Surj: Ontological Learning for Fast, Accurate, and Robust Hierarchical Multi-label Classification

Sean T. Yang
tyyang38@uw.edu
University of Washington
Seattle, WA, USA

Bill Howe
billhowe@uw.edu
University of Washington
Seattle, WA, USA

ABSTRACT

We consider multi-label classification in the context of complex hierarchical relationships organized into an ontology. These situations are ubiquitous in learning problems on the web and in science, where rich domain models are developed but labeled data is rare. Most existing solutions model the problem as a sequence of simpler problems: one classifier for each level in the hierarchy, or one classifier for each label. These approaches require more training data, which is often unavailable in practice: as the ontology grows in size and complexity, it becomes unlikely to find training examples for all expected combinations. In this paper, we learn offline representations of the ontology using a graph autoencoder and separately learn to classify input records, reducing dependence on training data: Since the relationships between labels are encoded independently of training data, the model can make predictions even for underrepresented labels, naturally generalize to DAG-structured ontologies, remain robust to low-data regimes, and, with minor offline retraining, tolerate evolving ontologies. We show empirically that our label predictions respect the hierarchy (predicting a descendant implies predicting its ancestors) and propose a method of evaluating hierarchy violations that properly ignores irrelevant violations. Our main result is that our model outperforms all state-of-the-art models on 17 of 20 datasets across multiple domains by a significant margin, even with limited training data.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Theory of computation** → **Graph algorithms analysis**.

KEYWORDS

Hierarchical Multi-label classification, Graph Learning, Ontology Learning

ACM Reference Format:
Sean T. Yang and Bill Howe. 2022. Surj: Ontological Learning for Fast, Accurate, and Robust Hierarchical Multi-label Classification. In *Companion Proceedings of the Web Conference 2022 (WWW '22 Companion), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3487553.3524723

1 INTRODUCTION

Hierarchy- and graph-structured domains are becoming ubiquitous for learning tasks on and off the web. In high-expertise domains, human attention tends to be invested in designing and curating application-agnostic ontologies rather than on hand-labeling application-specific datasets. WordNet, first introduced by Miller et al. [34] in 1998 and since adopted in over 200 languages and used in tens of thousands of papers, is characterized by rich thesaurus relationships between terms. ImageNet [13], designed as a WordNet analoogue for images, contains over 3 million images labeled with 60,942 terms organized into a hierarchy (actually a DAG, but represented as a hierarchy by repeating nodes). For learning tasks on the web, ontological relationships between labels have been used to improve document classification [36, 38], study zero-shot learning [21], and improve recommendation systems [17, 27, 39, 53].

Despite the opportunity for supervision, relationships among labels are often ignored. For example, the ImageNet dataset led to AlexNet [28], ResNet [23], and VGG [42], but all three models ignore the hierarchical structure of the labels. In response, an emerging community is studying hierarchical multi-label classification, aiming to use the hierarchy to help supervise multi-label learning.

Current hierarchical multi-label classification models, however, tend to emphasize particular domains and are computationally expensive to train. Wehrmann et al. [48] proposed HMCN, a cascade neural network that simultaneously optimizes local hierarchical label relationships and the global hierarchy while penalizing hierarchical violations. However, the amount of the parameter of HMCN-F grows with the number of hierarchical levels. It gets computational expensive when the hierarchies are large. Xu et al. [50] introduced a hierarchical classification model which represents the correlation among labels with the label distribution and learns a mapping function from the instance to the label distribution, but the model is only tested for single-label problem. HyperIM [10] learns label-aware document representations and model the word and label hierarchies in hyperbolic space, but the model is only suitable for the text domain.

Existing approaches to hierarchical multi-label classification also tend to assume the availability of data to provide balanced coverage of the ontology, a condition that becomes increasingly unlikely as the ontology grows in size, and becomes impossible when the ontology is bigger than the training data or when the ontology changes after training data has been collected. This latter situation is especially insidious: ontologies undergo constant revision in practice, meaning that training datasets face continual obsolescence.

While several methods aim to prevent hierarchy violations (recommending a child label without also recommending a parent label), there has been no principled model-agnostic evaluations of hierarchy violations. Despite the emphasis on preventing hierarchy violations by Wehrmann et al. [48] and Giunchiglia et al[22], no direct evaluation of hierarchy violations was conducted.

In this paper, we propose an algorithm for hierarchical multi-label classification that separates the problem into learning reusable embeddings of the ontology itself, then training a classifier using these embeddings. Since the structure of the ontology is encoded in the learned representation, trained classifiers can predict a correct label even with no examples of that label available, which in turn allows a trained classifier to be more robust when the ontology changes. We use a graph autoencoder to learn label representations from the structure of the ontology, then use a simple model to learn the function mapping the input space into the label embedding space using binary cross entropy loss. We call this framework Surj .

We show that Surj outperforms current state-of-the-art algorithms on 20 benchmark datasets, usually by a significant margin. We also propose one metric to quantify hierarchy violations, Global Hierarchy Violation. The metric is applied to Surj and it reveals that there is no hierarchy violation from Surj in all 20 datasets. Surj is also trained up to 40 times faster than the current state-of-the-art C-HMCNN. We demonstrate the effectiveness of the ontology learning with an ablation study. We also show that Surj is robust to lower data size and remain superior even when only half of the training data is provided.

We make the following contributions:

- We propose a framework Surj to perform hierarchical multi-label classification with ontology learning. The framework learns a representation for the label ontology and then uses the representation as the labels for multi-label classification learning.
- We introduce Global Hierarchy Violation to comprehensively measure whether predicted results from a HMC model violate hierarchical constraint.
- We show that the proposed method achieves the-state-of-the-art results in hierarchical multi-label classification learning in 17 out of 20 benchmark datasets and that it is hierarchy-violation-free and computation efficient compared to competitive methods.
- We show a number of auxiliary studies to show the individual effects of the graph encoding, performance in low-data regimes, tolerance for changing ontologies, and a significantly faster training time than competitive methods.

2 RELATED WORK

Hierarchical multi-label classification has been a long standing problem due to the ubiquity of ontologies. Ontologies are used to organize knowledge in wide variety of domains on the web[43, 44], in urban settings [1, 18], finance[45, 46], oceanography[39], and art[12]. Some models are tailored to Natural Language Processing for text multi-label classification [9, 10, 24, 31]. While these models present novel networks, they are limited to NLP applications and not applicable in our evaluation. We review studies for general hierarchical multi-label classification problem.

