Study on Deep Transfer Learning Methods for the Predictions of Protein Functions

Xin YUAN

June 2022

Waseda University Doctoral Dissertation

Study on Deep Transfer Learning Methods for the Predictions of Protein Functions

Xin YUAN

Graduate School of Information, Production and Systems
Waseda University

June 2022

Abstract

Carbohydrates, proteins and lipids are the three essential substances of life. Protein as one of them participates in almost all life activities. Protein function is the sum of all the behaviors of proteins and the behaviors that happen through proteins. Therefore, protein function prediction is of great significance for many biological studies. However, the entire experimental prediction methods have taken a lot to identify the function information but with little success. On the other hand, structural genomics projects exponentially increase the number of protein sequences by the high-throughput method in genome-wide strategies. Thus, more and more researches focus on computational prediction methods using protein sequences.

Protein function prediction can be carried out in many ways. This dissertation mainly discusses three cases: protein GO (gene ontology) annotation prediction, subcellular localization prediction, and PPI (protein-protein interaction) prediction. GO annotation is a description category of the whole functions. GO annotation prediction is a complicated multi-label classification task because there are thousands of GO classes and proteins usually have more than one annotation. Subcellular localization describes the proteins in different locations corresponding to the cellular functions. Finally, the protein interaction, as the critical point of forming protein macromolecular, includes PPI prediction and protein complex detection. They play an essential role in studying molecular functions.

Applying machine learning methods for protein function prediction has become popular in recent years. The superiority of machine learning has been proved many times. However, a powerful deep learning method is not yet successful in protein function prediction tasks. On one hand, since the prediction tasks are usually very complicated multi-label classification problems, it is crucial to implement deep neural networks for the tasks. However, experimentally annotated proteins are not available for most species. As a result, it requires to perform a transfer learning based on the experimentally annotated proteins available for some few type species. On the other hand, it has been known that it is difficult to implement a transfer learning in bioinformatics tasks because of the overfitting problems due to limited training samples.

To address the above problems, we develop a novel deep convolution neural network (CNN) model with multi-head and multi-end (MHME) to implement a class of classifiers in one model. The deep MHME CNN model shares a deep CNN feature extractor

in a class of different prediction tasks, that make the transfer learning possible. The proposed model is then applied for three protein function prediction tasks: GO annotation prediction, subcellular localization prediction and PPI prediction based on a transfer learning from different tasks and different species.

The dissertation contains five chapters as follows:

Chapter 1 first introduces the background of proteins and protein functions. Then we discuss the three prediction issues and the related researches and summarize the challenges. At last, the goal of deep modeling for protein function predictions and our proposal are listed.

Chapter 2 develops a deep hierarchical model for GO annotation prediction. GO annotation can be formulated as a very complicated hierarchical multilabel classification tasks consisting of a set of related local classifiers. A novel hierarchical multi-label classifier is proposed to implement the whole set of hierarchically organized local classifiers in one deep MHME CNN model. The proposed model consists of three parts: the body part of a deep CNN model shared by different local classifiers for feature extracting and mapping; the multi-end part of a set of autoencoders performing feature fusion transforming the input vectors of different local classifiers to feature vectors with the same length to share the feature mapping part; and the multi-head part of a set of linear multi-label classifiers. In this way, by sharing a deep CNN with multiple local classifiers, we can extract common feature and construct more powerful local classifiers for each level with limited training samples and achieve better classification performance. Experiment results on various benchmark datasets from Uniprot 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 transfer learning of same tasks, but different species.

Chapter 3 introduces a deep protein subcellular localization predictor enhanced with transfer learning of GO annotation. GO annotations have been shown to be helpful for the subcellular localization prediction. However, experimentally annotated proteins are not always available. It is motivated to perform deep learning of GO annotations on the available experimentally annotated proteins for some type species and transfer it to subcellular localization prediction on other species. The proposed deep protein subcellular localization predictor consists of a linear classifier and a deep CNN feature extractor. By using the deep MHME CNN model, a deep CNN feature extractor is first shared and pretrained in a deep GO annotation predictor, and then is transferred to the

subcellular localization predictor with fine-tuning using protein localization samples. In this way, we have a deep protein subcellular localization predictor enhanced with transfer learning of GO annotation. The proposed method has good performances on the Swiss-Prot datasets when transfer learning using the protein samples both within and out species. Moreover, it outperforms the state-of-the-art traditional methods on benchmark datasets.

Chapter 4 proposes a deep PPI predictor and reconstructs a PPI network based on deep transfer learning for protein complex detection. The completeness of a PPI network is crucial for the detection of protein complexes. However, complete PPI networks are not available for most species because experimentally identified PPIs are usually very limited. To solve the problem, a deep learning based PPI predictor is proposed to estimate the unknown PPIs, and construct a complete PPI network, from which protein complexes are detected using a spectral clustering method. Considering the facts that the similarities of GO annotations contribute to protein interactions, and the differences of subcellular localizations contribute to negative interactions, the deep MHME CNN model is used to pretrain a deep CNN feature extractor in a class of deep GO annotation and subcellular localization predictors using datasets from the type species, then transfer it to the PPI prediction model for fine-tuning, so as to have a deep PPI detector enhanced with transfer learning of GO annotation and subcellular localization prediction. Experimental results on benchmark datasets CYC2008 and MIPS show that the proposed method outperforms the state-of-the-art methods.

Chapter 5 concludes the contributions of this dissertation and provides future works. In summary, this dissertation proposes a deep MHME CNN model and applies it to the predictions of protein GO annotation, subcellular localization and PPI based on a transfer learning from different tasks and different species.

Preface

The general theme of this dissertation is to develop deep transfer learning methods for different protein function prediction problems. This dissertation is organized in five chapters. Most of the materials have been published in following listed journal papers and conference papers.

The materials in Chapter 2 can be found in

- [J4] X. Yuan, E. Pang, K.Lin and J.Hu. "Hierarchical Multi-label Classification for Gene Ontology Annotation using Multi-head and Multi-end Deep CNN model", *IEEJ Trans. on Electrical and Electronics Engineering*, Vol.15, No.7 pp.1057-1064, July, 2020.
- [P2] X. Yuan, W. Li, K. Lin and J. Hu, "A Deep Neural Network Based Hierarchical Multi-Label Classifier for Protein Function Prediction", in *Proc. of 2019 International Conference on Computer, Information and Telecommunication Systems (CITS 2019) (Beijing)*, pp.1-5, August, 2019.

The materials in Chapter 3 appeared in

- [J3] X. Yuan, E. Pang, K. Lin and J. Hu. "Deep Protein Subcellular Localization Predictor Enhanced with Transfer Learning of GO Annotation", *IEEJ Trans. on Electrical and Electronics Engineering*, Vol. 16, No. 4, pp.559-567, Apr, 2021.

The materials in Chapter 4 have been presented in

- [J2.] X. Yuan, H. Deng and J. Hu, "Constructing a PPI Network Based on Deep Transfer Learning for Protein Complex Detection", *IEEJ Trans. on Electrical and Electronics Engineering*, Vol.17, No.3, pp.436-444, March, 2022.
- [P1] X. Yuan, H. Deng and J. Hu, "Deep Transfer Learning Based PPI Prediction for Protein Complex Detection", in *Proc. of 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2021) (Melbourne)*, pp.321-326, October, 2021.

Acknowledgements

I am deeply grateful to the following people who have supported me in my PhD experience at Waseda University.

Firstly, I would like to extend my sincere thanks to my supervisor, professor Jinglu Hu, for his invaluable advice, continuous support and patience during my PhD study. His immense knowledge and plentiful experience have encouraged in all the time of my academic study and daily life.

I am very grateful to Professor Jinglu Hu, Professor Fujimura Shigeru and Professor Yoshie Osamu who proofread all the dissertation. Many thanks are for their valuable comments.

I also would like to thank all the group members in Neurocomputing Lab at Waseda University. Many thanks to Dr. Weite Li, Dr. Bo Zhou, Dr. Peifeng Liang, Mr Hangyu Deng, for their constructive discussions in the start of my research. I would also like to acknowledge my classmates and other students, too numerous to name.

Finally, I dedicate this dissertation to my parents Yifen Lv and Zining Yuan for their constant support and unconditional love.

Kitakyushu, Japan

Xin Yuan

November 5th, 2021

Contents

List of Figures

List of Tables

Abbreviations

AE	Autoencoder
BF	Biological Process
CNN	Convolutional Neural Network
CC	Cellular Component
DAG	Direct Acyclic Graph
DCNN	Deep Convolutional Neural Network
FN	False Negatives
FP	False Positive
GO	Gene Ontology
HMC	Hierarchical Multilabel Classification
LMLP	Local Multilayer Perceptron
LSTM	Long Short-Term Memory
MAcc	Multilabel Accuracy
MF	Molecular Functions
MF1	Multilabel F1-score
MG	Multilabel G-means
MHME	Multi-Heads and Multi-Ends
MLP	Multilayer Perceptron
MPre	Multilabel Precision
MRe	Multilabel Recall
MSE	Mean Squared Error
NPPI	Negative Protein-protein Interaction
PCA	Principal Components Analysis
PPI	Protein-Protein Interaction

ReLU	Rectified Linear Unit
SL	Subcellular Localization
SVM	Support Vector Machine
SCE	Sigmoid Cross-Entropy
TN	True Negatives
TP	True Positive

Symbols

seq	protein sequence
t	appearance time vector of protein sequence
x	input vector of protein sequence
x^T	transpose
y_g	label vector of Gene Ontology
y_s	label vector of subcellular localization
y_p	label vector of protein complex
$e(x_i, x_j)$	interaction of protein x_i and x_j
N	number of protein samples
D	number of GO terms
q	level number of hierarchical GO terms
L	number of subcellular locations
k	number of protein complexes
I_i	input from the level i of HMC
Z_i	output from the level i of HMC
A_n	output from the hidden-layer of autoencoder
Φ	output from the deep CNN feature extractor
$\hat{Z}_n$	prediction of output
$a_n(\cdot)$	Autoencoder
$f_n(\cdot)$	classifier
$\phi(\cdot)$	deep CNN feature extractor
ε	loss function
$\Omega_n^{(1)}$	parameter of the multi-end part of the MHME CNN model

$\Omega^{(2)}$	parameter of the body part of the MHME CNN model
$\Omega_n^{(3)}$	parameter of the multi-head part of the MHME CNN model
$K^T(\cdot)$	kernel function in SVM
b	the bias of SVM
α	the weight of SVM

Chapter 1

Introduction

1.1 Proteins

Protein, one of the three essential substances of life, participates in most of the life activities. We briefly introduce proteins from their synthesis, structure and function.

