.2352	0.2132	0.2409	0.2516	0.2525	**0.3041**
	600	0.0853	0.2512	0.2132	0.1590	0.3225	0.3135	0.3138	0.2750	0.3386	0.3401	0.3497	**0.4029**
	800	0.0943	0.3037	0.2303	0.1970	0.3711	0.3958	0.3657	0.3479	0.3830	0.3983	0.4117	**0.4652**
	1000	0.1016	0.3546	0.2985	0.2406	0.4129	0.4363	0.4223	0.3874	0.4611	0.4732	0.4854	**0.5315**
Wiki10	1k	0.2157	0.2269	0.2341	0.1196	0.3776	0.3853	0.4132	0.3094	0.3758	0.3626	0.4143	**0.4384**
	2k	0.2142	0.2304	0.2334	0.1261	0.3478	0.3909	0.3801	0.3171	0.3284	0.3927	0.4113	**0.4472**
	3k	0.2133	0.2491	0.2431	0.1356	0.3965	0.4087	0.4123	0.3253	0.3623	0.4076	0.4335	**0.4561**
	4k	0.2145	0.2573	0.2541	0.1548	0.4149	0.4252	0.4034	0.3464	0.3514	0.4265	0.4504	**0.4777**
	5k	0.2267	0.2922	0.2766	0.1750	0.4173	0.4184	0.4063	0.3581	0.3614	0.4388	0.4503	**0.4981**

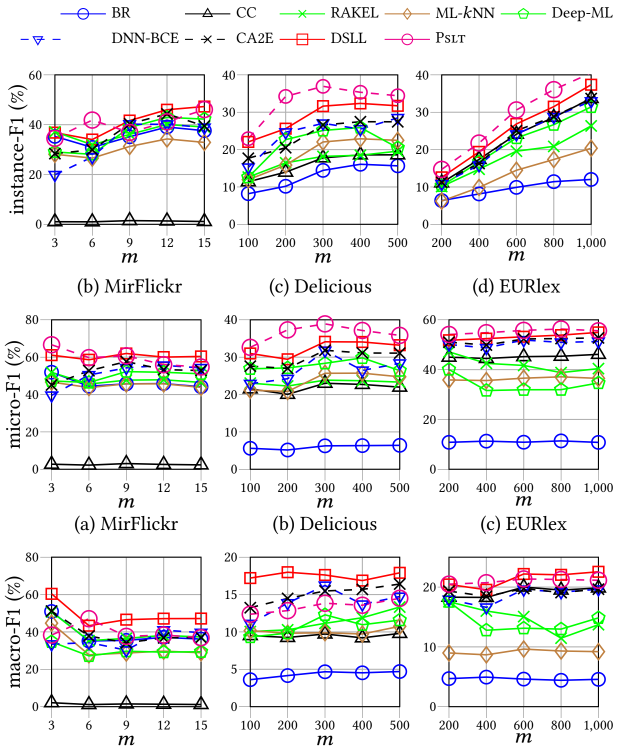
be found in the supplementary materials. It can be observed that PSLT achieves the best performance in most cases. Especially, PSLT outperforms others with a large margin on Delicious dataset. However, traditional multi-label learning approaches, such as BR and ML-*k*NN, perform poorly on datasets with a relatively large number of labels because interrelationships among labels are not effectively explored in their methods.

7 CONCLUSION

We study a novel problem setting called *streaming multi-label learning*, in which most extant multi-label learning approaches cannot directly applied. To solve this problem, we propose a new learning framework based on the probabilistic label tree. The benefits are mainly in twofolds. First, it can naturally incorporate new labels by taking advantage of label correlations between past and new labels. Second, the modification of node classifiers is efficient and we further theoretically derive an estimation error bound, which guarantees the learning performance of the proposal. Experiments in comparison with eleven baselines on four benchmark datasets verify the effectiveness of our method.

ACKNOWLEDGE

This research was supported by the National Key R&D Program of China (2017YFB1001903) and the NSFC (61772262). Yu-Feng Li is the corresponding author.