Hierarchical Multi-label Classification. Most HMC algorithms can be categorized into global and local approaches. Global approaches are designed to handle the entire hierarchy. Vens et al. proposed Clus-HMC [47], based on the concept of Predictive Clustering Trees and involving a decision tree learner to map the entire hierarchy. Schietgat et al. extended the idea from Clus-HMC in Clus-Ens [40], adopting a bagging strategy to create decision tree ensembles. MHC-CNN [4] consists of a Competitive Neural Network where each neuron represents a node in the hierarchy and the whole network is a clone of the hierarchy. Masera et al. [33] proposes AWX (Adjacency Wrapping Matrix), which incorporates the hierarchy into their model architecture and the learned knowledge propagates through each layer. C-HMCNN [22] has a hierarchy-coherent layer to produce coherent predictions by construction.

Local approaches break down the problem into smaller classification tasks, often by level [5–7, 30, 52] or by node[3, 8, 19, 29, 50]. Cerri et al. [5–7] proposed HMC-LMLP, a approach based on a chain of Multi-Layer Perceptrons (MLPs) with a single layer represents a level in the hierarchy. The input of a given MLP is the output of the previous MLP. Bi et al.[3] utilize Kernel Dependency Estimation (KDE) to transforms the number of labels in a hierarchy into a workable number of single-label learning problems. Condensing Sort and Select Algorithm (CSSA) is employed to find an optimal approximation subtree to preserve tree structure and they use ridge regression in the learning step. Wehrmann et al. [48] presented HMCN-R and HMCN-F, which are optimized by a global and a local loss. HMCN-F is a feed-forward network and HMCN-R is a recurrent architecture. While HMCN-F produces better overall results but the network parameters increase significantly as the hierarchy grows. While there has been significant progress on this problem, our approach of directly embedding the ontology using graph neural networks has not been considered, and as we will show, offers significant improvements. We explore how graph neural networks can contribute to hierarchical multi-label classification and review graph representation methods in the next subsection.

Deep Learning on Graphs. Graph representation learning with graph neural network has created opportunities for applications such as node classification, link prediction, and spatial–temporal graph forecasting. Jurisch et al. utilizes graph convolution networks to solve ontology alignment problem. Schlichtkrull et al. [41] models relational data with graph convolutional networks. Zang et al. [51] apply graph embeddings for gene ontology annotations to predict protein-protein interaction. In our work, we propose to learn node embeddings to capture the structure information of a hierarchy and use these embeddings to train a multi-label classifier.

3 METHOD

In this section, we define the problem of hierarchical multi-label classification and introduce our framework.

3.1 Preliminaries

We define a label hierarchy $H = (V, E)$ as a graph with labels V and edges E, where edges are typically parent-child relationships representing specificity. Given a space of input instances X where each element is a vector of features $x_1, x_2, ..., x_N$.

Our framework consists of a learned function $g : V \to Z^D$ that embeds each vertex in the ontology into a D-dimensional representation, and another learned function $m : X \to P(R)$ to

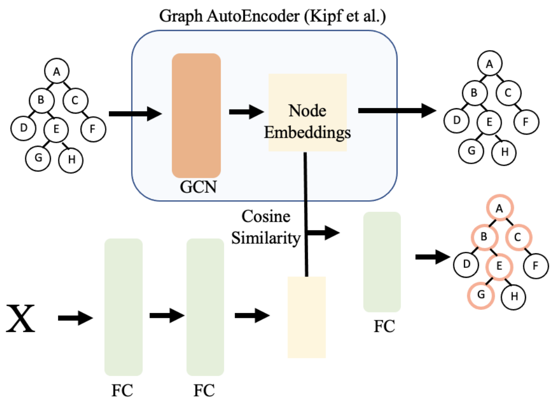

Figure 1: Illustration of our framework. We learn a representation for the label ontology using a graph autoencoder. Then, the model considers the node embeddings and maps the input instances X onto the node embedding space with cosine similarity. Finally, the model is optimized with binary cross entropy and produce probability confidence as output.

take input instances and produces a probability distribution over the learned embedding space R. The classifier is trained to map the input instances X into an embedding space R , followed by a cosine similarity layer to compute the cosine similarity between the embedding space and the ontology representation. The classifier is optimized with the binary cross entropy (BCE). The intuition behind this framework is that graph representation learning models allow us to capture the structural information from the label hierarchies: labels that are "close" in the hierarchy are "close" in the learned space. The learned embedding allows us to accommodate complex graphs, enforce parent-child relationships in predicted labels, make better use of limited training data, and tolerate ontology changes without requiring new training data.

3.2 Ontology Learning

For the label hierarchy learning, we use a graph auto-encoder [25] to learn the node embeddings. Graph auto-encoders are easy to implement and computationally efficient. We introduce an adjacency matrix A constructed from the label graph H and its degree matrix D. Node features are constructed in an $N \times D$ matrix S. The graph autoencoder includes a two-layer Graph Convolutional Network (GCN) defined as:

$$\hat{\mathrm{A}} = \sigma(\mathbf{Z}\mathbf{Z}^T), \text{ with } \mathbf{Z} = GCN(\mathrm{S}, \mathrm{A}) \tag{1}$$

The two-layer GCN is defined as:

$$\mathrm{GCN}(\mathrm{S}, \mathrm{A}) = \hat{\mathrm{A}}\mathrm{ReLU}(\hat{\mathrm{A}}\mathrm{S}\mathbf{W_0})\mathbf{W_1} \tag{2}$$

where $\mathrm{ReLU}(\cdots) = max(0, \cdots)$, $\mathbf{W_i}$ suggests weights, and $\hat{\mathrm{A}} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is the normalized adjacency matrix. Finally, $\sigma(\cdots)$ is the logistic sigmoid function.

Hierarchy nodes are frequently equipped with semantic features. For example, every node of WordNet consists of words which can be converted into word embeddings. Protein and gene possesses observed features within the hierarchies. While there is no feature vectors provided for the node in the benchmark dataset, the learned embeddings with feature vectors provided can be more representative and further improve the overall HMC performance. The graph autoencoder takes both dense feature vectors and an affinity matrix as inputs, allowing the ontology learner to learn the semantic and latent features of the nodes.

