# Deep Streaming Label with Knowledge Distillation

Zhen Wang, *Member, IEEE,* Liu Liu, *Member, IEEE,* and Dacheng Tao, *Fellow, IEEE*

**Abstract**—In multi-label learning, each instance can be associated with multiple and non-exclusive labels. Previous studies assume that all the labels in the learning process are fixed and static; however, they ignore the fact that the labels will emerge continuously in changing environments. In order to fill in these research gaps, we propose a novel deep neural network (DNN)-based framework, Deep Streaming Label Learning (DSLL), to classify instances with newly emerged labels effectively. DSLL with knowledge distillation can explore and incorporate the knowledge from past labels and historical models to understand and develop emerging new labels. DSLL consists of three components: 1) a streaming label mapping to extract deep relationships between new labels and past labels with a novel label correlation-aware loss; 2) a streaming feature distillation propagating feature-level knowledge from the historical model to a new model; 3) a *senior student* network to model new labels with the help of knowledge learned from the past. Theoretically, we prove that DSLL admits tight generalization error bounds for new labels in the DNN framework. Experimentally, extensive empirical results show that the proposed method performs significantly better than the existing state-of-the-art multi-label learning methods to handle the continually emerging new labels.

**Index Terms**—Streaming Label Learning, Multi-label Learning, Knowledge Distillation, Deep Learning.

## 1 Introduction

In traditional supervised learning, a single instance only possesses a single label; whereas, in many real-world tasks, one instance can be naturally associated with multiple and non-exclusive labels. For example, a document may belong to various topics [1, 2]; an image can contain several individuals' faces [3]; a gene may be related to multiple functions [4, 5]. Multi-label learning has been proposed and studied to handle such kind of tasks, with the goal of predicting a label vector $\boldsymbol{y} \in \{0,1\}^m$ for a given instance $\boldsymbol{x} \in \mathbb{R}^d$, where $m$ is the number of labels, and $d$ is the dimensionality of the input instance.

A straightforward solution to multi-label learning is binary relevance (BR) [6] and one-vs-all [7], which treat each label as an independent binary classification problem. However, in addition to a lack of correlations between label information, such approaches suffer from high computational and storage costs for large datasets, which would limit the prediction performance. As a result, there are two prevalent techniques to deal with multi-label learning: (1) exploring and exploiting the correlations across labels [1, 8, 9], and (2) reducing the label space by embedding original high-dimensional label vectors into the low-dimensional representation [10, 11, 12]. In particular, deep neural networks (DNNs) [13, 14] offer a good way to learn the label correlation and embedding simultaneously, significantly outperforming classical methods [2, 15, 16].

Overall, existing multi-label learning studies focus on a fixed set of labels. That is, they assume that all the labels in the learning process are given at once. However, this assumption is frequently violated in real-world changing environments. In practice, with more in-depth data exploration and understanding, the number of labels gradually increases. For example, when a photograph is posted on Facebook or Twitter, the photo will be labeled continuously and differently by users who are browsing, and the classification system needs to be updated accurately according to the new labels. In the multi-label learning setting, independently learning only for new labels ignores the correlated information from past labels; integrating past labels with emerging new labels to retrain a new multi-label model would be prohibitively computationally expensive. Hence, streaming label learning (SLL) [17] has been proposed to handle this problem, which assumes that the new label vectors are a linear combination of past label vectors and that the new classifier can inherit the linear combination to obtain the correlations across labels without retraining the historical classifier. Nevertheless, the linear representation between new labels and past labels limits SLL performance, and the training knowledge from the historical classifiers is neglected.

In this paper, we propose a novel DNN-based framework, Deep Streaming Label Learning (DSLL), to effectively model the emerging new labels. DSLL has the ability to explore and utilize deep correlations between new labels and past labels without computationally intensive retraining. Furthermore, inspired by knowledge distillation [18, 19], a technique that distills knowledge or feature representations from a large *teacher* model to a smaller *student* model, DSLL develops the approach to leverage the knowledge from previous learning to model new labels more accurately. DSLL has three components: (1) a streaming label mapping to exploit deep relationships between new labels and past labels with a novel label correlation-aware loss; (2) a streaming feature distillation framework to propagate the feature-level

• *Zhen Wang, Liu Liu and Dacheng Tao were with UBTECH Sydney AI Centre, School of Computer Science, Faculty of Engineering, The University of Sydney, Darlington, NSW 2008, Australia*
*E-mail: zwan4121@uni.sydney.edu.au*

*Manuscript received April 19, 2020*

knowledge from a past-label classifier (teacher) to a new-label classifier (student); (3) a *senior student* network to integrate the relationships and knowledge learned from the past to model the newly arrived labels.

In order to verify the effectiveness and performance of DSLL, both theoretical and experimental analyses are conducted. From the theoretical perspective, we present proofs that the generalizability of modeling emerging new labels can be significantly improved with the knowledge extracted from past labels. In contrast to linear frameworks [20], proofs for a nonlinear model can be challenging, especially in DNNs. We analyze the excess risk bounds for the proposed learning model by leveraging the latest deep learning theory. From the experimental perspective, we perform a comprehensive empirical evaluation of our model on a variety of benchmark datasets, using prevalent metrics and comparisons with well-established or state-of-the-art multi-label learning algorithms. The results show that our approach considerably outperforms other methods and demonstrates the significance of utilizing knowledge learned from past labels and their corresponding classifiers in modeling new labels.

The rest of the paper is organized as follows. Section 2 summarizes the previous related work. Section 3 formulates the problem and describes the proposed model in detail. Section 4 presents the theoretical analysis. Section 5 presents the experimental results validating the model and providing several insights. We conclude in Section 6. All detailed proofs are provided in the Appendix to the Supplementary Materials. The anonymized code is available here[1].

## 2 Related Work

**Multi-label Learning.** Obvious baselines for multi-label learning (MLL) are provided by the one-vs-all technique [7] and binary relevance [6], which seek to learn an independent classifier for each label. As expected, the huge training and prediction costs for a large number of labels and the lack of ability to discover the correlations between labels are limitations. Label embedding (LE) is a popular strategy for MLL that projects the label vectors onto a lower subspace with the latent embedding of corresponding information and utilizes a decompression mapping to lift the embedded label vectors back to the original label space [10, 21]. SLEEC [11] is the state-of-the-art LE method that captures the embedding of labels by preserving the local distances between a few nearest label neighbors, and SML [12] further improves the convergence rate of SLEEC. Deep learning is another popular end-to-end learning approach. BP-MLL [1] introduces the neural network architecture to learn relevant features automatically. Since then, other DNN techniques have been applied to multi-label learning [2]. C2AE [15] is a state-of-the-art DNN-based approach that integrates canonical correlation analysis [22] and autoencoder to utilize features and label embedding jointly. In summary, all these MLL studies explicitly and implicitly focus on a fixed set of labels, which is frequently violated in the changing environments. Although traditional multi-label algorithms can be adapted to handle emerging new labels, they can suffer from the lack of correlations between past labels and new labels for only learning new labels or prohibitive computational costs caused by retraining the model with whole labels.

**Streaming Label Learning.** To model newly arrived labels expediently and effectively, streaming label learning (SLL) [17] has been proposed, which aims to employ the knowledge from past labels. SLL 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. Note that while Zhu et al. [23] proposed a method, in which instances with emerging new labels are regarded as outliers, this is different to the SLL problem.

**Knowledge Distillation.** Knowledge distillation (KD) aims to transfer the knowledge from an existing powerful classifier (called a *teacher*) to a lightweight classifier (called a *student*). For instance, Ba and Caruana [24] trained a shallow student network to mimic the *logits* provided by a deep teacher network. Hinton et al. [18] proposed a more general framework by training a small student to predict softened labels of the output of a large teacher network. FitNet [19] employed the intermediate layer representation of the teacher as a *hint* to guide student training. Due to its effectiveness, KD has been being used in many fields such as object detection [25], model compression [26], image classification [27], machine translation [28], and meta-learning [29]. In this work, we propose *streaming* feature distillation, where the past-label classifier acted as a teacher helps guide a student network to model new labels. It not only extracts knowledge from the intermediate layer of the teacher, but also takes advantage of the softened output provided by the well-trained teacher to construct the relationship between new and past labels.

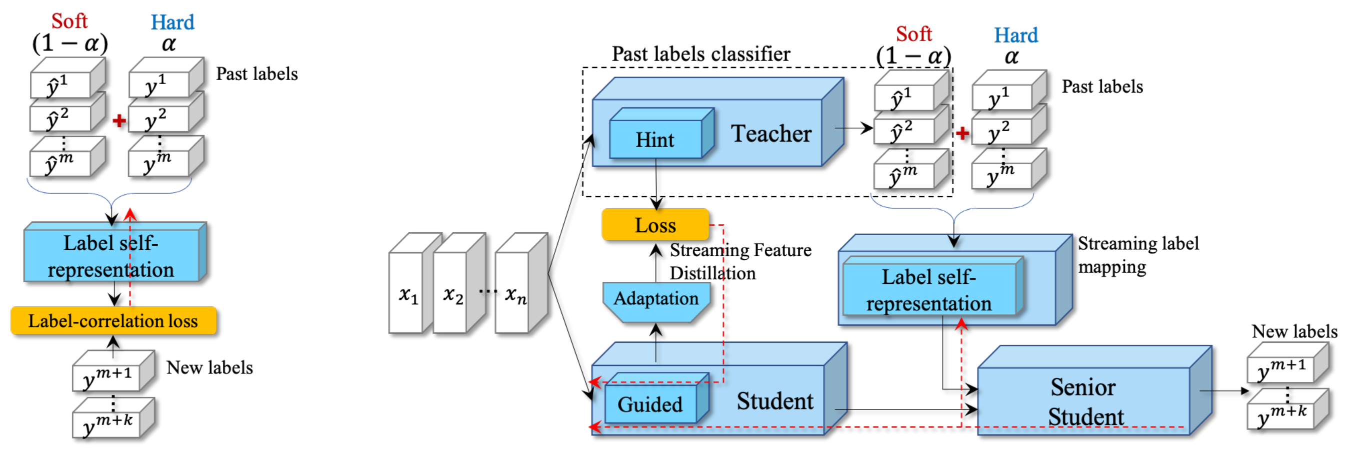
## 3 Deep Streaming Label Learning

Conventional multi-label learning encounters the following dilemmas in the face of newly-arrived labels: (1) integrating past labels and new labels to retrain a new multi-label model requires massive computational resources; (2) independently learning for the new labels would neglect the knowledge acquired from past labels and historical models. This section will detail the proposed Deep Streaming Label Learning to handle the continuously emerged labels, which can solve these challenging problems.

### 3.1 Problem Formulation

First, we state the MLL with emerging new labels problem. Assume an initial training data set is denoted by $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$, where $\boldsymbol{x}_i \in \mathcal{X} \subseteq \mathbb{R}^{d\times 1}$ is a real vector representing an input feature (instance) and $\boldsymbol{y}_i \in \mathcal{Y} \subseteq \{0,1\}^{m\times 1}$ is the corresponding output label vector ($i \in [n]$, defined as $i \in \{1, ..., n\}$). Moreover, $y_i^j = 1$ if the $j$-th label is assigned to the instance $\boldsymbol{x}_i$ and $y_i^j = 0$ otherwise. The input matrix is $\mathbf{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n] \in \mathbb{R}^{d\times n}$ and the initial output matrix is $\mathbf{Y} = [\boldsymbol{y}_1, ..., \boldsymbol{y}_n] \in \{0,1\}^{m\times n}$.

1. https://github.com/DSLLcode/DSLL



(a) Streaming Label Mapping (b) Deep Stream Label Learning

Fig. 1. The overall DSLL architecture. The proposed framework has three components: 1) streaming label mapping, designed to capture the relationship between new labels and past labels with the proposed label correlation-aware loss, where *hard* is the ground truth of past labels and *soft* is the prediction provided by the past-labels classifier (teacher); 2) streaming feature distillation, which transforms the knowledge from the teacher to the new-label classifier (student), employing the outputs of the intermediate layer of the teacher as *hint* to guide those of the student; 3) the senior student network, which leverages the relationship and knowledge to finally model new labels. The red dotted lines denote the backpropagation path during learning.

By observing $\mathcal{D}$, multi-label learning aims to derive a proper learning model $\mathbf{M}_m$ that generates the prediction on a label vector for a test instance. Then new labels data arrive in a streaming fashion. For simplicity, we denote $\boldsymbol{y}_i^{new} = [y_i^{m+1}, y_i^{m+2}, ..., y_i^{m+k}]$ as the new $k$-labels vector for instance $\boldsymbol{x}_i$, where $k \geq 1$. The MLL problem in the changing environment is to derive a learning model on the continually emerging new labels.

### 3.2 Overall Structure

We propose DSLL, designed to accommodate emerging new labels. Our overall learning framework is illustrated in Figure 1. DSLL can overall be regarded as a new label classifier consisting of three components. First, we construct streaming label mapping $\mathbf{S}_m$ to capture and maintain the relationships from past labels to new labels, followed by the proposed label correlation loss function to preserve cross-label dependency (Section 3.3). As shown in Figure 1(a), $\mathbf{S}_m$ projects past label vectors $\boldsymbol{y}$ to new label vectors $\boldsymbol{y}^{new}$. In the testing process, we can only use the prediction value $\hat{\boldsymbol{y}}$ provided by the past-label classifier as the input of $\mathbf{S}_m$, since label vectors are unknown. Therefore, in the training process, we jointly employ the true past label vectors $\boldsymbol{y}$ (hard) and the prediction $\hat{\boldsymbol{y}}$ (soft) as the input of the mapping $\mathbf{S}_m$ to improve its accuracy and robustness. The shallow label mapping network has much fewer parameters than the large classifier, which can extract label correlation without computationally intensive retraining. Second, we present the streaming feature distillation, a hint-based learning approach [19], where past-label classifier serves as a teacher network to guide the student network. Streaming feature distillation encourages the feature representation of the student to mimic that of the teacher network at the intermediate layer, which boosts the final performance of the new-label classifier (Section 3.4). Besides, streaming feature distillation network, as the initialization of the new-label classifier, facilitates the convergence of DSLL. Finally, we learn integrally the label relationships obtained by $\mathbf{S}_m$ and the knowledge acquired from the teacher by training a senior student network with the cross-entropy loss function to model the newly-arrived labels (Section 3.5).