1.1.1 Protein Biosynthesis

Protein biosynthesis is a step of gene expression, it occurs continuously in cells to maintain the normal life activities. As we known, cells are the smallest units of life. Different tissue cells that make up living organisms have different physiological functions. Gene is the basis unit that carries genetic information, it is a fragment from the sequence of Deoxyribonucleic Acid (DNA) or Ribonucleic Acid (RNA). These are thousands of gene fragments which expressed in specific tissue cells determine what the cell can do by synthesized different varieties or numbers of protein.

During gene expression, DNA first unwinds into two single strands, each stand is composed of four different bases (adenine [A], guanine [G], cytosine [C], and thymine [T]) and link in series. One DNA sequence *transcribes* into a messenger RNA (mRNA) sequence following the complementary base pairing rules (A pairs with U uracil, T pairs with A, C pairs with G, G pairs with C). In a eukaryote, this RNA sequence will export

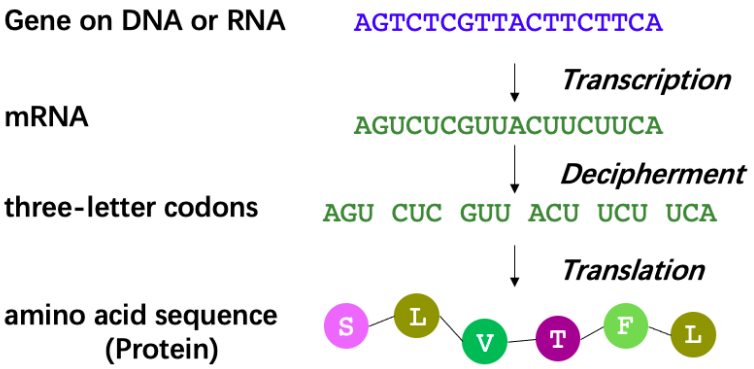


FIGURE 1.1: Protein biosynthesis.

TABLE 1.1: 20 Amino Acids and the corresponding RNA three-letter codons

Full name	Three-letter	One-letter	RNA codons					
Alanine	ala	A	GCU	GCC	GCA	GCG		
Arginine	arg	R	CGU	CGC	CGA	CGG	AGA	AGG
Asparagine	asn	N	AAU	AAC				
Aspartic acid	asp	D	GAU	GAC				
Cysteine	cys	C	UGU	UGC				
Glutamic acid	glu	E	GAA	GAG				
Glutamine	gln	Q	CAA	CAG				
Glycine	gly	G	GGU	GGC	GGA	GGG		
Histidine	his	H	CAU	CAC				
Isoleucine	ile	I	AUU	AUC	AUA			
Leucine	leu	L	UUA	UUG	CUU	CUC	CUA	CUG
lysine	lys	K	AAA	AAG				
Methionine	met	M	AUG					
Phenylalanine	phe	F	UUU	UUC				
Proline	pro	P	CCU	CCC	CCA	CCG		
Serine	ser	S	UCU	UCC	UCA	UCG	AGU	AGC
Threonine	thr	T	ACU	ACC	ACA	ACG		
Tryptophan	trp	W	UGG					
Tyrosine	tyr	Y	UAU	UAC				
Valine	val	V	GUU	GUC	GUA	GUG		

out of the nucleus to the cytoplasm, then decipher as a series of three-letter codons. And each codon corresponds to a particular amino acid [4, 5]. The process of protein biosynthesis shows in Figure 1.1. There are 20 common amino acids, the informations of corresponding RNA three-letter codons the amino acid names shows in Table 1.1.

1.1.2 Protein Structure

As mentioned previously, the 20 amino acids are translated from the mRNA and formed the protein molecules. And these big protein molecules could contain more than 2000

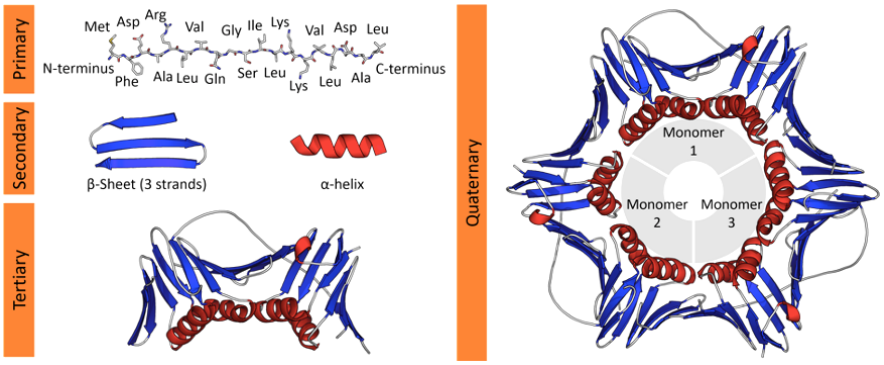


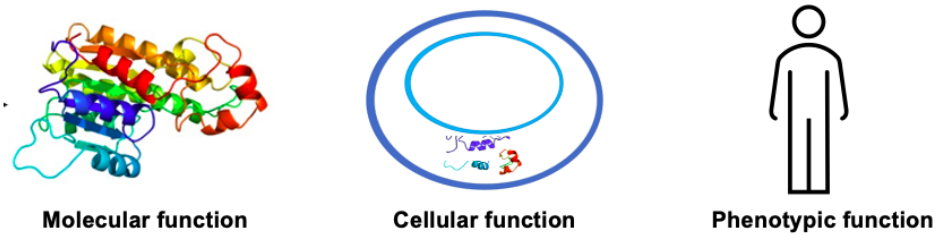
FIGURE 1.2: Four levels of protein structure.[1]

amino acids. Linked by chemical bonds, amino acids make up various protein structures, and these structures could help to understand the functions and other necessary information of proteins. The structures of protein are generally divided to four levels: the primary structure is the amino acid sequence; the secondary structure includes α-helix and β-sheet; the tertiary structure is the 3D structure of protein, and the quaternary structure is the protein complex [6]. The four levels of structures shown in Figure 1.2.

Protein structures could help to understand the function and other necessary information. Although the higher-level structures determines the functions easier, structure annotation is still an arduous task in protein bioinformatics field. Meanwhile, protein sequence analysis became popular as an approach to studying protein characteristics, function, and other bioinformatics. Due to the development of high-throughput [7] production methods for protein sequences, the speed of adding new sequences to the database has increased exponentially. Such a collection of sequences itself does not increase the biological understanding of researchers. However, comparing these new sequences with those functions annotated sequences is a meaningful way to study the biology of the organism from which the new sequences are derived, by the similarities between different sequences. The methods used include sequence alignment, searches for biological databases, and other methods.

1.1.3 Protein Functions

Protein function is the most important information for protein, since it is the sum of all the behaviors of proteins and the behaviors that happen through proteins [8, 9]. The main protein functions are known as: Antibodies, recognize antigens to help the immune system protect the body from viruses and bacteria. Enzymes, promote thousands of chemical reactions in cells [10]. Structural protein, confers stiffness and rigidity to otherwise-fluid biological components, such as hairs, nails, and some animal shells. Messenger proteins, transmit signals to coordinate biological processes between totally different cells, tissues, and organs, like some hormones. More importantly, DNA self-regulates its function also by adjusting the number and type of proteins it produces. So far, protein functions can be divided into three categories according to their properties:



- *Molecular function:* Biochemical functions of proteins are like connection, binding, catalytic biochemical reactions and conformational changes.
- *Cellular function:* Multiple proteins aggregate to perform complex physiological functions, such as the operation of metabolic pathways and signal transduction, to maintain the regular operation of various components of organisms.
- *Phenotypic function:* The integration of physiological subsystems composed of proteins that perform their cellular functions, and the interaction of the integrated system with environmental stimuli determine the phenotypic properties and behaviors of organisms.

Due to the wide range of protein functions, only a careful division can make each function work in biological research. Although the three categories of functions contain all

the behaviors of proteins, they are far away from a systematical, particular division of functions. The definition of protein function was a very subjective concept initially, and different researchers may have different representations of protein function. At the very beginning, natural language is used to label proteins [11]. Because it has great variability, the naming system is not suitable for analysis by human beings, much less computers. Therefore, the demand for standardized functional labeling schemes is crucial. Then Ouzounis et al. [12], Rison et al. [13], and the other works proposed many excellent classification schemes for standardized protein function annotation. Among them, the most popular schemes are those that are not designed for any particular species but are based on the general biological phenomena occurring in various species, including eukaryotes. Like FunCat [14], one of the frequently-used functional schemes, because it covers a wide range and the hierarchy is standardized. Gene ontology (GO) by Ashburner et al. [15, 16] is a recently proposed functional classification system based on reliable computer science and biological principles. It is quickly considered the most common solution for functional annotation technology across various biological data.