(b) MirFlickr (c) Delicious (d) EURlex

(a) MirFlickr (b) Delicious (c) EURlex

(b) MirFlickr (c) Delicious (d) EURlex

Figure 3: Performance comparison by considering 50% labels as past labels. *m* indicates the number of new labels.

REFERENCES

[1] Samy Bengio, Jason Weston, and David Grangier. 2010. Label Embedding Trees for Large Multi-Class Tasks. In *NIPS*. Curran Associates, Inc., 163–171.

[2] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain. 2015. Sparse local embeddings for extreme multi-label classification. In *Advances in Neural Information Processing Systems*. Montreal, Canada, 730–738.

[3] H. Daume III, N. Karampatziakis, J. Langford, and P. Mineiro. 2016. Logarithmic time one-against-some. *arXiv preprint arXiv:1606.04988* (2016).

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and Li F.-F. 2009. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Miami, FL, 248–255.

[5] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research* 9 (2008), 1871–1874.

[6] Huang Fang, Minhao Cheng, Cho-Jui Hsieh, and Michael P. Friedlander. 2019. Fast Training for Large-Scale One-versus-All Linear Classifiers using Tree-Structured Initialization. In *Proceedings of the 2019 SIAM International Conference on Data Mining, SDM 2019, Calgary, Canada*. SIAM, 280–288.

[7] D. J. Hsu, S. M. Kakade, J. Langford, and T. Zhang. 2009. Multi-label prediction via compressed sensing. In *Advances in Neural Information Processing Systems*. Vancouver, Canada, 772–780.

[8] Kalina Jasinska-Kobus, Marek Wydmuch, Krzysztof Dembczynski, Mikhail Kuznetsov, and Robert Busa-Fekete. 2020. Probabilistic Label Trees for Extreme Multi-label Classification. *arXiv preprint arXiv:2009.11218* (2020).

[9] Kalina Jasinska-Kobus, Marek Wydmuch, Devanathan Thiruvenkatachari, and Krzysztof Dembczyński. 2020. Online probabilistic label trees. *arXiv preprint arXiv:2007.04451* (2020).

[10] Sujay Khandagale, Han Xiao, and Rohit Babbar. 2019. Bonsai-Diverse and Shallow Trees for Extreme Multi-label Classification. *arXiv preprint arXiv:1904.08249* (2019).

[11] Weiwei Liu and Xiaobo Shen. 2019. Sparse Extreme Multi-label Learning with Oracle Property. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, CA (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). 4032–4041.

[12] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE transactions on information theory* 28, 2 (1982), 129–137.

[13] Maryam Majzoubi and Anna Choromanska. 2020. LdSM: Logarithm-depth Streaming Multi-label Decision Trees. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS, Online [Palermo, Italy]*, Vol. 108. 4247–4257.

[14] J. McAuley, R. Pandey, and J. Leskovec. 2015. Inferring networks of substitutable and complementary products. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Sydney, Australia, 785–794.

[15] Xin Mu, Kai Ming Ting, and Zhi-Hua Zhou. 2017. Classification under streaming emerging new classes: A solution using completely-random trees. *IEEE Transactions on Knowledge and Data Engineering* 29, 8 (2017), 1605–1618.

[16] Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz. 2014. Large-Scale Multi-label Text Classification — Revisiting Neural Networks. In *Machine Learning and Knowledge Discovery in Databases*. 437–452.

[17] I. Partalas, A. Kosmopoulos, N. Baskiotis, T. Artieres, G. Paliouras, E. Gaussier, I. Androutsopoulos, M.-R. Amini, and P. Galinari. 2015. LSHTC: A benchmark for large-scale text classification. *arXiv preprint arXiv:1503.08581* (2015).

[18] Y. Prabhu and M. Varma. 2014. FastXML: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, 263–272.

[19] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2011. Classifier chains for multi-label classification. *Mach. Learn.* 85, 3 (2011), 333–359.

[20] S. Si, H. Zhang, S. S. Keerthi, D. Mahajan, I. S. Dhillon, and C.-J. Hsieh. 2017. Gradient Boosted Decision Trees for High Dimensional Sparse Output. In *Proceedings of International Conference on Machine Learning*. 3182–3190.

