IEEJ TRANSACTIONS ON ELECTRICAL AND ELECTRONIC ENGINEERING
IEEJ Trans 2020
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI:10.1002/tee.23150

Paper

Hierarchical Multilabel Classifier for Gene Ontology Annotation Using Multihead and Multiend Deep CNN Model

Xin Yuan[*], Non-member
Erli Pang[**], Non-member
Kui Lin[**], Non-member
Jinglu Hu[*a], Member

Gene ontology annotation is known to be a very complicated multilabel classification task, and the hierarchical multilabel classification (HMC) approaches with local classifiers have been shown to be effective for the task. In a traditional HMC method, a set of hierarchically organized simple local classifiers are usually used, each of which for one hierarchical level separately. In this paper, we propose a novel hierarchical multilabel classifier implementing the whole set of hierarchically organized local classifiers in one deep convolution neural network (CNN) model with multiple heads and multiple ends (MHME). The proposed MHME CNN model consists of three parts: the body part of a deep CNN model shared by different local classifiers for feature extraction and feature mapping; the multiend part of a set of autoencoders performing feature fusion transforming the input vectors of different local classifiers to feature vectors with the same length so as to share the feature mapping part; and the multihead part of a set of linear multilabel classifiers. Furthermore, a sophisticated recursive algorithm is designed to train the MHME CNN model to realize the functions of a set of hierarchically organized local classifiers. In this way, by sharing a deep CNN with multiple local classifiers, we are able to construct more powerful local classifiers for each level with limited training samples, and to achieve better classification performance. Experiment results on various benchmark datasets show that the proposed deep CNN-based model has better performance than the state-of-the-art traditional models. Moreover, it gives rather good performance even under a transfer learning. © 2020 Institute of Electrical Engineers of Japan. Published by John Wiley & Sons, Inc.

Keywords: hierarchical multilabel classification; GO annotation; deep learning; autoencoder; feature extraction

Received 4 November 2019; Revised 27 February 2020

1. Introduction

As gene ontology (GO) reveals, individual genes contribute to the biology of an organism at the molecular, cellular, and organism levels. GO annotation plays an essential role in many biomedical studies [1]. However, with the GO functions continuing to be enriched, the number of GO functions, which could be annotated is more than 40 thousands. Therefore, fully experimental protein annotation becomes time-consuming and cost consuming; automatic prediction with the help of a machine learning method is widely considered to be more practical. Compared to the biological experiment methods, the machine learning method is more economical and convenient.

Automatic GO annotation method is a process to predict the GO functions (classes) as a multilabel classification task [2] by checking the combination of proteins. However, it could not be dealt with a normal multilabel method as GO annotation is very complicated, not only the number of GO functions is huge but also the functions are arranged in a hierarchy: typically a direct acyclic graph [3] where each node represents a function. These functions could be divided into different levels according to the grade. It is based on the 'is-a' relationship [4]. Hierarchical multilabel classification (HMC) is one kind of the algorithms known to have better performance [5], since it is a method ranking labels into a hierarchical structure, and performing classification on each level independently.

HMC [6–8], is a kind of methods which separate the labels into hierarchical levels and apply the classifiers to different levels, respectively. Multiform HMC methods could be divided into two circumstances, local hierarchical classification approach and global hierarchical classification. The local approach also called top-down approach [9–11] trains the hierarchical structure network from the higher level to the lower level. Global approach uses a classification algorithm, which learns a global classifier about the whole classes. For example, the authors of Ref. [12] investigated the flat classification algorithm naive Bayes applied in protein function prediction. Flat classification is the simplest approach in HMC methods, but it ignores class-relationship. However, the local approach is used to solve protein function prediction, as it used augmented versions of dealing with the problem which has hierarchical classes [13].

In Refs. [14–17], Cerri et al applied the HMC to protein function prediction problem with the local approach. It is a set of multilayer perceptron (MLP) classifiers trained for each level, separately. The local-based classification model trains MLPs for each level, called a HMC with local MLP (HMC-LMLP). In a HMC-LMLP, the predictions with MLP at a given level are used as inputs to the neural network responsible for the prediction in the next level. These methods focused on the optimizing model by training it level by level and propagating the positive signal from the previous output. Although the HMC-LMLP achieves a state-of-the-art performance, it is difficult to construct more powerful local

[a] Correspondence to: Jinglu Hu. E-mail: jinglu@waseda.jp

[*]Graduate School of Information, Production and Systems, Waseda University, Kitakyushu-shi, 808-0135, Japan

[**]College of Life Science, Beijing Normal University, 19 Xinjiekou Outer St, BeiTaiPingZhuang, Haidian Qu, Beijing, 100875, China

classifiers due to the limitation of training samples available in each class level.

Following the basic idea of HMC with local neural network models and the analyzing method of protein sequences, in this paper, we propose a novel hierarchical multilabel classifier implementing the whole set of hierarchically organized local classifiers in one deep convolution neural network (CNN) model with multiple heads and multiple ends (MHME). For this purpose, the proposed MHME CNN model consists of three parts: the body part, the multiend part, and the multihead part. The body part is a deep CNN model shared by different local classifiers for the feature extraction and feature mapping. The multiend part is a set of autoencoders performing feature fusion and dimension reduction which transforms the input vectors of different local classifiers to feature vectors with the same length so as to share the feature mapping of the deep CNN model. The multihead part is a set of linear multilabel classifiers realizing the multilabel protein function prediction. Furthermore, we design a sophisticated recursive algorithm to train the MHME CNN model in such way that it performs the functions of a set of hierarchically organized local classifiers. In this way, we are able to design a better hierarchical multilabel classifier by increasing the complexity of the feature mapping model via sharing a deep CNN with multiple local classifiers, and achieve better annotation performance.

