2023 IEEE International Conference on Big Data (BigData) | 979-8-3503-2445-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/BigData59044.2023.10386341

Hybrid Loss for Hierarchical Multi–label Classification Network

Wenting Qi
Department of Computer Science
University at Albany, SUNY
Albany, New York, USA
wqi@albany.edu

Charalampos Chelmis*†
Department of Computer Science
University at Albany, SUNY
Albany, New York, USA
cchelmis@albany.edu

***Abstract*—Machine learning models for hierarchical multi–label classification (HMC) typically achieve low accuracy. This is because such models need not only predict multiple labels for each data instance, but also ensure that predicted labels conform to a given hierarchical structure. Existing state–of–the–art strategies for HMC decouple the learning process from ensuring that predicted labels reside in a path of the hierarchy, thus inevitably degrading the overall classification accuracy. To address this limitation, we propose a novel loss function, which enables a model to encode both a global perspective of the class hierarchy, as well local class–relationships in adjacent hierarchical levels, to ensure that predictions align with the class hierarchy, both during training and testing. We demonstrate the superiority of the proposed approach against multiple state–of–the–art methods for HMC on 20 real–world datasets.**

***Index Terms*—Learning with constraints, local loss, global loss**

I. Introduction

Hierarchical multi–label classification (HMC) is defined as a classification in which data instances are associated with multiple classes that are not disjoint, but organized into hierarchical structures, such as trees [1]–[3] or directed acyclic graphs (DAG) [4]. The main difference between multi–label classification and HMC is the *hierarchy constraint* with respect to the class hierarchy. Hierarchy constraint refers that if a data instance belongs to a class, it must also be an instance of all of its predecessors in the class hierarchy [3]. HMC tasks have attracted increasing attention in the machine learning domain since they have numerous real–world applications including, but not limited to, bioinformatics [1], [5], [6], image annotation [2], and text classification [7]–[9].

From the perspective of the learning algorithm, existing solutions can be generally divided into two categories: algorithmic and neural networks. Methods in the first category include, but are not limited to CSSA [10] (a greedy approach), H–AdaBoost [11] (based on AdaBoost), H–SVM [12], C–SSVM [13] (based on SVM), and CLUS–HMC [14] (based on decision trees). More recently, neural network–based solutions have been proposed, such as CHMCNN [15] (feed–forward neural network) and HMCN–R (recurrent neural network) [16]. Both have been shown to outperform algorithmic approaches, especially for HMCN–R, which leveraged a long short-term memory (LSTM) network [17] and achieved comparable performance compared to CHMCNN, but requires fewer training parameters. Inspired by this result, this work focuses on further improving the performance of LSTM–based solutions for HMC tasks.

From the perspective of the hierarchy constraint, most neural network–based solutions [16], [18], [19] impose the constraint post–processing. However, addressing hierarchy violations for individual data instances post–processing hinders the ability of a model to learn the class hierarchy during the learning process. Methods, such as CHMCNN [15], which incorporate the hierarchy constraint directly into the learning process, have been shown to significantly outperform post–processing methods. The major drawback of CHMCNN is that it optimizes a loss function globally[1]. The global approach is more likely to be cheaper and avoid the well–known error–propagation, but it is less likely to capture local[2] information for any given level in the class hierarchy [3]. However, local information encodes class–relationships in adjacent hierarchical levels, which are essential for improving learning accuracy [16]. Meanwhile, a purely local approach is more likely to result in overfitting [16].

This work introduces a local loss function, that is used to incorporate local class relationships into the learning process, in a manner that complements the global loss proposed by [15]. The resulting framework, **Hybrid LSTM (HLSTM)**, concurrently optimizes both loss functions.

Our main contributions can be summarized as follows.

- First, we propose a local loss function that explicitly incorporates the hierarchy constraint into the learning process.

This material is based upon work supported by the National Science Foundation under Grant No. ECCS–1737443.

*Corresponding author.

†Both authors contributed equally.

[1]Global in HMC refers to discriminating all classes simultaneously. For example, the entire class hierarchy path is $8 \rightarrow 3 \rightarrow 1 \rightarrow 0$ in Figure 1(a).

[2]Local in HMC refers to class–relationships in adjacent hierarchical levels, such as $3 \rightarrow 1$ or $8 \rightarrow 3$ in Figure 1(a). Different from the global approach, the local approach emphasizes particular partial hierarchical class–relationships.

- Second, we propose a novel loss that can simultaneously optimize both local and global loss functions without any post–processing steps for hierarchy constraint.
- Finally, we experimentally evaluate the effectiveness of the proposed approach on 20 benchmark datasets against the state–of–the–art.

II. Background

A. HMCN–R

The hierarchical multi–label classification network (HMCN) in [16] for HMC problems uses both local and global optimization. However, HMCN allows the hierarchy to be violated in predictions, and adopts an additional independent loss to rectify hierarchy inconsistencies, which may lead to conflict with global and local loss when updating the gradient. Specifically, a feed–forward (HMCN–F) and a recurrent neural network (HMCN–R) architecture are exploited. Since HMCN–R is obtained by HMCN–F to reduce the number of learning parameters while maintaining high accuracy, we focus on HMCN–R hereafter.

In HMCN–R, the recurrent cell is designed as an LSTM network, and each iteration is concerned with an unrolled recurrent cell that represents a hierarchical level. The recurrent flow between the recurrent cells captures global information, whereas the unrolled recurrent cell captures local information. Therefore, at each iteration, the gradients flow between recurrent cells, as well as within each recurrent cell. The introduction of the LSTM structure enjoys the advantage of capturing the long–term dependency by using the forgetting gate and the input gate, such that each recurrent cell has access to the information contained in the previous recurrent cells. The loss function of HMCN–R comprises local, global, and hierarchy violation loss. Different from HMCN–R, this work directly incorporates the hierarchy constraint into the global and local losses, deeming the hierarchy violation loss completely unnecessary. The advantage of this approach is to ensure the prediction aligns with the class hierarchy.

B. CHMCNN

Coherent hierarchical multi–label classification neural network (CHMCNN) [15] is another popular model for HMC tasks. CHMCNN defines a max constraint module (MCM), which takes the output score of the learning model as input and imposes the hierarchy constraint. Specifically, assuming D_A denotes the set of A's sub–classes, and h denotes the learning model's output, the MCM for class A is defined as $MCM_A = max_{B \in D_A}(h_B)$. The global loss [15] is defined as $MCLoss_A = -y_A ln(\max_{B \in D_A}(y_B h_B)) - (1 - y_A) ln(1 - MCM_A)$. However, CHMCNN does not capture local information and cannot leverage existing hybrid approaches since MCM value is unobtainable from the local perspective. For instance, computing MCM$_A$ requires h_B for all the descendants of A. However, in the hybrid approach, the value of h_B is available only for the children in the nearest hierarchy level, rather than all descendants. This work proposes a novel loss function that encodes local hierarchy information and imposes the hierarchy constraint at the same time.