Nowadays, the knowledge of protein function is a critical link in many types of research, such as the development of pharmaceuticals and vaccines, prevention of complex diseases; better crops; and synthesizing biofuels or other biochemical products. However, two problems have to be solved before we apply protein function knowledge freely:

- Due to the complexity and quantity of protein functions, defining the functions of all proteins in organisms is one of the main objectives of biology in the present and future decades [17].
- More than 114 million proteins in NCBI data [18] are not identified with functions until now.

Therefore, protein function prediction has become one of the main tasks of bioinformatics, which can help solve a wide range of biological problems.

1.2 Protein Function Prediction

The research of various biological fields eventually constitutes the current status of protein function research, because the proteins have a large number of functions involving a wide range of life activities. Generally, the frequently-used function catalogues have been mentioned are the predicting tasks, like EcoCyc [19], FunCat or GO annotations. They investigate the functions though protein structure, sequences or the other features directly. These are still some related tasks. They are the bioinformatics point to functions, which could help to prediction, such as subcellular localization [20] for predicting cellular functions, protein-protein interaction [21] and ligand-protein interactions [22] for predicting molecular functions, and so on.

1.2.1 Prediction Tasks

In this dissertation, we investigate protein function prediction in three aspects: protein GO annotation prediction, localization prediction and interaction predictions.

1.2.1.1 GO Annotation

The GO (gene ontology) database was developed to systematically describe the functional properties of gene products (gene products include protein, complex, and RNA, this dissertation focuses on protein) across species and facilitate the computational prediction of gene function. It is a framework composed of controlled words describing the function of gene products in three aspects: Biological Process, Molecular Function, and Cell Component [23]. Biological Process (BP) describes biological goals accomplished by one or more ordered assemblies of molecular functions, like DNA repair or signal transduction. Molecular Function (MF) terms describe the activities that occur at the molecular level, such as catalysis or transport. Cellular Component (CC) describes locations at the levels of subcellular structures and macromolecular complexes, such as nucleus or organelles in cytoplasm [16].

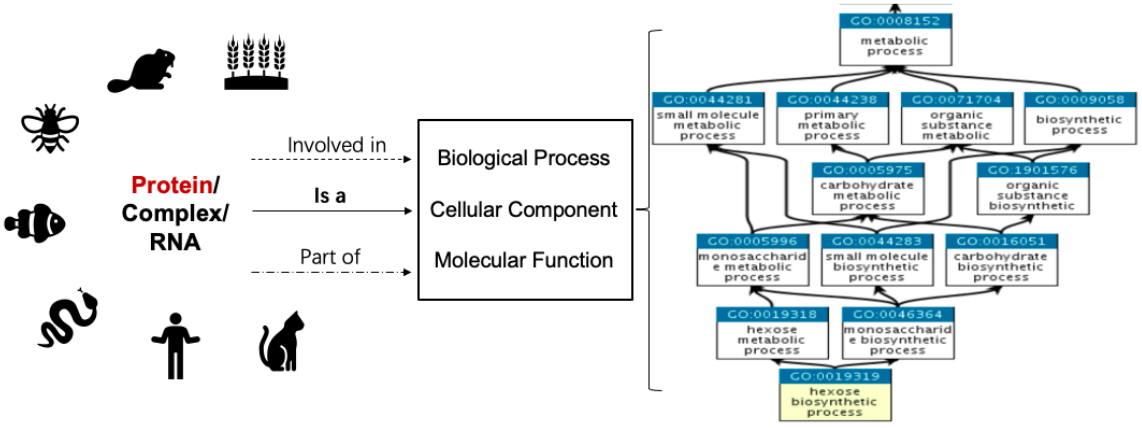


FIGURE 1.3: Hierarchical GO terms [2] describing the functional properties of gene products across species.

The structure of GO terms can be described by a directed acyclic graph (DAG) shown in the right part of Figure1.3, where each GO term is a node, and the relationship between terms is the edge between nodes. GO has a loose hierarchy, and the term 'child' is more specialized than the term 'parent'. However, unlike the strict hierarchy, a term may have multiple 'parent' terms. In addition, terms are structured to support the classification of 'is-a' , 'involved in' and 'part-of' relationships. There are more than 44,000 GO terms until now, and each protein is annotated by at least one term. Therefore, the protein GO annotation is a very complicated multi-label classification problem. In this dissertation, we will discuss the prediction of a part of GO annotations, there are about 4000 terms, and they are high-frequently annotated in proteins.

1.2.1.2 Protein Subcellular Localization

Protein function is determined by subcellular localization because the organelles provide different chemical environments and interaction partners. The locations are relative to cellular structures in which a gene product performs a function, either cellular compartments (e.g., mitochondrion), or stable macromolecular complexes of which they are parts (e.g., the ribosome). Therefore, many biological processes involve changes in protein subcellular localization to regulate protein activity [24]. Subcellular localization as an essential study area in molecular cell biology, it is closely related to protein function, metabolic pathway, signal transduction and biological process. Protein cell

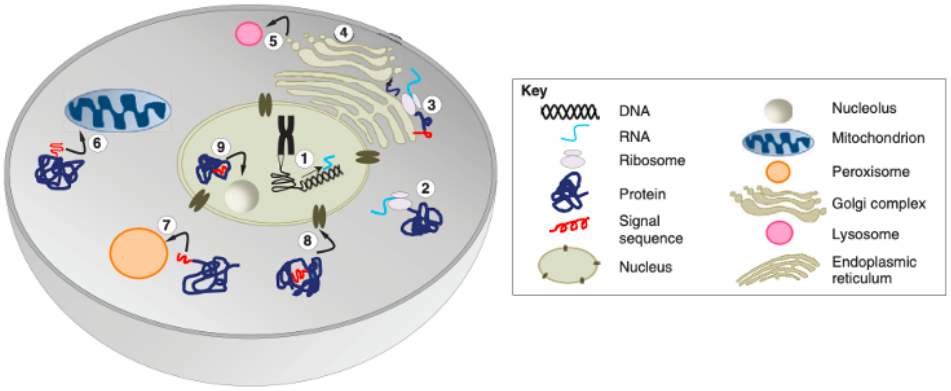


FIGURE 1.4: Proteins in the organelles. [3]

location information plays an important role in drug discovery, drug design, basic biological research and biomedicine research [25]. Like the studies of [26] visualized the subcellular locations of SARS-CoV-2 proteins (The proteins of COVID-19) and might provide novel insights into the viral proteins' biological functions.

Most of the biological activities of proteins take place in the organelles. The mark of eukaryotic cells is that they are divided into different membrane-bound organelles. There are 14 subcellular locations are usually discussed in eukaryotic subcelullar localization prediction: 'Membrane', 'Nucleus', 'Cytoplasm', 'Cell membrane', 'Endoplasmic reticulum', 'Golgi apparatus', 'Mitochondrion', 'Secreted', 'Chromosome', 'Plastid', 'Peroxisome', 'Lysosome', 'cytoskeleton' and 'endosome'. Figure 1.4 shows proteins works in the eukaryote. The same protein usually appears in different subcellular locations, representing different functional properties, so subcellular location prediction is also a multilabel classification problem.

1.2.1.3 Protein-Protein Interaction & Protein Complex

Proteins rarely act alone because their functions tend to regulate each other. Protein-protein interaction (PPI) is a particular physical or genetic contact between two or more protein molecules. Many of these interactions are related to the protein functions in the cell or specific biomolecular environment. On the other hand, proteins always play their functions in groups, so the detection of PPI and protein complexes is also a part of protein function prediction[27]. Figure 1.5 shows the proteins, PPI and protein complex.

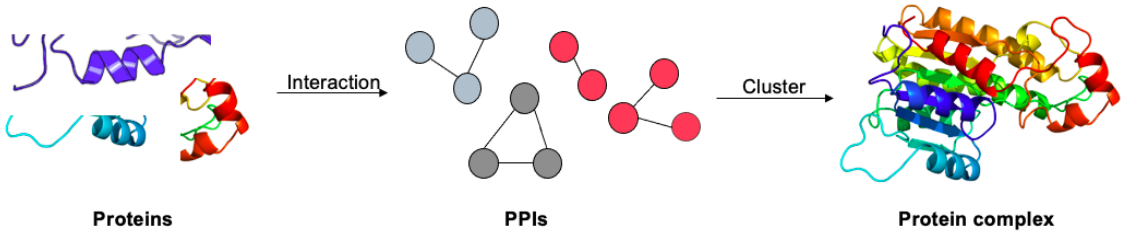


FIGURE 1.5: Protein, PPI and protein complex.

Many important life activities are inseparable from PPI, including DNA replication and transcription, protein synthesis and secretion, signal transduction and metabolism, etc. Abnormal PPI is the basis of many cluster-related diseases, such as Creutzfeldt-Jakob and Alzheimer's disease[28].