On the other hand, protein sequence as a very normal feature has been used in many types of research. Considering that protein sequence consists of 20 different amino acids, which is similar to nature language [18]. In this way, the n-gram algorithm utilized to encode proteins is defined as the appearance frequency of n consecutive amino acids [19,20]. The n-gram method generated from natural language processing is now developed in the bioinformatics field. In Ref. [21], the authors investigated larger numbers of protein GO function annotations dealing with convolutional neural networks. Protein sequences have been represented as one-hot encoding in units of the tripeptide, and a single-layer CNN model used as a feature extractor. In this paper, we use a '1 - gram + 2 - gram' encoding method to obtain an appearance frequency vector as the input of the MHME CNN model.

The rest of our paper is organized as follows. Section 2 introduces the '1 - gram + 2 - gram' encoding method and formulates local classifiers for each level. Section 3 describes the details of the MHME CNN model and the training algorithm. Section 4 carries out experiments on various benchmark datasets and compares the proposed MHME CNN model with the related existing methods. Finally, Section 5 gives the conclusions.

2. Problem Formulation

In this section, we first introduce the notations and then formulate the problem.

2.1. The notation and the problem

Suppose we denote the protein instances by $\mathbf{S} = \{seq_1, seq_2, seq_3, \ldots seq_i — i = N\}$, N is the number of instance, and seq_i represent the protein sequence of instance i. Protein functions are denoted by $\mathbf{C} = \{c_1, c_2, c_3, \ldots c_j — j = C\}$, C is the species number of functions. Different from sample binary classification and multiclass classification, in hierarchical classification databases used $D = \{(x_i, y_i), 1 \le i \le N\}$ to express the instance in different levels, if x_i belongs to function c_j, then $c_j = 1$, or $c_j = 0$. We use $y_n = \{c_1, c_2, c_3, c_4, \ldots c_j — j = C\}$ to express the label matrix. At label space, we assume that the number of level of the hierarchy classes is q, expressed as $\mathbf{Y} = \{Y_1, Y_2, Y_3, \ldots Y_n — n = q\}$. For label in level q is that $Y_q = \{c_{q1}, c_{q2}, \ldots c_{qn} — c_{qn} \in \mathbf{C}\}$.

2.2. '1 - gram + 2 - gram' encoding

The encoding method encodes the protein sequences of amino acids seq_i $(i = 1, \ldots, N)$ into appearance frequency vectors, x_i $(i = 1, \ldots, N)$, which are column vectors with a length of 420, the number of 20 amino acids, and 400 dipeptides.

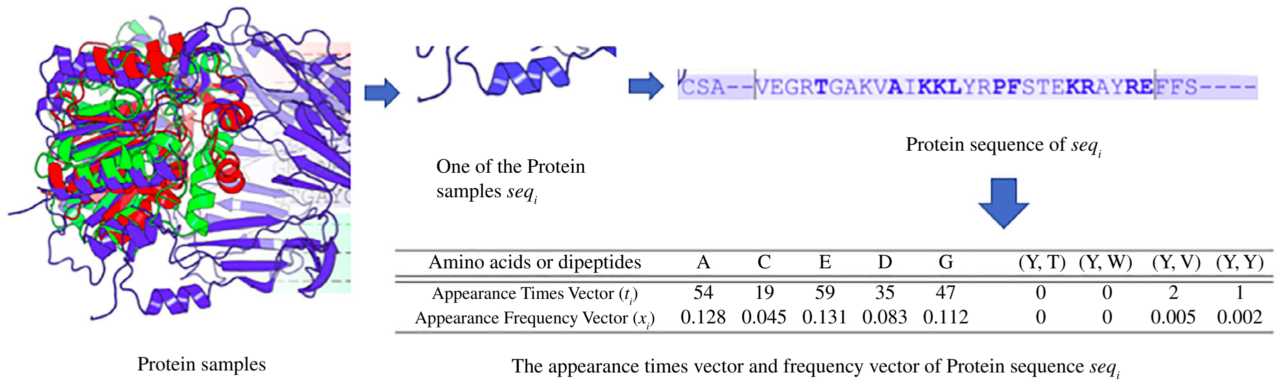
Figure 1 shows an image of the '1 - gram + 2 - gram' encoding method. For a given protein sequence seq_i, the appearance time vector t_i is first obtained by counting the number of times of each amino acid and dipeptide appearing in the protein sequence seq_i. Then the appearance frequency vector x_i is defined by

$$x_i = \frac{t_i}{\dim(x_i)} \tag{1}$$

where $\dim(x_i) = 20^1 + 20^2 = 420$. x_i $(i = 1, \ldots, N)$ is then used as an input vector of our MHME CNN model, where N is the number of protein samples.

2.3. Local classifiers for each level

Figure 2 shows the formulation of HMC with local classifiers. The local classifiers are designed to be in a hierarchically organized way so as to capture a top-down relationship of labels between different levels. The local classifier for level 1 (classifier 1) has the input vector of $I_1 = x$ and the output vector of $Z_1 = Y_1$. The local classifier for level 2 (classifier 2) has the input vector of $I_2 = [x, \widehat{Y_1}] = [x, \widehat{Z_1}]$, combined x with the output vector of classifier 1, and the output vector of $Z_2 = [Y_1, Y_2]$. In this way, the local classifier for level n $(n = 1, \ldots, q)$ (classifier n) will have the input vector of $I_n = [x, \widehat{Y_1}, \widehat{Y_2}, \ldots, \widehat{Y_{n-1}}] = [x, \widehat{Z_{n-1}}]$, combined x with the output vector of classifier $n - 1$, and the output vector of $Z_n = [Y_1, Y_2, \ldots, Y_n]$. Traditionally, the classifiers 1 to n are implemented by using simple MLP models due to the limitation of training samples in each level.