[21] W. Siblini, P. Kuntz, and F. Meyer. 2018. CRAFTML, an Efficient Clustering-based Random Forest for Extreme Multi-label Learning. In *Proceedings of the 35th International Conference on Machine Learning*. Stockholmsmässan, Stockholm Sweden, 4664–4673.

[22] Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*. 4077–4087.

[23] P. Szymański and T. K. 2017. A scikit-based Python environment for performing multi-label classification. *arXiv preprint arXiv:1702.01460* (2017).

[24] G. Tsoumakas, I. Katakis, and I. Vlahavas. 2011. Random k-Labelsets for Multilabel Classification. *IEEE Transactions on Knowledge and Data Engineering* 2s3, 7 (2011), 1079–1089.

[25] Zhen Wang, Liu Liu, and Dacheng Tao. 2020. Deep Streaming Label Learning. In *International Conference on Machine Learning (ICML)*. 378–387.

[26] Tong Wei, Lan-Zhe Guo, Yu-Feng Li, and Wei Gao. 2018. Learning safe multi-label prediction for weakly labeled data. *Machine Learning* 107, 4 (2018), 703–725.

[27] Tong Wei and Yu-Feng Li. 2018. Does Tail Label Help for Large-Scale Multi-Label Learning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. Stockholm, Sweden, 2847–2853.

[28] Tong Wei and Yu-Feng Li. 2019. Learning Compact Model for Large-Scale Multi-Label Data. In *AAAI*. 5385–5392.

[29] Tong Wei, Wei-Wei Tu, and Yu-Feng Li. 2019. Learning for Tail Label Data: A Label-Specific Feature Approach. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. Macau, China, 3842–3848.

[30] Xi-Zhu Wu and Zhi-Hua Zhou. 2017. A unified view of multi-label performance measures. In *International Conference on Machine Learning*. PMLR, 3780–3788.

[31] C.-K. Yeh, W.-C. Wu, W.-J. Ko, and Y.-C. F. Wang. 2017. Learning deep latent space for multi-label classification.. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. San Francisco, CA, 2838–2844.

[32] Ronghui You, Suyang Dai, Zihan Zhang, Hiroshi Mamitsuka, and Shanfeng Zhu. 2018. Attentionxml: Extreme multi-label text classification with multi-label attention based recurrent neural networks. *arXiv preprint arXiv:1811.01727* (2018).

[33] Shan You, Chang Xu, Yunhe Wang, Chao Xu, and Dacheng Tao. 2016. Streaming Label Learning for Modeling Labels on the Fly. *CoRR* abs/1604.05449 (2016).

[34] M.-L. Zhang, Y.-K. Li, X.-Y. Liu, and X. Geng. 2018. Binary relevance for multi-label learning: an overview. *Frontiers of Computer Science* (2018), 1–12.

[35] Min-Ling Zhang and Zhi-Hua Zhou. 2006. Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization. *IEEE Transactions on Knowledge and Data Engineering* 18, 10 (2006), 1338–1351.

[36] M.-L. Zhang and Z.-H. Zhou. 2007. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition* 40, 7 (2007), 2038–2048.

[37] M.-L. Zhang and Z.-H. Zhou. 2014. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering* 26, 8 (2014), 1819–1837.

[38] Y. Zhu, K. M. Ting, and Z. Zhou. 2018. Multi-Label Learning with Emerging New Labels. *IEEE Transactions on Knowledge and Data Engineering* 30, 10 (2018), 1901–1914.

[39] Yong-Nan Zhu and Yu-Feng Li. 2020. Semi-Supervised Streaming Learning with Emerging New Labels.. In *AAAI*. 7015–7022.