### 3.3 Streaming Label Mapping with Label-correlation Loss

Exploring and exploiting the relationship between labels has been demonstrated to be advantageous and crucial for boosting MLL performance [7, 15, 30]. We propose the streaming label mapping $\mathbf{S}_m : \mathbb{R}^{m\times 1} \rightarrow \{0,1\}^k$ to captures deep relationships between new labels and past labels effectively and inexpensively. Taking advantage of the relationships, DSLL can model new labels more accurately and avoids the huge computational cost associated with retraining all the labels to obtain label correlations. In the training process of $\mathbf{S}_m$, as shown in Figure 1(a), we use a combination of the *soft* label vector $\hat{\boldsymbol{y}}^j$ provided by the past-label classifier and the *hard* label vector $\boldsymbol{y}^j$ denoting ground truth of the past label as the input $\tilde{\boldsymbol{y}}^j$:

$$\tilde{\boldsymbol{y}}^j = \alpha \boldsymbol{y}^j + (1-\alpha)\hat{\boldsymbol{y}}^j,$$

where $j \in [m]$ and $\alpha \in [0,1]$ balance the importance of soft predictions and the true hard labels. Note that $[\tilde{\boldsymbol{y}}_1, ..., \tilde{\boldsymbol{y}}_i] = [\tilde{\boldsymbol{y}}^1, .., \tilde{\boldsymbol{y}}^j]^T$, $i \in [n]$. Distinct from KD, which places the soft labels as the target to learn [18, 26], here we employ both soft labels and hard labels as the input, aiming to make the streaming label mapping more robust in learning the relationship between labels. This is because, on the one hand, the hard past labels keep the unbroken latent relationship with newly-arrived labels, and on the other hand, in the testing process, the hard labels are unknown and $\mathbf{S}_m$ only relies on the soft labels provided by the past-label classifier. Therefore, joint training of hard labels and soft labels can improve the accuracy and robustness of the streaming label mapping and boosts DSLL performance (See Section 5.2). Specifically, $\mathbf{S}_m$ consists of a fully connected network with

ReLU activation function in the hidden layers and sigmoid function in the output layer.

Inspired by Zhang and Zhou [1], we present a novel label correlation-aware loss at the sigmoid output of the streaming label mapping, which is defined as follows:

$$\mathcal{L}_{\mathbf{S}}(\tilde{\boldsymbol{y}},\boldsymbol{y}^{new})= \sum_{i=1}^{n}\frac{1}{|Y_i^+||Y_i^-|}\sum_{(p,q)\in Y_i^+\times Y_i^-}\|(\mathbf{S}_m(\tilde{\boldsymbol{y}}_i)^p-\mathbf{S}_m(\tilde{\boldsymbol{y}}_i)^q)-b\|_2^2 \quad (1)$$

where $Y_i^+$ (or $Y_i^-$) denotes the index set of new labels associated (or non-associated) with $\boldsymbol{x}_i$. Formally, $Y_i^+ = \{j \mid y_i^j = 1\}$ and $Y_i^- = \{j \mid y_i^j = 0\}$, $j \in \{m+1,...,m+k\}$; $b = \max\mathcal{Y} - \min\mathcal{Y}$ is a constant[2]. Given the past label vector $\tilde{\boldsymbol{y}}_i$ as input, $\mathbf{S}_m(\tilde{\boldsymbol{y}}_i)^p-\mathbf{S}_m(\tilde{\boldsymbol{y}}_i)^q$ returns the discrepancy between the output of the mapping $\mathbf{S}_m$ on one label associated with $\boldsymbol{x}_i$ ($p \in Y_i^+$) and one label not associated with it ($q \in Y_i^-$). Note that $(\mathbf{S}_m(\tilde{\boldsymbol{y}}_i)^p) \in [-1,1]$ due to the sigmoid function after the output layer. Therefore, minimizing the label correlation-aware loss is equal to maximizing $\mathbf{S}_m(\tilde{\boldsymbol{y}}_i)^p-\mathbf{S}_m(\tilde{\boldsymbol{y}}_i)^q$, which implicitly learns the paired correlation between different labels and enforces the preservation of cross-label dependency.

### 3.4 Streaming Feature Distillation

Hinton et al. [18] proposed a knowledge distillation (KD) method using only the final output of the teacher network. Hint learning [19] demonstrates that using the intermediate representation of the teacher as a *hint* to guide that of the student can improve the final performance of the student. We propose the streaming feature distillation, a hint based learning, which transforms knowledge from the past-label classifier (as a teacher) to the new-label classifier (as a student) to boost the performance of the new-label classifier. As shown on the left side of Figure 1(b), the hint layer is represented as the output of a teacher's intermediate layer guiding a hidden layer of the student, the guided layer, to mimic it. Since the selected hint layer may have more outputs (neurons) than the guided layer, we add an adaptation matrix parameterized by $\mathbf{W}_{Adapt}$ after the guided layer to match the output size of the hint layer. Then, we train the student parameters up to the guided layer by minimizing the following loss function:

$$\mathcal{L}_H(\mathbf{R}_{Guided},\mathbf{W}_{Adapt}) = \|\mathbf{R}_{Hint} - \mathbf{W}_{Adapt}\mathbf{R}_{Guided}\|_2^2,$$

where $\mathbf{R}_{Hint}$ and $\mathbf{R}_{Guided}$ represent the output of the hint layer in the teacher and the output of the guided layer in the student, respectively. Finally, the student parameters up to the guided layer would be saved as the pre-trained parameters and kept fine-tuned with a slow learning rate in later training. Streaming feature distillation outperforms training from scratch because the pre-trained model already extracts a great deal of feature-level knowledge.

### 3.5 Learning from the Past

The *senior student* network is designed to integrally digest the knowledge learned from past labels and the historical classifier. As shown in Figure 1(b), the senior student receives the outputs of both the streaming label mapping and the student network as inputs and generates the prediction for new labels. In particular, we add one hidden layer after the streaming label mapping as a nonlinear transformation $\mathbf{S}_t$, which can transform the outputs of the streaming label mapping to a higher space. Overall, training the senior student can be considered as the process from instance $\boldsymbol{x}_i$ and the output of streaming label mapping to the corresponding new label vector $\boldsymbol{y}_i^{new}$. In the learning stage, the parameters of the senior student, the transformation $\mathbf{S}_t$ and the student (included the pre-trained guided layer) would be trained simultaneously by minimizing the cross-entropy loss as follows:

$$\mathcal{L}_{CE} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=m+1}^{m+k} -y_i^j\log(\hat{y}_i^j) - (1-y_i^j)\log(1-\hat{y}_i^j),$$

where $\hat{y}_i^j$ is the prediction of $j$th label corresponding to the instance $\boldsymbol{x}_i$ generated by the senior student, and $y_i^j$ indicates the ground truth.

The red dotted line from the senior student in Figure 1(b) denotes the backpropagation path. The streaming label mapping is kept frozen to preserve the relationships between new labels and past labels. The student parameters up to the guided layer could be trained at a slow learning rate to fine-tune. Interestingly, we find that having a transformation $\mathbf{S}_t$ is practical for achieving effective label mapping, especially in the case of dimension raising (see Section 5.2).

## 4 Theoretical Analysis

In this section, we theoretically analyze the excess risk bounds for our learning model and provide a generalization error bound for the new-label classifier DSLL. Our analysis demonstrates that the generalizability for modeling the emerging new labels can be improved with the knowledge extracted from past labels. In contrast to linear frameworks [20], it is a bigger challenge to provide proofs for nonlinear models, especially DNNs.

For notational simplicity, we denote $\boldsymbol{y}_i^{new} = [y_i^{m+1}, y_i^{m+2},...,y_i^{m+k}] \in \{0,1\}^k$ as the new $k$-labels vector for instance $\boldsymbol{x}_i$, and $\boldsymbol{x}_i^* = [x_i^1,x_i^2,...,x_i^d,y_i^1,y_i^2,...,y_i^m]$ as the $i$-th tuple input vector combined from $\boldsymbol{x}_i$ and the past label vector $\boldsymbol{y}_i$. We use $\phi(x)=\max\{0,x\}$ to represent the ReLU function, and extend it to vectors $v \in \mathbb{R}^d$ by letting $\phi(\boldsymbol{v}) = (\phi(v_1),...,\phi(v_d))$. We use $\mathbb{1}_{event}$ to denote the indicator function for *event*. We define the following network as the basic composition unit of the DSLL model,

$$\begin{aligned}
&g_{i,0} = \mathbf{A}\boldsymbol{x}_i^* && h_{i,0} = \phi(\mathbf{A}\boldsymbol{x}_i^*) && \text{for } i \in [n]\\
&g_{i,l} = \mathbf{W}_l h_i^{l-1} && h_i^l = \phi(\mathbf{W}_l h_i^{l-1}) && \text{for } i \in [n], l \in [L]\\
&\hat{\boldsymbol{y}}_i = \mathbf{B}h_i^L && && \text{for } i \in [n],
\end{aligned}$$

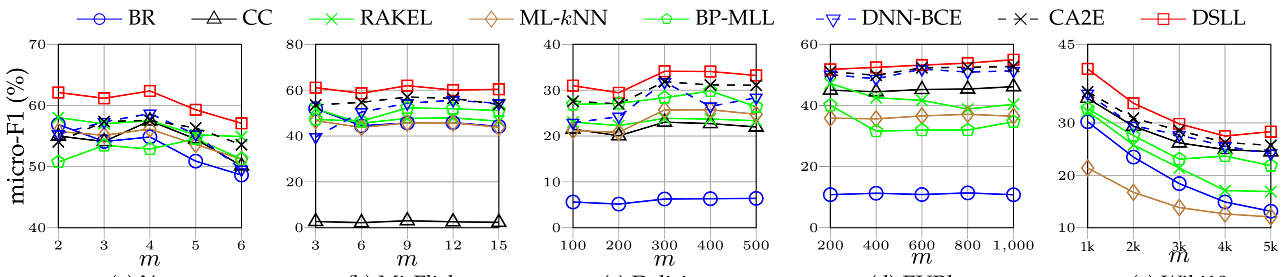
where $\mathbf{A} \in \mathbb{R}^{u_1\times(d+m)}$ is the weight matrix for the input layer, $\mathbf{W}_l \in \mathbb{R}^{u_{l+1}\times u_l}$ is the weight matrix for the $l$-th hidden layer, $\mathbf{B} \in \mathbb{R}^{k\times u_L}$ is the weight matrix for the output layer, and $u_l$ is the number of neurons in the hidden layer $l$. For notational convenience, we use $h_i^{-1}$ to denote input vector, $\mathbf{W}_0$ to denote the corresponding weight matrix for the input layer, and $u_0$ to denote the input data dimension.

Recently, there has been some work on deep learning theory [31, 32, 33]. Allen-Zhu et al. [34] present a more simple and intuitive analysis. To simplify the proof as [34], we

2. for current settings, $b = 1$

TABLE 1
Statistics of five real-world datasets.

| Dataset | Domain | #Training | #Testing | #Feature | #Labels | #Card-Features | #Card-Labels |
|---|---|---|---|---|---|---|---|
| yeast | biology | 1,500 | 917 | 103 | 14 | 103.00 | 4.24 |
| MirFlickr | image | 10,417 | 2,083 | 1,000 | 38 | 539.33 | 4.74 |
| Delicious | web | 12,920 | 3,185 | 500 | 983 | 18.17 | 19.03 |
| EURlex | text | 15,479 | 3,869 | 5,000 | 3,993 | 5.31 | 236.69 |
| Wiki10 | text | 14,146 | 6,616 | 101,938 | 30,938 | 673.45 | 18.64 |



(a) Yeast (b) MirFlickr (c) Delicious (d) EURlex (e) Wiki10

Fig. 2. Performance comparison of learning new labels with different batch sizes by considering 50% of labels as past labels. $m$ indicates the number of new labels.

define the following *diagonal sign matrix* to linearize the DNN corresponding to each instance $\boldsymbol{x}_i$.

**Definition 4.1** (diagonal sign matrix). *For each $i \in [n]$ and $l \in \{0,1,...,L\}$, we denote by $\mathbf{D}_i^l$ the diagonal sign matrix where $\left(\mathbf{D}_i^l\right)_k^k = \mathbb{1}_{(\mathbf{W}_l h_i^{l-1})_k \geq 0}$ for each $k \in [u_l]$.*

As a result, we have $h_i^l = \mathbf{D}_i^l \mathbf{W}_l h_i^{l-1} = \mathbf{D}_i^l g_i^l$ and $\left(\mathbf{D}_i^l\right)_k^k = \mathbb{1}_{(g_i^l)_k \geq 0}$. DSLL integrates three effective networks: Streaming Student $\mathbf{W}_{\mathcal{S}}$, Streaming Label Mapping $\mathbf{W}_{\mathcal{Y}}$, and Senior Student $\mathbf{W}_{\triangle}$. These three networks constitute the DSLL model $\mathbf{W}$ structure, defined as

$$\mathbf{W} = \mathbf{W}_{\triangle}\left[\mathbf{W}_{\mathcal{S}}^T, \mathbf{W}_{\mathcal{Y}}^T\right]^T,$$

which uses $\boldsymbol{x}^*$, the tuples of data $\boldsymbol{x}$ and past labels $\boldsymbol{y}$, as the input to predict the new labels $\boldsymbol{y}^{new}$. According to Definition 4.1, each input tuple $\boldsymbol{x}_i^*$ corresponds to a network parameter in which the prediction is obtained by

$$\hat{\boldsymbol{y}}_i{}^{new} = f_i(\boldsymbol{x}_i^*, \mathbf{W}) = \mathbf{B}\mathbf{D}_i^L \mathbf{W}_L, ..., D_i^1 \mathbf{W}_1 \mathbf{D}_i^0 \mathbf{A} \boldsymbol{x}_i^*, \quad i \in [n].$$

DSLL learns a classifier $\hat{\mathbf{W}}$ by minimizing the *empirical risk*,

$$\hat{\mathcal{L}}(\mathbf{W}) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=m+1}^{m+k} \ell(y_i^j, f_i^j(\boldsymbol{x}_i^*, \mathbf{W})),$$

where $\ell(\cdot)$ denotes the loss function, and we can obtain $\hat{\mathbf{W}} = \text{argmin}_{\mathbf{W} \in \mathcal{W}} \hat{\mathcal{L}}(\mathbf{W})$.

Our goal is to show that the learned $\hat{\mathbf{W}}$ is generalizable, i.e.,

$$\mathcal{L}(\hat{\mathbf{W}}) \leq \inf_{\mathbf{W} \in \mathcal{W}} \mathcal{L}(\mathbf{W}) + \epsilon,$$

where $\mathcal{L}(\mathbf{W})$ is the *population risk* of a classifier, defined as:

$$\mathcal{L}(\mathbf{W}) := \mathbb{E}_{(\boldsymbol{x}^*, \boldsymbol{y}^{new})}\left[\!\!\left[\sum_{j=m+1}^{m+k} \ell(\boldsymbol{y}^j, f^j(\boldsymbol{x}^*, \mathbf{W}))\right]\!\!\right].$$

And we give the following theorem.