III. Preliminaries and Problem Statement

Let $\mathcal{H}$ denote the class hierarchy structure of c classes in total, j denote the class index in $\mathcal{H}$ as $0, 1, \ldots, c-1$ in a top to bottom manner with the index of j, and d represent the depth of $\mathcal{H}$ with level index of h. For two classes c_A and c_B (i.e., c_A and $c_B \in \{0, 1, \ldots, c-1\}$), c_B is a parent of c_A, then we have $c_A \rightarrow c_B$. Additionally, let the training data be $\mathcal{D}_{train} = \{(\mathbf{x}_i, \mathbf{y}_i), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$, where $0 \leq i \leq n$, and the label vector $\mathbf{y}(i) = [y_i^0, y_i^1, y_i^2, \ldots, y_i^{c-1}] \in \{0,1\}^c$ denote the labels of $\mathbf{x}_i$ in $\mathcal{H}$. Given classification network l, let $l_{x_i}(j)$ denote l's output for data instance x_i. Specifically, each label j is assigned to x_i if $l_{x_i}(j)$ exceeds a pre–set threshold t_s (i.e., $j \in y$ if $l(j) \geq t_s$). $\hat{y}_i$ denotes the set of labels that l predicts for $\mathbf{x}_i$. Following [15], we define *hierarchy violation* for HMC tasks as follows. For label c_A and c_B, a hierarchy violation occurs when any $l(c_A) > l(c_B)$, if $c_A \rightarrow c_B$. Given $\mathcal{D}$ and $\mathcal{H}$, the goal is to learn a multi–label classification network l to predict the labels of $\mathbf{x}_i$, while ensuring the predicted labels follow the hierarchy structure. To ensure the hierarchy constraint is always satisfied, the output of $l(c_B)$ should always be larger than or equal to $l(c_A)$ if $c_A \rightarrow c_B$ [16].

IV. Hybrid Hierarchical Multi–label Classification Network

A. Global Optimization Using MCLoss

The majority of global approaches for HMC output prediction results (or scores) at once for all classes. [15] proposed a HMC network that globally optimizes neural network using the MCLoss function discussed in Section II-B. However, global methods such as the above are prone to errors when a wrong prediction only occurs in a class located in the lower hierarchy level. Figure 1 shows such an example. Specifically, Figure 1(a) presents the class hierarchy structure as well as the l's outcomes. The prediction perfectly aligns with the ground–truth for a threshold $t_s = 0.5$ (i.e., for $\forall\ j$ in $\mathcal{H}$, $l(j) \geq 0.5 \Rightarrow j \in \hat{y}_i$). However, if the output score for the $9th$ class happened to be 0.7, as illustrated in Figure 1(a), the MCM loss in [15] updates all scores of the $9th$ node's predecessors to 0.7, as illustrated in Figure 1(b), severely affecting the learning results of the $5th$ and $2nd$ classes. The problem is even more severe if the class hierarchy is deep because an error in a leaf node will affect all of its predecessors. Despite the inadequacy of the global approach to handling such cases, it is still beneficial in scenarios such as the one illustrated in Figure 2(a) where a wrong prediction occurs on the $1st$ and $3rd$ classes. In this case, incorporating the MCLoss into the learning process results in better aligning the predictions to the ground–truth, as shown in Figure 2(b). This work proposes to leverage the strength of the global approach, while avoiding its pitfalls, as discussed in Section IV-B.

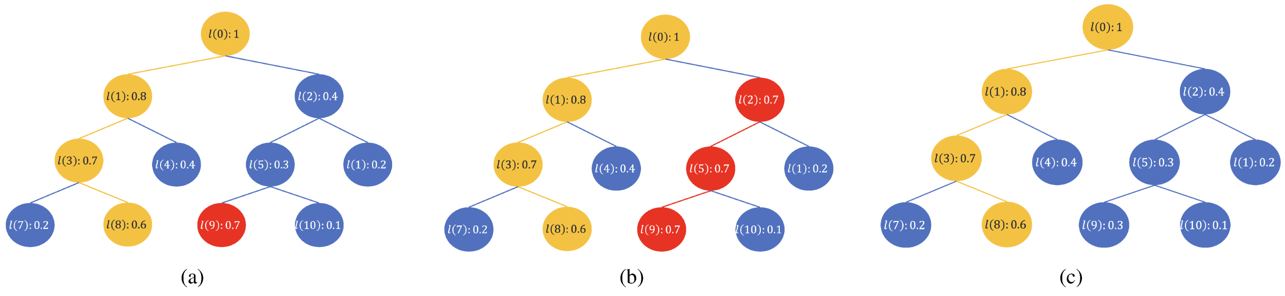


Fig. 1. Toy example (better seen in color) of a tree hierarchy. True classes and prediction errors are highlighted in yellow and red, accordingly. Three scenarios are shown: (a) a single prediction error (b) output after applying MCM (the whole predicted path is incorrect), and (c) output when applying LMC (proposed loss introduced in Section 4.2).

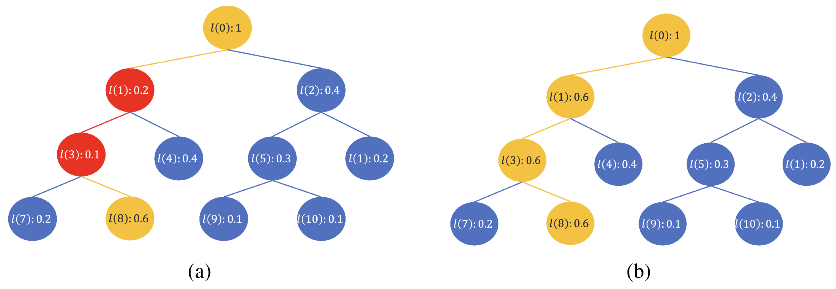


Fig. 2. Different scenario using the same tree hierarchy as in Figure 1. (a) shows mistakes (highlighted in red). (b) shows the output after applying MCM.

B. Local Hierarchical Multi–label Classification

Different from global approaches, local methods focus on the nearest hierarchy–level relationship and make predictions for each class hierarchy level. Based on this idea, we propose a method that imposes the local hierarchy constraint by using the output scores at higher hierarchy levels to calibrate the outputs of children classes located in the lower hierarchy levels in order to avoid hierarchy violation. We call this *local minimal constraint module* (LMC) and implement LMC as a single layer that takes the learning model's output as input and imposes the hierarchy constraint. Specifically, for any non–root class j, we define LMC as follows:

$$LMC(j) = \min_{z \in \{j, pa(j)\}} l(z), \tag{1}$$

where $pa(j)$ denotes the parent classes of j. To better explain LMC, consider the example shown in Figure 1(a). Different from the global approach's result (i.e., Figure 1(b)), LMC is able to correct the mistake by imposing the hierarchy constraint from the local perspective (i.e., $l(9) = 0.7 \Rightarrow l(9) = l(5) = 0.3$) as shown in Figure 1(c). The rationale behind LMC is to impose the hierarchical constraint in the lower level in the top–down manner.