We perform an experiment to verify this premise using the ontology from one of the benchmark datasets, cellcycle(FUN).

For this framework to succeed, the learned hierarchy embedding needs to reflect the structural information. To show a simple verification of the correspondence, we take the label hierarchy from the CellCycle(FUN) dataset and compare the shortest path of all node pairs and the cosine similarity of their learned embeddings from the graph autoencoder. The results are shown in Figure 2. The x-axis is the length of the shortest path and the y-axis is the cosine distance d_{cosine}, defined as:

$$d_{cosine} = 1 - cos(\boldsymbol{P}, \boldsymbol{Q}) \tag{3}$$

where

$$cos(\boldsymbol{P}, \boldsymbol{Q}) = \frac{\boldsymbol{P} \cdot \boldsymbol{Q}}{||\boldsymbol{P}|| \cdot ||\boldsymbol{Q}||} \tag{4}$$

$\boldsymbol{P}$ and $\boldsymbol{Q}$ are two dense vectors. We can simply observe from the figure that the closer the two nodes are in the ontology, the more similar they are in the learned embedding space.

3.3 Multi-label Classification

The framework leverages the learned representation by learning to map the input instances on to the learned node representations. This is achieved by inserting a cosine similarity layer to compute the cosine similarity between the output from the fully connected layers and the node embeddings. The cosine similarity layer follows equation 4.

The cosine similarity layer is followed by another fully connected layer and a sigmoid layer for multi-label classification. The model is optimized by Binary Cross Entropy Loss:

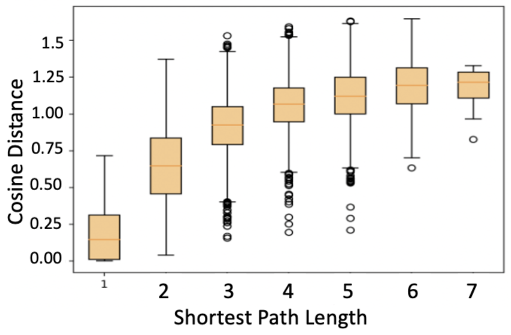

Figure 2: The relationships between the shortest path length of all node pairs and the cosine similarity of their learned embeddings. The closer the two nodes are in the tree, the more similar they are in the embedding space.

Table 1: Datasets used in the evaluation

Taxonomy	Dataset	# Classes	# Attributes	Depth	Train	Validation	Test
Tree	ENRON	56	1000	3	692	296	660
	DIATOMS	398	371	3	1085	464	1054
	IMCLEF07A	96	80	3	7000	3000	1006
	IMCLEF07D	46	80	3	2100	900	3000
FUNCAT (Tree)	CELLCYCLE	499	77	6	1628	848	1281
	DERSI	499	63	6	1608	842	1275
	EISEN	461	79	6	1058	529	837
	EXPR	499	551	6	1638	849	1291
	GASCH1	499	173	6	1634	846	1284
	GASCH2	499	52	6	1639	849	1291
	SEQ	499	478	6	1701	879	1339
	SPO	499	80	6	1600	837	1266
Gene Ontology	CELLCYCLE	4122	77	12	1625	848	1278
	DERSI	4116	63	12	1605	842	1272
	EISEN	3570	79	12	1055	528	835
	EXPR	4128	551	12	1636	849	1288
	GASCH1	4122	173	12	1631	846	1281
	GASCH2	4128	52	12	1636	849	1288
	SEQ	4130	478	12	1692	876	1332
	SPO	4116	80	12	1597	837	1263

$$L = -\frac{1}{V}\sum_{i=1}^{V} y_i \times \log(p(y_i) + (1 - y_i)) \times \log(1 - (p(y_i))) \quad (5)$$

where $y \in 0, 1$ is the label and $p(y)$ is the predicted probability of the node being true.

4 EXPERIMENTAL EVALUATION

In this section, we describe the empirical experimentation to verify the effectiveness of Surj for hierarchical multi-label classification. We evaluate Surj on 20 benchmark datasets across biological sequencing (protein function prediction), images, and text and compare our framework against six other state-of-the-art algorithms in the HMC space. While several recent HMC models are designed based on the premise of avoiding hierarchy violations, there is no metrics to quantitatively measure hierarchy violations. Global Hierarchy Violations proposed to determine the magnitude of the hierarchy violations of the predicted outputs. The implementation of the experiments are detailed in Sec. 4.4.

4.1 Datasets

We consider 20 datasets across multiple domains used in previous hierarchical multi-label classification studies [22, 48]. The datasets consist of protein function prediction [11], annotation of medical images [15, 16], or text classification [26]. The datasets are constructed as trees (MIPS functional Catalogue for protein function) or directed acyclic graphs (Gene Ontology). The statistics for the datasets are shown in 1.

As mentioned by Wehrmann et al [48], these datasets are challenging for neural networks for multiple reasons: (1) The training samples are relatively low. (2) The number of features vary significantly across the datasets (varying from 52 to 1000) (3) The hierarchies exhibit a wide range of depths and number of classes.

4.2 Competitive Methods

We compare Surj against several models that are considered the state-of-the-art for Multi-label Hierarchical Classification:

- C-HMCNN[22]: C-HMCNN leverages a constraint layer to ensure the predictions are coherent with the hierarchy constraints.
- HMCN[48]: Wehrmann et al. two neural network architectures, HMCN-F and HMCN-R, for HMC based on the concept of finding the local hierarhical class-relationships and entire class hierarchy with penalizing hierarchical violations. HMCN-F is a feed forward network designed for optimizing the hierarhical structure of the labeled data and HMCN-R is a recurrent network where the global flow shares weights throughout the hierarchy.
- HMC-LMLP[7]: HMC-LMLP is the first study to utilize neural networks for HMC problems. They associate one multilayer perception (MLP) to each hierarchical level and the MLP is only fed by the output from the previous MLP from the second level onwards.
- CLUS-HMC[47]: A global approach based on the concept of Predictive Clustering Trees (PCT) to generate a decision tree to cover the entire tree hierarchy.
- CLUS-HMC-Ens[40]: This algorithm considerably improves upon CLUS-HMC by integrating a bagging strategy for creating ensembles of Clus-HMC trees.