CSA--VEGRTGAKVAIKKLYRPFSTEKRAYREFFS----
One of the Protein samples seq_i
Protein sequence of seq_i
Protein samples
The appearance times vector and frequency vector of Protein sequence seq_i

Amino acids or dipeptides	A	C	E	D	G	(Y, T)	(Y, W)	(Y, V)	(Y, Y)
Appearance Times Vector (t_i)	54	19	59	35	47	0	0	2	1
Appearance Frequency Vector (x_i)	0.128	0.045	0.131	0.083	0.112	0	0	0.005	0.002

Fig. 1. An image of '1-gram + 2-gram' encoding-based feature extraction of protein sequences

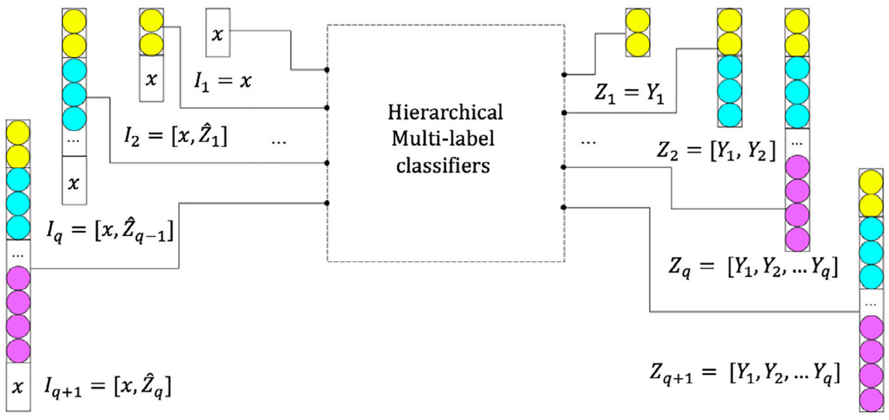

$I_1 = x$
$I_2 = [x, \hat{Z}_1]$
$I_q = [x, \hat{Z}_{q-1}]$
$I_{q+1} = [x, \hat{Z}_q]$
Hierarchical Multi-label classifiers
$Z_1 = Y_1$
$Z_2 = [Y_1, Y_2]$
$Z_q = [Y_1, Y_2, \ldots Y_q]$
$Z_{q+1} = [Y_1, Y_2, \ldots Y_q]$

Fig. 2. The formulation of HMC with local classifiers

2.4. An additional global classifier There exists not only top-down relationship of labels between different levels but also button-up label relationship. In order to capture both top-down relationship and button-up relationship of labels between different levels, in the proposed model, we add another global classifier which has the input vector of $I_{q+1} = [x, \widehat{Z}_q]$, combined x with the output vector of the last level classifier q, and the output vector of $Z_{q+1} = [Y_1, Y_2, \ldots, Y_q]$.†.where † denotes $\widehat{Z}_1, \widehat{Z}_2, \ldots, \widehat{Z}_q$ are the prediction values of $Z_1, Z_2, \ldots, Z_q$ by the local classifiers.

Traditionally, one constructs $q+1$ neural network classifiers separately for the above q local classifiers and one global classifier.

$$Z_n = F_n(I_n), \quad n = 1, \ldots, q+1 \tag{2}$$

where $I_1 = x$ and $I_n = [x, \widehat{Z}_{n-1}]$. In the next section, we will design a deep CNN model with MHME to implement both the q local classifiers and the additional global classifier in one deep neural network, and design a sophisticated recursive algorithm to train the MHME CNN model to perform a set of hierarchically organized powerful classifiers with the limited training samples.

3. The Deep MHME CNN Model

Figure 3 shows the structure of deep CNN model with multiple ends and multiple heads (MHME) to implement the $q+1$ hierarchically related classifiers using one deep CNN model. The MHME CNN model consists of three parts: the body part, the multiend part, and the multihead part, which realizes the $q+1$ classifiers with sharing the body part, referred to Fig. 3.

$$Z_n = f_n(\Omega_n^{(3)}, \phi(\Omega^{(2)}, A_n(\Omega_n^{(1)}, I_n))), \tag{3}$$

where $n = 1, \ldots, q+1$, $\phi(\Omega^{(2)}, \cdot)$, $A_n(\Omega_n^{(1)}, \cdot)$ and $f_n(\Omega_n^{(3)}, \cdot)$ denote the body part, the multiend part, and the multihead part, and $\Omega^{(2)}$, $\Omega_n^{(1)}$, $\Omega_n^{(3)}$ are the parameters, respectively.

3.1. The multiend part: A set of autoencoders The multiend part corresponds to the inputs of the $q+1$ hierarchically related classifiers. As can be seen from the description of the previous section, the local classifiers and the additional global classifier have their input vectors with different lengths. In order to share the body part for feature extraction, these input vectors should be transformed to a set of feature vectors with same length at the specific feature space.

$$\begin{cases} a_1 = x \\ a_n = A_n(\Omega_n^{(1)}, I_n), \quad n = 2, \ldots, q+1 \end{cases} \tag{4}$$

where a_n is the outputs of multiend part. For simplicity, we set $\dim(a_n) = \dim(x)$.