Based on LMC, we define *local hierarchical loss* (LHLoss) function as follows.

Definition 1: Let $C = \{1, 2, ..., j, ..., c-1\}$ (excluding the root class) and $LMC(1), ..., LMC(c-1)$ be defined as in Equation (1), for each $j \in C$, LHLoss is defined as:

$$LHLoss(j) = -y_i^j ln(\min_{z \in \{j, pa(j)\}} (y_i^z l(z))) - \tag{2}$$

$$(1 - y_i^j) ln(1 - LMC(j)). \tag{3}$$

The final local hierarchy loss is defined as:

$$LHLoss = \sum_{j \in C} LHLoss(j). \tag{4}$$

Next, we discuss the advantages of LHLoss compared to MCLoss [15] from a gradient descent perspective, using an illustrative example as shown in Figure 3, where for simplicity, $\mathbf{x}_i$ only has two classes, A and B, and $A \rightarrow B$.

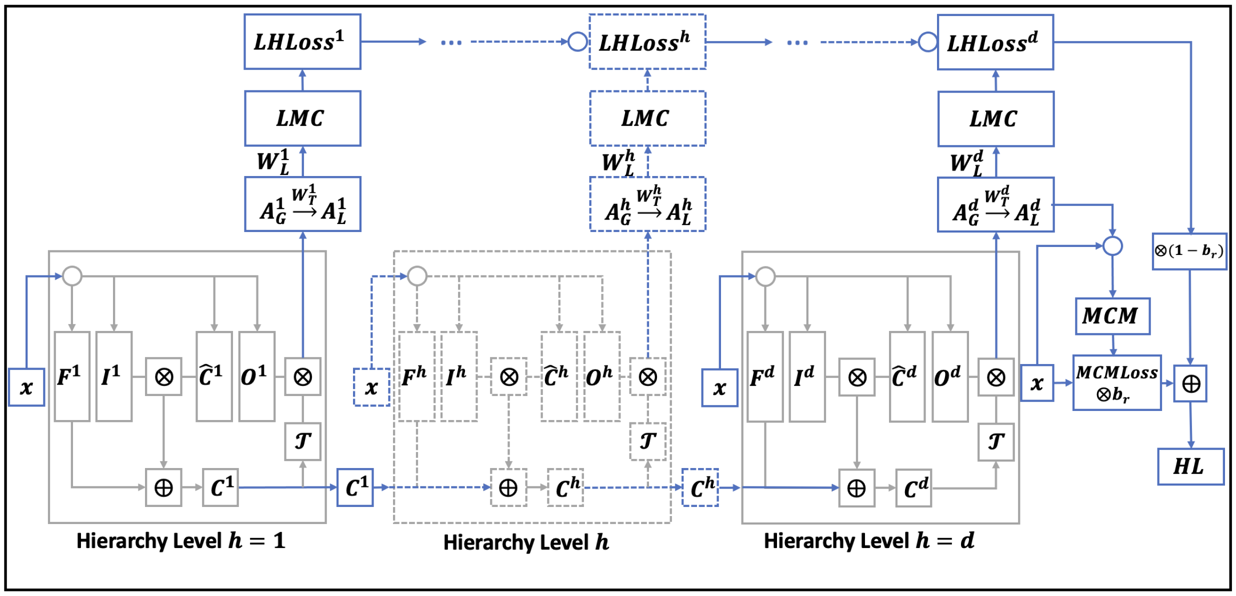
In Example 4.2, MCLoss learns to increase $l(A)$, whereas LHLoss learns to increase $l(B)$. Both MCLoss and LHLoss learn in the right direction (i.e., given $y^A = 1$ and $y^B = 1$), and can even complement each other. However, increasing $l(B)$ according to LHLoss is more beneficial, since it helps to **avoid hierarchy violation** in the next learning round. Conversely, MCLoss only learns to increase $l(A)$ while keeping $l(B)$ unchanged, potentially exacerbating the hierarchy violation in subsequent learning rounds, since $l(A)$ remains larger than $l(B)$. However, LHLoss addresses this issue by increasing $l(B)$ and keeps $l(A)$ to achieve the possible scenario that $l(A)$ will be smaller than $l(B)$. More examples to illustrate the benefits of MCLoss and LHLoss are discussed in the supplementary material. In summary, LHLoss leads to better update of the gradient by directly imposing hierarchy constraint on the "parent level". However, LHLoss alone cannot address all possible scenarios. Instead, it can benefit

Example 4.2. $l(A) = 0.3, y^A = 1; l(B) = 0.1, y^B = 1$. *In this example, a hierarchy violation exists because* $l(B) < l(A)$. *The MCLoss and LHLoss are computed as:*

$$\begin{aligned} MCLoss &= MCLoss(B) + MCLoss(A) \\ &= -ln(max(y^A l(A), l(B))) - ln(MCM(A)) \\ &= -ln(l(A)) - ln(l(A)), \\ \frac{\partial MCLoss}{\partial l(A)} &= -\frac{2}{l(A)} \approx -6.67, \\ \frac{\partial MCLoss}{\partial l(B)} &= 0. \end{aligned}$$

$$\begin{aligned} LHLoss &= LHLoss(B) + LHLoss(A) \\ &= -ln(LMC(B)) - ln(min(l(A), y^B l(B))) \\ &= -ln(l(B)) - ln(y^B l(B)), \\ \frac{\partial LHLoss}{\partial l(A)} &= 0 \\ \frac{\partial LHLoss}{\partial l(B)} &= -\frac{2}{l(B)} = -20. \end{aligned}$$

Fig. 3. Gradient calculation using MCLoss (left) and LHLoss (right).



$LHLoss^1$
$LHLoss^h$
$LHLoss^d$
LMC
W_L^1
W_L^h
W_L^d
$A_G^1 \xrightarrow{W_T^1} A_L^1$
$A_G^h \xrightarrow{W_T^h} A_L^h$
$A_G^d \xrightarrow{W_T^d} A_L^d$
$\otimes(1-b_r)$
MCM
MCMLoss $\otimes b_r$
HL
x
F^1 I^1 $\hat{C}^1$ O^1 $\mathcal{T}$ C^1
F^h I^h $\hat{C}^h$ O^h C^h
F^d I^d $\hat{C}^d$ O^d C^d
Hierarchy Level $h = 1$
Hierarchy Level h
Hierarchy Level $h = d$

Fig. 4. The proposed HLSTM framework for Hybrid Loss (HL).

by properly considering MCLoss as illustrated in the examples presented in Appendix A.

C. Hybrid Loss for Hierarchical Multi–label Classification Network

We propose **Hybrid Loss for LSTM (HLSTM)** hierarchical multi–label classification network. Figure 4 illustrates the structure of HLSTM, which is based on the original LSTM structure. The benefits of LSTM are (i) gradient regulation which can prevent the gradient vanishing or explosion issue, and (ii) limited parameters regardless of the hierarchy level [16]. The recurrent flow captures global information by incorporating the global hierarchy constraint (MCLoss). Each "time step" (iteration) corresponds to a particular hierarchy level, which captures local information. Considering the above two factors, we define the *Hybrid Loss* (HL) as follows:

$$HL = (1 - b_r)LHLoss + b_r MCLoss, \tag{5}$$

where hyperparameter b_r controls the trade–off between the local loss and global loss. The reason for linearly combining the global and local loss is the complementary gradient update flow for the global recurrent flows and unrolled local outputs, as discussed in the examples in Section IV-B.