4.3 Evaluation Metrics

Our framework and the competitive methods generate a probability distribution as output. Thresholding is a common practice to acquire binary prediction, but the selection of the threshold value is difficult to obtain and arbitrary. Following Wehrmann et al. [48] and Giunchiglia et al. [22], we provide quantitative evaluation using the area under the average precision-recall curve ($AU(\overline{PRC})$), whose points ($\overline{Prec}$, $\overline{Rec}$) is calculated as following:

$$\overline{Prec} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FP_i} \tag{6}$$

$$\overline{Rec} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FN_i} \tag{7}$$

where TP_i, FP_i, and FN_i are the number of true positives, false positives, and false negatives for class i, respectively.

In addition to $AU(\overline{PRC})$, we propose algorithms to detect global hierarchy violations. Hierarchy violations occur when the predicted probability of a child node is higher than a parent node. This definition is threshold-independent. In real applications, thresholds are required to generate predictions, and hierarchy violations occur when predictions do not include ancestors of a predicted node.

Wehrmann et al. [48] and Giunchiglia et al. [22] emphasized that a hierarchical multi-label classification model should not have any hierarchy violations and designed their models accordingly. Wehrmann et al. [48] penalize hierarchical violation by employing a regularizer to ensure the prediction score of a node is lower than its parent nodes. Giunchiglia et al. [22] proposed a modified binary cross-entropy loss (MCLoss), which constrains the predicted probability of a child node to only be as high as its parent node. Both papers demonstrate the improvement in $AU(\overline{PRC})$ from the hierarchical structural constraints, but do not evaluate hierarchy violations. We introduce **Global Hierarchy Violation** to measure hierarchy violations.

Consider the example in Fig. 3(a). We have a subtree of 6 nodes. The letters identify the nodes and the p annotations indicate the predicted probability scores. The right branch of this subtree has no hierarchy violation because all child nodes have lower predicted outputs than their parent nodes. On the other hand, we can observe hierarchy violations on the left branch where node B, as a parent node, has a lower score than node D and node E. Node E also has higher predicted probability than node A.

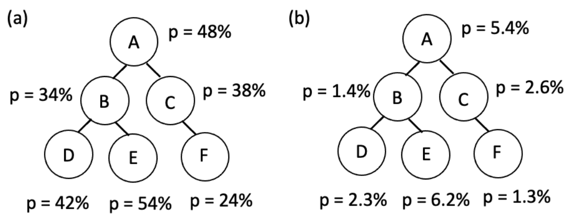

Figure 3: Demonstration of a hierarchy violation with predicted probabilities. Letters identify nodes and p annotations indicate predicted probabilities (pp). Hierarchy violations occur when the pp of a descendant node is higher than that of one of its ancestors. (a) B-D, B-E, and A-E pairs are hierarchy violations. (b) While the same hierarchy violation pairs are present, they are irrelevant due to low predicted confidence.

Global Hierarchy Violation compares all valid node pairs and computes the number of hierarchical violation occurrences. Given a label hierarchy $H = (V, E)$ as a tree or ontology with labels V and edges E. Each label v_i is associated with a predicted probability p_i. A valid node pair (v_i, v_j) is defined as the shortest path between v_i and v_j does not go through the root node. Global Hierarchy Violation is defined as the total number of occurrences of the ancestor node associated with lower probability in all valid node pairs. Ancestor node is the node that has shorter shortest path to the root node in a valid node pair. For the example in Fig. 3(a), the algorithm would consider 8 node pairs in this subtree: A-B, A-D, A-E, A-C, A-F, B-D, B-E, and C-F, and compute the number of violation occurs in these pairs. The Global Hierarchy Violation would be 3 for this scenario. However, Global Hierarchy Violation might have a blind spot. Consider the scenario in Fig. 3(b), it remains the similar predicted probability pattern as Fig. 3(a) but with much lower probability confidence. In real life applications, most set the thresholds around 50%. The violations happen in this scenario would be irrelevant. Global Hierarchy Violation is designed to detect all hierarchy violations and we encourage future HMC research to incorporate this metric in the evaluation purpose.

4.4 Implementation

To conduct a fair comparison, we adopt the code[1] provided by Giunchiiglia et al [22] for dataset pre-processing and evaluation. For pre-processing, all nominal features were converted to numeric values via one-hot encoding. The feature vectors were then normalized. All missing values were replaced by the corresponding mean. For evaluation, we remove root nodes ("root" for FUN datasets, "root", "GO0003674", "GO0005575", and "GO0008150" for GO datasets[2]). All the experiments were trained with 32 CPU cores and all reported results are the average of 10 trials. Code for all experiments will be published on github.

The hyper-parameters in our framework are the dimensions of the fully connected layers and the learning rate. These hyper-parameters are optimized with the validation set. We find that the dimension of the fully connected layers marginally impact the overall performance and the framework produces the best overall results with learning rate = 0.001. The hyper-parameters are consistent across datasets.

5 EXPERIMENTAL RESULTS

We report the evaluation results against the state-of-the-art models in this section. We first assess the overall performance of our model with $AU(\overline{PRC})$ against six other models in 20 real-life benchmark datasets (Sec.5.1). We also compute the Global Hierarchy Violation to verify that the generated predictions follow the hierarchical constraints (Sec. 5.2). Computation cost analysis is provided in Sec. 5.3. Finally, the ablation study is conducted to demonstrate the impact of the ontology learning process (Sec. 5.5).

[1]https://github.com/EGiunchiglia/C-HMCNN
[2]https://dtai.cs.kuleuven.be/clus/hmcdatasets/

Table 2: Quantitative comparison with the state-of-the-art in the hierarchical multi-label classification. The numbers reported are $AU(\overline{PRC})$. Average ranking is the average of the rankings compared to other competitive algorithms among all datasets. Higher numbers are better. Our models produce superior results in 17 of 20 real-life benchmark datasets and have 1.3 average ranking.