**Theorem 1.** *Suppose we learn a new classifier $\mathbf{W}$ in terms of $k$ new labels using formulation $\hat{\mathbf{W}} = \text{argmin}_{\mathbf{W} \in \mathcal{W}} \hat{\mathcal{L}}(\mathbf{W})$ over a set of $n$ training instances. Then, with a probability of at least $1-\delta$, we have*

$$\mathcal{L}(\hat{\mathbf{W}}) \leq \inf_{\mathbf{W} \in \mathcal{W}} \mathcal{L}(\mathbf{W}) + \mathcal{O}\left(\sqrt{\frac{k}{n}}\left(\gamma_x + \gamma_y\sqrt{k}\right)\right) + \mathcal{O}\left(k\sqrt{\frac{\log\frac{1}{\delta}}{n}}\right),$$

*where $\gamma_x$ and $\gamma_y$ are real constant numbers related to the number of neurons in the output layer. We assume, without loss of generality, that $\mathbb{E}[\![\|\boldsymbol{x}\|_2^2]\!] \leq 1$, $\mathbb{E}[\![\|\boldsymbol{y}^{new}\|^2]\!] \leq k$.*

Refer to Appendix A in the Supplementary Materials for the proof. Theorem 1 demonstrates that knowledge learned from past labels helps to improve the generalizability to model new labels without compromising model accuracy.

## 5 EXPERIMENTS

In this section, we conduct extensive experiments to demonstrate the performance of the proposed method with respect to handling new labels. All the computations are performed on a 64-Bit Linux workstation with 10-core Intel Core CPU i7-6850K 3.60GHz processor, 256 GB memory, and 4 Nvidia GTX 1080 Ti GPUs.

### 5.1 Experimental Setup

**Datasets.** We use five readily available multi-label benchmark datasets from different application domains, including three regular-scale datasets[3,4] (Yeast, MirFlickr and Delicious) and two large-scale datasets[5] (EURlex and Wiki10). The statistics of these five real-world data sets are summarized in Table 1.

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

3. http://mulan.sourceforge.net/
4. https://github.com/chihkuanyeh/C2AE
5. http://manikvarma.org/downloads/XC/XMLRepository.html

TABLE 2
Ranking performance of each comparison algorithm for learning new labels with different batch sizes by regarding 50% of labels as past labels. #label denotes the number of new labels. ↓(↑) means the smaller (larger) the value, the better the performance.

| Datasets | #label | micro-AUC ↑ | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | BR | CC | RAKEL | ML-$k$NN | SLEEC | SML | SLL | BP-MLL | DNN-BCE | C2AE | DSLL |
| yeast | 2 | 0.6703 | 0.6750 | 0.6914 | 0.6731 | 0.7003 | 0.7114 | 0.7209 | 0.7228 | 0.7616 | 0.7624 | **0.7793** |
| | 3 | 0.7140 | 0.6972 | 0.7162 | 0.7074 | 0.8003 | 0.8287 | 0.7302 | 0.8437 | 0.8521 | 0.8590 | **0.8641** |
| | 4 | 0.6978 | 0.7096 | 0.7113 | 0.7027 | 0.7751 | 0.7955 | 0.7401 | 0.8216 | 0.8383 | 0.8384 | **0.8530** |
| | 5 | 0.6883 | 0.6979 | 0.7061 | 0.6964 | 0.7775 | 0.7889 | 0.7203 | 0.8139 | 0.8325 | 0.8318 | **0.8507** |
| | 6 | 0.6768 | 0.6743 | 0.7059 | 0.6834 | 0.7645 | 0.7751 | 0.7133 | 0.8115 | 0.8079 | 0.8199 | **0.8229** |
| MirFlickr | 3 | 0.6589 | 0.5049 | 0.6572 | 0.6296 | 0.5476 | 0.6123 | 0.7291 | 0.7448 | 0.7592 | 0.8051 | **0.8122** |
| | 6 | 0.7100 | 0.5045 | 0.7187 | 0.6624 | 0.6400 | 0.6959 | 0.8388 | 0.8476 | 0.8606 | 0.8628 | **0.8713** |
| | 9 | 0.7311 | 0.5071 | 0.7170 | 0.6754 | 0.7001 | 0.7331 | 0.8671 | 0.8692 | 0.8778 | 0.8890 | **0.8972** |
| | 12 | 0.7151 | 0.5057 | 0.7223 | 0.6692 | 0.6862 | 0.7116 | 0.8630 | 0.8600 | 0.8763 | 0.8807 | **0.8886** |
| | 15 | 0.7157 | 0.5051 | 0.7253 | 0.6611 | 0.6805 | 0.7015 | 0.8624 | 0.8584 | 0.8787 | 0.8749 | **0.8873** |
| Delicious | 100 | 0.7133 | 0.5658 | 0.6382 | 0.5707 | 0.7813 | 0.8274 | 0.7981 | 0.8776 | 0.8615 | 0.8545 | **0.8820** |
| | 200 | 0.7080 | 0.5606 | 0.6400 | 0.5692 | 0.7815 | 0.8258 | 0.7956 | 0.8626 | 0.8677 | 0.8321 | **0.8888** |
| | 300 | 0.7177 | 0.5719 | 0.6493 | 0.5925 | 0.7941 | 0.8387 | 0.8068 | 0.8792 | 0.8655 | 0.8777 | **0.9037** |
| | 400 | 0.7159 | 0.5705 | 0.6515 | 0.5916 | 0.7964 | 0.8141 | 0.8059 | 0.8736 | 0.8909 | 0.8656 | **0.9048** |
| | 500 | 0.7144 | 0.5679 | 0.6445 | 0.5868 | 0.7926 | 0.8124 | 0.8030 | 0.8762 | 0.8697 | 0.8657 | **0.9011** |
| EURlex | 200 | 0.6936 | 0.7308 | 0.7015 | 0.6255 | 0.8591 | 0.8216 | 0.8813 | 0.8130 | 0.8201 | 0.8561 | **0.8884** |
| | 400 | 0.6738 | 0.7294 | 0.6821 | 0.6228 | 0.8481 | 0.8611 | 0.8905 | 0.8257 | 0.8423 | 0.8633 | **0.8928** |
| | 600 | 0.6769 | 0.7321 | 0.6743 | 0.6256 | 0.8547 | 0.8735 | 0.8995 | 0.8218 | 0.8472 | 0.8414 | **0.9001** |
| | 800 | 0.6825 | 0.7349 | 0.6575 | 0.6283 | 0.8548 | 0.8775 | 0.9016 | 0.8289 | 0.8491 | 0.8637 | **0.9034** |
| | 1000 | 0.6733 | 0.7361 | 0.6708 | 0.6258 | 0.8406 | 0.8691 | **0.9115** | 0.8246 | 0.8431 | 0.8456 | 0.9106 |
| Wiki10 | 1k | 0.6293 | 0.6535 | 0.6403 | 0.5694 | 0.8092 | 0.8113 | 0.8345 | 0.6978 | 0.7830 | 0.8274 | **0.8406** |
| | 2k | 0.5928 | 0.6244 | 0.6029 | 0.5510 | 0.7505 | 0.7557 | 0.7876 | 0.6545 | 0.6968 | 0.7937 | **0.8013** |
| | 3k | 0.5703 | 0.6078 | 0.5815 | 0.5405 | 0.7567 | 0.7192 | 0.7417 | 0.6505 | 0.7197 | 0.7939 | **0.8023** |
| | 4k | 0.5567 | 0.6008 | 0.5687 | 0.5364 | 0.7198 | 0.7105 | 0.7366 | 0.6556 | 0.7307 | 0.7944 | **0.7996** |
| | 5k | 0.5502 | 0.5984 | 0.5651 | 0.5345 | 0.7087 | 0.7194 | 0.7350 | 0.6463 | 0.7228 | 0.7824 | **0.7865** |
| Datasets | #label | Ranking loss ↓ | | | | | | | | | | |
| | | BR | CC | RAKEL | ML-$k$NN | SLEEC | SML | SLL | BP-MLL | DNN-BCE | C2AE | DSLL |
| yeast | 2 | 0.3064 | 0.2912 | 0.2519 | 0.2966 | 0.1527 | 0.1493 | 0.1668 | 0.1538 | 0.1494 | 0.1483 | **0.1439** |
| | 3 | 0.3059 | 0.3217 | 0.2863 | 0.3064 | 0.0927 | 0.0901 | 0.1674 | 0.0840 | 0.0818 | 0.0812 | **0.0807** |
| | 4 | 0.3409 | 0.3722 | 0.3535 | 0.3433 | 0.1137 | 0.1093 | 0.1891 | 0.1111 | 0.1077 | 0.1114 | **0.1065** |
| | 5 | 0.3694 | 0.4113 | 0.3787 | 0.3731 | 0.1245 | 0.1179 | 0.2189 | 0.1230 | 0.1163 | 0.1242 | **0.1147** |
| | 6 | 0.4080 | 0.4606 | 0.4049 | 0.4136 | 0.1558 | 0.1508 | 0.2382 | 0.1492 | 0.1536 | 0.1514 | **0.1475** |
| MirFlickr | 3 | 0.3032 | 0.5336 | 0.2842 | 0.3157 | 0.2393 | 0.2253 | 0.1179 | 0.1051 | 0.1025 | 0.0881 | **0.0809** |
| | 6 | 0.2868 | 0.5786 | 0.2735 | 0.3337 | 0.2488 | 0.2189 | 0.0586 | 0.0571 | 0.0565 | 0.0569 | **0.0562** |
| | 9 | 0.3366 | 0.6854 | 0.3194 | 0.3981 | 0.2186 | 0.2386 | 0.0672 | 0.0663 | 0.0609 | 0.0580 | **0.0570** |
| | 12 | 0.4011 | 0.7816 | 0.3784 | 0.4690 | 0.2692 | 0.2528 | 0.0877 | 0.0865 | 0.0857 | 0.0772 | **0.0750** |
| | 15 | 0.4209 | 0.7966 | 0.3873 | 0.4929 | 0.2857 | 0.2710 | 0.0899 | 0.0909 | 0.0837 | 0.0858 | **0.0812** |
| Delicious | 100 | 0.4135 | 0.6551 | 0.4550 | 0.6521 | 0.2333 | 0.2235 | 0.1438 | 0.0916 | 0.0890 | 0.0929 | **0.0830** |
| | 200 | 0.5115 | 0.8147 | 0.5603 | 0.7988 | 0.2712 | 0.2523 | 0.1771 | 0.1168 | 0.1217 | 0.1224 | **0.1140** |
| | 300 | 0.5248 | 0.8303 | 0.5710 | 0.7918 | 0.2759 | 0.2594 | 0.1758 | 0.1093 | 0.1071 | 0.1073 | **0.1031** |
| | 400 | 0.5311 | 0.8360 | 0.5711 | 0.7978 | 0.2661 | 0.2608 | 0.1778 | 0.1102 | 0.1079 | **0.1050** | 0.1082 |
| | 500 | 0.5383 | 0.8435 | 0.5841 | 0.8096 | 0.2784 | 0.2679 | 0.1822 | 0.1109 | 0.1154 | 0.1094 | **0.1067** |
| EURlex | 200 | 0.1473 | 0.1281 | 0.1403 | 0.1787 | 0.0654 | 0.0743 | 0.0198 | 0.0879 | 0.0337 | 0.0205 | **0.0138** |
| | 400 | 0.2493 | 0.2048 | 0.2386 | 0.2880 | 0.1101 | 0.0901 | 0.0299 | 0.1062 | 0.0380 | 0.0274 | **0.0226** |
| | 600 | 0.3468 | 0.2898 | 0.3494 | 0.4041 | 0.1552 | 0.1325 | 0.0404 | 0.1848 | 0.0562 | 0.0531 | **0.0313** |
| | 800 | 0.4072 | 0.3400 | 0.4384 | 0.4780 | 0.1871 | 0.1618 | 0.0483 | 0.1530 | 0.0609 | 0.0495 | **0.0330** |
| | 1000 | 0.4861 | 0.3960 | 0.4939 | 0.5619 | 0.2375 | 0.2085 | 0.0585 | 0.2326 | 0.0754 | 0.0721 | **0.0370** |
| Wiki10 | 1k | 0.4746 | 0.4375 | 0.4213 | 0.5520 | 0.0918 | 0.1082 | 0.0867 | 0.2259 | 0.1011 | 0.0483 | **0.0421** |
| | 2k | 0.6205 | 0.5642 | 0.5705 | 0.6818 | 0.2385 | 0.2052 | 0.1259 | 0.3373 | 0.2225 | 0.0648 | **0.0606** |
| | 3k | 0.7324 | 0.6577 | 0.6823 | 0.7766 | 0.3306 | 0.2573 | 0.1897 | 0.3727 | 0.1911 | 0.0715 | **0.0642** |
| | 4k | 0.8167 | 0.7246 | 0.7569 | 0.8424 | 0.4037 | 0.3029 | 0.2123 | 0.3899 | 0.1831 | 0.0751 | **0.0687** |
| | 5k | 0.8538 | 0.7495 | 0.7941 | 0.8698 | 0.4313 | 0.4368 | 0.2234 | 0.4525 | 0.1972 | 0.0833 | **0.0816** |

- *Binary relevance* (BR) [6] is a set of $m$ independent logistic regression classifiers.
- *Classifier chains* (CC) [8] transforms the multi-label learning problem into a chain of binary classification problems to incorporate label dependencies into the classification process.
- RAKEL [10] considers a set of $k$ labels in a multi-label training set as one of the classes of a new single-label classification task.
- ML-kNN [35] identifies each unseen instance's $k$ nearest neighbors in the training set and utilizes the maximum a posteriori principle to determine the label set for the unseen instance.
- SLEEC [11] is a well-known embedding-based method, which learns the label embedding by preserving the pairwise distances between a few nearest label neighbors.
- SML [12] is the state-of-the-art embedding-based method, which further improves the convergence rate of SLEEC.
- SLL [17] is the original streaming label learning method, which leverages the linear relationship between past labels and new labels.
- BP-MLL [1] 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 [2] makes use of ReLU, AdaGrad, Dropout, and other deep learning techniques to train a DNN-based model.

- C2AE [15] is the state-of-the-art deep learning method for multi-label classification, which integrates the DNN architectures of canonical correlation analysis and autoencoder to learn the deep latent space.

The codes of baseline methods are provided by the authors or scikit-multilearn [36].

**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., Hamming loss, 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 [37, 38]. Note that the coverage score is 1 greater than the one given in [6] to extends it to handle the cases where an instance has zero true labels; it is then normalized by the number of labels such that all the evaluation metrics vary between [0,1]. The details of these metrics can be referred in Appendix B.1.

**Settings.** To handle newly arrived labels, we investigated the models with different batch sizes following previous settings [17]. The batch size refers to the number of new labels emerging in a dynamic environment. In particular, for each dataset, we selected randomly 50% (for Delicious 483) labels as past labels and constructed a classifier for the past labels. Then, the remaining labels were treated as new labels with different batch sizes. MLL methods can be extended to handle these new labels through independent modeling. More detailed settings are provided in the Appendix B.

### 5.2 Results

The results will show the performance evaluation in terms of ranking and classification. Then, we shall describe the ablation studies for different components in DSLL.