As HLSTM is a variant of the original LSTM, we adopt conventional LSTM notations. Let F^d denote the forget gate, which decides which parts of the long-term memory have less weight given the previous hidden state and the new data instance in the sequence. Note that the "new" data instance input at each hierarchy level is always the given data instance $\mathbf{x}_i$. I^d is the input gate that decides what new information should be added to the LSTM. $\boldsymbol{C}^h$ is a "conveyor belt" in which the past information directly flows to the future. Combining with the input gate, $\hat{\boldsymbol{C}}^h$ decides which value needs to be added to the "conveyor belt". In HLSTM, $\boldsymbol{C}^h$ is used for capturing global hierarchy information. The mathematical descriptions for F^d, I^d, $\hat{\boldsymbol{C}}^h$, and $\boldsymbol{C}^h$ are shown below:

$$\boldsymbol{F}^h = \sigma(\boldsymbol{W}_F(\boldsymbol{A}^{h-1} \odot \boldsymbol{x}) + \boldsymbol{b}_F), \tag{6}$$

$$I^h = \sigma(\boldsymbol{W}_I(\boldsymbol{A}^{h-1} \odot \boldsymbol{x}) + \boldsymbol{b}_I), \tag{7}$$

$$\hat{\boldsymbol{C}}^h = \mathcal{T}(\boldsymbol{W}_C(\boldsymbol{A}^{h-1} \odot \boldsymbol{x}) + \boldsymbol{b}_C), \tag{8}$$

$$\boldsymbol{C}^h = \boldsymbol{F}^h \boldsymbol{C}^{h-1} + \boldsymbol{I}^h \hat{\boldsymbol{C}}^h, \tag{9}$$

where $\mathcal{T}$ and σ denote the $tanh$ and $sigmoid$ functions, respectively. $\boldsymbol{O}^h$ denotes the output gate that decides the new hidden state (A_G^h in Figure 4). W_T^h is used as a transition weights matrix to project A_G^h to A_L^h. After projecting global features into a particular local level, the weight parameter W_L^h makes predictions for classes located in the corresponding hierarchy level. Thus, the hidden state for capturing the local hierarchy information. The mathematical descriptions for $\boldsymbol{O}^h$ and $\boldsymbol{A}^h$ are expressed as follows:

$$\boldsymbol{O}^h = \sigma(\boldsymbol{W}_O(\boldsymbol{A}^{h-1} \cdot \boldsymbol{x}) + \boldsymbol{b}_O), \tag{10}$$

$$\boldsymbol{A}^h = \boldsymbol{O}^h \mathcal{T}(\boldsymbol{C}^h). \tag{11}$$

In both the training and testing phases of HLSTM output,we define $P = MCM(\sigma(\boldsymbol{W}^{d+1}(\boldsymbol{A}_G^d + \boldsymbol{b}^{d+1}))$, where $\boldsymbol{W}^{d+1}$ is weight matrix that maps the hierarchical output of the bottom layer to the global output with total C classes. Local outputs (LMC) are not directly used in the model's prediction phase. This is because LMC requires an additional round to achieve consistency, transitioning from multiple partial local parent-child paths to a complete path.

V. Experimental Setting

A. Datasets

We evaluate HLSTM on 20 benchmark datasets [15], [16] that are widely used to evaluate HMC methods. The datasets can be divided into three categories, including 16 functional genomics datasets [12], two medical datasets [2], one textual dataset [20], and a dataset of microalgae images [21]. Table I provides the dataset name (Dataset), the number of classes (# classes) and the number of features (# features), and the size of the training and test datasets for each benchmark. All 20 datasets are preprocessed so that missing values are replaced with the mean for numeric features, and a vector of all zeros for categorical features. Moreover, data are Z–normalized to eliminate the influence of the magnitude difference among features on the learning process.

B. Baselines

HMC–LMLP [19] is the first work that uses neural network (NN) in HMC tasks. HMC–LMLP consists of multiple Multi-Layer Perceptrons (MLP), and each MLP represents a hierarchical level. **CHMCNN** [15] uses a global approach for HMC tasks based on the feed–forward NN without any post–processing step for hierarchy constraint. **HMCN–R** [16] uses a hybrid approach based on the LSTM architecture, and applies additional procedures to address hierarchy violation. **HMCN–F** [16] is similar to HMCN–R, but based on the feed–forward NN architecture. **Clus–Ens** [22] is a decision tree–based approach, where each node located in the tree corresponds to a classifier that contains all the training examples belonging to its parent's node. **HMCN–CHMC** is based on HMCN–R [16] with the difference that the global loss function is changed to MCLoss [15]. This baseline is chosen to explore the effect of the proposed local loss function (i.e., LHLoss). **CHMC–LSTM** is based on CHMCNN but changes the architecture from feed–forward NN to LSTM. This baseline is chosen to explore the effect of proposed NN architecture.

C. Evaluation Metrics

The outputs of the baselines and HLSTM are probability values for each class. Therefore, a threshold is required to make a final prediction that indicates whether the input data instance belongs to a particular class. To avoid using a predefined threshold, we employ precision–recall curves (i.e., PR–curves) to compare the different approaches, and report the area under the average precision and recall curve, denoted as $AU(\overline{PRC})$, which is accepted as the standard evaluation metric for HMC models [10], [14], [15]. Let TP_i, FP_i, and FN_i denote the number of true positive predictions, false positive predictions, and false negative predictions for class i, respectively. The precision and recall scores are defined as

$$\bar{P_r} = (\sum_{i=0}^{c-1} TP_i)/(\sum_{i=0}^{c-1} TP_i + \sum_{i=0}^{c-1} FP_i) \tag{12}$$

and

$$barR_e = (\sum_{i=0}^{c-1} TP_i)/(\sum_{i=0}^{c-1} TP_i + \sum_{i=0}^{c-1} FN_i), \tag{13}$$

respectively. For a specific prediction threshold, a precision and recall score can be obtained, which corresponds to a point in the $\bar{P}_r - \bar{R}_e$ plane. By varying the prediction threshold, a precision–recall curve can be plotted, under which the area is computed as $AU(\overline{PRC})$. A high $AU(\overline{PRC})$ indicates both high $\bar{P}_r$ and $\bar{R}_e$.