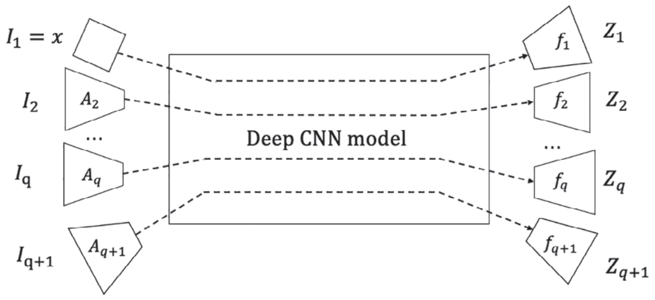


Fig. 3. The structure of the multihead and multiend deep CNN model

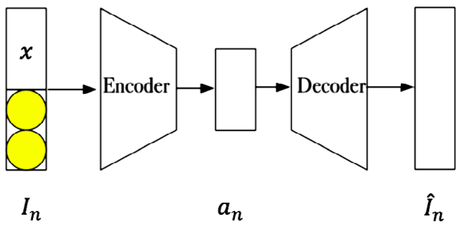


Fig. 4. The structure of autoencoder A_n

Due to no teacher signals for a_n, we apply autoencoders [22], which realizes the transforming of (4) by minimizing the information loss. As shown in Fig. 4, an autoencoder neural network is an unsupervised learning algorithm that applies backpropagation [23], by setting the target values to be equal to the inputs. The general idea of autoencoder is to represent the data through a nonlinear encoder to a hidden layer and use the hidden units as the new feature representations. The major purpose of autoencoder is to learn an approximation to the identity of original protein representation, by encouraging its output to be as similar to its input as possible. Recently, the variants of original autoencoder have drawn increasing attention as an effective feature extractor or descriptor [24]. The autoencoder plays two roles: feature fusion and dimension reduction.

3.2. The body part: A deep CNN model The rectangle in the middle of Fig. 3 represents the body part, which realizes a feature mapping or feature extraction. CNN outperforms in dealing with the spatial information like graph, text, etc. Recently, it is also shown to be effective in extracting feature from protein sequences in the some related work [21,25]. To demonstrate the ordered arrangement of protein sequences, the proteins are described as the frequency of amino acids and dipeptides. Moreover, the GO terms (predictions of linear classifiers, mentioned in the second section) added in input during the training makes the features very high-dimensional. Compared with the one-layer CNN model used in Ref. [21], a deep CNN model is more reasonable for our purposes of feature extraction and feature mapping. The details of deep CNN model used in our experiments will be described in Subsection 5.2.1.

$$\phi_n = \phi(\Omega^{(2)}, a_n), \quad n = 1, \ldots, q+1 \tag{5}$$

where ϕ_n is the output of the body part.

3.3. The multihead part: A set of linear classifiers

The multihead part consists of a set of linear multilabel classifiers with input vectors from the outputs of body part. As the linear multilabel classifiers, we simply use a linear network with logistic sigmoid outputs.

$$Z_n = f_n(\Omega_n^{(3)}, \phi_n), \quad n = 1, \ldots, q+1 \tag{6}$$

4. Training Method

The output of hierarchical multilabel classifier Z_{q+1} is calculated recursively by

$$Z_1 = f_1(\Omega_1^{(3)}, \phi(\Omega^{(2)}, A_1(\Omega_1^{(1)}, I_1))), I_1 = x$$

$$Z_2 = f_2(\Omega_2^{(3)}, \phi(\Omega^{(2)}, A_2(\Omega_2^{(1)}, I_2))), I_2 = [x, Z_1]$$

$$Z_{q+1} = f_{q+1}(\Omega_{q+1}^{(3)}, \phi(\Omega^{(2)}, A_{q+1}(\Omega_{q+1}^{(1)}, I_{q+1}))),$$

$$I_{q+1} = [x, Z_q]. \tag{7}$$

4.1. Training of parameters $\Omega_n^{(1)}$

$\Omega_n^{(1)}$ $(n = 1, \ldots, q+1)$ are the parameters of autoencoders. As shown in Fig. 4, they are trained in an unsupervised manner by optimizing the mean squared error (MSE) loss function $E_n = \varepsilon_{MSE}(\widehat{I}_n, I_n)$. Let z be the input of encoder and $\widehat{z}$ be the output of decoder. Then ε_{MSE} is defined by

$$\varepsilon_{MSE}(\widehat{z}, z) = \frac{1}{N} \sum_{i=1}^{N} (\widehat{z}_i - z_i)^2 \tag{8}$$

where N is the number of samples.

4.2. Training of parameters $\Omega^{(2)}$ and $\Omega_n^{(3)}$

$\Omega^{(2)}$ and $\Omega_n^{(3)}$ $(n = 1, \ldots, q+1)$ are the parameters of deep CNN and the linear network with logistic outputs. They are trained in a supervised manner by optimizing the sigmoid cross-entropy (SCE) loss function $E = \sum_{i=1}^{n} \varepsilon_{SCE}(\widehat{Z}_n, Z_n)$. In this formulation, ε_{SCE} is defined by

$$\varepsilon_{SCE}(\widehat{z}, z) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C_n} [z_{ij} \times \log(\widehat{z}_{ij}) + (1 - z_{ij}) \times \log(1 - \widehat{z}_{ij})] \tag{9}$$

and $C_n = \dim(Z_n)$.

However, as we can see from (7) that the $q+1$ classifiers are hierarchically related, a sophisticated training algorithm is needed to train the parameters of $\Omega_n^{(1)}$, $\Omega^{(2)}$, and $\Omega_n^{(3)}$ $(n = 1, \ldots, q+1)$. The training algorithm consists of the following three steps.