Protein-protein interactions forms a PPI network essential for many protein studies. Seattle Center for Structural Genomics [29] Infectious disease scientists model more than 1,000 complex proteins to help develop treatment methods for infectious diseases. However, these centrality calculation methods have defects, because the proportion of false positive and false negative data in the protein-protein interaction network is very high. More importantly, these methods ignore the intrinsic biological significance of essential proteins. Therefore, high-throughput calculation method is needed to identify PPI with high quality and accuracy. Predicting the protein pairs are interacted or not is a binary classification problem.

1.2.2 Prediction Approaches

This part introduces the main approaches of function predictions: from experimental identification to computational approach. Protein sequence analysis as the most popular part of computational approach will be introduced individually.

1.2.2.1 Experimental Identification

Earlier, protein function predictions were entire experimental identification. They focused on a specific target gene or protein or a small set of proteins forming natural

groups such as protein complexes. These approaches included gene knockout, targeted mutations and the inhibition of gene expression [30]. Experimental identifications cost substantial financial resources, however, even large-scale experimental annotation programs like the EUROFAN project [31] are not sufficient to annotate important parts of proteins that have become available due to the rapid development of genome sequencing technologies.

Structural Genomics Project [32] determined the sequences of many proteins by the high-throughput method in genome-wide strategies. Compared with the experiments of sequencing proteins, the experimental procedure of protein bioinformatic annotation is essentially low throughput. This led to a widening sequence-function gap in the proteins found. Figure 1.6 shows the growing gap of protein sequences and GO function annotations from UniProt [33] and Gene Ontology Consortium [2] in these 20 years.

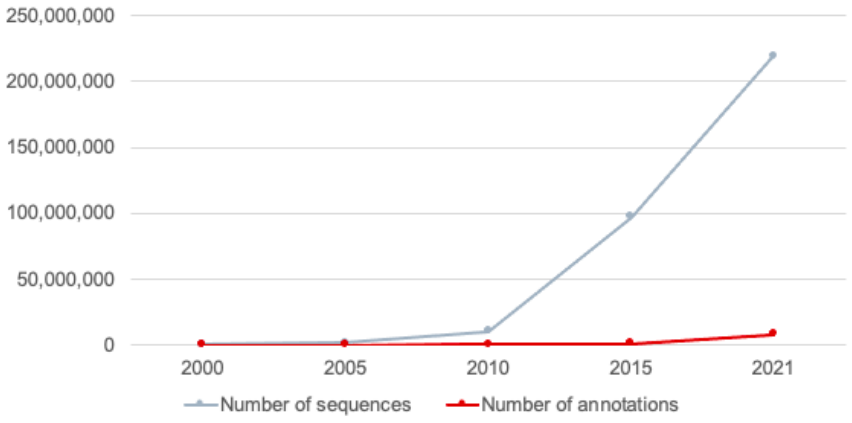


FIGURE 1.6: The numbers of protein sequence and annotation.

Thus, developing computational approaches that analyze simple features, like protein sequences, can have enough discrimination to build practical algorithms that are expected to solve the problems in function prediction.

1.2.2.2 Computational Approach

The existing computational approach methods used to predict protein function from protein sequence, micro-array expression [34], protein families [35], especial protein structure [36]. Jacob et al. [22] propose a systematic method to predict ligand-protein

interactions, even for targets with no known 3D structure and few or no known ligands. Besides sequences, however, the other features are difficult to obtain or unavailable for most of proteins.

Although protein structure information is essential for understanding the function, the speed of protein structure determination lags far behind the increase of sequences due to the technical difficulties and laborious nature of structural biology experiments. On the one hand, the high-throughput sequencing technique makes protein sequences cheaply available, and many computational models are based on protein primary sequences only in computational proteomics. On the other hand, data integration has become a popular method to integrate diverse biological data. Thus, developing efficient algorithms that can predict protein functions by primary sequence-based data is probably the avenue to fill the gap. More and more algorithms are used to predict functions from the protein sequences. For this reason, some recent predictive models are deliberately designed to integrate multiple heterogeneous data sources for exploiting multi-aspect protein feature information. However, protein sequences as one of the main data sources, the algorithms that focus on protein sequence analysis have been paid more attention.

1.2.2.3 Protein Sequence Analysis

In the domain of automated prediction, sequences have been heavily utilized in both direct homology-based and indirect subsequence-and feature-based approaches. Specifically, techniques that predict protein function, localization, and interaction from sequences can be categorized into three classes, namely, sequence homology-based approaches, subsequence-based approaches and feature-based approaches, which are explained below:

- *Homology-based approaches*

  Protein bioinformatics commonly used tools, like BLAST [37, 38], FASTA [39], which are related to searching sequence databases by sequence similarity. Based on the biological rationale for homology transfer, If two sequences are highly similar, they evolved from a common ancestor and had similar annotations [40]. With

the increase in the number of sequences in the database, however, homology-based transfer fails in three aspects: high sequence similarity can be wrong; sequence-based tools are not sensitive enough; propagation of incorrect annotation in the whole database [41, 42].

- *Subsequence-based Approaches*

  Many studies have shown that it's not usually the entire sequence but rather specific segments of it that are important for determining the function of a given protein. Therefore, the method in this category regards these fragments or subsequences as the features of protein sequences and constructs a model to map these features to protein functions. Many approaches based on this idea have been proposed, starting with the methods of Hannenhalli et al [43] which tried to identify regions of a sequence that best distinguish a certain function or sub-type.

- *Feature-based Approaches*

  Homology-based and subsequence-based approaches discussed above predict protein function from original sequences, namely, as strings. However, these sequences can be transformed into more biologically meaningful features, making it easier to divide proteins with different functional categories. This is the viewpoint of feature-based methods. It uses the standard classification algorithm to learn the functional class model from the transformed feature set and then uses the model to predict uncharacterized proteins. These methods attempt to take advantage of the view that amino acid sequences are unique proteins and identify several of their physical and functional characteristics. These features are used to construct a prediction model, which can map the eigenvalue vector of the query protein to its function.

Our research focus on the feature-based approaches combining with machine learning method, and we will discuss these approaches in the following section particularly.

1.3 Related Works

In the era of big data, it becomes more and more critical to transform massive data into valuable knowledge in various fields [44], and bioinformatics is no exception. In order to extract knowledge from big data of bioinformatics, machine learning has become widely used and achieved success. Machine learning algorithms train data to discover potential patterns, build models and predict based on the best fitting model. For example, State-of-the-arts protein function predictors have used some well-known algorithms, like, support vector machine (SVM) [45, 46], decision tree [47], hidden Markov model [48], clustering[49].

Machine learning methods for predicting functions involve two major aspects: to derive protein features and to design a predictive model. Deep learning, as a branch of machine learning, can independently analyze features from big data and has been successful in many fields. However, there is still much room for the development of deep models for protein detection. This section will introduce some effective algorithms applied to GO annotation, Localization, and interaction predictions.

1.3.1 Sequence Encoding Methods

Vectorized protein sequence is the first step of protein sequence analysis by machine learning methods. Considering that a protein sequence consists of 20 different amino acids, which is similar to a natural language [50]. There are some frequently-used way for encoding sequences:

- The n-gram algorithm utilized to encode proteins is defined as the appearance frequency of n consecutive amino acids [51, 52]. The n-gram method generated from natural language processing is now developed in the bioinformatics field.
- Protein sequences also have been represented as one-hot encoding [53] in units of the 20 amino acids with the sequence length of 1000 (input: $20 * 1000$, the work ignored the sequences larger than 1000).

- Amino acids background frequency (AABF) [54] is utilized for presenting the biological feature of a protein by calculating the frequency of each amino acid in the whole sequence, and combined them to be a frequency vector [55].

Comparing with the other two methods, n-gram encoding is the most popular method because it takes account of the entire sequences with units of amino acids, dipeptides or even more.

1.3.2 Hierarchical Multilabel Classification

As mentioned in Section 1.1.1, GO annotation prediction is a multi-label classification problem which has complex functional tags that make up an undirected graph. Many conventional methods have tried to solve it with hierarchical multilabel classification (HMC). HMC is a classification task where the classes to be predicted are hierarchically organized. Each instance can be assigned to classes belonging to more than one path in the hierarchy.

In the work of Cerri et al. [56, 57], a preliminary HMC-LMLP was built, where a multilayer perceptron (MLP) for each level was associated together. For the first level, they used the instances as input; and from the second level onwards, each MLP was fed only with the output provided by the previous MLP. However, there is a problem in the series of HMC-LMLP methods, the structure of MLP is too simple to address this issues. GO annotation, as a very complex hierarchical multilabels classification problem with thousands of classes, applying deep learning models to extract features would be more reasonable.

1.3.3 Deep Learning Methods

Due to the lack of annotated protein sequences, a good sequence analysis model becomes essential. In this way, a deep learning model which could extract better features is needed. With the vectorized protein sequence, a feature extractor is usually needed to extract and map the feature to space where it is more linearly separable. A neural

network, especially deep convolutional neural network (CNN), can be used as such a feature extractor.

The DeepGO model [58] is an effective deep learning model for GO annotation. 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 tripeptide units, and a single layer CNN model is used as a feature extractor. After that, it is improved by the work of DeepGOPlus [59]. Compared with DeepGO, the one CNN layer with fixed a filter length which was extended to several CNN layers with different filter lengths. DeepGOPlus also used a flat classification layer instead of hierarchical classifier in DeepGO. The method of *CONV A-BLSTM* [53]achieve a success in subcellular localization prediction. The authors investigated protein sequences dealing with some deep learning models. The method consists of four parts, one layer of CNN, one hidden state layer and one attention decoding layer of long short-term memory (LSTM), one dense layer of the fully-connected network and the predictor of hierarchical tree likelihood. And the study of Sun et al.[60] built a deep predictor which consist a stacked autoencoder with a SVM classifier on sequence-based feature.