D. Setup

Experiments for CHMC–LSTM and HLSTM are performed with Pytorch. For all neural network–based baselines and HLSTM, we minimize the loss function via mini–batch gradient descent, and batch size of 4 as suggested by [16]. For LSTM–based baselines and HLSTM, the time step is equal to the depth of the hierarchy structure. The hyperparameter b_r is chosen as 0.5, which is discussed in Section VI-C. The global weight (e.g., parameters inside LSTM cell) is uniformly initialized, and the local weight (W_L^d) is initiated with small weights which is detailed discussed in the supplementary material. Besides, the Adam optimizer is utilized to learn the parameters with a learning rate of 1×10^{-4}, and weight decay of 1×10^{-5}. We report results using the model at the last epoch. All experiments were conducted on NVIDIA GeForce RTX–3080 Ti GPU.

TABLE I
SUMMARY OF 20 REAL-WORLD DATASETS.

Dataset	# classes	# features	D_{train}	D_{test}	taxonomy	label cardinality	label density	depth	average classes per level
CELLCYCLE FUN	499	77	1625	1281	Tree	8.72	0.019	6	83.16
DERISI FUN	499	63	1605	1272	Tree	8.76	0.019	6	83.16
EISEN FUN	461	79	1055	835	Tree	9.20	0.022	6	76.8
EXPR FUN	499	551	1636	1288	Tree	8.69	0.019	6	83.16
GASCH1 FUN	499	173	1631	1281	Tree	8.70	0.019	6	83.16
GASH2 FUN	499	52	1636	1288	Tree	8.69	0.019	6	83.16
SEQ FUN	499	478	1692	1332	Tree	8.53	0.019	6	83.16
SPO FUN	499	80	1597	1263	Tree	8.74	0.019	6	83.16
CELLCYCLE GO	4122	77	1625	1281	DAG	34.67	0.008	6	687
DERISI GO	4116	63	1605	1272	DAG	34.76	0.008	6	694
EISEN GO	3570	79	1055	835	DAG	37.92	0.01	6	595
EXPR GO	4128	551	1636	1288	DAG	34.61	0.008	6	688
GASCH1 GO	4122	173	1631	1281	DAG	34.63	0.008	6	687
GASCH2 GO	4128	52	1636	1288	DAG	34.61	0.008	6	688
SEQ GO	4130	478	1692	1332	DAG	34.40	0.008	6	688
SPO GO	4166	80	1597	1263	DAG	34.73	0.008	6	694
DIATOMS	398	371	1085	1054	Tree	2	0.007	2	199
ENRON	56	1000	692	660	Tree	5	0.10	5	11.2
IMCLEF07A	96	80	7000	1006	Tree	3	0.04	3	32
IMCLEF07D	46	80	7000	1006	Tree	3	0.08	3	15.3

VI. EXPERIMENTAL RESULTS

A. Comparison with the State–of–the–art

This section compares HLSTM with the state–of–the–art (see Section V-B). Table II shows the comparison of HLSTM with the state–of–the–art with respect to $AU(\overline{PRC})$. The results for CHMCNN come from [15], and the results for HMCN–R, HMCN–F and HMC–LMLP come from [16].

The average rank of HLSTM is 1.50, which means it outperforms baselines. First, we compare HMCN–R and HLSTM. Among all 20 datasets, HLSTM outperforms HMCN–R in all cases except for EISEN FUN, although the difference with HLSTM is small (i.e., 0.09). This illustrates the efficiency of incorporating the hierarchy constraint into the hybrid loss. Specifically, outputting prediction results that align with the hierarchy constraint contributes to the improvement of the overall performance. Next, we compare CHMCNN and HLSTM. CHMCNN performs the best for particular datasets, such as Eisen Fun and IMCEF07A/07D. However, its difference with HLSTM is relatively small. Besides, HLSTM shows its superiority on GO datasets with a more complex hierarchy (i.e., # classes) structure, as shown in Table II. This illustrates the benefit of jointly considering the proposed local loss (i.e., LHLoss) with global loss into a hybrid framework.

B. Statistical Significance of Results

To compare the statistical difference between HLSTM and the state-of-the-art, we perform the Friedman test. Indeed, the models are statistically different with a p-value of 2.76×10^{-12}. The Nemenyi test is carried out to further investigate the statistical difference. To visualize the statistical difference, we employ the critical difference diagram [23], as shown in Figure 5. Specifically, the horizontal line represents the classifiers' average rank on the 20 real–world datasets. The classifiers that are not statistically different are connected by a bold horizontal line. The threshold for determining the statistical difference is $CD = 2.01$. Evidently, the average rank of HLSTM is higher than the other models with statistical significance.

C. Ablation Study

To investigate the importance of the proposed LHLoss, we separately compare CHMC–LSTM with HLSTM. The only difference between CHMC–LSTM and HLSTM lies in the loss function. HLSTM employs the proposed Hybrid Loss, while CHMC–LSTM uses MCLoss without incorporating the proposed local Loss (LHLoss). We calculate the lift percentage for HLSTM as compared to CHMC–LSTM as:

$$\frac{AU(\overline{PRC})(HLSTM) - AU(\overline{PRC})(CHMC-LSTM)}{AU(\overline{PRC})(CHMC-LSTM)} \times 100\%. \quad (14)$$

The results are shown in Figure 6. Positive lift is observed for all datasets with the exception of three datasets in FUN (i.e., DERISI, EXPR, and SPO). This means that HLSTM outperforms CHMC–LSTM, demonstrating the effectiveness and importance of learning from a local perspective. In the FUN datasets, learning becomes more challenging with lower performance for all baselines and HLSTM. However, incorporating the proposed local loss contributes to improved performance in the majority of datasets in FUN.

To further investigate the influence of the local component, we focus on parameter b_r in Equation (5) which controls the tradeoff between the local and global losses. We assign b_r with different values in $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. $b_r < 0.5$ means HL is biased towards local LHLoss, and global MCLoss otherwise. Figure 7 shows the results for different b_r on Cellcycle Fun, Cellcycle Go, and Imclef07A datasets, respectively. Both too small (i.e. $b_r < 0.3$) and too big b_r (i.e., $b_r > 0.7$) values are not good options. Small b_r values are likely to cause overfitting because HL is biased more on the local loss, as shown by the

TABLE II
COMPARISON OF $AU(\overline{PCR})$ OF HLSTM AND STATE–OF–THE-ARTS ON 20 REAL-WORLD DATASETS. THE RESULTS OF HMC–LMLP FOR DIATOMS, ENRONM, IMCLEF07A AND IMCLEF07D ARE NOT BEEN PROVIDED IN [15].