**Ranking performance evaluation.** First, we evaluate the results obtained by DSLL and other methods for dealing with new labels using the ranking-based metrics. Limited by space, we only show the average results of two measures: ranking loss and micro-AUC (Table 2). The results evaluated by Precision@$k$, AP, macro-AUC, and coverage can be found in the Appendix B.2 of the Supplementary Materials. The results show that DSLL largely outperforms the other approaches with respect to handling newly arrived labels. In this setting, traditional multi-label learning methods could not utilize the knowledge from past labels and classifiers. Although SLL takes advantage of the relationship between past labels and new labels, the linear representation limits its performance. Nevertheless, DSLL not only learns the correlations across new and past labels but also distills the knowledge from the past classifier.

**Classification performance evaluation.** Second, we compare the classification results as evaluated by bipartition-based metrics, where the prediction value of a label belongs to $\{0,1\}$ rather than a ranking value or a probability value. Figure 2 shows the micro-F1 results of learning new labels with different batch sizes. Other results of the bipartition-based metrics (Hamming loss, macro-F1, and instance-F1) can be found in the Supplementary Materials. We can see that DSLL consistently outperforms other multi-label learning methods with respect to all measures. Several machine learning approaches, e.g., BR and ML-$k$NN, perform poorly on datasets with a large number of labels because the interrelationships between labels are not effectively employed. Note that SLEEC, SML, and SLL only focus on ranking results; if adopting a naive thresholding policy (e.g., a threshold of 0.5), they underperform compared to baselines, i.e., BR [39]. Hence, classification performance evaluation excludes the results of SLEEC, SML, and SLL.

**Ablation study.** Finally, as shown in Table 3, we compare different strategies for learning knowledge from the past in DSLL to evaluate the effectiveness of various parts of our proposed framework. AP and coverage [37] are used to measure the performance on the MirFlickr dataset, where 19 labels are used as past labels and 9 labels as new labels. We choose a fully-connected deep network as the baseline, and another network distills the knowledge at the intermediate layer of the past-label classifier to encourage the network to converge to a better value, denoted KD in Table 3. We compare the label correlation-aware loss in Equation (1), denoted LC, with the cross-entropy loss (CE) to achieve slightly better performance. Moreover, we find that the transformation layer $\mathbf{S}_t$ proposed in Section 3.5 is critical for effective relationship representations, which can be used with LC to obtain a 1.7% improvement. Note that a 1% improvement in multi-label classification is considered significant. In the process of streaming label mapping, we discover that the performance of using both hard and soft label vectors denoted $\mathbf{L}_{soft}^{hard}$, is better than using only one of the two, which is supported by the results shown in Table 3. This is because that the labels are unknown in the testing stage, the soft labels predicted by the past-label classifier are sent to the streaming label mapping as the input. Therefore, learning the combination of hard labels and soft labels can improve the robustness of the model in the testing. Finally, we find that our proposed overall method (KD+LC-$\mathbf{S}_t$+$\mathbf{L}_{soft}^{hard}$) significantly outperforms the baseline and others with part strategies.

## 6 Conclusion

In this paper, we propose a novel framework, Deep Streaming Label Learning (DSLL), to address the multi-label learning problem with new labels. DSLL distills the knowledge from past labels and their classifiers to effectively model new emerging labels without retraining the whole multi-label classifier, which has three components: a streaming label mapping, a streaming feature distillation, and a senior student. Our theoretical derivations prove that DSLL framework can provide a tight excess risk bound for new labels. Extensive empirical results show that DSLL significantly outperforms current state-of-the-art methods, and demonstrate the significance of leveraging knowledge learned from the past when modeling new labels.

## Acknowledgements

This work was supported in part by Australian Research Council Projects FL-170100117, DP-180103424, IH-180100002, IC-190100031, and LE-200100049

TABLE 3
The proposed method component comparison, including streaming feature distillation (KD, Section 3.4), cross-entropy loss (CE), our proposed label correlation-aware loss (LC) for label mapping (Section 3.3), and transformation layer $\mathbf{S}_t$ after label mapping (Section 3.5). $\mathbf{L}_{soft}^{hard}$ represents using both hard and soft label vectors in label mapping, while $\mathbf{L}_{hard}$ and $\mathbf{L}_{soft}$ respectively represent the cases where only one of the label vectors is used. ↓(↑) means the smaller (larger) the value, the better the performance.

| | Baseline | KD | CE | LC | LC-$\mathbf{S}_t$ | $\mathbf{L}_{hard}$ | $\mathbf{L}_{soft}$ | $\mathbf{L}_{soft}^{hard}$ | KD+LC-$\mathbf{S}_t$+$\mathbf{L}_{soft}^{hard}$ |
|---|---|---|---|---|---|---|---|---|---|
| Average precision ↑ | 0.3824 | 0.4018 | 0.4113 | 0.4182 | 0.4287 | 0.4159 | 0.4427 | 0.4592 | 0.4925 |
| Coverage ↓ | 0.2415 | 0.2349 | 0.2245 | 0.2221 | 0.2188 | 0.2352 | 0.2289 | 0.2237 | 0.2153 |

## References

[1] M.-L. Zhang and Z.-H. Zhou, “Multilabel neural networks with applications to functional genomics and text categorization,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 10, pp. 1338–1351, 2006.

[2] J. Nam, J. Kim, E. Loza Mencía, I. Gurevych, and J. Fürnkranz, “Large-scale multi-label text classification — revisiting neural networks,” in *Machine Learning and Knowledge Discovery in Databases*, 2014, pp. 437–452.

[3] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, “Sun database: Large-scale scene recognition from abbey to zoo,” in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2010, pp. 3485–3492.

[4] Z. Barutcuoglu, R. E. Schapire, and O. G. Troyanskaya, “Hierarchical multi-label prediction of gene function,” *Bioinformatics*, vol. 22, no. 7, pp. 830–836, 2006.

[5] N. Cesa-Bianchi, M. Re, and G. Valentini, “Synergy of multi-label hierarchical ensembles, data fusion, and cost-sensitive methods for gene functional inference,” *Machine Learning*, vol. 88, no. 1, pp. 209–241, 2012.

[6] G. Tsoumakas, I. Katakis, and I. Vlahavas, *Mining Multi-label Data*. Springer US, 2010, pp. 667–685.

[7] A. K. Menon, A. S. Rawat, S. Reddi, and S. Kumar, “Multilabel reductions: what is my loss optimising?” in *NeurIPS*, 2019, pp. 10 599–10 610.

[8] J. Read, B. Pfahringer, G. Holmes, and E. Frank, “Classifier chains for multi-label classification,” *Machine Learning*, vol. 85, no. 3, p. 333, Jun 2011.

[9] J. Nam, Y.-B. Kim, E. L. Mencia, S. Park, R. Sarikaya, and J. Fürnkranz, “Learning context-dependent label permutations for multi-label classification,” in *ICML*, vol. 97, 2019, pp. 4733–4742.

[10] G. Tsoumakas, I. Katakis, and I. Vlahavas, “Random k-labelsets for multilabel classification,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 2s3, no. 7, pp. 1079–1089, 2011.

[11] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain, “Sparse local embeddings for extreme multi-label classification,” in *NeurIPS*, 2015, pp. 730–738.

[12] W. Liu and X. Shen, “Sparse extreme multi-label learning with oracle property,” in *ICML*, vol. 97, 2019, pp. 4032–4041.

[13] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[14] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85 – 117, 2015.

[15] C.-K. Yeh, W.-C. Wu, W.-J. Ko, and Y.-C. F. Wang, “Learning deep latent spaces for multi-label classification,” in *AAAI*, 2017, pp. 2838–2844.

[16] S.-F. Chen, Y.-C. Chen, C.-K. Yeh, and Y.-C. Wang, “Order-free rnn with visual attention for multi-label classification,” 2018.

[17] S. You, C. Xu, Y. Wang, C. Xu, and D. Tao, “Streaming Label Learning for Modeling Labels on the Fly,” *arXiv preprint arXiv:1604.05449*, 2016.

[18] G. Hinton, O. Vinyals, and J. Dean, “Distilling the Knowledge in a Neural Network,” in *NeurIPS workshop*, 2014.

[19] A. Romero, S. E. Kahou, P. Montréal, Y. Bengio, U. D. Montréal, A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, “Fitnets: Hints for thin deep nets,” in *ICLR*, 2015.

[20] H.-F. Yu, P. Jain, P. Kar, and I. S. Dhillon, “Large-scale multi-label learning with missing labels,” in *ICML*, 2014, pp. 593–601.

[21] Y. Prabhu and M. Varma, “Fastxml: a fast, accurate and stable tree-classifier for extreme multi-label learning,” in *KDD*, 2014.

[22] G. Andrew, R. Arora, J. Bilmes, and K. Livescu, “Deep canonical correlation analysis,” in *ICML*, vol. 28, no. 3, 2013, pp. 1247–1255.

[23] Y. Zhu, K. M. Ting, and Z. Zhou, “Multi-label learning with emerging new labels,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 10, pp. 1901–1914, Oct 2018.

[24] J. Ba and R. Caruana, “Do deep nets really need to be deep?” in *NeurIPS*. Curran Associates, Inc., 2014, pp. 2654–2662.

[25] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, “Learning efficient object detection models with knowledge distillation,” in *NeurIPS*, 2017, pp. 742–751.

[26] B. Heo, J. Kim, S. Yun, H. Park, N. Kwak, and J. Y. Choi, “A comprehensive overhaul of feature distillation,” in *ICCV*, October 2019.

[27] Y. Liu, L. Sheng, J. Shao, J. Yan, S. Xiang, and C. Pan, “Multi-label image classification via knowledge distillation from weakly-supervised detection,” in *ACM International Conference on Multimedia*, 2018, pp. 700–708.

[28] C. Zhou, J. Gu, and G. Neubig, “Understanding knowledge distillation in non-autoregressive machine translation,” in *ICLR*, 2020.

[29] H. Bai, J. Wu, I. King, and M. Lyu, “Few Shot Network Compression via Cross Distillation,” in *AAAI*, 2020.

[30] M. Zhang and Z. Zhou, “A review on multi-label learning algorithms,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 8, pp. 1819–1837, 2014.

[31] D. Zou, Y. Cao, D. Zhou, and Q. Gu, “Gradient descent optimizes over-parameterized deep relu networks,” *Machine Learning*, vol. 109, pp. 467 – 492, 2019.

[32] Y. Cao and Q. Gu, “Generalization error bounds of

gradient descent for learning over-parameterized deep relu networks." in *AAAI*, 2020, pp. 3349–3356.

[33] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai, "Gradient descent finds global minima of deep neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, vol. 97, 2019, pp. 1675–1685.

[34] Z. Allen-Zhu, Y. Li, and Z. Song, "A convergence theory for deep learning via over-parameterization," in *ICML*, vol. 97, 2019, pp. 242–252.

[35] M.-L. Zhang and Z.-H. Zhou, "Ml-knn: A lazy learning approach to multi-label learning," *Pattern Recogn.*, vol. 40, no. 7, pp. 2038–2048, 2007.

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

[37] X.-Z. Wu and Z.-H. Zhou, "A unified view of multi-label performance measures," in *ICML*, vol. 70, 2017, pp. 3780–3788.

[38] H. Jain, Y. Prabhu, and M. Varma, "Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications," in *KDD*. ACM, 2016, pp. 935–944.

[39] J. Nam, E. Loza Mencía, H. J. Kim, and J. Fürnkranz, "Maximizing subset accuracy with recurrent neural networks in multi-label classification," in *NeurIPS*, 2017, pp. 5413–5423.

[40] C. Sammut and G. I. Webb, Eds., *Encyclopedia of Machine Learning: McDiarmid's Inequality*, 2010, pp. 651–652.

[41] M. Ledoux and M. Talagrand, *Probability in Banach Spaces: Rademacher Averages*. Springer Berlin Heidelberg, 1991, pp. 89–121.

[42] A. Maurer, "A vector-contraction inequality for rademacher complexities," in *Algorithmic Learning Theory*, 2016, pp. 3–17.

[43] S. M. Kakade, S. Shalev-Shwartz, and A. Tewari, "Regularization techniques for learning with matrices," *J. Mach. Learn. Res.*, vol. 13, pp. 1865–1890, 2012.

**Zhen Wang** received the B.Sc. degree from the Hunan University, Changsha, China, in 2016, and the M.Sc. degree from Tsinghua University, Beijing, China, in 2014. He is currently pursuing the Ph.D. degree with the UBTECH Sydney Artificial Intelligence Centre, School of Information Technologies, Faculty of Engineering and Information Technologies, The University of Sydney, Darlington, NSW, Australia. His current research interest include data science, machine learning and deep learning.

**Liu Liu** received the B.Sc. degree from the Hebei University of Technology, Tianjin, China, in 2011, and the M.Eng. degree from Beihang University, Beijing, China, in 2014. She is currently working toward the Ph.D. degree with the UBTECH Sydney Artificial Intelligence Centre and the School of Computer Science, The University of Sydney, Darlington, NSW, Australia. Her current research interests include machine learning and optimization.

**Dacheng Tao** (F'15) is a Professor of computer science and an ARC Laureate Fellow with the School of Computer Science and the Faculty of Engineering and Information Technologies, and the Inaugural Director of the UBTECH Sydney Artificial Intelligence Centre, The University of Sydney. He mainly applies statistics and mathematics to artificial intelligence and data science. His research results have expounded in one monograph and in more than 200 publications at prestigious journals and prominent conferences, such as IEEE T-PAMI, T-IP, T-NNLS, IJCV, JMLR, NIPS, ICML, CVPR, ICCV, ECCV, ICDM, and ACM SIGKDD. He was the recipient of the several best paper awards, such as the best theory/algorithm paper runner up award in IEEE ICDM07, the best student paper award in IEEE ICDM13, the distinguished paper award in the 2018 IJCAI, the 2014 ICDM 10-year highest-impact paper award, and the 2017 IEEE Signal Processing Society Best Paper Award. He was also the recipient of the 2015 Austrlian Scopus-Eureka Prize and the 2018 IEEE ICDM Research Contributions Award. He is a Fellow of the Australian Academy of Science, AAAS, IAPR, OSA and SPIE.

