A. Proof of Theorem 1

In the section, we provide a proof of Theorem 1. Different from the linear framework presented in (Yu et al., 2014), 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 (Sammut & Webb, 2010), 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 (Ledoux & Talagrand, 1991; Yu et al., 2014; Maurer, 2016).

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 (Allen-Zhu et al., 2019). Then, we reduce the estimation of the Rademacher average to the estimation of the norm by matrix-based regularization techniques (Kakade et al., 2012; Maurer, 2016).

A.1. Bounding Excess Risk by Expected Supremum Deviation

We first investigate, by using McDiarmid's inequality (Sammut & Webb, 2010), 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 (Ledoux & Talagrand, 1991; Yu et al., 2014; Maurer, 2016). 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] \\
\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] \\
&+ \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 ϵ_i is a Rademacher variable and ϵ_i^j is an independent doubly indexed Rademacher sequence (Ledoux & Talagrand, 1991; Yu et al., 2014). Note that the first inequality employs Jensen inequality (Xi et al., 2020). 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 Maurer, 2016, corollary 4), and under the assumption that the loss ℓ is bounded and C-Lipschitz.

A.3. Estimating the Rademacher Average

Now we estimate the Rademacher average (Maurer, 2016) 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 (Allen-Zhu et al., 2019). 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 $\mathbf{W}_\Delta$. These three networks constitute the DSLL model $\mathbf{W}$ structurally, which is defined as $\mathbf{W} = \mathbf{W}_\Delta \begin{bmatrix} \mathbf{W}_\mathcal{S} \\ \mathbf{W}_\mathcal{Y} \end{bmatrix}$. 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 $\mathbf{W}_\Delta$ above, we use $\boldsymbol{w}_{\mathcal{S}_i}$, $\boldsymbol{w}_{\mathcal{Y}_i}$, $\boldsymbol{w}_{\Delta_i}$ denote the networks parameters of streaming student, label mapping and senior student with respect to $\boldsymbol{x}_i^*$. We define $\boldsymbol{w}_{\Delta_i} := \left[\boldsymbol{w}_{\Delta\mathcal{S}_i}, \boldsymbol{w}_{\Delta\mathcal{Y}_i}\right]$ to denote the two components of $\boldsymbol{w}_{\Delta_i}$ corresponding streaming student and label mapping by column. Then, we have

$$\begin{aligned}
\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} \epsilon_i^j \mathbf{B}^j \boldsymbol{w}_i \boldsymbol{x}_i^* \right\} \right]\!\!\right] \\
=& \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} \epsilon_i^j \mathbf{B}^j \boldsymbol{w}_{\Delta_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} \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} \epsilon_i^j \mathbf{B}^j \left[\boldsymbol{w}_{\Delta\mathcal{S}_i}, \boldsymbol{w}_{\Delta\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} \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} \epsilon_i^j \mathbf{B}^j \left(\boldsymbol{w}_{\Delta\mathcal{S}_i} \boldsymbol{w}_{\mathcal{S}_i} \boldsymbol{x}_i + \boldsymbol{w}_{\Delta\mathcal{Y}_i} \hat{\boldsymbol{y}}_i^{new}\right) \right\} \right]\!\!\right] \\
=& \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} \left(\epsilon_i^j \mathbf{B}^j \boldsymbol{w}_{\Delta\mathcal{S}_i} \boldsymbol{w}_{\mathcal{S}_i} \boldsymbol{x}_i + \epsilon_i^j \mathbf{B}^j \boldsymbol{w}_{\Delta\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} \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} \left(\epsilon_i^j \mathbf{B}^j \boldsymbol{w}_{\Delta\mathcal{S}_i} \boldsymbol{w}_{\mathcal{S}_i} \boldsymbol{x}_i + \epsilon_i^j \mathbf{B}^j \boldsymbol{w}_{\Delta\mathcal{Y}_i} \left(\xi_i + \boldsymbol{y}_i^{new}\right)\right) \right\} \right]\!\!\right] \\
=& \frac{1}{n} \mathop{\mathbb{E}}_{\boldsymbol{x}_i^*, \epsilon_i^j} \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}\Delta_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}_{\Delta\mathcal{Y}_i} \left(\xi_i + \boldsymbol{y}_i^{new}\right) \right\rangle \right\} \right]\!\!\right],
\end{aligned} \tag{4}$$

where ξ_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; $\boldsymbol{w}_{\Delta\mathcal{S}_i}\boldsymbol{w}_{\mathcal{S}_i}$ is denoted by $\boldsymbol{w}_{\mathcal{S}\Delta_i}$.