Dataset	CHMCNN	HMCN–R	HMCN–F	HMC–LMLP	CLUS–ENS	HMCN–CHMC	HLSTM
CELLCYCLE FUN	0.255	0.247	0.252	0.207	0.227	0.263	**0.267**
DERISI FUN	0.195	0.189	0.193	0.182	0.187	**0.253**	0.199
EISEN FUN	**0.306**	0.298	0.298	0.245	0.286	0.224	0.289
EXPR FUN	**0.302**	0.300	0.301	0.242	0.271	0.278	0.288
GASCH1 FUN	0.286	0.283	0.284	0.235	0.267	0.304	**0.349**
GASCH2 FUN	0.258	0.249	0.254	0.211	0.231	0.281	**0.328**
SEQ FUN	0.292	0.290	0.291	0.236	0.284	0.301	**0.311**
SPO FUN	0.215	0.210	0.211	0.186	0.211	0.246	**0.272**
CELLCYCLE GO	0.413	0.395	0.400	0.361	0.387	0.420	**0.452**
DERISI GO	0.370	0.368	0.369	0.343	0.361	0.395	**0.454**
EISEN GO	0.455	0.435	0.440	0.406	0.433	0.412	**0.487**
EXPR GO	0.447	0.450	0.452	0.373	0.422	0.379	**0.457**
GASCH1 GO	0.436	0.416	0.428	0.380	0.415	0.428	**0.479**
GASCH2 GO	0.414	0.463	0.465	0.371	0.395	0.437	**0.474**
SEQ GO	0.446	0.443	0.447	0.370	0.438	0.473	**0.491**
SPO GO	0.382	0.375	0.376	0.342	0.371	0.410	**0.450**
DIATOMS	0.758	0.514	0.530	–	0.501	0.719	**0.776**
ENRON	0.756	0.710	0.724	–	0.696	0.764	**0.781**
IMCLEF07A	**0.956**	0.904	0.950	–	0.803	0.681	0.932
IMCLEF07D	**0.927**	0.897	0.920	–	0.881	0.872	0.926
Average Rank	2.65	4.47	3.37	6.95	5.67	3.37	**1.50**

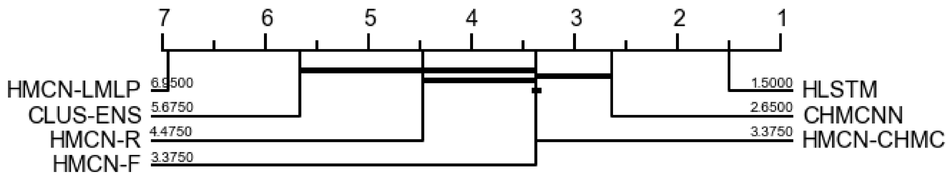


Fig. 5. The improved ratio of $AU(\overline{PRC})$ of HLSTM compared with CHMC–LSTM.

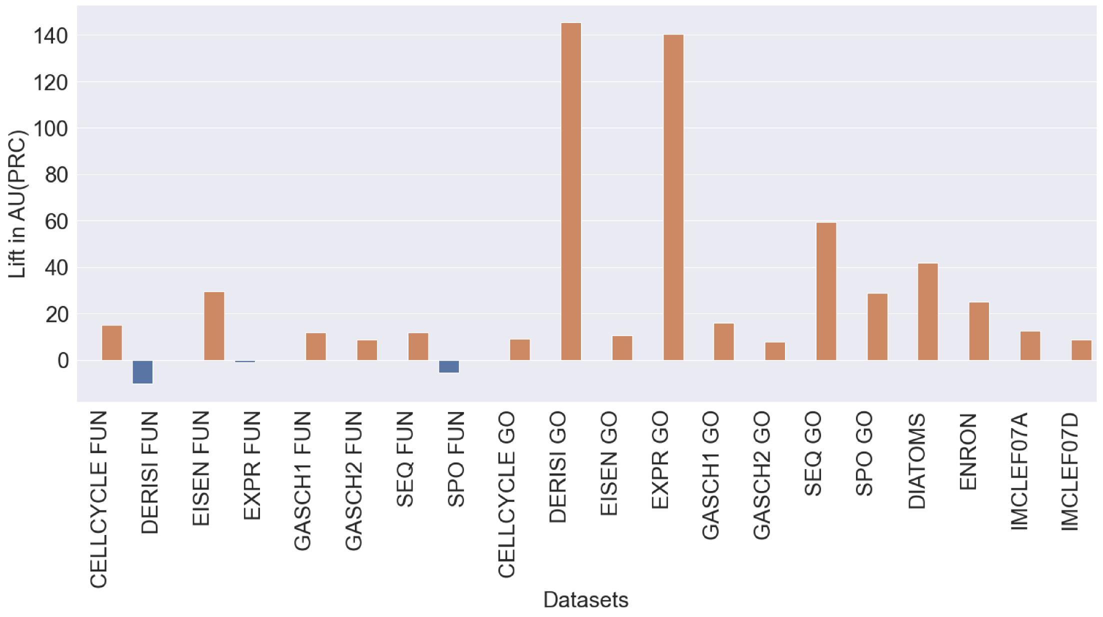


Fig. 6. Lift precentage in $AU(\overline{PRC})$ across all evaluation datasets for HLSTM as compared to CHMC–LSTM. Positive lift indicates increase in $AU(\overline{PRC})$.

blue curve in Figure 7(a) (i.e., $b_r = 0.1$). On the other hand, large b_r values limit the overall performance as HL will omit the local information, as illustrated in Figures 7(b) and 7(c). Therefore, we take $b_r = 0.5$ as the best option based on the results shown in Figure 7.

VII. Conclusions and Future Work

We proposed a novel approach for hierarchical multi-label classification tasks. Specifically, we first proposed a local loss function, which captures the local information and imposes the hierarchy constraint at the same time. Then, we proposed a hybrid loss that simultaneously optimizes for local and global constraints. We use several simple but informative examples to illustrate the rationality of this loss. Our experiments on 20 real–world datasets demonstrated the superiority of the proposed approach. In future work, we wish to extend the proposed solution to an online learning setting, in which the model can be updated as new data instances become available.

Appendix A
Discussion on MCLoss and LHLoss

We present more examples to present the difference between MCLoss and LHLoss. From the local perspective, there are only three possible scenarios for the combination of labels of parent and child. Specifically, assuming x_i only has two classes, A and B, and that $A \rightarrow B$, then, only three scenarios are possible as follows: $(y^A = 0, y^B = 0)$, $(y^A = 1, y^B = 1)$, and $(y^A = 0, y^B = 1)$. Note that $(y^A = 1, y^B = 0)$ is invalid in hierarchical classification task.

We have already discussed the example with respect to $(y^A = 1, y^B = 1)$ in the main text. We therefore consider the rest two scenarios here.