## APPENDIX A PROOF OF THEOREM 1

In the section, we provide a proof of Theorem 1. Different from the linear framework presented in [20], we prove the generalizability of the nonlinear model, especially the DNN based model. We will show that with the help of knowledge learned from past labels, the generalizability to model the new labels can be improved without compromising model accuracy. Given training data, the proposed DSLL learns a classifier $\hat{\mathbf{W}}$ by minimizing the *empirical risk*,

$$\hat{\mathcal{L}}(\mathbf{W}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\boldsymbol{y}_i, f_i(\boldsymbol{x}_i^*, \mathbf{W})) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=m+1}^{m+k} \ell(y_i^j, f_i^j(\boldsymbol{x}_i^*, \mathbf{W})),$$

where $\ell(\cdot)$ denotes the loss function; $f_i^j(\boldsymbol{x}_i^*, \mathbf{W})$ is the predicted value of the $j$-th label corresponding to the $i$-th input data $\boldsymbol{x}_i^*$. And we can obtain

$$\hat{\mathbf{W}} = \arg\min_{\mathbf{W}\in\mathcal{W}} \hat{\mathcal{L}}(\mathbf{W}).$$

The goal of the proof is to show that the learned $\hat{\mathbf{W}}$ is generalizable, i.e.,

$$\mathcal{L}(\hat{\mathbf{W}}) \leq \inf_{\mathbf{W}\in\mathcal{W}} \mathcal{L}(\mathbf{W}) + \epsilon,$$

where $\mathcal{L}(\mathbf{W})$ is the *population risk* of a classifier, defined as:

$$\mathcal{L}(\mathbf{W}) = \mathop{\mathbb{E}}_{(\boldsymbol{x}^*, \boldsymbol{y}^{new})} [\![\ell(\boldsymbol{y}, f(\boldsymbol{x}^*, \mathbf{W}))]\!] = \mathop{\mathbb{E}}_{(\widetilde{\boldsymbol{x}}_i^*, \widetilde{\boldsymbol{y}}_i^{new})} \left[\!\!\left[ \frac{1}{n}\sum_{i=1}^{n} \ell(\widetilde{\boldsymbol{y}}_i^{new}, f_i(\widetilde{\boldsymbol{x}}_i^*, \mathbf{W})) \right]\!\!\right].$$

We perform our analysis in three setps:

Step 1: By using McDiarmid's inequality [40], the excess risk of the learned model can be bounded by the expected supremum deviation of empirical risks.

Step 2: We bound the expected supremum deviation by application of Rademacher averages [20, 41, 42].

Step 3: Since we use diagonal sign matrix to denote ReLU activation (Definition 4.1), the network parameters corresponding to each input data can be expressed in the form of a matrix [34]. Then, we reduce the estimation of the Rademacher average to the estimation of the norm by matrix-based regularization techniques [42, 43].

### A.1 Bounding Excess Risk by Expected Supremum Deviation

We first investigate, by using McDiarmid's inequality [40], that the excess risk $\mathcal{L}(\hat{\mathbf{W}}) - \hat{\mathcal{L}}(\hat{\mathbf{W}})$ can be bounded by the expected supremum deviation of empirical risks,

$$\begin{aligned}\mathcal{L}(\hat{\mathbf{W}}) - \hat{\mathcal{L}}(\hat{\mathbf{W}}) &\leq \sup_{\mathbf{W}\in\mathcal{W}} \{\mathcal{L}(\mathbf{W}) - \hat{\mathcal{L}}(\mathbf{W})\} \\ &= \sup_{\mathbf{W}\in\mathcal{W}} \left\{ \mathop{\mathbb{E}}_{(\widetilde{\boldsymbol{x}}_i^*, \widetilde{\boldsymbol{y}}_i^{new})} \left[\!\!\left[ \frac{1}{n}\sum_{i=1}^{n} \ell(\widetilde{\boldsymbol{y}}_i^{new}, f_i(\widetilde{\boldsymbol{x}}_i^*, \mathbf{W})) \right]\!\!\right] - \frac{1}{n}\sum_{i=1}^{n} \ell(\boldsymbol{y}_i^{new}, f_i(\boldsymbol{x}_i^*, \mathbf{W})) \right\} \\ &\triangleq g((\boldsymbol{x}_1^*, \boldsymbol{y}_1^{new}), ..., (\boldsymbol{x}_n^*, \boldsymbol{y}_n^{new})).\end{aligned}$$

Since the decomposable loss function $\ell(\boldsymbol{y}^{new}, f(\boldsymbol{x}^*, \mathbf{W}))= \sum_{j=m+1}^{m+k} \ell(\boldsymbol{y}^j, f^j(\boldsymbol{x}^*, \mathbf{W})$ are bounded, the change in any $(\boldsymbol{x}_i^*, \boldsymbol{y}_i^{new})$ would induce a perturbation of $g((\boldsymbol{x}_1^*, \boldsymbol{y}_1^{new}), ..., (\boldsymbol{x}_n^*, \boldsymbol{y}_n^{new}))$ at most $\mathcal{O}(\frac{k}{n})$. Then by applying McDiarmid's inequality, the sum of squared perturbations is bounded by $\frac{2k^2}{n}$, and thus the excess risk is bounded by a term related to the expectation of $g((\boldsymbol{x}_1^*, \boldsymbol{y}_1^{new}), ..., (\boldsymbol{x}_n^*, \boldsymbol{y}_n^{new}))$, the expected supremum deviation. Therefore, we have established that with probability at least $1 - \delta$,

$$\mathcal{L}(\hat{\mathbf{W}}) - \hat{\mathcal{L}}(\hat{\mathbf{W}}) \leq \mathop{\mathbb{E}}_{(\boldsymbol{x}_i^*, \boldsymbol{y}_i^{new})} [\![ g((\boldsymbol{x}_1^*, \boldsymbol{y}_1^{new}), ..., (\boldsymbol{x}_n^*, \boldsymbol{y}_n^{new})) ]\!] + \mathcal{O}\left( k\sqrt{\frac{\log\frac{1}{\delta}}{n}} \right).$$

Next, the upper bound of the expected supremum deviation will be investigated.

### A.2 Bounding Expected Supremum Deviation by Rademacher Averages

We now bound the expected supremum deviation by Rademacher averages [20, 41, 42]. We have

$$\begin{aligned}&\mathop{\mathbb{E}}_{(\boldsymbol{x}_i^*, \boldsymbol{y}_i^{new})} [\![ g((\boldsymbol{x}_1^*, \boldsymbol{y}_1^{new}), ..., (\boldsymbol{x}_n^*, \boldsymbol{y}_n^{new})) ]\!] \\ &= \mathop{\mathbb{E}}_{(\boldsymbol{x}_i^*, \boldsymbol{y}_i^{new})} \left[\!\!\left[ \sup_{\mathbf{W}\in\mathcal{W}} \left\{ \mathop{\mathbb{E}}_{(\widetilde{\boldsymbol{x}}_i^*, \widetilde{\boldsymbol{y}}_i^{new})} \left[\!\!\left[ \frac{1}{n}\sum_{i=1}^{n} \ell(\widetilde{\boldsymbol{y}}_i^{new}, f_i(\widetilde{\boldsymbol{x}}_i^*, \mathbf{W})) - \ell(\boldsymbol{y}_i^{new}, f_i(\boldsymbol{x}_i^*, \mathbf{W})) \right]\!\!\right] \right\} \right]\!\!\right]\end{aligned}$$

$$
\begin{aligned}
&\leq \mathop{\mathbb{E}}_{\substack{(\boldsymbol{x}_i^*,\boldsymbol{y}_i^{new})\\(\widetilde{\boldsymbol{x}}_i^*,\widetilde{\boldsymbol{y}}_i^{new})}} \left[\!\!\left[ \sup_{\mathbf{W}\in\mathcal{W}} \left\{ \frac{1}{n}\sum_{i=1}^{n} \ell(\widetilde{\boldsymbol{y}}_i^{new}, f_i(\widetilde{\boldsymbol{x}}_i^*,\mathbf{W})) - \frac{1}{n}\sum_{i=1}^{n}\ell(\boldsymbol{y}_i^{new}, f_i(\boldsymbol{x}_i^*,\mathbf{W}))\right\}\right]\!\!\right] \\
&= \mathop{\mathbb{E}}_{\substack{(\boldsymbol{x}_i^*,\boldsymbol{y}_i^{new})\\(\widetilde{\boldsymbol{x}}_i^*,\widetilde{\boldsymbol{y}}_i^{new}),\epsilon_i}} \left[\!\!\left[ \sup_{\mathbf{W}\in\mathcal{W}} \left\{ \frac{1}{n}\sum_{i=1}^{n} \epsilon_i \left(\ell(\widetilde{\boldsymbol{y}}_i^{new}, f_i(\widetilde{\boldsymbol{x}}_i^*,\mathbf{W})) - \ell(\boldsymbol{y}_i^{new}, f_i(\boldsymbol{x}_i^*,\mathbf{W}))\right)\right\}\right]\!\!\right] \\
&\leq \mathop{\mathbb{E}}_{(\boldsymbol{x}_i^*,\boldsymbol{y}_i^{new}),(\widetilde{\boldsymbol{x}}_i,\widetilde{\boldsymbol{y}}_i^{new}),\epsilon_i} \left[\!\!\left[ \sup_{\mathbf{W}\in\mathcal{W}} \left\{ \frac{1}{n}\sum_{i=1}^{n} \epsilon_i \ell(\widetilde{\boldsymbol{y}}_i^{new}, f_i(\widetilde{\boldsymbol{x}}_i^*,\mathbf{W}))\right\}\right]\!\!\right] \\
&\quad + \mathop{\mathbb{E}}_{(\boldsymbol{x}_i^*,\boldsymbol{y}_i^{new}),(\widetilde{\boldsymbol{x}}_i,\widetilde{\boldsymbol{y}}_i^{new}),\epsilon_i} \left[\!\!\left[ \sup_{\mathbf{W}\in\mathcal{W}} \left\{ \frac{1}{n}\sum_{i=1}^{n} \epsilon_i \ell(\boldsymbol{y}_i^{new}, f_i(\boldsymbol{x}_i^*,\mathbf{W}))\right\}\right]\!\!\right] \\
&=2 \mathop{\mathbb{E}}_{(\boldsymbol{x}_i^*,\boldsymbol{y}_i^{new}),\epsilon_i} \left[\!\!\left[ \sup_{\mathbf{W}\in\mathcal{W}} \left\{ \frac{1}{n}\sum_{i=1}^{n} \epsilon_i \ell(\boldsymbol{y}_i^{new}, f_i(\boldsymbol{x}_i,\mathbf{W}))\right\}\right]\!\!\right] \\
&=2 \mathop{\mathbb{E}}_{(\boldsymbol{x}_i^*,\boldsymbol{y}_i^{new}),\epsilon_i} \left[\!\!\left[ \sup_{\mathbf{W}\in\mathcal{W}} \left\{ \frac{1}{n}\sum_{i=1}^{n} \epsilon_i \sum_{j=m+1}^{m+k} \ell(\boldsymbol{y}_i^{j}, f_i^j(\boldsymbol{x}_i^*,\mathbf{W}))\right\}\right]\!\!\right] \\
&\leq \frac{2C}{n} \mathop{\mathbb{E}}_{\boldsymbol{x}_i^*,\epsilon_i^j} \left[\!\!\left[ \sup_{\mathbf{W}\in\mathcal{W}} \left\{ \sum_{i=1}^{n} \sum_{j=m+1}^{m+k} f_i^j(\epsilon_i^j \boldsymbol{x}_i^*,\mathbf{W})\right\}\right]\!\!\right] = 2C\mathcal{R}_n(\mathcal{W}),
\end{aligned}
$$

where $\epsilon_i$ is a Rademacher variable and $\epsilon_i^j$ is an independent doubly indexed Rademacher sequence [20, 41]. Note that the first inequality employs Jensen inequality. The second inequality is based on the convexity of the supremum function. In the last inequality, we use the contraction inequality for Rademacher averages (see 42, corollary 4), and under the assumption that the loss $\ell$ is bounded and $C$-Lipschitz.

### A.3 Estimating the Rademacher Average

Now we estimate the Rademacher average [42] to bound the following quantity:

$$
\mathcal{R}_n(\mathcal{W}) := \frac{1}{n} \mathop{\mathbb{E}}_{\boldsymbol{x}_i^*,\epsilon_i^j} \left[\!\!\left[ \sup_{\mathbf{W}\in\mathcal{W}} \left\{ \sum_{i=1}^{n} \sum_{j=m+1}^{m+k} f_i^j(\epsilon_i^j \boldsymbol{x}_i^*,\mathbf{W})\right\}\right]\!\!\right]. \tag{2}
$$

Before that, we have to express the nonlinear network in matrix form [34]. Note that the deep streaming label learning (DSLL) integrates three effective networks: Streaming Student $\mathbf{W}_{\mathcal{S}}$, Label Self-representation $\mathbf{W}_{\mathcal{Y}}$ and Senior Student $\mathbf{W}_{\triangle}$. These three networks constitute the DSLL model $\mathbf{W}$ structurally, which is defined as $\mathbf{W} = \mathbf{W}_{\triangle} \begin{bmatrix}\mathbf{W}_{\mathcal{S}}\\ \mathbf{W}_{\mathcal{Y}}\end{bmatrix}$. Specifically, $\mathbf{W}$ consists of a fully connected network with the ReLU activation function in the hidden layers and the sigmoid activation function in the output layer.

$$
\text{ReLU} \qquad \phi(x) = max(0, x)
$$

$$
\text{sigmoid} \qquad S(x) = \frac{1}{1+e^{-x}}
$$

According to Definition 4.1, the neural network with ReLU can be expressed in the form of linear matrices corresponding to each input data, i.e.,

$$
\hat{\boldsymbol{y}}_i = f_i(\boldsymbol{x}_i^*, \mathbf{W}) = \mathbf{B}\mathbf{D}_i^L\mathbf{W}_L, ..., D_i^1\mathbf{W}_1\mathbf{D}_i^0\mathbf{A}\boldsymbol{x}_i^*, \tag{3}
$$

where $\mathbf{A} \in \mathbb{R}^{u_1\times(d+m)}$ is the weight matrix for the input layer, $\mathbf{W}_l \in \mathbb{R}^{u_{l+1}\times u_l}$ is the weight matrix for the $l$-th hidden layer, $\mathbf{B} \in \mathbb{R}^{k\times u_L}$ is the weight matrix for the output layer, and $u_l$ is the number of neurons in the hidden layer $l$. $\mathbf{D}_i^l$ is diagonal sign matrix defined in Definition 4.1, which denotes the ReLU function of $l$-th layer corresponding to $\boldsymbol{x}_i^*$. We define the network matrix

$$
\boldsymbol{w}_i = \mathbf{D}_i^L\mathbf{W}_L, ..., D_i^1\mathbf{W}_1\mathbf{D}_i^0\mathbf{A},
$$

where $\boldsymbol{w}_i$ is related to each input data $\boldsymbol{x}_i^*$ since different inputs correspond to different diagonal sign matrix $\mathbf{D}_i^l$ to represent the effect of the nonlinear ReLU function. Then, Equation 3 can be rewrote:

$$
\hat{\boldsymbol{y}}_i = f_i(\boldsymbol{x}_i^*, \mathbf{W}) = \mathbf{B}\boldsymbol{w}_i\boldsymbol{x}_i^*.
$$