Dataset		Ours	C-HMCNN	HMCN-F	HMCN-R	HMC-LMLP	CLUS-HMC	CLUS-ENS
FUNCAT	CELLCYCLE	**0.269**	0.255	0.252	0.247	0.207	0.172	0.227
	DERSI	**0.231**	0.195	0.193	0.189	0.182	0.175	0.188
	EISEN	**0.392**	0.306	0.298	0.298	0.245	0.204	0.271
	EXPR	**0.382**	0.302	0.301	0.300	0.242	0.210	0.271
	GASCH1	**0.369**	0.286	0.284	0.283	0.235	0.205	0.267
	GASCH2	**0.273**	0.258	0.254	0.249	0.211	0.195	0.227
	SEQ	**0.341**	0.292	0.291	0.290	0.236	0.211	0.284
	SPO	**0.241**	0.215	0.211	0.210	0.186	0.186	0.210
GO	CELLCYCLE	**0.460**	0.413	0.400	0.395	0.361	0.357	0.387
	DERSI	**0.445**	0.370	0.369	0.368	0.343	0.355	0.363
	EISEN	**0.487**	0.455	0.440	0.435	0.406	0.380	0.433
	EXPR	**0.477**	0.447	0.452	0.450	0.373	0.368	0.418
	GASCH1	**0.481**	0.436	0.428	0.416	0.380	0.371	0.415
	GASCH2	**0.473**	0.414	0.465	0.463	0.371	0.369	0.395
	SEQ	**0.478**	0.446	0.447	0.443	0.370	0.386	0.435
	SPO	**0.439**	0.382	0.376	0.375	0.342	0.345	0.372
ENRON		0.743	**0.756**	0.724	0.710	-	0.638	0.681
DIATOMS		**0.772**	0.758	0.530	0.514	-	0.167	0.379
IMCLEF07A		0.943	**0.956**	0.950	0.904	-	0.574	0.777
IMCLEF07D		0.917	**0.927**	0.920	0.897	-	0.749	0.863
AVERAGE RANKING		1.3	2.05	2.75	3.85	6.19	6.6	4.95

5.1 Overall Performance vs. State-of-the-Art

Table 2 shows the empirical results for the current state-of-the-art models on 20 real-world benchmark HMC datasets. The results for C-HMCNN and HMC-LMLP are adopted from those published by Giunchiglia et al. [22] and results for HMCN and CLUS models are adopted from those of Wehrmann et al. [48]. For C-HMCNN, we ran their published code and verified that the results are consistent with the published results in the paper.

Our method outperforms other algorithms on 17 of 20 datasets by a significant margin. We also have the best average ranking (1.3). To demonstrate the statistical significance of the reported results, we follow previous work [22, 48] to perform Friedman test [20] and Wilcoxon Test [49]. Friedman test is a non-parametric test to compare three or more matched groups by ranks. It is used to determine if a particular factor has an effect. Wilcoxon Test calculates the difference between sets of pairs and analyze whether these differences establish statistically significant differences between the two groups. Friedman test indicates statistically significant difference with p-value of $1.06x10^{-17}$. We then apply Wilcoxon Test to our results and C-HMCNN and it concludes that there is a statistical significant difference between the performance of our model and C-HMCNN with p-value of $3.05x10^{-05}$

To better visualize the dominant performance of our model, we create a dot chart (Figure 4 to show the margins to the-state-of-the-art (sota) performance. Our model outperforms all other models in FUN and GO datasets. Among the three datasets (Enron_corr, ImCLEF07A, and ImCLEF07D) that our models do not produce the best results, we still remain competitive (within 1.5%) with the best performance.

5.2 Hierarchy Violation Analysis

We then apply Global Hierarchy Violation to our model to examine if the predictions follow the hierarchical structure. There are no occurrences of hierarchy violations in any of the 20 datasets. We also perform Global Hierarchy Violation for C-HCMNN in FUN datasets and C-HMCNN also achieves zero hierarchy violations. This analysis shows that our model achieves the superior performance with perfect hierarchy constraints.

5.3 Comparing Training Time

In this subsection, we consider training time of Surj relative to C-HMCNN[22]. C-HMCNN is assumed to be significantly faster than other competitive neural models: Unlike HMCN-F, HMCN-R, and HMC-LMLP, the training time of C-HMCNN does not depend on the size of the ontology – the main contribution is a post-processing step to avoid hierarchy violations. The training time in seconds for both models on FUN datasets is shown in Table 3. We report results on the FUN datasets because they are smaller; C-HMCNN training is exhorbitantly expensive on larger datasets. We include training time for both ontology learning and classification learning steps. The weak correlation between the training times of the two models is due to high variance in the number of epochs needed for C-HMCNN. Our model takes significantly less time to train compared to C-HMCNN, ranging from 5x to 40x less time. In practice, ontologies

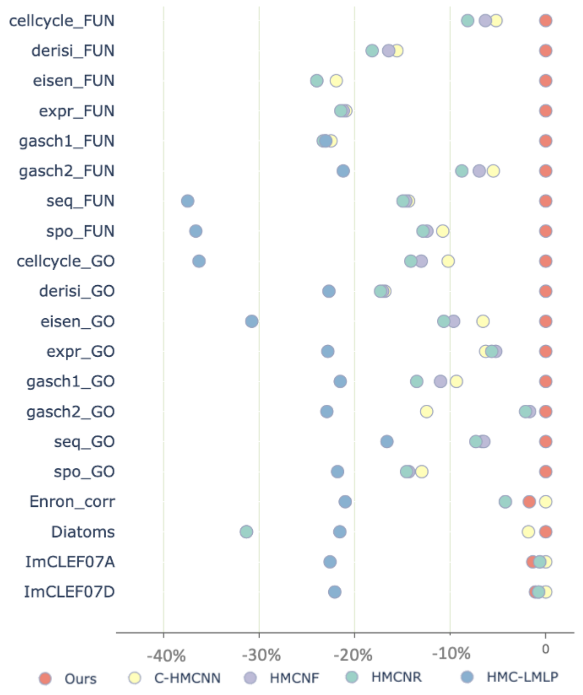

Figure 4: Margins to the state-of-the-art (sota) performance among 20 benchmark datasets. The x axis is the margin of between a model performance and the sota number. We can observe that our model (red dots) demonstrate dominance among FUN and GO datasets. Our model remains competitive (within 1.5%) even when we are not the best.