Although the existing methods used some deep learning techniques for protein sequence analysis like CNN, RNN, etc., shallow or even single layer networks are unable to map protein features into separable spaces. Compared with the deep models of the mature fields, the algorithms of protein function prediction have much room for improvement. Admittedly, the deep model cannot be applied to the hierarchical multi-label model, since the situation will make the computational burden problems of hierarchical classification even worse.

1.3.4 Data Integration Methods

More and more researches proved that predicting function using sequence-based feature alone is imperfect.

Many predictive models are deliberately designed to integrate heterogeneous data sources [61, 62]. As the entire function category across species, Gene Ontology uses a controlled

vocabulary to depict biological molecules or gene products in terms of biological process, molecular function, and cellular component. On the other hand, with the rapid expansion of annotated proteins, Gene ontology has been shown to help improve the prediction accuracy of protein localization and interaction [63, 64]. Figure 1.7 shows the scope of Localization, PPI and complex described in the GO function catalog. The blue shadow covers the function of describing Localization, and the green shadow covers the function of describing PPI and Complex.

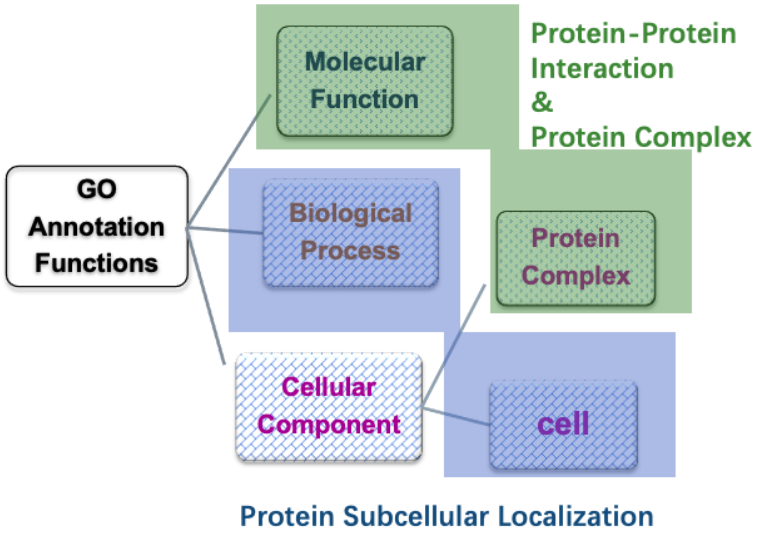


FIGURE 1.7: The scopes of the three special functions described in the GO catalog respectively.

The subcellular localization prediction methods achieved better performance by protein sequence and GO annotations [65, 66]. X. Cheng et al. (2018) [67] and K.C. Chou et al.(2001) [68] investigated the cases like the proteins of human beings, fungus where GO annotation is available by applying both sequence-based and annotation-based feature for localization prediction and found that it is better than using only sequence feature. Furthermore, Gene ontology as the well-organized biological knowledge also participates in subcellular localization [69, 70]. For PPI prediction, similar GO functions can confirm interaction, and different subcellular localizations are non-interactive [71]. Gavin et al.(2006) [72] suggests that a protein complex consists of two parts, protein-complex core, and attachments. Proteins in a core generally are highly co-expressed and share a high functional similarity. Since proteins in the same complex are highly interactive with each other, protein complexes generally correspond to dense subgraphs

in the PPI network [73, 74, 75] and many previous studies have been proposed based on this observation.

Multiple heterogeneous data sources increases the prior knowledge for prediction, while in the same-time, increases the difficulties of feature extraction and selection. However, as mentioned in the previous section, the deep learning methods for protein function prediction are not yet able to deal with complex features. Furthermore, the experimentally identified annotations are not always available, there are only some type species have complete annotations.

1.4 Challenges

These years, the superiority of deep learning in data analysis fields has been proved, such as in image processing, semantic segmentation. Compared with these mature fields, the deep learning methods of protein sequence analysis has not obtained a significant improvement based on the related works. Even if the number of protein sequencing has already reached a certain scale. That is because the following challenges need to be faced:

- *Prediction tasks are complicated:*
  Function predictions are usually complicated multi-label classification problems. Especially GO annotation prediction has thousands of classes.
  Solution: Building a deep learning model to deal with these kind of problem is easy to considered.

- *Training samples are limited:*
  Training an efficient deep model need large numbers of training samples. Actually, most of the proteins don't have experimental annotated function labels, only a very little part samples from some type species have been annotated.
  Solution: It may be natural to consider employing transfer learning in the cases when training samples is limited.

- *Transfer learning is hard to implement:*
  However, transfer learning of protein data is much harder than the other mature fields. Protein function prediction has too many tasks and each task has very few instances. That makes the implementing model is easy to overfit and hard to extract common features.

We can see that, how to build an efficient deep learning model to implement transfer learning for function prediction is the biggest problem.

1.5 Motivation and Formulation

Unlike most of the existing methods only applied the normal machine learning models for prediction, we propose that developing a deep learning model which focused on function prediction via protein sequence analysis. This dissertation develops a novel deep convolution neural network (CNN) model with multi-head and multi-end (MHME) to implement a class of classifiers in one model. The deep CNN feature extractor plays two roles: extracting features from the vectorized protein sequence and mapping the feature onto a feature space where it is more linearly separable. Thus, this deep CNN model with MHME shares this deep CNN feature extractor in a class of different prediction tasks and makes the transfer learning possible. This novel model is employed to the three function prediction tasks, protein GO annotation, localization and interaction predictions, based on a transfer learning from different tasks and different species.

This novel deep model is defined by

$$y_i = f_n(\phi(x_i)) \tag{1.1}$$

where x_i is the input, the vectorized protein sequences calculated by a '1-gram + 2-gram' encoding method which shows in the following; y_i is the output, the function labels of x_i, and $f_n(\phi(\cdot))$ represents the predictors f_n for different tasks with a sharing deep feature extractor ϕ.

This dissertation uses a '1-gram + 2-gram' encoding method to obtain an appearance frequency vector as the input of the machine learning model. The encoding method encodes the protein sequences of amino acids seq_i $(i = 1, ..., N)$ into appearance frequency vectors, x_i $(i = 1, ..., N)$, which are column vectors with a length of 420, the number of 20 amino acids and 400 dipeptides.

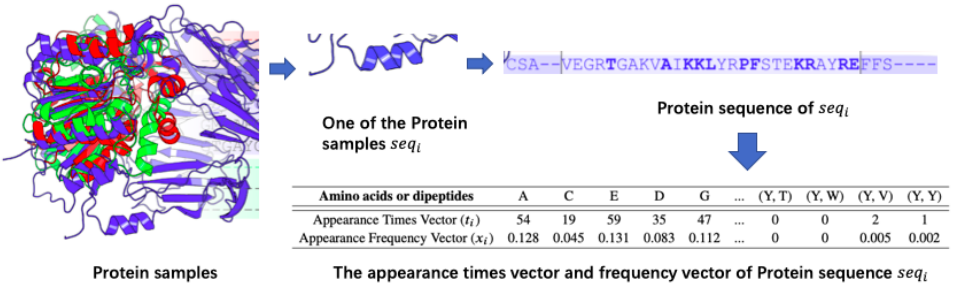


FIGURE 1.8: An image of '1-gram + 2-gram' encoding based feature extraction of protein sequences.

Figure 1.8 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

$$x_i = \frac{t_i}{\dim(x_i)} \tag{1.2}$$

where $\dim(x_i) = 20^1 + 20^2 = 420$. x_i $(i = 1, ..., N)$ is the vectorized sequence, where N is the number of protein samples.

1.6 Thesis Outlines and Main Contributions

This dissertation presents my cumulative works over my doctor career through five chapters. Chapter 1 shows the background materials, authors' proposals and a dissertation outline. Chapter 2 develops a deep CNN model with Multi-head and Multi-end to realize a deep hierarchical multi-classifier for predicting protein GO annotation. Chapter 3 proposes a new protein subcellular localization predictor which is enhanced by

transfer learning of GO annotation. Chapter 4 detects protein complex from a reconstructing PPI network which is built via a transfer learning PPI predictor. Chapter 5 summarizes the whole dissertation and discusses the potential applications and future study. The flow of this dissertation is depicted in Fig.1.9, in which the indexes represent the published paper in Publication List.