Note that $\mathbf{B} \in \mathbb{R}^{k\times u_L}$ is the ouput matrix independent of input data $\boldsymbol{x}_i^*$. Moreover,

$$
\hat{y}_i^j = \mathbf{B}^j\boldsymbol{w}_i\boldsymbol{x}_i^*,
$$

where $\mathbf{B}^j \in \mathbb{R}^{1\times u_L}$ denotes the $j$-th label component of output matrix, $j \in [m+1, ..., m+k]$. Corresponding to $\mathbf{W}_{\mathcal{S}}$, $\mathbf{W}_{\mathcal{Y}}$, and $\mathbf{W}_{\triangle}$ above, we use $\boldsymbol{w}_{\mathcal{S}_i}$, $\boldsymbol{w}_{\mathcal{Y}_i}$, $\boldsymbol{w}_{\triangle_i}$ denote the networks parameters of streaming student, label mapping and senior student with respect to $\boldsymbol{x}_i^*$. We define $\boldsymbol{w}_{\triangle_i} := \left[\boldsymbol{w}_{\triangle\mathcal{S}_i}, \boldsymbol{w}_{\triangle\mathcal{Y}_i}\right]$ to denote the two components of $\boldsymbol{w}_{\triangle_i}$ corresponding streaming student and label mapping by column. Then, we have

$$
\begin{aligned}
\mathcal{R}_n(\mathcal{W}) =& \frac{1}{n} \underset{\boldsymbol{x}_i^*, \epsilon_i^j}{\mathbb{E}} \left[\!\!\left[ \sup_{\mathbf{W}\in\mathcal{W}} \left\{ \sum_{i=1}^{n} \sum_{j=m+1}^{m+k} \epsilon_i^j \mathbf{B}^j \boldsymbol{w}_i \boldsymbol{x}_i^* \right\} \right]\!\!\right] \\
=& \frac{1}{n} \underset{\boldsymbol{x}_i^*, \epsilon_i^j}{\mathbb{E}} \left[\!\!\left[ \sup_{\mathbf{W}\in\mathcal{W}} \left\{ \sum_{i=1}^{n} \sum_{j=m+1}^{m+k} \epsilon_i^j \mathbf{B}^j \boldsymbol{w}_{\triangle_i} \begin{bmatrix} \boldsymbol{w}_{\mathcal{S}_i} \boldsymbol{x}_i \\ \boldsymbol{w}_{\mathcal{Y}_i} \boldsymbol{y}_i \end{bmatrix} \right\} \right]\!\!\right] \\
=& \frac{1}{n} \underset{\boldsymbol{x}_i^*, \epsilon_i^j}{\mathbb{E}} \left[\!\!\left[ \sup_{\mathbf{W}\in\mathcal{W}} \left\{ \sum_{i=1}^{n} \sum_{j=m+1}^{m+k} \epsilon_i^j \mathbf{B}^j \left[\boldsymbol{w}_{\triangle\mathcal{S}_i}, \boldsymbol{w}_{\triangle\mathcal{Y}_i}\right] \begin{bmatrix} \boldsymbol{w}_{\mathcal{S}_i} \boldsymbol{x}_i \\ \hat{\boldsymbol{y}}_i^{new} \end{bmatrix} \right\} \right]\!\!\right] \\
=& \frac{1}{n} \underset{\boldsymbol{x}_i^*, \epsilon_i^j}{\mathbb{E}} \left[\!\!\left[ \sup_{\mathbf{W}\in\mathcal{W}} \left\{ \sum_{i=1}^{n} \sum_{j=m+1}^{m+k} \epsilon_i^j \mathbf{B}^j \left(\boldsymbol{w}_{\triangle\mathcal{S}_i} \boldsymbol{w}_{\mathcal{S}_i} \boldsymbol{x}_i + \boldsymbol{w}_{\triangle\mathcal{Y}_i} \hat{\boldsymbol{y}}_i^{new}\right) \right\} \right]\!\!\right] \\
=& \frac{1}{n} \underset{\boldsymbol{x}_i^*, \epsilon_i^j}{\mathbb{E}} \left[\!\!\left[ \sup_{\mathbf{W}\in\mathcal{W}} \left\{ \sum_{i=1}^{n} \sum_{j=m+1}^{m+k} \left(\epsilon_i^j \mathbf{B}^j \boldsymbol{w}_{\triangle\mathcal{S}_i} \boldsymbol{w}_{\mathcal{S}_i} \boldsymbol{x}_i + \epsilon_i^j \mathbf{B}^j \boldsymbol{w}_{\triangle\mathcal{Y}_i} \left((\hat{\boldsymbol{y}}_i^{new} - \boldsymbol{y}_i^{new}) + \boldsymbol{y}_i^{new}\right)\right) \right\} \right]\!\!\right] \\
=& \frac{1}{n} \underset{\boldsymbol{x}_i^*, \epsilon_i^j}{\mathbb{E}} \left[\!\!\left[ \sup_{\mathbf{W}\in\mathcal{W}} \left\{ \sum_{i=1}^{n} \sum_{j=m+1}^{m+k} \left(\epsilon_i^j \mathbf{B}^j \boldsymbol{w}_{\triangle\mathcal{S}_i} \boldsymbol{w}_{\mathcal{S}_i} \boldsymbol{x}_i + \epsilon_i^j \mathbf{B}^j \boldsymbol{w}_{\triangle\mathcal{Y}_i} \left(\xi_i + \boldsymbol{y}_i^{new}\right)\right) \right\} \right]\!\!\right] \\
=& \frac{1}{n} \underset{\boldsymbol{x}_i^*, \epsilon_i^j}{\mathbb{E}} \left[\!\!\left[ \sup_{\mathbf{W}\in\mathcal{W}} \left\{ \sum_{j=m+1}^{m+k} \left\langle \mathbf{B}^j, \sum_{i=1}^{n} \epsilon_i^j \boldsymbol{w}_{\mathcal{S}\triangle_i} \boldsymbol{x}_i \right\rangle + \sum_{j=m+1}^{m+k} \left\langle \mathbf{B}^j, \sum_{i=1}^{n} \epsilon_i^j \boldsymbol{w}_{\triangle\mathcal{Y}_i} \left(\xi_i + \boldsymbol{y}_i^{new}\right) \right\rangle \right\} \right]\!\!\right],
\end{aligned} \tag{4}
$$

where $\xi_i$ is the discrepancy between $\hat{\boldsymbol{y}}_i^{new}$ and $\boldsymbol{y}_i^{new}$, which can be sufficiently small $\mathbb{E}[\![\|\xi_i\|]\!] = \mathcal{O}(1)$ if the label representation performs accurately; $\boldsymbol{w}_{\triangle\mathcal{S}_i} \boldsymbol{w}_{\mathcal{S}_i}$ is denoted by $\boldsymbol{w}_{\mathcal{S}\triangle_i}$.

Let $F$ be a Hilbert-space, and let $\mathcal{B}(F, \mathbb{R}^k)$ be the set of bounded transformation from $F$ to $\mathbb{R}^k$. We denote $|||\cdot|||$ as a norm of on $\mathcal{B}(F, \mathbb{R}^k)$ with dual norm $|||\cdot|||_*$. Fix some real number $\gamma$, and define a class $\mathcal{W}$ of functions from $F$ to $\mathbb{R}^k$ by

$$
\mathcal{W} = \{\boldsymbol{x} \to \mathbf{W}\boldsymbol{x} : \mathbf{W} \in \mathcal{B}(F, \mathbb{R}^k), |||\mathbf{W}||| \leq \gamma\}.
$$

Equation (4) can be rewrote as follow.

$$
\begin{aligned}
\mathcal{R}_n(\mathcal{W}) =& \frac{1}{n} \underset{\boldsymbol{x}_i^*, \epsilon_i^j}{\mathbb{E}} \left[\!\!\left[ \sup_{\mathbf{W}\in\mathcal{W}} \left\{ tr(H_x^* \mathbf{W}_{\mathcal{S}\triangle}) + tr(H_y^* \mathbf{W}_{\triangle\mathcal{Y}}) \right\} \right]\!\!\right] \\
\leq& \frac{1}{n} \underset{\boldsymbol{x}_i^*, \epsilon_i^j}{\mathbb{E}} [\![\gamma|||H_x^*|||_* + \gamma|||H_y^*|||_*]\!],
\end{aligned} \tag{5}
$$

where $H_x, H_y \in \mathcal{B}(F, \mathbb{R}^k)$ are the random transformations:

$$
\begin{aligned}
H_x: \quad & v \to \left( \left\langle v, \sum_{i=1}^{n} \epsilon_i^j \boldsymbol{w}_{\mathcal{S}\triangle_i}^j \boldsymbol{x}_i \right\rangle, ..., \left\langle v, \sum_{i=1}^{n} \epsilon_i^j \boldsymbol{w}_{\mathcal{S}\triangle_i}^j \boldsymbol{x}_i \right\rangle \right), \\
H_y: \quad & v \to \left( \left\langle v, \sum_{i=1}^{n} \epsilon_i^j \boldsymbol{w}_{\triangle\mathcal{Y}}^j \left(\xi_i + \boldsymbol{y}_i^{new}\right) \right\rangle, ..., \left\langle v, \sum_{i=1}^{n} \epsilon_i^j \boldsymbol{w}_{\triangle\mathcal{Y}}^j \left(\xi_i + \boldsymbol{y}_i^{new}\right) \right\rangle \right).
\end{aligned}
$$

Bounding $\mathbb{E}\,|||\cdot|||_*$ depends on the nature of the norm $|||\cdot|||$. In Equation (5), we use the Hilbert-Schmidt or Frobenius norm, denoted $\|\cdot\|_F$, where

$$\begin{aligned}
&\frac{1}{n}\underset{\boldsymbol{x}_i^*,\epsilon_i^j}{\mathbb{E}}[\![\gamma|||H_x^*|||_* + \gamma|||H_y^*|||_*]\!] \\
=&\frac{1}{n}\underset{\boldsymbol{x}_i^*,\epsilon_i^j}{\mathbb{E}}\left[\!\!\left[\gamma\sqrt{\sum_{j=m+1}^{m+k}\left\|\sum_{i=1}^{n}\epsilon_i^j\boldsymbol{w}_{\mathcal{S}\Delta_i}\boldsymbol{x}_i\right\|^2} + \gamma\sqrt{\sum_{j=m+1}^{m+k}\left\|\sum_{i=1}^{n}\epsilon_i^j\boldsymbol{w}_{\Delta\mathcal{Y}_i}\left(\xi_i+\boldsymbol{y}_i^{new}\right)\right\|^2}\right]\!\!\right] \\
\leq&\frac{1}{n}\underset{\boldsymbol{x}_i^*,\epsilon_i^j}{\mathbb{E}}\left[\!\!\left[\gamma\left\|\mathbf{W}_{\mathcal{S}\Delta_i}\right\|_F\sqrt{\sum_{j=m+1}^{m+k}\left\|\sum_{i=1}^{n}\epsilon_i^j\boldsymbol{x}_i\right\|^2} + \gamma\left\|\mathbf{W}_{\Delta\mathcal{Y}_i}\right\|_F\sqrt{\sum_{j=m+1}^{m+k}\left\|\sum_{i=1}^{n}\epsilon_i^j\left(\xi_i+\boldsymbol{y}_i^{new}\right)\right\|^2}\right]\!\!\right] \\
=&\frac{1}{n}\underset{\boldsymbol{x}_i^*,\epsilon_i^j}{\mathbb{E}}\left[\!\!\left[\gamma_x\sqrt{\sum_{j=m+1}^{m+k}\left\|\sum_{i=1}^{n}\epsilon_i^j\boldsymbol{x}_i\right\|^2} + \gamma_y\sqrt{\sum_{j=m+1}^{m+k}\left\|\sum_{i=1}^{n}\epsilon_i^j\left(\xi_i+\boldsymbol{y}_i^{new}\right)\right\|^2}\right]\!\!\right] \\
\leq&\frac{1}{n}\underset{\boldsymbol{x}_i^*,\epsilon_i^j}{\mathbb{E}}\left[\!\!\left[\gamma_x\sqrt{k\sum_{i=1}^{n}\|\boldsymbol{x}_i\|^2} + \gamma_y\sqrt{k\sum_{i=1}^{n}\|(\xi_i+\boldsymbol{y}_i^{new})\|^2}\right]\!\!\right] .
\end{aligned} \tag{6}$$

In Equation (6), the first inequality applies the Cauchy-Schwarz inequality; $\gamma_x$ and $\gamma_y$ are real constant, which are the upper bounds of the norm of trained network parameters [34]; The second inequality estimates the Rademacher complexity for vector-valued classes (see 42, Section 4.2). Without loss of generality, we assume that $\mathbb{E}[\![\|\boldsymbol{x}\|^2]\!] \leq 1,\ \mathbb{E}[\![\|\xi_i+\boldsymbol{y}_i^{new}\|^2]\!] \leq 4k$. This proves

$$\mathcal{R}_n(\mathcal{W}) \leq \sqrt{\frac{k}{n}}\left(\gamma_x + \gamma_y\sqrt{4k}\right),$$

which establishes Theorem 1.

TABLE 4
Definitions of ten multi-label performance measures.