- *Step* 1: Pre-training of $\Omega^{(2)}$ and $\Omega_n^{(3)}$
- Set $a_n = x$ $(n = 1, \ldots, q+1)$. Then train $\Omega^{(2)}$ and $\Omega_n^{(3)}$ based on (5), (6), and (9). After the pre-training, we are able to calculate the approximated predictions $\widehat{Z}_n$ $(n = 1, \ldots, q+1)$ using (5) and (6);
- *Step* 2: Training of $\Omega_n^{(1)}$
- Let $I_1 = x$ and $I_n = [x, \widehat{Z}_{n-1}]$ $(n = 2, \ldots q+1)$. Then train $\Omega_n^{(1)}$ based on (4) and (8).
- *Step* 3: Training of $\Omega^{(2)}$ and $\Omega_n^{(3)}$
- Let $I_1 = x$ and $I_n = [x, \widehat{Z}_{n-1}]$ $(n = 2, \ldots q+1)$. Then train $\Omega^{(2)}$ and $\Omega_n^{(3)}$ based on (7) and (9). After the training, we calculate the predictions $\widehat{Z}_n$ $(n = 1, \ldots, q+1)$ using (7);
- *Step* 4: Repeat *Steps* 2 and 3, until an appropriate stop condition is satisfied.

5. Experiments

In this section, the proposed MHME CNN model is applied to various banchmark protein function datasets, and the results are compared with the state-of-the-art traditional methods.

5.1. Evaluation metrics

The evaluation metrics for multilabel classification include the multilabel precision (MPrecision), the multilabel recall (MRecall), the multilabel F-score (MF-score) [26], and the multilabel G-mean(MG-mean). Considering that the multilabel classification method in our work is a combination of multibinary classifier for each class, MPrecision and MRecall are the averages for the precious and recall of all classes. They correspond to the micro average of the multilabel precision and the multilabel recall. Let us assume that D is a multilabel dataset, and the sample in $|D|$ is (x_i, y_i), $i = 1 \ldots |D|$, $y_i \in \{0, 1\}^m$ is the set of labels. Suppose $z_i \in \{0, 1\}^m$ be the set of labels predicted by HMC classifier for sample x_i. The definitions of the evaluation metrics are given by

$$\text{MPrecision} = \frac{1}{m} \sum_{i=1}^{|D|} \frac{|\, y_i \cap z_i \,|}{|\, z_i \,|} \tag{10}$$

$$\text{MRecall} = \frac{1}{m} \sum_{i=1}^{|D|} \frac{|\, y_i \cap z_i \,|}{|\, y_i \,|} \tag{11}$$

$$\text{MF-Score} = \frac{2 * \text{MPrecision} * \text{MRecall}}{\text{MPrecision} + \text{MRecall}} \tag{12}$$

$$\text{MG-mean} = \sqrt{\text{MPrecision} * \text{MRecall}} \tag{13}$$

In the multilabel classification, considering that the number of labels is imbalanced, so the result of MF-score and MG-means are more important.

5.2. Experiment settings

5.2.1. Details of the deep CNN model

Figure 5 shows the structure of deep CNN model used in the experiments, which has six convolutional layers and one flatten layer. Each layer of the CNN model is a 1-d CNN with ReLU activation function. Smaller size filters are used in layers 1 to 3 to extract the micro information of amino acids and dipeptides. And larger size filters are used in layers 4 to 6 so as to learn the ranking rule of amino acids. The size of filters and the number of channels are shown above each layer. For example, 'Conv 3: 1*5*256' denotes that the convolution layer 3 has a filter of size 1*5, and 256 channels. The pooling layers are used in the convolution layers 2, 4, 6, which are max-pooling with pool-size: 1*2. The notation such as

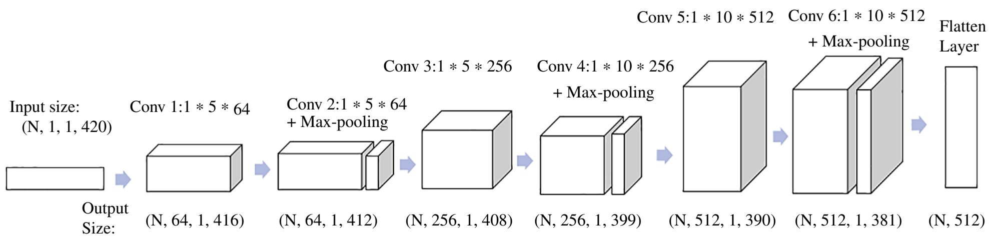


Fig. 5. The structure of deep CNN model consisting of six convolutional layers and one flatten layer

Table I. Datasets used in the paper of DeepGO

Datasets	Training	Validation	Test	Classes	Levels	Classes per level
DeepGO(BP)	20 000	9000	10 000	932	5	30/180/292/477/932
DeepGO(CC)	20 000	8800	10 000	439	4	18/126/260/439
DeepGO(MF)	20 000	6300	5000	589	5	23/160/261/392/589

Table II. Comparison results of the proposed MHME-CNN model with the HMC-LMLP, the DeepGO, and the simple DCNN models

		Evaluation metrics			
Datasets	Methods	MPrecision	MRecall	MF-score	MG-mean
DeepGO(BP)	DeepGO	0.3900	0.3400	0.3600	0.3700
	simple DCNN	0.4841	0.5001	0.4917	0.4920
	HMC-LMLP	0.4847	0.5005	0.4921	0.4935
	MHME-CNN	0.8028	0.5226	**0.5315**	**0.6418**
	DeepGO	0.6600	0.6100	0.6300	0.6400
DeepGO(CC)	simple DCNN	0.5028	0.5005	0.4959	0.5015
	HMC-LMLP	0.4973	0.5068	0.5014	0.5017
	MHME-CNN	0.9149	0.6122	**0.6619**	**0.7456**
	DeepGO	0.6000	0.3800	0.4600	0.5100
DeepGO(MF)	simple DCNN	0.5005	0.4998	0.4974	0.5001
	HMC-LMLP	0.4941	0.5010	0.4975	0.4998
	MHME-CNN	0.5451	0.5125	**0.5093**	**0.5261**

Bold values are shown the best result in each group of the experiments.