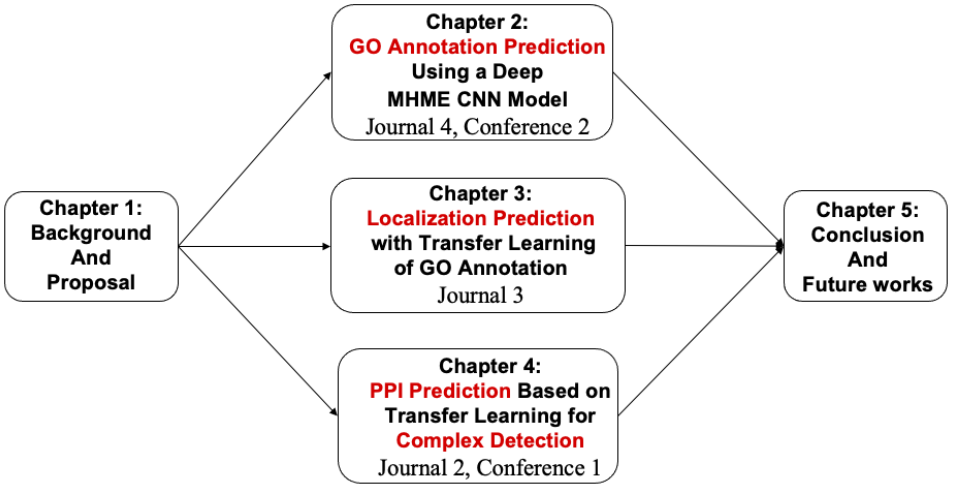


FIGURE 1.9: Flow diagram of this dissertation

Chapter 2 proposes a novel hierarchical multi-label classifier implementing the 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 multi-end part of a set of autoencoders performing feature fusion transforming the input vectors of different local classifiers to feature vectors with the same length to share the feature mapping part; and the multi-head part of a set of linear multi-label 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 model with multiple local classifiers, we can construct more powerful local classifiers for each level with limited training samples and realize a deep transfer learning from different species to achieve better classification performance.

The main contributions related to the deep MHME CNN model are shown as follows:

- This proposed deep CNN feature extractor maps the feature onto a feature space where it is more linear separable.
- Multi-head and multi-end parts realize the local classifiers and a global classifier to capture the two-way relationship of hierarchy GO annotations.

Chapter 3 proposes a deep protein subcellular localization predictor which is enhanced in a transfer learning way. The deep predictor consists a linear classifier and a deep feature extractor of a convolution neural network. GO annotations have been proved that it is helpful in subcellular localization prediction. So, the deep CNN feature extractor is first pretrained by a deep GO annotation predictor which realized by the deep MHME CNN model. It is then transferred to the subcellular localization predictor with fine-tuning using protein localization samples. In this way, we have a deep protein subcellular localization predictor enhanced with transfer learning of GO annotation.

The main contributions related to this enhanced deep predictor are shown as follows:

- A deep localization predictor is trained successfully with a large set of experimentally annotated proteins which we collect from various species.
- Pretraining the deep CNN model by the GO annotation predictor is to transfer the common feature from different tasks and species, which improves the performance of localization prediction.

Chapter 4 introduces a new method of detecting protein complexes from a reconstructed PPI network using spectral clustering. In order to complete the raw PPI network, we propose a deep PPI predictor consisting of a semi-supervised SVM classifier and a deep CNN feature extractor to predict the unknown PPIs. This deep CNN feature extractor is enhanced by a deep GO & localization annotation predictor in a transfer learning way by the deep MHME CNN model, because the similarities of GO annotations contribute to protein interactions and the difference in subcellular localizations contribute to negative interactions. In this way, we got a deep PPI detector enhanced with transfer features from different tasks and different species.

The main contributions related to this transfer learning model for PPI prediction and protein complex detection are shown as follows:

- We develop a deep PPI predictor which contains a transfer DCNN feature extractor and an inductive semi-supervised SVM classifier.
- Deep CNN feature extractor learns the knowledge from GO and localization annotation tasks and protein from type species in a transfer learning way.
- Spectral clustering is applied to detect complex from the reconstructed PPI network and achieve a better performance.

Chapter 5 concludes this work, summarizes the dissertation and gives suggestions for further research.

Chapter 2

GO Annotation Prediction Using a Deep MHME CNN Model

2.1 Background

[1] Chapter 2 investigates protein function by gene ontology (GO) annotation, the most commonly used function catalog system. As gene ontology 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 [76]. However, with the GO functions being enriched, the number of GO functions that could be annotated is more than 40 thousand. Therefore, entirely experimental protein annotation becomes time-consuming and cost-consuming; automatic prediction with the help of a machine learning method is widely considered more practical. Compared to the biological experiment methods, the machine learning method is more economical and convenient.

The automatic GO annotation method is a process to predict the GO functions (classes) as a multilabel classification task [77] by checking the combination of proteins. It could be impossible to address this issue with a standard classification method, however, since the GO annotation classes are very complicated. Not only the number of GO functions is

[1] This chapter mainly extends the Journal paper: X.Yuan, E.Pang, K.Lin and J.Hu. "Hierarchical Multilabel Classification for Gene Ontology Annotation using Multi-head and Multi-end Deep CNN model", *IEEJ Trans. on Electrical and Electronics Engineering*, Vol.15, No.7 pp.1057-1064, July, 2020.

huge, but also the functions are arranged in a hierarchy: typically a direct acyclic graph (DAG) [78] 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 [79]. Hierarchical multilabel classification is one kind of algorithms known to have better performance [80], since it is a method ranking labels into a hierarchical structure and performing classification on each level independently.

Hierarchical multilabel classification (HMC) [81, 82, 83], is a kind of method which separates 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 [84, 85, 86] trains the hierarchical structure network from the higher level to the lower level. The global approach uses a classification algorithm that learns a global classifier about the whole classes. For example, the authors of Ref. [87] 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 applied to solve protein function prediction, as it is used augmented versions of dealing with the problem which has hierarchical classes [88].

With a local approach, R. Cerri et al. [56, 57, 89, 90] applied the hierarchical multilabel classification to the protein function prediction problem. 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 hierarchical multilabel classification with local MLP (HMC-LMLP). In this HMC-LMLP method, MLP predictions at a given level are used as inputs to the neural network responsible for the prediction in the next level. These methods focused on optimizing the model by training it level-by-level and propagating the positive signal from the previous output. Although the HMC-LMLP achieves state-of-the-art performance, it is difficult to construct more powerful local classifiers due to the limited 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 chapter, 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 multi-end part, and the multi-head part. The body part is a deep CNN model shared by different local classifiers for feature extraction and mapping. The multi-end 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 to share the feature mapping of the deep CNN model. Finally, the multi-head part is a set of linear multilabel classifiers realizing the multilabel protein function prediction. Furthermore, we design a sophisticated recursive algorithm that trains the MHME CNN model to perform 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.

The rest of chapter is organized as follows. Section 2.2 introduces the '1-gram + 2-gram' encoding method and formulates local classifiers for each level. Section 2.3 describes the details of the MHME CNN model and the training algorithm. Section 2.4 introduce the training process of the proposed model. Section 2.5 carries out experiments on various benchmark datasets and compares the proposed MHME CNN model with the related existing methods. Finally, Section 2.6 has a summary showing the conclusions of this chaper.

2.2 Problem Formulation

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

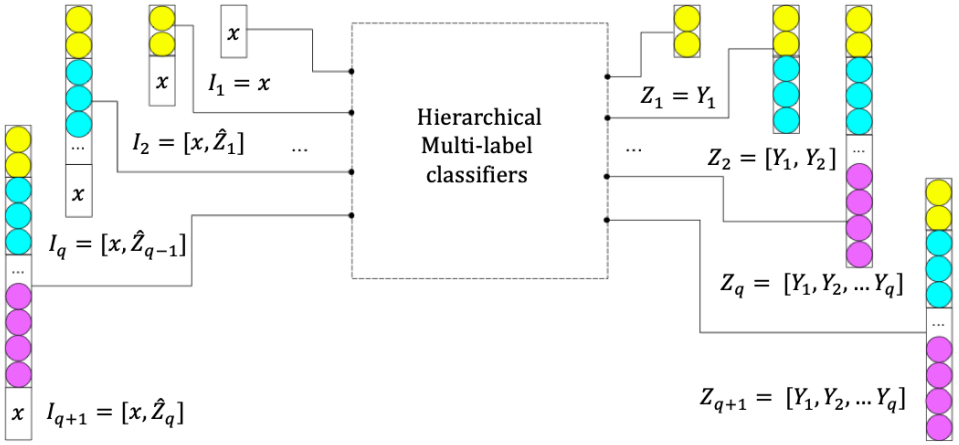


FIGURE 2.1: The formulation of HMC with local classifers.

2.2.1 The Notation and the Problem

Suppose we denote the protein instances by $\mathbf{S} = \{seq_1, seq_2, seq_3, ...seq_i | i = N\}$, N is the number of instance, and seq_i represent the protein sequence of instance i. Protein functions are denoted by $\mathbf{G} = \{g_1, g_2, g_3, ...g_j | j = G\}$, G is the category number of functions. Here we use a x_i to represent the vectorized seq_i by a '1-gram + 2-gram' encoding introduced in Section 1.5 Different from the normal binary classification and multiclass classification, in hierarchical classification databases used $D = \{(x_i, y_i), 1 \leq i \leq N\}$ to express the instance in different levels, if x_i is belong to function g_j, then $g_j = 1$, or $g_j = 0$. We use $y_n = \{g_1, g_2, g_3, g_4, ...g_j | j = G\}$ to express the label matrix. At label space, we assume that the number of level of the hierarchy classes is q, expressed as $\mathbf{Y} = \{Y_{g1}, Y_{g2}, Y_{g3}, ...Y_{gn} | n = q\}$. For label in level q is that $Y_{gq} = \{g_{q1}, g_{q2}, ...g_{qn} | g_{qn} \in \mathbf{G}\}$.