Example 1: $l(A) = 0.3, y^A = 0; l(B) = 0.1, y^B = 0$. The hierarchy violation exists in this example because $l(B) < l(A)$. The MCLoss is calculated as follows:

$$\begin{aligned}MCLoss &= MCLoss(B) + MCLoss(A)\\ &= -ln(1 - MCM(B)) - ln(1 - MCM(A))\\ &= -ln(1 - l(A)) - ln(1 - l(A)),\\ \frac{\partial MCLoss}{\partial l(A)} &= -\frac{2}{l(A) - 1} \approx 2.85,\\ \frac{\partial MCLoss}{\partial l(B)} &= 0.\end{aligned}$$

In contrast, LHLoss is calculated as:

$$\begin{aligned}LHLoss &= LHLoss(B) + LHLoss(A)\\ &= -ln(1 - LMC(B)) - ln(1 - LMC(A))\\ &= -ln(1 - l(B)) - ln(1 - l(B)),\\ \frac{\partial LHLoss}{\partial l(A)} &= 0,\\ \frac{\partial MCLoss}{\partial l(B)} &= -\frac{1}{l(B) - 1} - \frac{1}{l(B) - 1} \approx 2.22.\end{aligned}$$

Example 2: $l(A) = 0.3, y^A = 0; l(B) = 0.1, y^B = 1$. MCLoss is calculated as follows:

$$\begin{aligned}MCLoss &= MCLoss(B) + MCLoss(A)\\ &= -ln(max(y^A l(A), l(B))) - ln(1 - MCM(A))\\ &= -ln(l(B)) - ln(1 - l(A)),\\ \frac{\partial MCLoss}{\partial l(A)} &= -\frac{1}{l(A) - 1} \approx 1.4,\\ \frac{\partial MCLoss}{\partial l(B)} &= -\frac{1}{h(B)} = -10.\end{aligned}$$

The LHLoss is calculated as:

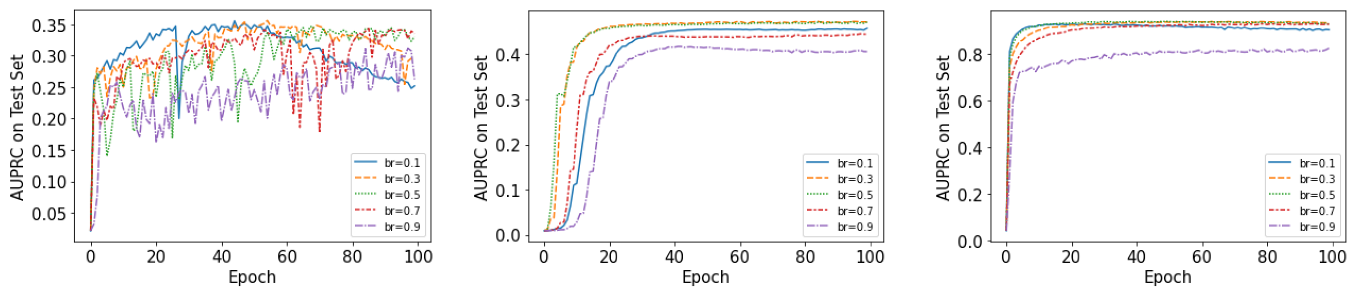
$$\begin{aligned}LHLoss &= LHLoss(B) + LHLoss(A)\\ &= -ln(LMC(B)) - ln(1 - LMC(A))\\ &= -ln(l(B)) - ln(1 - l(B)),\\ \frac{\partial LHLoss}{\partial l(A)} &= 0,\\ \frac{\partial MCLoss}{\partial l(B)} &= -\frac{1}{l(B) - 1} - \frac{1}{l(B)} \approx -8.89.\end{aligned}$$

The above Examples show the MCLoss and LHLoss are complementary to each other. In summary, combining MCLoss and LHLoss seems to address all possible scenarios well.

Appendix B
Impact of Local Weight Initialization

To investigate the influence of initialization strategy on the model's performance, we have trained our model with six distinct initialization methods, including zeros initialization, ones initialization, constant initialization, normal initialization (rand), xavier initialization (Xavier) [24], and He normal initialization (He) [25]. For completeness, we briefly introduce these initialization methods. The zeros and ones initialization are straightforward. The weights are initialized as all zeros or one, accordingly. The constant initialization adopts a similar idea, but all weights are set to a constant α. In our experiments, we consider $\alpha \in \{0.1, 0.2, 0.4, 0.6, 0.8\}$. The normal initialization method initializes the weight parameters with random variables drawn from Gaussian distribution with zero mean and unit variance. The Xavier normalization initializes the parameters with random variables from a uniform distribution in $[-\sqrt{\frac{6}{n_{in}+n_{out}}}, \sqrt{\frac{6}{n_{in}+n_{out}}}]$, where n_{in} (n_{out}) is the number of inputs (outputs) for a neuron. The He normal initialization initializes the parameters as random variables drawn from a Gaussian distribution with zero mean and $\sqrt{1/n_{in}}$ standard deviation. We use these methods to initialize the local weights (i.e., W_L^h in Figure 4).

Figure 8(a) shows the experiment results for the constant initialization method. Initializing the local weights with a small constant value (i.e., $LW = 0.1$) yields the best performance. This leads us to the question if a smaller constant would further improve the performance. However, as Figure 8(b) illustrates, zeros initialization causes an overfitting problem, as the $AU(\overline{PRC})$ drops for large training epochs. Similarly, random, He and Xavier initializations lead to an overfitting



(a) (b) (c)

Fig. 7. HLSTM's $AU(\overline{PRC})$ for different b_r values on the (a) Cellcycle FUN, (b) Cellcycle GO, and (c) IMCLEF07A datasets.

problem for training epochs greater than 60, as shown in Figure 8(c). In summary, initializing local weights with a constant value of 0.1 yields the best performance.

Appendix C
$AU(\overline{PRC})$ of HLSTM and CHMC–LSTM

We present the $AU(\overline{PRC})$ of HLSTM and CHMC–LSTM in Tables III, IV, and V.

Appendix D
Discussion on the Learning Speed of HLSTM

Here, we compare HLSTM with the state–of–the–art (see Section VI-A) with respect to learning speed. Considering the influence of different ways of local weight initialization, we present two versions of HLSTM here, HLSTM$_{rw}$ (i.e., Random initialization) and HLSTM$_{sw}$ (i.e., Constant initialization with $LW = 0.1$). For both HLSTM$_{rw}$ and HLSTM$_{sw}$, we set $b_r = 0.5$. Figure 9 shows the learning speech of HLSTM and state–of–the–art methods within limited learning epoches on Eisen Fun dataset. It's obvious that the learning speed for HLSTM (i.e., HLSTM$_{rw}$ and HLSTM$_{sw}$) is considerably faster than CHMCNN.

References

[1] A. Freitas and A. Carvalho, "A tutorial on hierarchical classification with applications in bioinformatics," in *Research and trends in data mining technologies and applications*. IGI Global, 2007, pp. 175–208.

[2] I. Dimitrovski, D. Kocev, S. Loskovska, and S. Džeroski, "Hierarchical annotation of medical images," *Pattern Recognition*, vol. 44, no. 10-11, pp. 2436–2449, 2011.

[3] C. N. Silla and A. A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery*, vol. 22, no. 1-2, pp. 31–72, 2011.

[4] F. Wu, J. Zhang, and V. Honavar, "Learning classifiers using hierarchically structured class taxonomies," in *International symposium on abstraction, reformulation, and approximation*. Springer, 2005, pp. 313–320.

[5] N. Cesa-Bianchi and G. Valentini, "Hierarchical cost-sensitive algorithms for genome-wide gene function prediction," in *Machine Learning in Systems Biology*, 2009, pp. 14–29.

[6] A. Sokolov and A. Ben-Hur, "Hierarchical classification of gene ontology terms using the gostruct method," *Journal of bioinformatics and computational biology*, vol. 8, no. 02, pp. 357–376, 2010.