(N, 64, 1, 416) describes an input or output of a CNN layer, which denotes that there are N training samples, 64 channels, and each channel with the size of [1, 416]. Finally, the output of the deep CNN model is a vector with the size of $1 * 512$. Obviously, the proposed MHME CNN model has a more complicated network compared to the traditional HMC-LMLP model. We are expecting it to have much better performance.

5.2.2. Details on the supervised and unsupervised training algorithms As the deep learning framework, we use Chainer developed by Community, Preferred Networks, Inc [27]. The training algorithm described in Section 4 is implemented by using Python programming language. In the experiments, each dataset is divided into three sets: training set, validation set, and test set.

The training set is used to train the models. Since the hyper-parameters are not so sensitive for the proposed model training in our experiments, the default values suggested by the deep learning framework, Chainer are used. When supervised leaning of the body part and the multihead part by using the algorithm described as *Step* 1 and *Step* 3 in Section 4, the initial learning rate used is 0.0075 and training process stops after 120 epochs where the performance is not improved anymore for the validation set. On the other hand, when the unsupervised learning of the autoencoders of the multiend part by using the algorithm described as the *Step* 2, the initial learning rate used is 0.05 and the learning was stopped after 30 epochs where the performance for validation set stops improving.

The *Step* 2 and *Step* 3 of training process was repeated 2 times where the performance for validation set stopped improving. Finally, the trained models are evaluated on the test set. In the experiments, efforts are made to prevent the training of models stuck at a local minimum. The proposed model and the conventional models are compared under a condition of being well-trained. All the results shown are an average of 10 trials, but the standard deviations are ignored since they are rather small in our experiments.

5.3. Comparison results with the State-of-the-Art methods Three datasets of protein sequences are used in this comparison, which are from the paper of DeepGO [21]. Table I describes the details of datasets. For the classes, GO functions are divided into three independent groups: the molecular functions (MF) that represent the activities performed by gene products; the cellular component (CC) functions, that describe the locations relative to cellular structures; and the biological process (BP) functions, that describe some ‘biological programs’ [1].

When applying the proposed MHME CNN model, according to the hierarchy of GO functions, the labels are divided into five levels for BP and MF functions, four levels for CC functions. Table I shows the numbers of classes per level for each dataset. The autoencoder used for each level is a one-layer MLP with a size of $\dim(I_n) \times \dim(x)$. The CNN model used is described in Fig. 5.

After training, the MHME CNN model is evaluated on test data. Table II shows the experimental results on the three datasets. For

Table III. UniProt datasets (number of samples)

Gene ontology	Human	Mouse	Thaliana	Cerevisiae
BP	3229	2743	488	488
CC	7194	6055	2149	2149
MF	3024	2756	776	776

comparison, the first row is the results of DeepGO model copied from the DeepGO paper [21]. The second row shows the results of simple DCNN (deep CNN) model, whose structure is similar to

$$Z_{q+1} = f_{q+1}(\Omega^{(3)}_{q+1}, \phi(\Omega^{(2)}, x)), \tag{14}$$

a deep CNN model without considering the label hierarchical relationship; and the third row is the result of the HMC-LMLP method [16]. The last row is the results of the proposed MHME CNN model.

From the results of Table II, we can see that the proposed MHME CNN model performs better than the state-of-the-art models. The results of HMC-LMLP stand out when the dataset has more GO terms (BP, 932). Because it is the model-trained GO terms level by level, whose result is influenced by GO terms more. The DeepGO or simple DCNN model does well when the dataset has less GO terms (CC, 439; MF, 589), as these two methods are concentrating on the feature extraction of protein sequences. The proposed MHME CNN model improves the performance by taking into account both hierarchy GO training and feature extraction.

5.4. Performance with transfer learning The proposed MHME CNN models trained on the three DeepGO datasets in the previous subsection are applied to other datasets picked up from the UniProt [28]. Table III shows the details of these datasets. These four new datasets are not used in the training and only used for testing.

Table IV shows the results. The proposed MHME CNN model is compared with the simple DCNN model and the HMC-LMLP model. From Table IV, we can see that the proposed MHME CNN model performs much better than other two methods. Moreover, the simple DCNN model outperforms over the HMC-LMLP model in transfer learning, which also proves that the deep learning-based method has better performance in the case of transfer learning. Further studies are needed to show the potential applications of our proposed MHME CNN model based on transfer learning.