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 γ, 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.

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

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

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

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; γ_x and γ_y are real constant, which are the upper bounds of the norm of trained network parameters (Allen-Zhu et al., 2019); The second inequality estimates the Rademacher complexity for vector-valued classes (see Maurer, 2016, 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.

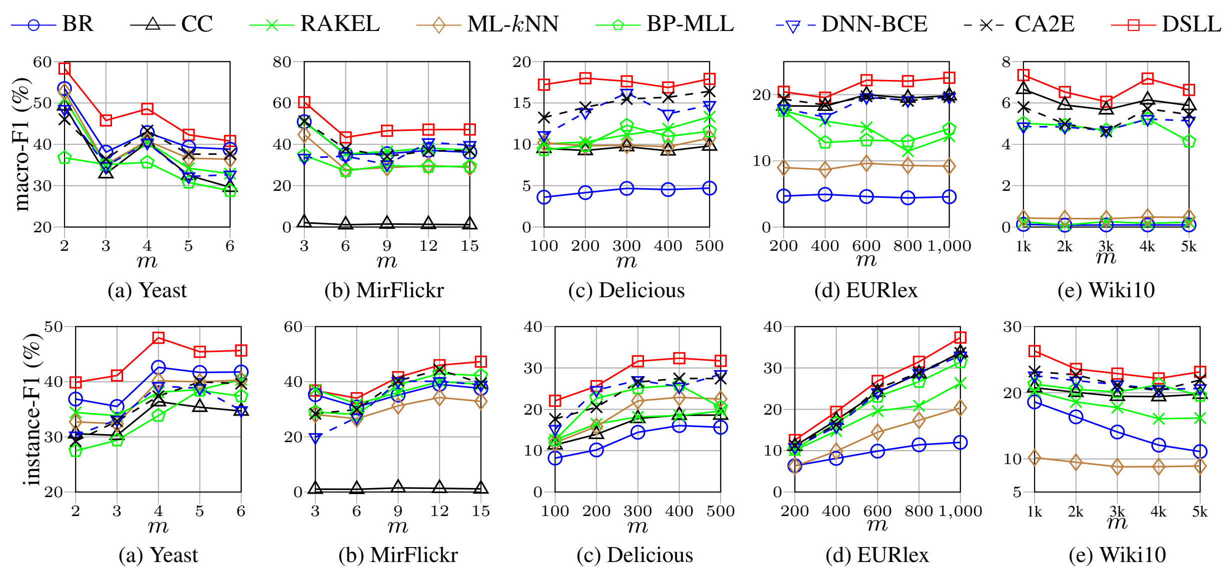
B. Experimental Supplementary Material

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 $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$, 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. For notational simplicity, we use $Y_{i\cdot}^{+}$ to denote the index set of associated (non-associated) labels of $\boldsymbol{y}_i$. Formally, $Y_{i\cdot}^{+} = \{j | y_i^j = 1\}$ and $Y_{i\cdot}^{-} = \{j | y_i^j = 0\}$. With respect to j-th column of label matrix, $Y_{\cdot j}^{+} = \{i | y_i^j = 1\}$ denotes the index set of associated instance of the j-th label and $Y_{\cdot j}^{-} = \{i | y_i^j = 0\}$ denotes the set of non-associated instances similarly. We use $|\cdot|$ 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 (Wu & Zhou, 2017; Jain et al., 2016; Wang et al., 2019; Tsoumakas et al., 2010). We assume that $H : \mathbb{R}^d \to \{0,1\}^m$ is the multi-label classifier and predicts which labels an instance is associated with. H can be decomposed as $\{h^1, ..., h^m\}$ and $h^j(\boldsymbol{x}_i)$ represents the prediction of y_i^j. The results of H can be evaluated by bipartition-based metrics. $F : \mathbb{R}^d \to \mathbb{R}^m$ is the multi-label predictor, whose predicted value could be regarded as the confidence of association. $F = \{f^1, ..., f^m\}$ 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.