2.2.2 Local Classifiers for Each Level

Figure 2.1 shows the formulation of HMC with local classifiers. The local classifiers are designed to be 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 $Z_{g1} = Y_{g1}$. The local classifier for level 2 (classifier 2) has the input vector of $I_2 = [x, \hat{Y}_{g1}] = [x, \hat{Z}_{g1}]$, combined x with the output vector of classifier 1, and the output vector of $Z_2 = [Y_{g1}, Y_{g2}]$. In this way, the local classifier for level n ($n = 1, ..., q$) (classifier n) will have the input vector of

$I_n = [x, \hat{Y}_{g1}, \hat{Y}_{g2}, ..., \hat{Y}_{gn-1}] = [x, \hat{Z}_{gn-1}]$, combined x with the output vector of classifier $n-1$, and the output vector of $Z_{gn} = [Y_{g1}, Y_{g2}, ..., Y_{gn}]$. Traditionally, the classifiers 1 to n are implemented by using simple MLP models due to the limitation of training samples in each level.

2.2.3 An Additional Global Classifier

There exists not only a top-down relationship of labels between different levels but also a 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 $I_{q+1} = [x, \hat{Z}_{gq}]$, combined x with the output vector of the last level classifier q, and the output vector of$Z_{gq+1} = [Y_{g1}, Y_{g2}, ..., Y_{gq}]$.[2]

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

$$Z_{gn} = F_n(I_n), \quad n = 1, ..., q+1 \tag{2.1}$$

where $I_1 = x$ and $I_n = [x, \hat{Z}_{gn-1}]$. In the next section, we will design a deep CNN model with multiple heads and multiple ends 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.

2.3 The Deep MHME CNN Model

Figure 2.2 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

[2] $\hat{Z}_{g1}, \hat{Z}_{g2}, ..., \hat{Z}_{gq}$ are the prediction values of $Z_{g1}, Z_{g2}, ..., Z_{gq}$ by the local classifiers.

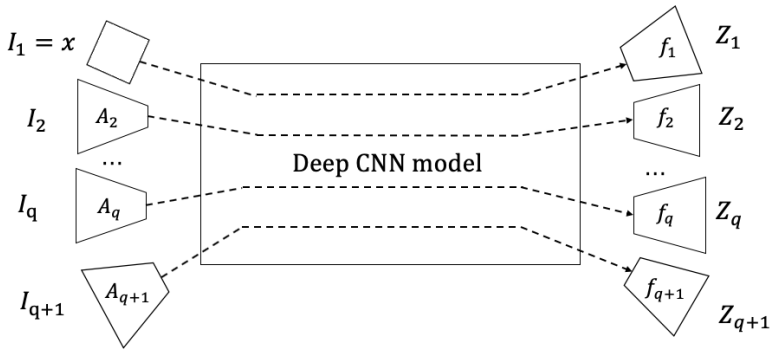


FIGURE 2.2: The structure of the multi-head and multi-end deep CNN model.

CNN model. The MHME CNN model consists of three parts: the body part, the multi-end part and the multi-head part, which realizes the $q+1$ classifiers with sharing the body part, referred to Figure 2.2.

$$Z_{gn} = f_n\left(\Omega_n^{(3)}, \phi\left(\Omega^{(2)}, A_n\left(\Omega_n^{(1)}, I_n\right)\right)\right), \tag{2.2}$$

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

2.3.1 The Multi-End Part: A Set of Autoencoders

The multi-end 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. Therefore, to share the body part for feature extraction, these input vectors should be transformed into a set of feature vectors with the same length at the specific feature space.

$$\begin{cases} A_1 = x \\ A_n = a_n(\Omega_n^{(1)}, I_n), \quad n = 2, ..., q+1 \end{cases} \tag{2.3}$$

where A_n is the outputs of multi-end part. For simplicity, we set $\dim(A_n) = \dim(x)$.

Due to no teacher signals for A_n, we apply autoencoders [91], which realizes the transforming of Eq.(2.3) by minimizing the information loss. As shown in Figure 2.3, an

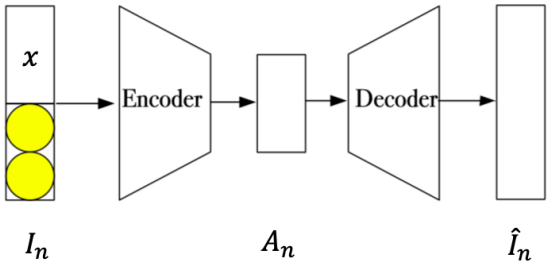


FIGURE 2.3: The structure of autoencoder a_n.

autoencoder neural network is an unsupervised learning algorithm that applies back-propagation [92] by setting the target values equal to the inputs. The general idea of the 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 the original autoencoder have drawn increasing attention as a compelling feature extractor or descriptor [93]. The autoencoder plays two roles: feature fusion and dimension reduction.

2.3.2 The Body Part: A Deep CNN Model

The rectangle in the middle of Figure 2.2 represents the body part, which realizes a feature mapping or feature extraction. CNN outperforms in dealing with spatial information like graphs, texts, etc. Recently, it has also been shown to be effective in extracting features from protein sequences in some related work [58, 94]. The proteins are described as the frequency of amino acids and dipeptides to demonstrate the ordered arrangement of protein sequences. Moreover, the GO terms (predictions of linear classifiers, mentioned in the second section) added in input during the training make the features very high-dimensional. Compared with the one-layer CNN model used in Ref. [58], a deep CNN model is more reasonable for our purposes of feature extraction and feature mapping. The structure of the deep CNN model in the experiments will describe in Subsection2.5.2.1.

$$\Phi_n = \phi(\Omega^{(2)}, A_n), \quad n = 1, ..., q+1 \tag{2.4}$$

where ϕ_n is the output of the body part.

2.3.3 The Multi-Head Part: A Set of Linear Classifiers

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

$$Z_{gn} = f_n(\Omega_n^{(3)}, \Phi_n), \quad n = 1, ..., q+1 \tag{2.5}$$

2.4 Training Method

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

$$\begin{aligned} Z_{g1} &= f_1\left(\Omega_1^{(3)}, \phi\left(\Omega^{(2)}, a_1\left(\Omega_1^{(1)}, I_1\right)\right)\right), \; I_1 = x \\ Z_{g2} &= f_2\left(\Omega_2^{(3)}, \phi\left(\Omega^{(2)}, a_2\left(\Omega_2^{(1)}, I_2\right)\right)\right), \; I_2 = [x, Z_{g1}] \\ &\cdots \\ Z_{gq+1} &= f_{q+1}\left(\Omega_{q+1}^{(3)}, \phi\left(\Omega^{(2)}, a_{q+1}\left(\Omega_{q+1}^{(1)}, I_{q+1}\right)\right)\right), \\ & \quad I_{q+1} = [x, Z_{gq}]. \end{aligned} \tag{2.6}$$

1) Training of Parameters $\Omega_n^{(1)}$

$\Omega_n^{(1)}$ $(n = 1, ..., q+1)$ are the parameters of autoencoders. As shown in Figure 2.3, they are trained in an unsupervised manner by optimizing the mean squared error (MSE) loss function $E_n = \varepsilon_{MSE}(\hat{I}_n, I_n)$. Let z be the input of encoder and $\hat{z}$ be the output of decoder. Then ε_{MSE} is defined by

$$\varepsilon_{MSE}(\hat{z}, z) = \frac{1}{N} \sum_{i=1}^{N} (\hat{z}_i - z_i)^2 \tag{2.7}$$

where N is the number of samples.

2) Training of Parameters $\Omega^{(2)}$ and $\Omega_n^{(3)}$

$\Omega^{(2)}$ and $\Omega_n^{(3)}$ $(n = 1, ..., q+1)$ 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 $E = \sum_{i=1}^{n} \varepsilon_{SCE}(\hat{Z}_{gn}, Z_{gn})$. In this formulation, ε_{SCE} is defined by

$$\begin{aligned} \varepsilon_{SCE}(\hat{z}, z) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C_n} [z_{ij} \times log(\hat{z}_{ij}) \\ + (1 - z_{ij}) \times log(1 - \hat{z}_{ij})] \end{aligned} \tag{2.8}$$

and $C_n = \dim(Z_{gn})$.

However, as we can see from Eq.(2.6) 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 $\Omega_n^{(3)}$ $(n = 1, ..., q+1)$. The training algorithm consists of the following three steps.

- *Step* 1: Pretraining of $\Omega^{(2)}$ and $\Omega_n^{(3)}$
  Set $a_n = x$ $(n = 1, ..., q+1)$. Then train $\Omega^{(2)}$ and $\Omega_n^{(3)}$ based on Eqs. (2.4), (2.5) and (2.8). After the pretraining, we are able to calculate the approximated predictions $\hat{Z}_{gn}$ $(n = 1, ..., q+1)$ using Eqs. (2.4) and (2.5);

- *Step* 2: Training of $\Omega_n^{(1)}$
  Let $I_1 = x$ and $I_n = [x, \hat{Z}_{gn-1}]$ $(n = 2, ...q+1)$. Then train $\Omega_n^{(1)}$ based on Eqs. (2.3) and (2.7).

- *Step* 3: Training of $\Omega^{(2)}$ and $\Omega_n^{(3)}$
  Let $I_1 = x$ and $I_n = [x, \hat{Z}_{gn-1}]$ $(n = 2, ...q+1)$. Then train $\Omega^{(2)}$ and $\Omega_n^{(3)}$ based on Eqs. (2.6) and (2.8). After the training, we calculate the predictions $\hat{Z}_{gn}$ $(n = 1, ..., q+1)$ using Eq. (2.6);

- *Step* 4: Repeat *Steps* 2 and 3, until an appropriate stop condition is satisfied.