6. Conclusions

In this paper, we proposed a multihead and multiend (MHME) deep CNN model for GO annotation to improve the performance of

Table IV. Results of the proposed MHME CNN model under transfer learning

			Evaluation metrics			
Datasets	Functions	Methods	MPrecision	MRecall	MF-score	MG-mean
Human		Simple DCNN	0.5127	0.5081	0.5085	0.5102
	BP	HMC-LMLP	0.4968	0.4989	0.4978	0.4979
		MHME-CNN	0.6967	0.6999	**0.6979**	**0.6983**
		Simple DCNN	0.5339	0.5308	0.5276	0.5314
	CC	HMC-LMLP	0.4929	0.4977	0.4953	0.4957
		MHME-CNN	0.6592	0.6651	**0.6613**	**0.6615**
		Simple DCNN	0.5249	0.5202	0.5208	0.5225
	MF	HMC-LMLP	0.4952	0.5000	0.4976	0.4977
		MHME-CNN	0.6565	0.6612	**0.6587**	**0.6588**
		Simple DCNN	0.5119	0.5069	0.5080	0.5093
	BP	HMC-LMLP	0.4977	0.4994	0.4981	0.4985
		MHME-CNN	0.6873	0.6907	**0.6887**	**0.6890**
		Simple DCNN	0.5273	0.5251	0.5218	0.5252
Mouse	CC	HMC-LMLP	0.4927	0.4977	0.4952	0.4951
		MHME-CNN	0.6601	0.6663	**0.6623**	**0.6626**
		Simple DCNN	0.5201	0.5167	0.5165	0.5183
	MF	HMC-LMLP	0.4949	0.5000	0.4975	0.4974
		MHME-CNN	0.6452	0.6502	**0.6476**	**0.6477**
		Simple DCNN	0.5273	0.5162	0.5207	0.5216
	BP	HMC-LMLP	0.4974	0.4989	0.4981	0.4982
		MHME-CNN	0.8793	0.8815	**0.8802**	**0.8803**
		Simple DCNN	0.5509	0.5498	0.5473	0.5496
Thaliana	CC	HMC-LMLP	0.4940	0.4977	0.4958	0.4959
		MHME-CNN	0.7833	0.7881	**0.7849**	**0.7851**
		Simple DCNN	0.5538	0.5479	0.5497	0.5507
	MF	HMC-LMLP	0.4958	0.5001	0.4979	0.4980
		MHME-CNN	0.8205	0.8242	**0.8222**	**0.8023**
		Simple DCNN	0.5265	0.5199	0.5223	0.5231
	BP	HMC-LMLP	0.4973	0.4989	0.4980	0.4981
		MHME-CNN	0.8788	0.8809	**0.8797**	**0.8798**
		Simple DCNN	0.5518	0.5499	0.5473	0.5498
Cerevisiae	CC	HMC-LMLP	0.4940	0.4976	0.4957	0.4956
		MHME-CNN	0.7822	0.7870	**0.7838**	**0.7840**
		Simple DCNN	0.5539	0.5510	0.5514	0.5523
	MF	HMC-LMLP	0.4962	0.5000	0.4981	0.4981
		MHME-CNN	0.8187	0.8225	**0.8105**	**0.8206**

Bold values are shown the best result in each group of the experiments.

complex HMC. The proposed MHME CNN model shares one deep CNN model with a set of linear classifiers, in which autoencoders are applied to fusing the features and reducing the dimension of feature vectors. The MHME CNN model is trained level by level to realize a set of local classifiers corresponding to the hierarchy labels, and one global classifier in order to capture the two-way relationship of labels. The simulation results show that our proposed MHME CNN model works well and has much better performance than the state of the art traditional methods, and our pre-built MHME CNN model also achieves a good performance on other GO annotation examples based on transfer learning.

In this paper, we only used the hierarchy GO dataset up to five levels. In our future work, we will apply the proposed model to GO datasets with more levels [29]. Moreover, we will carry out a study on applying other deep learning models, especially transfer learning for protein function predictions.

References

(1) Gene Ontology Consortium. The gene ontology (GO) database and informatics resource. *Nucleic Acids Research* 2004; **32**(suppl 1):D258–D261.

(2) Borges HB and Julio CN. "Multi-label hierarchical classification using a competitive neural network for protein function prediction," in *International Joint Conference on Neural Networks (IJCNN' 2012)*. IEEE, 2012, 1–8.

(3) Feng S, Fu P, Zheng W. A hierarchical multi-label classification algorithm for gene function prediction. *Algorithms* 2017; **10**(4):138.

(4) Wu F, Zhang J, Honavar V. Learning classifiers using hierarchically structured class taxonomies. In International Symposium on Abstraction, Reformulation, and Approximation(AAAI'*2005*). Berlin, Heidelberg: Springer; 2005; 313–320.

(5) Chen B, Hu J. Hierarchical multi-label classification based on over-sampling and hierarchy constraint for gene function prediction. *IEEJ Trans. on Electical and Electronic Engineering* 2012; **7**(2):183–189.

(6) Zhu X, Bain M. B-cnn: Branch convolutional neural network for hierarchical classification. *arXiv preprint arXiv:1709.09890* 2017.

(7) Silla CN and Freitas AA. "Novel top-down approaches for hierarchical classification and their application to automatic music genre classification," in 2009 *IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 2009, 3499–3504.

(8) Zhang L, Shah SK, Kakadiaris IA. Hierarchical multi-label classification using fully associative ensemble learning. *Pattern Recognition* 2017; **70**:89–103.

(9) Cerri R and de Carvalho André Carlos PLF. "Hierarchical multi-label classification using top-down label combination and artificial neural networks," in 2010 *Eleventh Brazilian Symposium on Neural Networks*. IEEE, 2010, 253–258.

(10) Wang X.-L, Zhao H, and Lu B.-L. "Enhance top-down method with meta-classification for very large-scale hierarchical classification," in *Proceedings of 5th International Joint Conference on Natural Language Processing*, 2011, pp. 1089–1097.