Appendices

A PROOF OF LEMMA 1

Lemma 1. *Assume $\ell(\cdot)$ in Problem (2) is α-Lipschitz and the optimal solution before and after incorporating emerging labels be $\boldsymbol{w}^*$ and $\tilde{\boldsymbol{w}}^*$ respectively, which satisfy $||\boldsymbol{w}^*||_2 \leq B$ and $||\tilde{\boldsymbol{w}}^*||_2 \leq B$. Then*

$$f\left(\tilde{\boldsymbol{y}}, \boldsymbol{w}^*\right) - f\left(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{w}}^*\right) \leq 4C\alpha B \sum_{i=1}^{N} \mathbb{I}(\tilde{y}_i = 1) \wedge \mathbb{I}(y_i = -1) \tag{6}$$

Proof. We can rewrite the LHS of Inequation (6) as

$$\begin{aligned} f\left(\tilde{\boldsymbol{y}}, \boldsymbol{w}^*\right) - f\left(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{w}}^*\right) = & f\left(\tilde{\boldsymbol{y}}, \boldsymbol{w}^*\right) - f\left(\boldsymbol{y}, \boldsymbol{w}^*\right) \\ & + f\left(\boldsymbol{y}, \boldsymbol{w}^*\right) - f\left(\tilde{\boldsymbol{y}}, \boldsymbol{w}^*\right) \\ & + f\left(\tilde{\boldsymbol{y}}, \boldsymbol{w}^*\right) - f\left(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{w}}^*\right) \end{aligned} \tag{7}$$

Note that the second term in Equation (7) is less than 0 because $\boldsymbol{w}^*$ minimizes the objective in Problem (2) with respect to $\boldsymbol{y}$. We first derive an upper bound for the first term and the bound for the last term can be similarly achieved.

$$\begin{aligned} f\left(\tilde{\boldsymbol{y}}, \boldsymbol{w}^*\right) - f\left(\boldsymbol{y}, \boldsymbol{w}^*\right) = & \mathcal{R}\left(\boldsymbol{w}^*\right) - \mathcal{R}\left(\boldsymbol{w}^*\right) \\ & + C \sum_{i=1}^{N} \left(\ell\left(\tilde{y}_i \boldsymbol{x}_i^\top \boldsymbol{w}^*\right) - \ell\left(y_i \boldsymbol{x}_i^\top \boldsymbol{w}^*\right)\right) \\ \leq & C \sum_{i=1}^{N} \left|\ell\left(\tilde{y}_i \boldsymbol{x}_i^\top \boldsymbol{w}^*\right) - \ell\left(y_i \boldsymbol{x}_i^\top \boldsymbol{w}^*\right)\right| \\ = & C \sum_{i=1}^{N} \mathbb{I}\left(y_i \neq \tilde{y}_i\right) \left|\ell\left(\boldsymbol{x}_i^\top \boldsymbol{w}^*\right) - \ell\left(-\boldsymbol{x}_i^\top \boldsymbol{w}^*\right)\right| \\ \leq & C \sum_{i=1}^{N} \mathbb{I}\left(y_i \neq \tilde{y}_i\right) \alpha \left(2\left|\boldsymbol{x}_i^\top \boldsymbol{w}^*\right|\right) \\ \leq & 2C \sum_{i=1}^{N} \mathbb{I}\left(y_i \neq \tilde{y}_i\right) \alpha B \\ = & 2C\alpha B \sum_{i=1}^{N} \mathbb{I}(\tilde{y}_i = 1) \wedge \mathbb{I}(y_i = -1) \end{aligned}$$

The first inequality uses the fact $a - b \leq |a - b|$ and the second inequality uses the definition that $\ell(\cdot)$ is α-Lipschitz. The third inequality is since $|\boldsymbol{x}_i^\top \boldsymbol{w}^*| \leq ||\boldsymbol{x}_i||_2 \cdot ||\boldsymbol{w}^*||_2$ and the assumption $||\boldsymbol{w}_2^* \leq B$. By considering that only additional positive samples would be appended to $\boldsymbol{y}$, we have $\ell_H\left(\boldsymbol{y}, \tilde{\boldsymbol{y}}\right) = \mathbb{I}(\tilde{y}_i = 1) \wedge \mathbb{I}(y_i = -1)$. Lastly, we derive the same result for the third term in Equation (7) and complete the proof. □