Table 3: Training Time Analysis. We measure the training time in seconds of our model and C-HMCNN on FUN datasets. We ran both models on a virtual machine with 32 cores. Our model is 5X to 40X faster to train.

	Ours			C-HMCNN
	Ontology Learning	classification Learning	total	total
CELLCYLE	0.8	45.3	46.1	1937
DERSI	0.4	44.2	44.4	1147
EISEN	0.3	24.4	24.7	1254
EXPR	0.4	55.7	56.1	694
GASCH1	0.3	56.8	57.1	748
GASCH2	0.6	58.6	59.2	2113
SEQ	0.2	61.3	61.5	320
SPO	0.3	44.3	44.6	1927

change very frequently (for example, the gene ontology database releases monthly updates [3]). Low training cost allows users to adopt and integrate new ontologies more efficiently.

[3]http://geneontology.org/docs/downloads/

Table 4: Response to an Evolving Ontology. Surj is more tolerant to ontology changes than a naive baseline (in parentheses). "% changes" indicates delta of the performance on evolving ontology.

	V1 data V1 ont.	V1 data V2 ont.	% changes
CELLCYCLE	0.269 (0.211)	0.213 (0.137)	-20.8% (-35.1%)
DERISI	0.231(0.209)	0.144(0.114)	-37.6% (-45.5%)
EISEN	0.392(0.281)	0.277(0.179)	-29.3% (-36.2%)
EXPR	0.382(0.223)	0.288(0.149)	-24.6% (-33.1%)
GASCH1	0.369(0.256)	0.271(0.175)	-26.6% (-31.6%)
GASCH2	0.273(0.208)	0.213(0.118)	-21.9% (-43.2%)
SEQ	0.341(0.218)	0.242(0.146)	-29.0% (-33.0%)
SPO	0.241(0.209)	0.207(0.129)	-14.1% (-38.2%)

5.4 Tolerance for Evolving Ontologies

We simulate an evolving ontology by removing 20% of the leaf labels from the ontology, and removing all references to the removed leaf nodes in the training data. We consider this reduced ontology the Version 1 (V1) ontology and the corresponding training data the V1 training data. We then train our model (and a competitive baseline) as usual on V1 ontology and V1 data. Then, we restore the missing nodes to the ontology to simulate Version 2 (V2), but we do not replace the labels in the training data. That is, we now have a V2 ontology (with the 20% of nodes restored) and an (out-of-date) V1 training dataset. We relearn the ontology embeddings on the V2 ontology, and retrain the model on V1 data to simulate the situation where a new ontology version has been released, but new training data has not yet been created.

The results of this experiment are show in Table 4. We trained a 2-layer fully connected network with binary cross entropy loss as our naive baseline. Surj is more tolerant of the evolving graph than the baseline. As part of our future work, we are considering more realistic simulations of ontology evolution, as well as real change histories, to develop methods of improving performance on unseen data.

5.5 Ablation Analysis

Finally, we analyze the impact of the ontology learning process. In Table 5, we show results using the baseline (3 fully connected layers trained with binary cross-entropy loss and conventional one-hot encoded vectors) against our model (same baseline trained with embeddings learned from the graph-autoencoder). The performance is measured as $AU(\overline{PRC})$. Ontology learning produces significant improvement, accounting for the majority of the difference between Surj and its closest competitors.

5.6 Varying Data Size

Ontologies often exist in high-value applications where training data is difficult or expensive to acquire. It is important for HMC models to capture meaningful signal with minimum data provided. While the data sizes for the benchmarks are already small, we challenge Surj with extreme cases to evaluate its robustness to

Table 5: Ablation Analysis. We measure the benefit of learning ontology representations. The performance is measured in $AU(\overline{PRC})$ and we can observe that ontology learning produces significant improvement, accounting for most of the difference between our model and competitors.

	w/o ontology learning	with ontology learning	% improvement
CELLCYLE	0.211	0.269	6.7%
DERSI	0.209	0.231	10.5%
EISEN	0.281	0.392	39.5%
EXPR	0.223	0.382	71.3%
GASCH1	0.256	0.369	44.1%
GASCH2	0.208	0.273	31.2%
SEQ	0.218	0.341	21.3%
SPO	0.209	0.241	15.3%

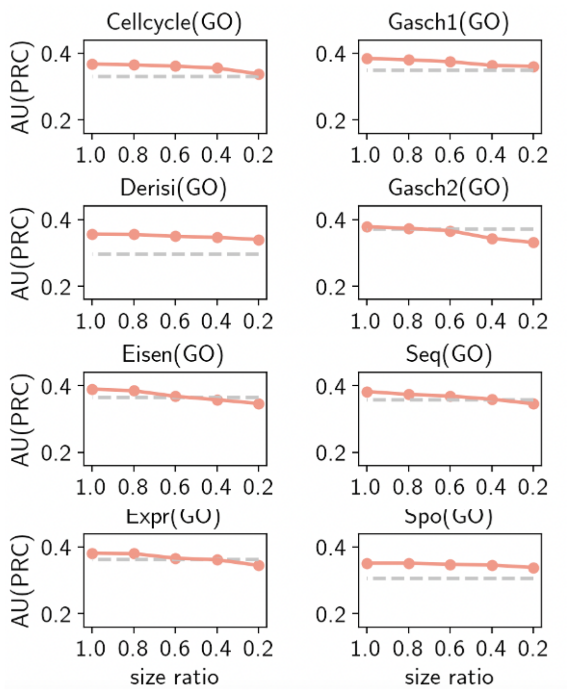

Figure 5: Evaluation on Surj 's robustness to varying data size. We test the Surj 's sensitivity to low data scenarios. We perform the experiments using GO datasets with 80%, 60%, 40%, 20% of the training data. The gray dashed lines on the background indicate the performance of the next best model with full training data. Surj remains superior even with only half of the training data provided compared to competitors with full training data.

little data environment. Figure 5 shows the results of varying data size environment. We train Surj with varying training data sizes, 80%, 60%, 40%, and 20%, from GO datasets. The red lines are Surj 's performance and the gray dashed lines indicate the next best model with full training data. We can see that the performance deteriorates gracefully as the data size shrinks and it remains superior in all datasets to the next best model even with only half of the training data (around 800 data records). All predictions in this experiment remain hierarchy violation-free. The robustness of Surj to low data regimes affords broad use.