(11) Mahdi A, Qin J, Crosby G. *Deepfeat: A Bottom-Up and Top-Down Saliency Model Based on Deep Features of Convolutional Neural Nets*. arXiv preprint arXiv:1709.02495 (2017).

(12) Silla CN Jr and Freitas AA. "A global-model naive bayes approach to the hierarchical prediction of protein functions," in *International Conference on Data Mining (ICDM' 2009)*. IEEE, 2009, 992–997.

(13) Silla CN, Alex AF. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery* 2011; **22**(1–2):31–72.

(14) Cerri R, Rodrigo CB, André C d C. Hierarchical multi-label classification using local neural networks. *Journal of Computer and System Sciences* 2014; **80**(1):39–56.

(15) R. Cerri, C. B. Rodrigo, and C. d. C. André. "Hierarchical classification of gene ontology-based protein functions with neural networks," in *International Joint Conference on Neural Networks (IJCNN' 2015)*. IEEE, 2015, 1–8.

(16) Cerri R, Rodrigo CB, André C d C, Jin Y. Reduction strategies for hierarchical multi-label classification in protein function prediction. *BMC Bioinformatics* 2016; **17**(1):373.

(17) Wehrmann J, Cerri R, and Barros R. "Hierarchical multi-label classification networks," in *International Conference on Machine Learning*, 2018, 5225–5234.

(18) Cao R, Freitas C, Chan L, Sun M, Jiang H, Chen Z. ProLanGO: Protein function prediction using neural machine translation based on a recurrent neural network. *Molecules* 2017; **22**(10): 1732–1745.

(19) Zeng J, Li D, Wu Y, Zou Q, Liu X. An empirical study of features fusion techniques for protein-protein interaction prediction. *Current Bioinformatics* 2016; **11**(1):4–12.

(20) Wei L, Xing P, Zeng J, Chen J, Su R, Guo F. Improved prediction of protein–protein interactions using novel negative samples, features, and an ensemble classifier. *Artificial Intelligence in Medicine* 2017; **83**:67–74.

(21) Kulmanov M, Khan MA, Hoehndorf R. Deepgo: Predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics* 2017; **34**(4):660–668.

(22) Goodfellow I, Bengio Y, Courville A. *Deep Learning*. Cambridge, MA: MIT press; 2016.

(23) Makhzani A, Frey BJ. *A Winner-Take-all Method for Training Sparse Convolutional Autoencoders*. Citeseer: NIPS Deep Learning Workshop; 2014.

(24) Makhzani A, Frey BJ. Winner-take-all autoencoders. *Advances in Neural Information Processing Systems*, New York: Curran Associates, Inc.; 2015;**28**:2791–2799.

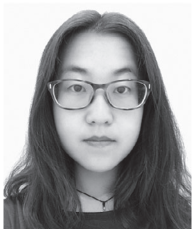
(25) Pan X, Rijnbeek P, Yan J, Shen H-B. Prediction of rna-protein sequence and structure binding preferences using deep convolutional and recurrent neural networks. *BMC Genomics* 2018; **19**(1):511.

(26) Tsoumakas G, Ioannis K. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)* 2007; **3**(3):1–13.

(27) Tokui S, Oono K, Hido S, and Clayton J. "Chainer: a next-generation open source framework for deep learning," in *Proceedings of Workshop on Machine Learning Systems (LearningSys) in the Twenty-Ninth Annual Conference on Neural Information Processing Systems (NIPS)*, vol. 5, 2015, 1–6.

(28) *UniProt*. [Online]. https://www.uniprot.org/uniprot//.

(29) Nakano FK, Lietaert M, Vens C. Machine learning for discovering missing or wrong protein function annotations. *BMC Bioinformatics* 2019; **20**(1):485.



Xin Yuan (Nonmember) received the B.E. degree in Measurement and Control Technology and Instrument from Beijing University of Chemical Technology, China in 2015 and the M.Sci. degree in Waseda University in 2017. Since 2017 September, she is currently working toward the Ph.D. degree in the Graduate School of Information, Production, and System, Waseda University. Her research interests include hierarchical multilabel classification, feature extraction, and bioinformatics.

Erli Pang (Nonmember) received the B.Sci. degree and the Ph.D. degree from Beijing Normal University, China. Now she works as an Associate Professor at College of Life, Beijing Normal University. Her research interests are on the areas of computational molecular biology.

Kui Lin (Nonmember) received the B.Sci. degree in 1983, the M.Sci. degree in 1990, and the Ph.D. degree in 1997 from Lanzhou University, China. From 1998 to 2002, he worked as a Research Fellow at BIC, NUS. From 2002 to 2007, he worked as an Associate Professor and Since July 2007, he has been a Professor at College of Life, Beijing Normal University. His research interests are on the areas of computational molecular biology.

Jinglu Hu (Member) received the M.Sci. degree in Electronic Engineering from Sun Yat-Sen University, China in 1986, and a Ph.D. degree in Computer Science and System Engineering from Kyushu Institute of Technology, Japan in 1997. From 1986 to 1993, he worked as a Research Associate and Lecturer at Sun Yat-Sen University. From 1997 to 2003, he worked as a Research Associate at Kyushu University. From 2003 to 2008, he worked as an Associate Professor and Since April 2008, he has been a Professor at Graduate School of Information, Production, and Systems of Waseda University. His research interests include Computational Intelligence such as neural networks and genetic algorithms, and their applications to system modeling and identification, bioinformatics, time series prediction, and so on. Dr. Hu is a member of IEEE, IEEJ, SICE, and IEICE.