[7] L. Cai and T. Hofmann, "Hierarchical document categorization with support vector machines," in *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, 2004, pp. 78–87.

[8] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni, "Incremental algorithms for hierarchical classification," *Journal of Machine Learning Research*, vol. 7, no. Jan, pp. 31–54, 2006.

[9] K. Kowsari, D. E. Brown, M. Heidarysafa, K. J. Meimandi, M. S. Gerber, and L. E. Barnes, "Hdltex: Hierarchical deep learning for text classification," in *2017 16th IEEE international conference on machine learning and applications (ICMLA)*. IEEE, 2017, pp. 364–371.

[10] W. Bi and J. T. Kwok, "Multilabel classification on tree-and dag-structured hierarchies," in *ICML*, 2011.

[11] C. Chelmis and W. Qi, "Hierarchical multiclass adaboost," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021, pp. 5063–5070.

[12] A. Clare, "Machine learning and data mining for yeast functional genomics," Ph.D. dissertation, Citeseer, 2003.

[13] Z. Ren, M.-H. Peetz, S. Liang, W. Van Dolen, and M. De Rijke, "Hierarchical multi-label classification of social text streams," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, 2014, pp. 213–222.

[14] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel, "Decision trees for hierarchical multi-label classification," *Machine learning*, vol. 73, no. 2, pp. 185–214, 2008.

[15] E. Giunchiglia and T. Lukasiewicz, "Coherent hierarchical multi-label classification networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9662–9673, 2020.

[16] J. Wehrmann, R. Cerri, and R. Barros, "Hierarchical multi-label classification networks," in *International conference on machine learning*. PMLR, 2018, pp. 5075–5084.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] R. Cerri, R. C. Barros, and A. C. De Carvalho, "Hierarchical multi-label classification using local neural networks," *Journal of Computer and System Sciences*, vol. 80, no. 1, pp. 39–56, 2014.

[19] R. Cerri, R. C. Barros, A. C. PLF de Carvalho, and Y. Jin, "Reduction strategies for hierarchical multi-label classification in protein function prediction," *BMC bioinformatics*, vol. 17, no. 1, pp. 1–24, 2016.

[20] B. Klimt and Y. Yang, "The enron corpus: A new dataset for email classification research," in *European conference on machine learning*. Springer, 2004, pp. 217–226.

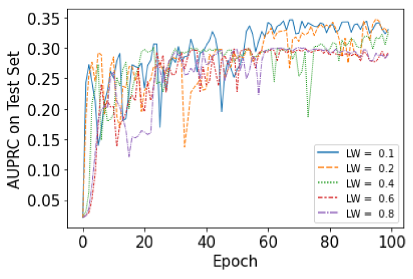
[21] I. Dimitrovski, D. Kocev, S. Loskovska, and S. Džeroski, "Hierarchical classification of diatom images using ensembles of predictive clustering trees," *Ecological Informatics*, vol. 7, no. 1, pp. 19–29, 2012.

[22] L. Schietgat, C. Vens, J. Struyf, H. Blockeel, D. Kocev, and S. Džeroski, "Predicting gene function using hierarchical multi-label decision tree ensembles," *BMC bioinformatics*, vol. 11, no. 1, pp. 1–14, 2010.

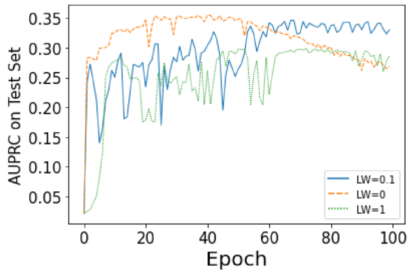
[23] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine learning research*, vol. 7, pp. 1–30, 2006.

[24] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

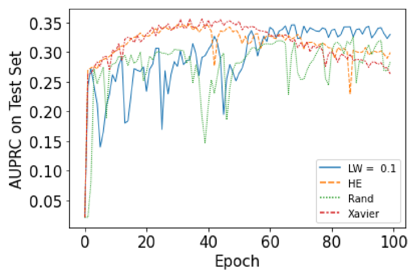
[25] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.



(a)



(b)



(c)

Fig. 8. Study of local weight initialization on HLSTM's $AU(\overline{PRC})$ on the Cellcycle FUN dataset. (a) constant local weight (LW) initialization, (b) ones and zeros LW initialization, and (c) random LW initialization. In all cases, x-axis is training epochs.

TABLE III
COMPARISON OF $AU(\overline{PCR})$ OF HLSTM AND CHMC–LSTM ON FUN.

Dataset / Model	CELLCYCLE FUN	DERISI FUN	EISEN FUN	EXPR FUN	GASCH1 FUN	GASCH2 FUN	SEQ FUN	SPO FUN
HLSTM	**0.267**	0.199	**0.289**	0.288	**0.349**	**0.328**	**0.311**	0.272
CHMC–LSTM	0.232	**0.222**	0.223	**0.291**	0.312	0.302	0.278	**0.288**

TABLE IV
COMPARISON OF $AU(\overline{PCR})$ OF HLSTM AND CHMC–LSTM ON GO.

Dataset / Model	CELLCYCLE GO	DERISI Go	EISEN GO	EXPR GO	GASCH1 GO	GASCH2 GO	SEQ GO	SPO GO
HLSTM	**0.452**	**0.454**	**0.487**	**0.457**	**0.479**	**0.474**	**0.491**	**0.450**
CHMC–LSTM	0.433	0.185	0.440	0.190	0.413	0.440	0.308	0.349

TABLE V
COMPARISON OF $AU(\overline{PCR})$ OF HLSTM AND CHMC–LSTM ON OTHER DATASETS.

Dataset / Model	DIATOMS	ENRON	IMCLEF07A	IMCLEF07D
HLSTM	**0.776**	**0.781**	**0.932**	**0.926**
CHMC–LSTM	0.547	0.624	0.828	0.851

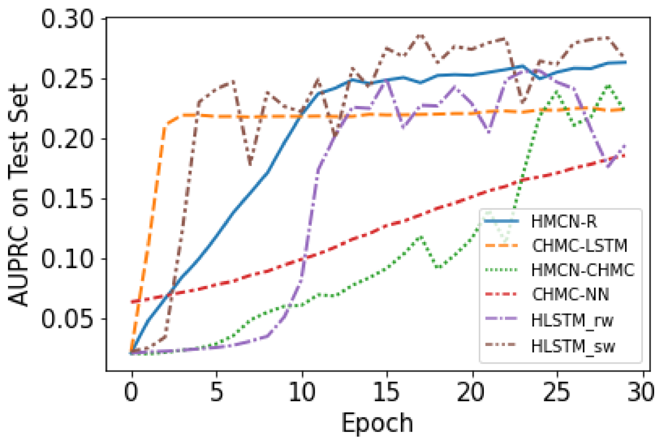


Fig. 9. Learning speed comparison of HLSTM with state–of–the–art methods for Eisen Fun dataset.