| Measure | Formulation | Note |
|---|---|---|
| Hamming loss | $hloss(H)=\frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\mathbb{1}\{h_i^j\neq y_i^j\}$ | The fraction of misclassified labels. |
| ranking loss | $rloss(F)=\frac{1}{n}\sum_{i=1}^{n}\frac{\lvert\mathcal{S}_{\text{rank}}^i\rvert}{\lvert Y_{i\cdot}^+\rvert\lvert Y_{i\cdot}^-\rvert}$ $\mathcal{S}_{\text{rank}}^i=\{(u,v)\mid f_u(\boldsymbol{x}_i)\leq f_v(\boldsymbol{x}_i),(u,v)\in Y_{i\cdot}^+\times Y_{i\cdot}^-\}$ | The average fraction of reversely ordered label pairs of each instance. |
| coverage | $coverage(F)=\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}\{\max_{j\in Y_{i\cdot}^+} rank_F(\boldsymbol{x}_i,j)-1\}$ | The number of more labels on average should include to cover all relevant labels |
| average precision | $avgprec(F)=\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\lvert Y_{i\cdot}^+\rvert}\sum_{j\in Y_{i\cdot}^+}\frac{\lvert\mathcal{S}_{\text{precision}}^{ij}\rvert}{rank_F(\boldsymbol{x}_i,j)}$ $\mathcal{S}_{\text{precision}}^{ij}=\{k\in Y_{i\cdot}^+\mid rank_F(\boldsymbol{x}_i,k)\leq rank_F(\boldsymbol{x}_i,j)\}$ | The average fraction of relevant labels ranked higher than one other relevant label. |
| Precision@$k$ | $Precision@k(F)=\frac{1}{n}\sum_{i=1}^{n}\frac{\lvert Y_{i\cdot}^+\cap\top_k(f(\boldsymbol{x}_i))\rvert}{k}$ $\top_k(f(\boldsymbol{x}_i))=\{j\mid f^j(\boldsymbol{x}_i)\in \underset{k}{Top}(f^1(\boldsymbol{x}_i),\ldots,f^m(\boldsymbol{x}_i))\}$ | It is the fraction of correct predictions among the first $k$ predicted labels. |
| macro-F1 | $\textit{macro-F1}(H)=\frac{1}{m}\sum_{j=1}^{m}\frac{2\sum_{i=1}^{n}y_{ij}h_{ij}}{\sum_{i=1}^{n}y_{ij}+\sum_{i=1}^{n}h_{ij}}$ | F-measure averaging on each label. |
| micro-F1 | $\textit{micro-F1}(H)=\frac{2\sum_{j=1}^{m}\sum_{i=1}^{n}y_{ij}h_{ij}}{\sum_{j=1}^{m}\sum_{i=1}^{n}y_{ij}+\sum_{j=1}^{m}\sum_{i=1}^{n}h_{ij}}$ | F-measure averaging on the prediction matrix. |
| instance-F1 | $\textit{instance-F1}(H)=\frac{1}{n}\sum_{i=1}^{n}\frac{2\sum_{j=1}^{m}y_{ij}h_{ij}}{\sum_{j=1}^{m}y_{ij}+\sum_{j=1}^{m}h_{ij}}$ | F-measure averaging on each instance. |
| macro-AUC | $\textit{macro-AUC}(F)=\frac{1}{m}\sum_{j=1}^{m}\frac{\lvert\mathcal{S}_{\text{macro}}^j\rvert}{\lvert Y_{\cdot j}^+\rvert\lvert Y_{\cdot j}^-\rvert}$ $\mathcal{S}_{\text{macro}}^j=\{(a,b)\in Y_{\cdot j}^+\times Y_{\cdot j}^-\mid f_j(\boldsymbol{x}_a)\geq f_j(\boldsymbol{x}_b)\}$ | AUC averaging on each label. $\mathcal{S}_{\text{macro}}$ is the set of correctly ordered instance pairs on each label. |
| micro-AUC | $\textit{micro-AUC}(F)=\frac{\lvert\mathcal{S}_{\text{micro}}\rvert}{(\sum_{i=1}^{n}\lvert Y_{i\cdot}^+\rvert)\cdot(\sum_{i=1}^{n}\lvert Y_{i\cdot}^-\rvert)}$ $\mathcal{S}_{\text{micro}}=\{(a,b,i,j)\mid(a,b)\in Y_{\cdot i}^+\times Y_{\cdot j}^-,\ f_i(\boldsymbol{x}_a)\geq f_j(\boldsymbol{x}_b)\}$ | AUC averaging on prediction matrix. $\mathcal{S}_{\text{micro}}$ is the set of correct quadruples. |

## B Experimental Supplementary Material

In this section, we provide more detailed experimental settings and results. Section B.1 presents definition of ten evaluation metrics and experimental settings. Section B.2 presents more extensive empirical evaluation for modeling new labels.

### B.1 Evaluation Metrics and Settings.

Given a test data set denoted by $\mathcal{D}=\{(\boldsymbol{x}_1,\boldsymbol{y}_1),\ldots,(\boldsymbol{x}_n,\boldsymbol{y}_n)\}$, where $\boldsymbol{x}_i\in\mathcal{X}\subseteq\mathbb{R}^{d\times 1}$ is a real vector representing an input feature (instance) and $\boldsymbol{y}_i\in\mathcal{Y}\subseteq\{0,1\}^{m\times 1}$ is the corresponding output label vector ($i\in[n]$, defined as $i\in\{1,\ldots,n\}$). Moreover, $y_i^j=1$ if the $j$-th label is assigned to the instance $\boldsymbol{x}_i$ and $y_i^j=0$ otherwise. For notational simplicity, we use $Y_{i\cdot}^+$ to denote the index set of associated (non-associated) labels of $\boldsymbol{y}_i$. Formally, $Y_{i\cdot}^+=\{j\mid y_i^j=1\}$ and $Y_{i\cdot}^-=\{j\mid y_i^j=0\}$. With respect to $j$-th column of label matrix, $Y_{\cdot j}^+=\{i\mid y_i^j=1\}$ denotes the index set of associated instance of the $j$-th label and $Y_{\cdot j}^-=\{i\mid y_i^j=0\}$ denotes the set of non-associated instances similarly. We use $\lvert\cdot\rvert$ to represent the cardinality of a set.

Table 4 summarizes the ten popular multi-label evaluation metrics used in this paper, which can be divided into bipartition-based metrics, i.e., Hamming loss, 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 [37, 38]. We assume that $H:\mathbb{R}^d\rightarrow\{0,1\}^m$ is the multi-label classifier and predicts which labels an instance is associated with. $H$ can be decomposed as $\{h^1,\ldots,h^m\}$ and $h^j(\boldsymbol{x}_i)$ represents the prediction of $y_i^j$. The results of $H$ can be evaluated by bipartition-based metrics. $F:\mathbb{R}^d\rightarrow\mathbb{R}^m$ is the multi-label predictor, whose predicted value could be regarded as the confidence of association. $F=\{f^1,\ldots,f^m\}$ and $f^j(x_i)$ denotes the predicted value of $y_i^j$, which can be evaluated by ranking-based metrics. $H$ can be induced from $F$ by thresholding techniques. For example, $h^j(\boldsymbol{x}_i)=\mathbb{1}\{f^j(\boldsymbol{x}_i)>t(\boldsymbol{x}_i)\}$, where we use $\mathbb{1}\{event\}$ to denote the indicator function for $event$. In the experiment, we simply use 0.5 as the threshold for the output of DSLL model.

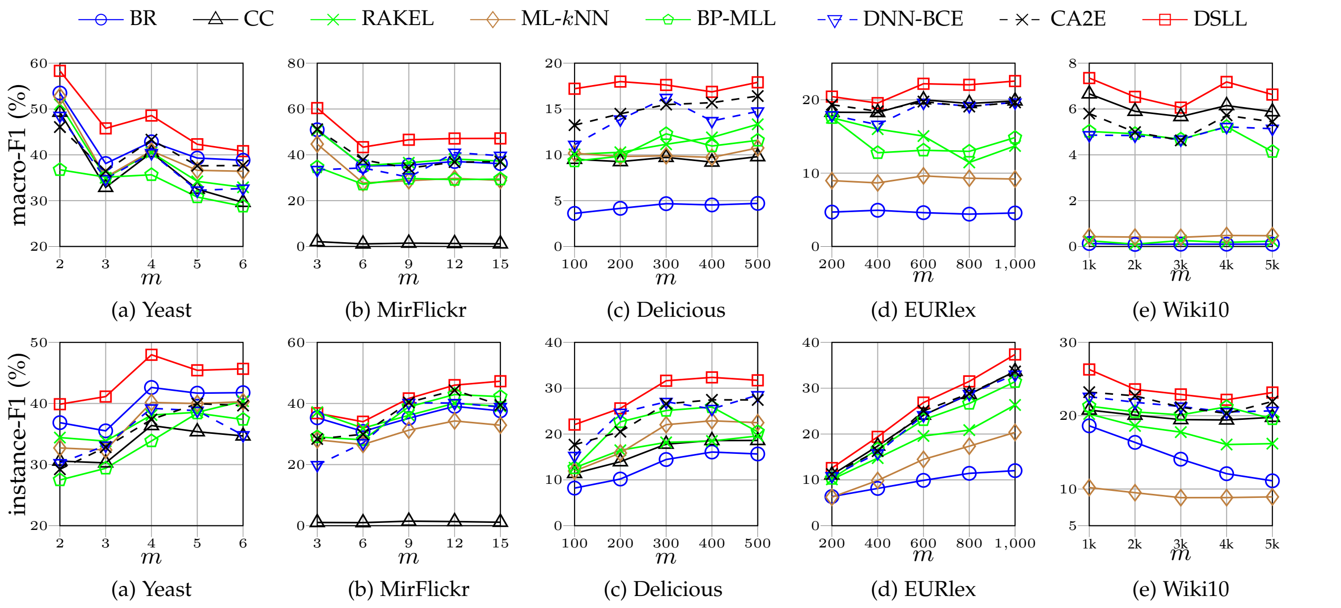


Fig. 3. Comparison of modeling new labels with different batch sizes by considering 50% labels as past labels. $m$ indicates the number of new labels.

## B.2 Detailed Results with Emerging New Labels

- Figure 3 shows the macro-F1 and instance-F1 results for learning new labels with different batch sizes. 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 scores.
- Table 5 shows the Coverage and Precision@1 results for learning new labels with different batch sizes.
- Table 6 shows the Average precision and macro-AUC results for learning new labels with different batch sizes. Note that due to the sparsity of text datasets, we could not use Average precision and macro-AUC to evaluate results on EURlex and Wiki10.
- Table 7 shows the Hamming loss results for learning new labels with different batch sizes. 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. In addition, due to the sparsity of text datasets, the performance of Hamming loss is homogenized on EURlex and Wiki10.

TABLE 5
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 | BP-MLL | DNN-BCE | C2AE | DSLL |
| yeast | 2 | 0.4826 | 0.4749 | 0.4553 | 0.4776 | 0.4057 | 0.4049 | 0.4128 | 0.4124 | 0.4062 | 0.4035 | **0.4013** |
| | 3 | 0.4217 | 0.4326 | 0.4097 | 0.4220 | 0.2846 | 0.2811 | 0.3341 | 0.2803 | 0.2788 | 0.2784 | **0.2777** |
| | 4 | 0.5057 | 0.5172 | 0.5025 | 0.4981 | 0.3408 | 0.3384 | 0.4008 | 0.3424 | 0.3394 | 0.3408 | **0.3351** |
| | 5 | 0.5278 | 0.5429 | 0.5178 | 0.5186 | 0.3302 | 0.3259 | 0.4131 | 0.3335 | 0.3269 | 0.3291 | **0.3230** |
| | 6 | 0.5676 | 0.5901 | 0.5549 | 0.5643 | 0.3595 | **0.3482** | 0.4340 | 0.3550 | 0.3571 | 0.3542 | 0.3499 |
| MirFlickr | 3 | 0.4530 | 0.5721 | 0.4468 | 0.4570 | 0.4117 | 0.3913 | 0.3568 | 0.3435 | 0.3442 | 0.3334 | **0.3236** |
| | 6 | 0.3983 | 0.5815 | 0.3916 | 0.4249 | 0.3602 | 0.3472 | 0.2242 | 0.2247 | 0.2216 | 0.2213 | **0.2132** |
| | 9 | 0.4611 | 0.6909 | 0.4552 | 0.5212 | 0.3651 | 0.3256 | 0.2321 | 0.2342 | 0.2286 | 0.2267 | **0.2153** |
| | 12 | 0.5654 | 0.7882 | 0.5639 | 0.6372 | 0.4714 | 0.4246 | 0.2836 | 0.2859 | 0.2751 | 0.2732 | **0.2601** |
| | 15 | 0.5817 | 0.8029 | 0.5733 | 0.6593 | 0.4954 | 0.4592 | 0.2862 | 0.2916 | 0.2809 | 0.2781 | **0.2663** |
| Delicious | 100 | 0.4587 | 0.6937 | 0.5420 | 0.6992 | 0.3084 | 0.2515 | 0.2208 | 0.1510 | 0.1454 | 0.1499 | **0.1302** |
| | 200 | 0.6147 | 0.8873 | 0.7399 | 0.8854 | 0.4399 | 0.3693 | 0.3408 | 0.2461 | 0.2281 | 0.2383 | **0.2210** |
| | 300 | 0.6937 | 0.9518 | 0.8500 | 0.9486 | 0.5460 | 0.4366 | 0.4529 | 0.3167 | 0.2854 | 0.2934 | **0.2512** |
| | 400 | 0.738 | 0.9679 | 0.8871 | 0.9673 | 0.6044 | 0.5074 | 0.5147 | 0.3669 | 0.3277 | 0.3266 | **0.3245** |
| | 500 | 0.7716 | 0.9782 | 0.9232 | 0.9776 | 0.6626 | 0.4664 | 0.5708 | 0.3868 | 0.4150 | 0.3590 | **0.3189** |
| EURlex | 200 | 0.1517 | 0.1356 | 0.1461 | 0.1823 | 0.0729 | 0.0813 | 0.0223 | 0.0932 | 0.0374 | 0.0234 | **0.0160** |
| | 400 | 0.2588 | 0.2211 | 0.2529 | 0.2978 | 0.1250 | 0.1023 | 0.0360 | 0.1176 | 0.0444 | 0.0325 | **0.0270** |
| | 600 | 0.3697 | 0.3242 | 0.3793 | 0.4280 | 0.1867 | 0.1439 | 0.0534 | 0.2127 | 0.0709 | 0.0669 | **0.0401** |
| | 800 | 0.4457 | 0.3950 | 0.4839 | 0.5200 | 0.2317 | 0.1817 | 0.0686 | 0.1874 | 0.0812 | 0.0670 | **0.0461** |
| | 1000 | 0.5408 | 0.4801 | 0.5679 | 0.6228 | 0.3102 | 0.2190 | 0.0988 | 0.3000 | 0.1076 | 0.1053 | **0.0948** |
| Wiki10 | 1k | 0.5116 | 0.4810 | 0.4577 | 0.5757 | 0.3551 | 0.2942 | 0.1222 | 0.2812 | 0.1413 | 0.0713 | **0.0635** |
| | 2k | 0.6751 | 0.6366 | 0.6251 | 0.7168 | 0.5469 | 0.3524 | 0.1954 | 0.4428 | 0.3217 | 0.1107 | **0.1037** |
| | 3k | 0.7963 | 0.7573 | 0.7461 | 0.8188 | 0.6819 | 0.4835 | 0.3184 | 0.5405 | 0.3322 | 0.1441 | **0.1317** |
| | 4k | 0.8838 | 0.8477 | 0.8339 | 0.8922 | 0.7937 | 0.5126 | 0.3933 | 0.6152 | 0.3726 | 0.1795 | **0.1674** |
| | 5k | 0.9187 | 0.8868 | 0.8798 | 0.9226 | 0.8463 | 0.6424 | 0.4423 | 0.7134 | 0.4236 | **0.2083** | 0.2125 |
| Datasets | #label | Precision@$k$ ($k$=1) ↑ | | | | | | | | | | |
| | | BR | CC | RAKEL | ML-$k$NN | SLEEC | SML | SLL | BP-MLL | DNN-BCE | C2AE | DSLL |
| yeast | 2 | 0.4046 | 0.4286 | 0.4493 | 0.4188 | 0.4689 | 0.4652 | 0.4515 | 0.4646 | 0.4689 | 0.4700 | **0.4844** |
| | 3 | 0.3119 | 0.3195 | 0.3544 | 0.3272 | 0.4591 | 0.4478 | 0.4253 | 0.4678 | 0.4719 | 0.4722 | **0.4820** |
| | 4 | 0.4449 | 0.4460 | 0.4427 | 0.4755 | 0.5420 | 0.5461 | 0.5158 | 0.5300 | 0.5420 | 0.5344 | **0.5551** |
| | 5 | 0.3697 | 0.3948 | 0.4166 | 0.4373 | 0.5256 | 0.5375 | 0.4831 | 0.5256 | 0.5398 | 0.5322 | **0.5442** |
| | 6 | 0.3904 | 0.4417 | 0.4558 | 0.4504 | 0.5333 | **0.5437** | 0.4798 | 0.5311 | 0.5213 | 0.5344 | 0.5420 |
| MirFlickr | 3 | 0.3120 | 0.1723 | 0.3361 | 0.3269 | 0.3255 | 0.3574 | 0.4383 | 0.4580 | 0.4508 | 0.4642 | **0.4729** |
| | 6 | 0.1906 | 0.0134 | 0.2189 | 0.2813 | 0.2967 | 0.3353 | 0.4402 | 0.4508 | 0.4575 | 0.4558 | **0.4614** |
| | 9 | 0.3908 | 0.3341 | 0.3937 | 0.4249 | 0.3658 | 0.3589 | 0.5031 | 0.5122 | 0.5305 | 0.5386 | **0.5511** |
| | 12 | 0.2780 | 0.1263 | 0.3068 | 0.3980 | 0.3845 | 0.4053 | 0.5842 | 0.5924 | 0.5972 | 0.6028 | **0.6035** |
| | 15 | 0.2708 | 0.1440 | 0.2765 | 0.3826 | 0.4369 | 0.3946 | 0.5785 | 0.5655 | 0.6001 | 0.6015 | **0.6020** |
| Delicious | 100 | 0.0776 | 0.1783 | 0.1852 | 0.1805 | 0.2914 | 0.2962 | 0.2967 | 0.3052 | 0.2659 | 0.3091 | **0.3099** |
| | 200 | 0.0553 | 0.1991 | 0.2251 | 0.2232 | 0.3694 | 0.3697 | 0.3704 | 0.3706 | 0.3651 | 0.3623 | **0.3712** |
| | 300 | 0.0333 | 0.2791 | 0.2474 | 0.3108 | 0.4672 | 0.4798 | 0.4893 | 0.4386 | 0.4650 | 0.4898 | **0.4904** |
| | 400 | 0.0936 | 0.3463 | 0.2424 | 0.3425 | 0.5102 | 0.5382 | 0.5338 | 0.5268 | 0.5394 | 0.5130 | **0.5501** |
| | 500 | 0.0424 | 0.3312 | 0.2041 | 0.3372 | 0.5171 | 0.5216 | **0.5504** | 0.5140 | 0.5024 | 0.5278 | 0.5498 |
| EURlex | 200 | 0.0501 | 0.1124 | 0.1042 | 0.0636 | 0.1612 | 0.1601 | 0.1548 | 0.1357 | 0.1571 | 0.1641 | **0.1696** |
| | 400 | 0.0649 | 0.1755 | 0.1548 | 0.1031 | 0.2357 | 0.2474 | 0.2352 | 0.2132 | 0.2409 | 0.2516 | **0.2525** |
| | 600 | 0.0853 | 0.2512 | 0.2132 | 0.1590 | 0.3225 | 0.3135 | 0.3138 | 0.2750 | 0.3386 | 0.3401 | **0.3497** |
| | 800 | 0.0943 | 0.3037 | 0.2303 | 0.1970 | 0.3711 | 0.3958 | 0.3657 | 0.3479 | 0.3830 | 0.3983 | **0.4117** |
| | 1000 | 0.1016 | 0.3546 | 0.2985 | 0.2406 | 0.4129 | 0.4363 | 0.4223 | 0.3874 | 0.4611 | 0.4732 | **0.4854** |
| Wiki10 | 1k | 0.2157 | 0.2269 | 0.2341 | 0.1196 | 0.3776 | 0.3853 | 0.4132 | 0.3094 | 0.3758 | 0.3626 | **0.4143** |
| | 2k | 0.2142 | 0.2304 | 0.2334 | 0.1261 | 0.3478 | 0.3909 | 0.3801 | 0.3171 | 0.3284 | 0.3927 | **0.4113** |
| | 3k | 0.2133 | 0.2491 | 0.2431 | 0.1356 | 0.3965 | 0.4087 | 0.4123 | 0.3253 | 0.3623 | 0.4076 | **0.4335** |
| | 4k | 0.2145 | 0.2573 | 0.2541 | 0.1548 | 0.4149 | 0.4252 | 0.4034 | 0.3464 | 0.3514 | 0.4265 | **0.4504** |
| | 5k | 0.2267 | 0.2922 | 0.2766 | 0.1750 | 0.4173 | 0.4184 | 0.4063 | 0.3581 | 0.3614 | 0.4388 | **0.4503** |