6 DISCUSSION AND FUTURE WORK

Our results show Surj outperforms competitive methods on a wide variety of datasets and is robust to low data / large ontology regimes. The design of Surj is based on a simple fully connected network and can be easily adapted to a more advanced neural network architecture targeting specific applications. For example, Surj can be attached to ResNet [23] to classify images or BERT [14] to classify web documents, depending on the given ontology[44]. Surj has also been used by a non-profit organization to classify heterogeneous social media posts (including crawls of Twitter, Reddit, and Facebook threads) and long-form survey responses into the Sustainable Development Goals Ontology (SDG) [2] [4] and the Social Progress Index (SPI)[5]. We trained Surj on 13k discourses labeled with SPI and SDG, and achieved a F1 score of 0.353 on 2k holdout posts, compared to 0.171 with baseline multi-label classification without ontology learning. Surj is deployed within an online dashboard serving policymakers and entrepreneurs and has processed over 2M posts in specific areas. Being easy to implement and quick to train allows people with limited technical experience or access to computational resources to achieve state of the art Surj . We expect Surj to have a broader impact across different disciplines.

As future work, we intend to evaluate Surj for different data modalities, such as raw image and text. We would like to explore how Surj compares to models tailored for natural language processing [9, 10, 24, 31]. We also plan to more thoroughly investigate how Surj performs in zero-shot or few-shot scenarios. Since the relationships between labels are encoded independently of training data, Surj can predict "nearby" labels in the embedding space even without seeing labeled examples.

7 CONCLUSION

We propose Surj , a lightweight neural framework for hierarchical (and graph-structured) multi-label classification. The framework learns a representation of a label hierarchy and maps the input instances onto the label representation space. We also present an algorithm Global Hierarchy Violation to measure the hierarchy violations from the predicted outputs. To our knowledge, no existing HMC work presents any evaluation on the hierarchy violations. Surj outperforms HMC models in 17 out of 20 datasets from three domains by significant margins and do not violate the hierarchical constraints in all datasets. We also show that Surj trains up to 40 times faster than the current state-of-the-art model C-HCMNN. Easy and quick to train allows Surj to be adaptable to ontology changes. For evaluation purpose, we focus on genomics datasets, but Surj is generalized and is flexible to different data modalities. We foresee Surj as relevant for applications for the web [32, 35, 43], document processing [37], finance [45, 46], and more.

[4]https://www.unep.org/explore-topics/sustainable-development-goals/what-we-do/monitoring-progress/sdg-interface-ontology
[5]https://www.socialprogress.org/2020-Social-Progress-Index-Methodology.pdf

REFERENCES

[1] Tarek Abid, Hafed Zarzour, Mohamed Ridda Laouar, and Mohamed Tarek Khadir. 2016. Towards a smart city ontology. In *AICCSA 2016*. IEEE, 1–6.

[2] Ki-moon Ban. 2016. Sustainable Development Goals. (2016).

[3] Wei Bi and James T Kwok. 2011. Multilabel classification on tree-and dag-structured hierarchies. In *ICML 2011*.

[4] Helyane Bronoski Borges and Julio Cesar Nievola. 2012. Multi-label hierarchical classification using a competitive neural network for protein function prediction. In *IJCNN 2012*. IEEE, 1–8.

[5] Ricardo Cerri, Rodrigo C Barros, and André CPLF de Carvalho. 2011. Hierarchical multi-label classification for protein function prediction: A local approach based on neural networks. In *ISDA 2011*. IEEE, 337–343.

[6] Ricardo Cerri, Rodrigo C Barros, and André CPLF De Carvalho. 2014. Hierarchical multi-label classification using local neural networks. *J. Comput. System Sci.* 80, 1 (2014), 39–56.

[7] Ricardo Cerri, Rodrigo C Barros, André CPLF de Carvalho, and Yaochu Jin. 2016. Reduction strategies for hierarchical multi-label classification in protein function prediction. *BMC bioinformatics* 17, 1 (2016), 1–24.

[8] Nicolo Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. 2006. Incremental algorithms for hierarchical classification. *The Journal of Machine Learning Research* 7 (2006), 31–54.

[9] Soumya Chatterjee, Ayush Maheshwari, Ganesh Ramakrishnan, and Saketha Nath Jagaralpudi. 2021. Joint Learning of Hyperbolic Label Embeddings for Hierarchical Multi-label Classification. In *EACL 2021*. 2829–2841.

[10] Boli Chen, Xin Huang, Lin Xiao, Zixin Cai, and Liping Jing. 2020. Hyperbolic interaction model for hierarchical multi-label classification. In *AAAI 2020*, Vol. 34. 7496–7503.

[11] Amanda Clare. 2003. *Machine learning and data mining for yeast functional genomics*. Ph. D. Dissertation. Citeseer.

[12] Gregory Currie. 1989. *An ontology of art*. Springer.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR 2009*. Ieee, 248–255.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[15] Ivica Dimitrovski, Dragi Kocev, Suzana Loskovska, and Sašo Džeroski. 2011. Hierarchical annotation of medical images. *Pattern Recognition* 44, 10-11 (2011), 2436–2449.

[16] Ivica Dimitrovski, Dragi Kocev, Suzana Loskovska, and Sašo Džeroski. 2012. Hierarchical classification of diatom images using ensembles of predictive clustering trees. *Ecological Informatics* 7, 1 (2012), 19–29.

[17] W Ford Doolittle. 1999. Phylogenetic classification and the universal tree. *Science* 284, 5423 (1999), 2124–2128.

[18] Paola Espinoza-Arias, María Poveda-Villalón, Raúl García-Castro, and Oscar Corcho. 2019. Ontological representation of smart city data: From devices to cities. *Applied Sciences* 9, 1 (2019), 32.

[19] Shou Feng, Ping Fu, and Wenbin Zheng. 2018. A hierarchical multi-label classification method based on neural networks for gene function prediction. *Biotechnology & Biotechnological Equipment* 32, 6 (2018), 1613–1621.

[20] Milton Friedman. 1937. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association* 32, 200 (1937), 675–701.