B PROOF OF THEOREM 1

Theorem 1. *Assume $\ell(\cdot)$ is α-Lipschitz and we have a solver $\mathcal{A}$ with sublinear convergence rate that can solve each subproblem k with ϵ precision in $T_k = O\left(\frac{f\left(\tilde{\boldsymbol{y}}_k, \boldsymbol{w}_k^*\right) - f\left(\tilde{\boldsymbol{y}}_k, \tilde{\boldsymbol{w}}_k^*\right)}{\epsilon^p}\right)$ iterations. We can bound the total number of iterations of the node classifier modification procedure by $T_{total} = \sum_{k=1}^{K} T_k = O\left(\frac{\sum_{k=1}^{K} \ell_H\left(\tilde{\boldsymbol{y}}_k, \boldsymbol{y}_k\right)}{\epsilon^p}\right) = O\left(\frac{\Delta_{\boldsymbol{y}_k \tilde{\boldsymbol{y}}_k}}{\epsilon^p}\right)$, where $\Delta_{\boldsymbol{y}\tilde{\boldsymbol{y}}} = \sum_{i=1}^{N} \mathbb{I}(\tilde{y}_i = 1) \wedge \mathbb{I}(y_i = -1)$. However, under a naive zeros initialization with $\tilde{\boldsymbol{w}}_k^* = \boldsymbol{0}$, the upper bound is $T_{total} = \sum_{k=1}^{K} T_k = O\left(\frac{NK}{\epsilon^p}\right)$.*

Proof. According to Lemma 1, we have

$$\begin{aligned} T_{\text{total}} = \sum_{k=1}^{K} T_k = & O\left(\frac{\sum_{k=1}^{K} \left(f\left(\tilde{\boldsymbol{y}}_k, \boldsymbol{w}_k^*\right) - f\left(\tilde{\boldsymbol{y}}_k, \boldsymbol{w}_k^*\right)\right)}{\epsilon^p}\right) \\ = & O\left(\frac{\sum_{k=1}^{K} 4\ell_H\left(\tilde{\boldsymbol{y}}_k, \boldsymbol{y}_k\right) C\alpha B}{\epsilon^p}\right) \\ = & O\left(\frac{\sum_{i=1}^{N} \mathbb{I}(\tilde{Y}_{k,i} = 1) \wedge \mathbb{I}(Y_{k,i} = -1)}{\epsilon^p}\right) \end{aligned}$$

By denoting $\Delta_{\boldsymbol{y}\tilde{\boldsymbol{y}}} = \sum_{i=1}^{N} \mathbb{I}(\tilde{y}_i = 1) \wedge \mathbb{I}(y_i = -1)$, we complete our proof for the first iteration complexity. To prove the bound for naive zeros initialization, recall that $f\left(\tilde{\boldsymbol{y}}, \boldsymbol{0}\right) - f\left(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{w}}^*\right) \leq 4\ell_H\left(\boldsymbol{y}, \tilde{\boldsymbol{y}}\right) C\alpha B$ which is upper bounded by $\leq CN\ell(\tilde{\boldsymbol{y}}, 0)$. Since C and $\ell(\tilde{\boldsymbol{y}}, 0)$ are typical far less than N in multi-label learning, the best upper bound for $f\left(\tilde{\boldsymbol{y}}, \boldsymbol{0}\right) - f\left(\tilde{\boldsymbol{y}}, \tilde{\boldsymbol{w}}^*\right)$ is $O(N)$. Therefore, combining the bound for T_k, we have $T_{\text{total}} = \sum_{k=1}^{K} T_k = O\left(\frac{NK}{\epsilon^p}\right)$. □

C ADDITIONAL EXPERIMENTAL RESULTS

- Figure 3 shows the macro-F1 and instance-F1 results for learning new labels with different batch size. Note that SLEEC, SML and SLL could not properly generate bipartite classification results, hence it is not possible to evaluate the results with the F1 score.
- Table **??** shows the performance in terms of hamming loss. Note that SLEEC, SML, and SLL could not properly generate bipartite classification results, hence it is not possible to evaluate the results with hamming loss. Additionally, due to the sparsity of feature space, hamming loss is homogenized on EURlex and Wiki10.