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

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

Figure 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

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	$rloss(F) = \frac{1}{n}\sum_{i=1}^{n}\frac{\lvert\mathcal{S}_{\text{rank}}^i\rvert}{\lvert Y_{i\cdot}^+\rvert\lvert Y_{i\cdot}^-\rvert}$ $\mathcal{S}_{\text{rank}}^i = \{(u,v)\lvert f_u(\boldsymbol{x}_i) \leq f_v(\boldsymbol{x}_i), (u,v) \in Y_{i\cdot}^+ \times Y_{i\cdot}^-\}$	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	$avgprec(F) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\lvert Y_{i\cdot}^+\rvert}\sum_{j \in Y_{i\cdot}^+}\frac{\lvert\mathcal{S}_{\text{precision}}^{ij}\rvert}{rank_F(\boldsymbol{x}_i, j)}$ $\mathcal{S}_{\text{precision}}^{ij} = \{k \in Y_{i\cdot}^+\lvert rank_F(\boldsymbol{x}_i, k) \leq rank_F(\boldsymbol{x}_i, j)\}$	The average fraction of relevant labels ranked higher than one other relevant label.
Precision@k	$Precision@k(F) = \frac{1}{n}\sum_{i=1}^{n}\frac{\lvert Y_{i\cdot}^+ \cap \top_k(f(\boldsymbol{x}_i))\rvert}{k}$ $\top_k(f(\boldsymbol{x}_i)) = \{j\lvert f^j(\boldsymbol{x}_i) \in \underset{k}{Top}(f^1(\boldsymbol{x}_i), \ldots, f^m(\boldsymbol{x}_i))\}$	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	$\textit{macro-AUC}(F) = \frac{1}{m}\sum_{j=1}^{m}\frac{\lvert\mathcal{S}_{\text{macro}}^j\rvert}{\lvert Y_{\cdot j}^+\rvert\lvert Y_{\cdot j}^-\rvert}$ $\mathcal{S}_{\text{macro}}^j = \{(a,b) \in Y_{\cdot j}^+ \times Y_{\cdot j}^-\lvert f_j(\boldsymbol{x}_a) \geq f_j(\boldsymbol{x}_b)\}$	AUC averaging on each label. $\mathcal{S}_{\text{macro}}$ is the set of correctly ordered instance pairs on each label.
micro-AUC	$\textit{micro-AUC}(F) = \frac{\lvert\mathcal{S}_{\text{micro}}\rvert}{(\sum_{i=1}^{n}\lvert Y_{i\cdot}^+\rvert)\cdot(\sum_{i=1}^{n}\lvert Y_{i\cdot}^-\rvert)}$ $\mathcal{S}_{\text{micro}} = \{(a,b,i,j)\lvert(a,b) \in Y_{\cdot i}^+ \times Y_{\cdot j}^-,\ f_i(\boldsymbol{x}_a) \geq f_j(\boldsymbol{x}_b)\}$	AUC averaging on prediction matrix. $\mathcal{S}_{\text{micro}}$ is the set of correct quadruples.

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-kNN	SLEEC	SML	SLL	BP-MLL	DNN-BCE	C2AE	DSLL
	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**
yeast	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
	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**
MirFlickr	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**
	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**
Delicious	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**
	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**
EURlex	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**
	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**
Wiki10	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-kNN	SLEEC	SML	SLL	BP-MLL	DNN-BCE	C2AE	DSLL
	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**
yeast	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
	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**
MirFlickr	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**
	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**
Delicious	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
	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**
EURlex	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**
	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**
Wiki10	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-kNN	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-kNN	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-kNN	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