[21] Yuxia Geng, Jiaoyan Chen, Zhuo Chen, Jeff Z. Pan, Zhiquan Ye, Zonggang Yuan, Yantao Jia, and Huajun Chen. 2021. OntoZSL: Ontology-enhanced Zero-shot Learning. *Proceedings of the Web Conference 2021* (2021).

[22] Eleonora Giunchiglia and Thomas Lukasiewicz. 2020. Coherent Hierarchical Multi-Label Classification Networks. *NeurIPS 2020* 33 (2020).

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR 2016*. 770–778.

[24] Wei Huang, Enhong Chen, Qi Liu, Yuying Chen, Zai Huang, Yang Liu, Zhou Zhao, Dan Zhang, and Shijin Wang. 2019. Hierarchical multi-label text classification: An attention-based recurrent network approach. In *CIKM 2019*. 1051–1060.

[25] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. In *Bayesian Deep Learning Workshop (NeurIPS 2016)*.

[26] Bryan Klimt and Yiming Yang. 2004. The enron corpus: A new dataset for email classification research. In *ECML 2004*. Springer, 217–226.

[27] Adila Krisnadhi, Yingjie Hu, Krzysztof Janowicz, Pascal Hitzler, Robert Arko, Suzanne Carbotte, Cynthia Chandler, Michelle Cheatham, Douglas Fils, Timothy Finin, et al. 2015. The GeoLink modular oceanography ontology. In *ISWC 2015*. Springer, 301–309.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *NeurIPS 2012* 25 (2012), 1097–1105.

[29] Maxat Kulmanov, Mohammed Asif Khan, and Robert Hoehndorf. 2018. DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics* 34, 4 (2018), 660–668.

[30] Yu Li, Sheng Wang, Ramzan Umarov, Bingqing Xie, Ming Fan, Lihua Li, and Xin Gao. 2018. DEEPre: sequence-based enzyme EC number prediction by deep learning. *Bioinformatics* 34, 5 (2018), 760–769.

[31] Federico López and Michael Strube. 2020. A Fully Hyperbolic Neural Model for Hierarchical Multi-class Classification. In *EMNLP 2020*. 460–475.

[32] Gjorgji Madjarov, Vedrana Vidulin, Ivica Dimitrovski, and Dragi Kocev. 2019. Web genre classification with methods for structured output prediction. *Information Sciences* 503 (2019), 551–573.

[33] Luca Masera and Enrico Blanzieri. 2018. Awx: An integrated approach to hierarchical-multilabel classification. In *ECML-PKDD 2018*. Springer, 322–336.

[34] George A Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

[35] Antonio Montieri, Domenico Ciuonzo, Giampaolo Bovenzi, Valerio Persico, and Antonio Pescapé. 2019. A dive into the dark web: Hierarchical traffic classification of anonymity tools. *IEEE Transactions on Network Science and Engineering* 7, 3 (2019), 1043–1054.

[36] Elias Moons, Tinne Tuytelaars, and Marie-Francine Moens. 2018. Text-Enriched Representations for News Image Classification. *Companion Proceedings of the The Web Conference 2018* (2018).

[37] Katariina Nyberg, Tapani Raiko, Teemu Tiinanen, and Eero Hyvönen. 2010. Document classification utilising ontologies and relations between documents. In *Proceedings of the eighth workshop on mining and learning with graphs*. 86–93.

[38] Katariina Nyberg, Tapani Raiko, Teemu Tiinanen, and Eero Hyvönen. 2010. Document Classification Utilising Ontologies and Relations between Documents. (09 2010), 86–93. https://doi.org/10.1145/1830252.1830264

[39] Tomer Sagi, Yoav Lehahn, Koby Bar, and Lisa A Miller. 2020. Artificial intelligence for ocean science data integration: current state, gaps, and way forward. *Elementa: Science of the Anthropocene* 8 (2020).

[40] Leander Schietgat, Celine Vens, Jan Struyf, Hendrik Blockeel, Dragi Kocev, and Sašo Džeroski. 2010. Predicting gene function using hierarchical multi-label decision tree ensembles. *BMC bioinformatics* 11, 1 (2010), 1–14.

[41] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC 2018*. Springer, 593–607.

[42] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[43] MuHee Song, SooYeon Lim, DongJin Kang, and SangJo Lee. 2006. Ontology-based automatic classification of web documents. In *ICIC 2006*. Springer, 690–700.

[44] Mu-Hee Song, Soo-Yeon Lim, Dong-Jin Kang, and Sang-Jo Lee. 2005. Automatic classification of web pages based on the concept of domain ontology. In *APSEC 2005*. IEEE, 7–pp.

[45] Timen Stepišnik Perdih, Senja Pollak, and Blaž Škrlj. 2021. JSI at the FinSim-2 task: Ontology-Augmented Financial Concept Classification. In *The Web Conference 2021*. 298–301.

[46] Ke Tian and Hua Chen. 2021. aiai at the FinSim-2 task: Finance Domain Terms Automatic Classification Via Word Ontology and Embedding. In *The Web Conference 2021*. 320–322.

[47] Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. 2008. Decision trees for hierarchical multi-label classification. *Machine learning* 73, 2 (2008), 185.

[48] Jonatas Wehrmann, Ricardo Cerri, and Rodrigo Barros. 2018. Hierarchical multi-label classification networks. In *ICML 2018*. 5075–5084.

[49] Frank Wilcoxon. 1992. Individual comparisons by ranking methods. In *Breakthroughs in statistics*. Springer, 196–202.

[50] Changdong Xu and Xin Geng. 2019. Hierarchical classification based on label distribution learning. In *AAAI 2019*, Vol. 33. 5533–5540.

[51] Xiaoshi Zhong and Jagath C Rajapakse. 2020. Graph embeddings on gene ontology annotations for protein–protein interaction prediction. *BMC bioinformatics* 21, 16 (2020), 1–17.

[52] Zhenzhen Zou, Shuye Tian, Xin Gao, and Yu Li. 2019. mldeepre: Multi-functional enzyme function prediction with hierarchical multi-label deep learning. *Frontiers in genetics* 9 (2019), 714.

[53] Emile Zuckerkandl and Linus Pauling. 1965. Evolutionary divergence and convergence in proteins. In *Evolving genes and proteins*. Elsevier, 97–166.