The details of these metrics are described as follows. Given a data set $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$, where $\boldsymbol{x}_i \in \mathbb{R}^{d \times 1}$ is a real vector representing features and $\boldsymbol{y}_i \in \{0, 1\}^{m \times 1}$ is the corresponding output label vector. For notational simplicity, we use $Y_{i\cdot}^+$ to denote the index set of associated labels of $\boldsymbol{y}_i$. Formally, $Y_{i\cdot}^+ = \{j | Y_{ij} = 1\}$ and $Y_{i\cdot}^- = \{j | Y_{ij} = 0\}$. With respect to j-th column of label matrix, $Y_{\cdot j}^+ = \{i | Y_{ij} = 1\}$ denotes the index set of associated instance of the j-th label and $Y_{\cdot j}^- = \{i | Y_{ij} = 0\}$ denotes the set of non-associated instances. Table 5 summarizes evaluation metrics used in this paper. Let $H : \mathbb{R}^d \rightarrow \{0, 1\}^m$ be a multi-label classifier and can be decomposed as $\{h^1, ..., h^m\}$. H can be evaluated by bipartition-based metrics. F is the multi-label predictor, whose predicted value could be regarded as the confidence of relevance and can be evaluated by ranking-based metrics.

Table 5: Definitions of multi-label performance measures.

Measure	Formulation
Hamming loss	$hloss(H) = \frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\mathbb{I}\{h_i^j \neq y_i^j\}$
ranking loss	$\textit{rank-loss}(F) = \frac{1}{n}\sum_{i=1}^{n}\frac{\lvert\mathcal{S}_{\text{rank}}^{i}\rvert}{\lvert Y_{i\cdot}^{+}\rvert\lvert Y_{i\cdot}^{-}\rvert}$ $\mathcal{S}_{\text{rank}}^{i} = \{(u,v)\lvert f_u(\boldsymbol{x}_i) \leq f_v(\boldsymbol{x}_i), (u,v) \in Y_{i\cdot}^{+} \times Y_{i\cdot}^{-}\}$
coverage	$coverage(F) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}\{\max_{j\in Y_{i\cdot}^{+}} rank_F(\boldsymbol{x}_i, j)\}$
precision@k	$precision@k(F) = \frac{1}{n}\sum_{i=1}^{n}\frac{\lvert Y_{i\cdot}^{+}\cap\top_k(f(\boldsymbol{x}_i))\rvert}{k}$ $\top_k(f(\boldsymbol{x}_i)) = \{j\lvert f^j(\boldsymbol{x}_i) \in \top_k(f^1(\boldsymbol{x}_i), ..., f^m(\boldsymbol{x}_i))\}$
macro-F1	$\textit{macro-F1}(H) = \frac{1}{m}\sum_{j=1}^{m}\frac{2\sum_{i=1}^{n} y_{ij}h_{ij}}{\sum_{i=1}^{n} y_{ij}+\sum_{i=1}^{n} h_{ij}}$
micro-F1	$\textit{micro-F1}(H) = \frac{2\sum_{j=1}^{m}\sum_{i=1}^{n} y_{ij}h_{ij}}{\sum_{j=1}^{m}\sum_{i=1}^{n} y_{ij}+\sum_{j=1}^{m}\sum_{i=1}^{n} h_{ij}}$
instance-F1	$\textit{instance-F1}(H) = \frac{1}{n}\sum_{i=1}^{n}\frac{2\sum_{j=1}^{m} y_{ij}h_{ij}}{\sum_{j=1}^{m} y_{ij}+\sum_{j=1}^{m} h_{ij}}$
micro-AUC	$\textit{micro-AUC}(F) = \frac{\lvert\mathcal{S}_{\text{micro}}\rvert}{(\sum_{i=1}^{n}\lvert Y_{i\cdot}^{+}\rvert)\cdot(\sum_{i=1}^{n}\lvert Y_{i\cdot}^{-}\rvert)}$ $\mathcal{S}_{\text{micro}} = \{(a,b,i,j)\lvert(a,b) \in Y_{\cdot i}^{+} \times Y_{\cdot j}^{-}, f_i(\boldsymbol{x}_a) \geq f_j(\boldsymbol{x}_b)\}$

Table 6: Comparison of modeling new labels with different batch sizes by considering 50% labels as past labels. #label denotes the number of new labels. ↓(↑) means the smaller (larger) the value is, the performance will be the better.