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 | BP-MLL | DNN-BCE | C2AE | DSLL |
| yeast | 2 | 0.4307 | 0.4475 | 0.4504 | 0.4427 | 0.5162 | 0.5315 | 0.5350 | 0.5049 | 0.5486 | 0.5550 | **0.5719** |
| | 3 | 0.3206 | 0.3030 | 0.3042 | 0.2998 | 0.3510 | 0.4113 | 0.3847 | 0.4010 | 0.3832 | 0.4172 | **0.4230** |
| | 4 | 0.3787 | 0.3606 | 0.3573 | 0.3504 | 0.3824 | 0.4412 | 0.4576 | 0.4424 | 0.4543 | 0.4741 | **0.4811** |
| | 5 | 0.3273 | 0.3086 | 0.3075 | 0.3073 | 0.3418 | 0.3718 | 0.4032 | 0.3834 | 0.3948 | 0.4294 | **0.4341** |
| | 6 | 0.3111 | 0.2895 | 0.2970 | 0.2985 | 0.3319 | 0.3593 | 0.3825 | 0.3857 | 0.3934 | 0.3922 | **0.3961** |
| MirFlickr | 3 | 0.3832 | 0.2791 | 0.3580 | 0.3432 | 0.2836 | 0.3251 | 0.4602 | 0.5238 | 0.5278 | 0.6314 | **0.6341** |
| | 6 | 0.2370 | 0.1715 | 0.2365 | 0.2129 | 0.1717 | 0.2435 | 0.3366 | 0.3587 | 0.3978 | 0.4143 | **0.4558** |
| | 9 | 0.2391 | 0.1599 | 0.2444 | 0.2111 | 0.1718 | 0.2519 | 0.3840 | 0.3738 | 0.3986 | 0.4307 | **0.4925** |
| | 12 | 0.2529 | 0.1711 | 0.2611 | 0.2243 | 0.1837 | 0.2373 | 0.4118 | 0.3876 | 0.4507 | 0.4517 | **0.4952** |
| | 15 | 0.2408 | 0.1594 | 0.2460 | 0.2117 | 0.1764 | 0.2391 | 0.4056 | 0.3861 | 0.4566 | 0.4660 | **0.4985** |
| Delicious | 100 | 0.0408 | 0.0595 | 0.0631 | 0.0531 | 0.0684 | 0.0631 | 0.1180 | 0.1028 | 0.1183 | 0.1553 | **0.1563** |
| | 200 | 0.0393 | 0.0580 | 0.0599 | 0.0512 | 0.0699 | 0.0707 | 0.1128 | 0.0994 | 0.1250 | 0.1420 | **0.1428** |
| | 300 | 0.0394 | 0.0612 | 0.0608 | 0.0524 | 0.0717 | 0.0689 | 0.1193 | 0.1220 | 0.1561 | 0.1596 | **0.1604** |
| | 400 | 0.0405 | 0.0584 | 0.0563 | 0.0528 | 0.0724 | 0.0774 | 0.1164 | 0.1113 | 0.1363 | 0.1514 | **0.1520** |
| | 500 | 0.0439 | 0.0622 | 0.0559 | 0.0580 | 0.0760 | 0.0819 | 0.1224 | 0.1290 | 0.1107 | **0.1634** | 0.1631 |
| Datasets | #label | macro-AUC ↑ | | | | | | | | | | |
| | | BR | CC | RAKEL | ML-$k$NN | SLEEC | SML | SLL | BP-MLL | DNN-BCE | C2AE | DSLL |
| yeast | 2 | 0.6507 | 0.6486 | 0.6567 | 0.6555 | 0.6791 | 0.6963 | 0.7140 | 0.6974 | 0.7248 | 0.7235 | **0.7542** |
| | 3 | 0.6269 | 0.5991 | 0.6040 | 0.6037 | 0.6167 | 0.6314 | 0.7292 | 0.7188 | 0.7349 | 0.7453 | **0.7525** |
| | 4 | 0.6358 | 0.6270 | 0.6271 | 0.6225 | 0.6259 | 0.6711 | 0.7428 | 0.6902 | 0.7381 | 0.7541 | **0.7641** |
| | 5 | 0.6302 | 0.6013 | 0.6051 | 0.6103 | 0.6421 | 0.6698 | 0.7219 | 0.6140 | 0.6765 | 0.7475 | **0.7536** |
| | 6 | 0.6275 | 0.5887 | 0.6026 | 0.6083 | 0.6325 | 0.6472 | 0.7090 | 0.7010 | 0.7122 | **0.7377** | 0.7354 |
| MirFlickr | 3 | 0.6625 | 0.5035 | 0.6522 | 0.6111 | 0.5185 | 0.5614 | 0.7151 | 0.7471 | 0.7472 | 0.7966 | **0.8063** |
| | 6 | 0.7088 | 0.5015 | 0.6854 | 0.5729 | 0.5093 | 0.5751 | 0.7666 | 0.7834 | 0.8013 | 0.8069 | **0.8114** |
| | 9 | 0.7414 | 0.5028 | 0.7202 | 0.5789 | 0.5307 | 0.5853 | 0.8059 | 0.8125 | 0.8238 | 0.8369 | **0.8636** |
| | 12 | 0.7284 | 0.5020 | 0.7080 | 0.5804 | 0.5302 | 0.5825 | 0.8007 | 0.8001 | 0.8110 | 0.8305 | **0.8423** |
| | 15 | 0.7290 | 0.5018 | 0.7066 | 0.5788 | 0.5379 | 0.5919 | 0.8077 | 0.8085 | 0.8276 | 0.8336 | **0.8420** |
| Delicious | 100 | 0.6260 | 0.5338 | 0.6271 | 0.5376 | 0.6806 | 0.6894 | 0.7331 | 0.7592 | 0.7383 | 0.7718 | **0.8109** |
| | 200 | 0.6249 | 0.5339 | 0.6187 | 0.5355 | 0.6805 | 0.6912 | 0.7279 | 0.7388 | 0.7633 | 0.7506 | **0.7941** |
| | 300 | 0.6229 | 0.5358 | 0.6203 | 0.5361 | 0.6850 | 0.7015 | 0.7280 | 0.7472 | 0.7543 | 0.7844 | **0.8093** |
| | 400 | 0.6185 | 0.5335 | 0.6179 | 0.5360 | 0.6863 | 0.6931 | 0.7309 | 0.7178 | 0.7720 | 0.7549 | **0.7991** |
| | 500 | 0.6211 | 0.5364 | 0.6200 | 0.5391 | 0.6885 | 0.6997 | 0.7335 | 0.7411 | 0.7290 | 0.7616 | **0.8021** |

TABLE 7
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 | Hamming loss ↓ | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | BR | CC | RAKEL | ML-$k$NN | BP-MLL | DNN-BCE | C2AE | DSLL |
| yeast | 2 | 0.319 | 0.2579 | 0.2557 | 0.2797 | 0.2721 | 0.2612 | 0.2557 | **0.2534** |
| | 3 | 0.2399 | 0.1767 | 0.1765 | 0.1912 | 0.1854 | 0.1759 | **0.1756** | 0.1852 |
| | 4 | 0.2568 | 0.1979 | 0.2067 | 0.2143 | 0.2037 | 0.1955 | 0.1971 | **0.1927** |
| | 5 | 0.2580 | **0.1788** | 0.1893 | 0.1954 | 0.1889 | 0.1830 | 0.1830 | 0.1804 |
| | 6 | 0.2681 | 0.1845 | 0.1827 | 0.1950 | **0.1818** | 0.1832 | 0.1821 | 0.1879 |
| MirFlickr | 3 | 0.3807 | 0.2744 | 0.2708 | 0.3066 | 0.2487 | 0.2447 | 0.2262 | **0.2175** |
| | 6 | 0.3201 | 0.1695 | 0.1689 | 0.1946 | 0.1542 | 0.1441 | 0.1477 | **0.1393** |
| | 9 | 0.2749 | 0.1553 | 0.1555 | 0.1642 | 0.1459 | 0.1373 | 0.1277 | **0.1203** |
| | 12 | 0.2839 | 0.1679 | 0.1691 | 0.1718 | 0.1523 | 0.1355 | 0.1351 | **0.1294** |
| | 15 | 0.2845 | 0.1569 | 0.1579 | 0.1627 | 0.1482 | **0.1179** | 0.1305 | 0.1261 |
| Delicious | 100 | 0.4042 | **0.0140** | 0.0142 | 0.0155 | 0.0357 | 0.0154 | 0.0158 | 0.0146 |
| | 200 | 0.4069 | **0.0131** | 0.0135 | 0.0144 | 0.0321 | 0.0140 | 0.0141 | 0.0139 |
| | 300 | 0.4035 | 0.0156 | 0.0164 | 0.0176 | 0.0458 | 0.0172 | 0.0160 | **0.0153** |
| | 400 | 0.4103 | 0.0167 | 0.0179 | 0.0178 | 0.0401 | 0.0174 | 0.0172 | **0.0166** |
| | 500 | 0.4139 | 0.0173 | 0.0199 | 0.0181 | 0.0411 | 0.0174 | 0.0175 | **0.0170** |
| EURlex | 200 | 0.0095 | 0.0016 | 0.0013 | 0.0013 | 0.0018 | **0.0012** | **0.0012** | **0.0012** |
| | 400 | 0.0069 | 0.0014 | 0.0012 | 0.0011 | 0.0023 | **0.0010** | **0.0010** | **0.0010** |
| | 600 | 0.0077 | 0.0015 | 0.0013 | **0.0010** | 0.0024 | **0.0010** | **0.0010** | **0.0010** |
| | 800 | 0.0074 | 0.0014 | 0.0013 | 0.0011 | 0.0023 | **0.0010** | 0.0011 | 0.0011 |
| | 1000 | 0.0078 | 0.0015 | 0.0013 | 0.0012 | 0.0021 | **0.0010** | **0.0010** | **0.0010** |
| Wiki10 | 1k | **0.0010** | 0.0011 | **0.0010** | **0.0010** | **0.0010** | 0.0011 | 0.0011 | **0.0010** |
| | 2k | **0.0007** | 0.0008 | **0.0007** | **0.0007** | **0.0007** | 0.0008 | **0.0007** | **0.0007** |
| | 3k | **0.0007** | 0.0008 | **0.0007** | **0.0007** | **0.0007** | 0.0008 | **0.0007** | **0.0007** |
| | 4k | **0.0007** | 0.0008 | **0.0007** | **0.0007** | **0.0007** | 0.0008 | **0.0007** | **0.0007** |
| | 5k | 0.0007 | 0.0008 | 0.0007 | **0.0006** | 0.0007 | 0.0008 | 0.0007 | 0.0007 |