Datasets	#label	Average precision ↑											
		BR	CC	RAKEL	ML-*k*NN	SLEEC	SML	SLL	Deep-ML	DNN-BCE	C2AE	DSLL	PSLT
MirFlickr	3	0.3832	0.2791	0.3580	0.3432	0.2836	0.3251	0.4602	0.5238	0.5278	0.6314	**0.6341**	0.3530
	6	0.2370	0.1715	0.2365	0.2129	0.1717	0.2435	0.3366	0.3587	0.3978	0.4143	**0.4558**	0.3542
	9	0.2391	0.1599	0.2444	0.2111	0.1718	0.2519	0.3840	0.3738	0.3986	0.4307	**0.4925**	0.3567
	12	0.2529	0.1711	0.2611	0.2243	0.1837	0.2373	0.4118	0.3876	0.4507	0.4517	**0.4952**	0.4040
	15	0.2408	0.1594	0.2460	0.2117	0.1764	0.2391	0.4056	0.3861	0.4566	0.4660	**0.4985**	0.3880
Delicious	100	0.0408	0.0595	0.0631	0.0531	0.0684	0.0631	0.1180	0.1028	0.1183	0.1553	0.1563	**0.1576**
	200	0.0393	0.0580	0.0599	0.0512	0.0699	0.0707	0.1128	0.0994	0.1250	0.1420	0.1428	**0.1563**
	300	0.0394	0.0612	0.0608	0.0524	0.0717	0.0689	0.1193	0.1220	0.1561	0.1596	0.1604	**0.1625**
	400	0.0405	0.0584	0.0563	0.0528	0.0724	0.0774	0.1164	0.1113	0.1363	0.1514	0.1520	**0.1572**
	500	0.0439	0.0622	0.0559	0.0580	0.0760	0.0819	0.1224	0.1290	0.1107	**0.1634**	0.1631	0.1606
Datasets	#label	macro-AUC ↑											
		BR	CC	RAKEL	ML-*k*NN	SLEEC	SML	SLL	Deep-ML	DNN-BCE	C2AE	DSLL	PSLT
MirFlickr	3	0.6625	0.5035	0.6522	0.6111	0.5185	0.5614	0.7151	0.7471	0.7472	0.7966	0.8063	**0.8081**
	6	0.7088	0.5015	0.6854	0.5729	0.5093	0.5751	0.7666	0.7834	0.8013	0.8069	0.8114	**0.8220**
	9	0.7414	0.5028	0.7202	0.5789	0.5307	0.5853	0.8059	0.8125	0.8238	0.8369	**0.8636**	0.8202
	12	0.7284	0.5020	0.7080	0.5804	0.5302	0.5825	0.8007	0.8001	0.8110	0.8305	0.8423	**0.8495**
	15	0.7290	0.5018	0.7066	0.5788	0.5379	0.5919	0.8077	0.8085	0.8276	0.8336	**0.8420**	0.8040
Delicious	100	0.6260	0.5338	0.6271	0.5376	0.6806	0.6894	0.7331	0.7592	0.7383	0.7718	**0.8109**	0.7975
	200	0.6249	0.5339	0.6187	0.5355	0.6805	0.6912	0.7279	0.7388	0.7633	0.7506	0.7941	**0.7961**
	300	0.6229	0.5358	0.6203	0.5361	0.6850	0.7015	0.7280	0.7472	0.7543	0.7844	**0.8093**	0.7962
	400	0.6185	0.5335	0.6179	0.5360	0.6863	0.6931	0.7309	0.7178	0.7720	0.7549	**0.7991**	0.7979
	500	0.6211	0.5364	0.6200	0.5391	0.6885	0.6997	0.7335	0.7411	0.7290	0.7616	**0.8021**	0.7997