2.5 Experiments

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

2.5.1 Evaluation Metrics

The evaluation metrics for multi-label classification include the multi-label precision(MPre), the multi-label recall(MRe), the multi-label F1-score(MF1) [95] and the multi-label G-means(MG). Considering that the multi-label classification method in our work is a combination of multi-binary classifiers for each class, MPre and MRe 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 $(x_i, y_i), i = 1...|D|, y_i \in \{0,1\}^m$ 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

$$MPre = \frac{1}{m}\sum_{i=1}^{|D|}\frac{|y_i \cap z_i|}{|z_i|} \tag{2.9}$$

$$MRe = \frac{1}{m}\sum_{i=1}^{|D|}\frac{|y_i \cap z_i|}{|y_i|} \tag{2.10}$$

$$MF1 = \frac{2 * MPrecision * MRecall}{MPrecision + MRecall} \tag{2.11}$$

$$MG = \sqrt{MPrecision * MRecall} \tag{2.12}$$

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

2.5.2 Experiment Settings

2.5.2.1 Details of the deep CNN model

By referring VGG16 network [96], we design a 1-d deep CNN as the feature extractor, consisting of six blocks as shown in Fig 2.4. Block 1 is the input block with two convolution layers with the size of 1×420 and 64 channels (1×420×64). Block 2 consists of a max-pooling layer (pool-size: 1×2) and two convolutional layers of 1×210×128. Blocks 3 to 5 consist of a max-pooling layer and three convolutional layers of 1×105×256, 1×53×512, 1×27×512, respectively. The final block is a fully connected layer of 1×1×512. All the convolutional layers and one fully connected layer use the ReLU activation function. Furthermore, regularization [97] is introduced to the deep network to reduce independence by adding batch normalization [98] and 0.4-ratio dropout to each network layer. In this way, the deep CNN feature extractor is a model with an input size 1×420 and output size 1×512.

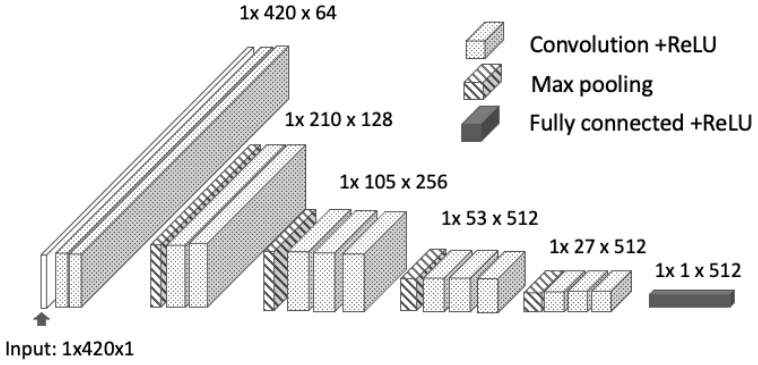


FIGURE 2.4: The structure of the deep CNN model.

2.5.2.2 Details on the supervised and unsupervised training algorithms

For the modeling environment, this training algorithm is implemented in Python. The deep CNN model is built via Chainer, developed by Community, Preferred Networks, Inc. [99]. 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 multi-head 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 multi-end 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 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.

2.5.3 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 [58]. Table 2.1 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' [16].

TABLE 2.1: Datasets used in the paper of DeepGO

Datasets	Training	Validation	Testing	Classes	Levels	Classes per level
DeepGO(BP)	20000	9000	10000	932	5	30/180/292/477/932
DeepGO(CC)	20000	8800	10000	439	4	18/126/260/439
DeepGO(MF)	20000	6300	5000	589	5	23/160/261/392/589

TABLE 2.2: Comparison results of the proposed MHME-CNN model with the HMC-LMLP, the DeepGO and the simple DCNN models

Datasets	Evaluation Metrics	Methods			
		MPre	**MRe**	**MF1**	**MG**
DeepGO(BP)	DeepGO[58]	0.3900	0.3400	0.3600	0.3700
	simple DCNN	0.4885±0.0043	0.4971±0.0013	0.4906±0.0095	0.4951±0.0045
	HMC-LMLP[57]	0.4847±0.0081	0.5005±0.0042	0.4921±0.0037	0.4935± 0.0021
	DCNN-MHME	0.7934±0.0059	0.5530±0.0098	**0.6317±0.0091**	**0.6605±0.0034**
DeepGO(CC)	DeepGO[58]	0.6600	0.6100	0.6300	0.6400
	simple DCNN	0.5047±0.0032	0.4909±0.0046	0.5014±0.0047	0.5017±0.0023
	HMC-LMLP[57]	0.4973±0.0035	0.5068±0.0089	0.5066±0.0067	0.5021± 0.0021
	DCNN-MHME	0.9101±0.0003	0.6646±0.0064	**0.6788±0.0034**	**0.7231±0.0032**
DeepGO(MF)	DeepGO[58]	0.6000	0.3800	0.4600	0.5100
	simple DCNN	0.5005±0.0010	0.4998±0.0073	0.4974±0.0089	0.5001±0.0035
	HMC-LMLP[57]	0.4941±0.0024	0.5010±0.0052	0.4975±0.0046	0.4998 ±0.0035
	DCNN-MHME	0.5987±0.0007	0.5238± 0.0042	**0.5158±0.0052**	**0.5548±0.0014**

When applying the proposed MHME CNN model, according to the hierarchy of GO functions, the labels are divided into 5 levels for BP and MF functions, 4 levels for CC functions. Table 2.1 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 Figure 2.3.

After training, the MHME CNN model is evaluated on test data. Table 2.2 shows the experimental results on the three datasets. For comparison, the first row is the results of DeepGO model copied from the DeepGO paper [58]. The second row shows the results of simple DCNN (deep CNN) model, whose structure is similar to

$$Z_{gq+1} = f_{q+1}\left(\Omega^{(3)}_{q+1}, \phi\left(\Omega^{(2)}, x\right)\right), \tag{2.13}$$

a deep CNN model without considering the label hierarchical relationship; and the third row is the result of the HMC-LMLP method [57]. The last row is the results of the proposed MHME CNN model.

For each model, these were calculated from the 10 resampled estimates produced by repeated cross-validation, and Table 2.2 shows the results: mean ± standard deviation. From the results of Table 2.2, 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

TABLE 2.3: 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

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.

2.5.4 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 [100]. Table 2.3 shows the details of these datasets. These 4 new datasets are not used in the training and only used for testing.

Table 2.4 shows the results. Same to previous subsection, 10 resampled estimates produced by repeated cross-validation. The proposed MHME CNN model is compared with the simple DCNN model and the HMC-LMLP model. From Table 2.4, 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.

TABLE 2.4: Results of the proposed MHME CNN model under transfer learning

Datasets	Functions	Methods	Evaluation Metrics			
			MPre	**MRe**	**MF1**	**MG**
Human	BP	simple DCNN	0.5214±0.0003	0.5107±0.0006	0.5025±0.0004	0.5232±0.0003
		HMC-LMLP	0.4968±0.0003	0.4989±0.0003	0.4978±0.0001	0.4979±0.0002
		MHME-CNN	0.7131±0.0005	0.6981±0.0002	**0.6999±0.0006**	**0.7003±0.0009**
	CC	simple DCNN	0.5322±0.0002	0.5301±0.0001	0.5166±0.0010	0.5314 ±0.0006
		HMC-LMLP	0.4929±0.0002	0.4977±0.0005	0.4953±0.0006	0.4957±0.0005
		MHME-CNN	0.6702±0.0003	0.6616±0.0005	**0.6773±0.0002**	**0.6610±0.0004**
	MF	simple DCNN	0.5190±0.0004	0.5002±0.0002	0.5178±0.0004	0.5244±0.0003
		HMC-LMLP	0.4952±0.0005	0.5000±0.0004	0.4976±0.0006	0.4977±0.0005
		MHME-CNN	0.6955±0.0002	0.6412±0.0001	**0.6777±0.0002**	**0.6670±0.0002**
Mouse	BP	simple DCNN	0.5103±0.0003	0.4940±0.0001	0.4989±0.0006	0.5002±0.0005
		HMC-LMLP	0.4977±0.0003	0.4994±0.0004	0.4981±0.0002	0.4985±0.0002
		MHME-CNN	0.6888±0.0004	0.7011±0.0002	**0.6899±0.0002**	**0.6823±0.0003**
	CC	simple DCNN	0.5222±0.0003	0.5049±0.0005	0.5113±0.0002	0.5232±0.0002
		HMC-LMLP	0.4927±0.0003	0.4977±0.0007	0.4952±0.0005	0.4951±0.0005
		MHME-CNN	0.6716±0.0003	0.6600±0.0004	**0.6621±0.0002**	**0.6599±0.0004**
	MF	simple DCNN	0.5002±0.0004	0.5375±0.0001	0.5202±0.0004	0.5181±0.0003
		HMC-LMLP	0.4949±0.0002	0.5000±0.0001	0.4975±0.0005	0.4974±0.0003
		MHME-CNN	0.6434±0.0002	0.6610±0.0001	**0.6404±0.0002**	**0.6398±0.0002**
Thaliana	BP	simple DCNN	0.5398±0.0003	0.5020±0.0001	0.5219±0.0001	0.5160±0.0003
		HMC-LMLP	0.4974±0.0002	0.4989±0.0003	0.4981±0.0006	0.4982±0.0002
		MHME-CNN	0.8708±0.0002	0.8903±0.0004	**0.8761±0.0002**	**0.8732±0.0005**
	CC	simple DCNN	0.5594±0.0002	0.5241±0.0005	0.5335±0.0003	0.5306±0.0007
		HMC-LMLP	0.4940±0.0002	0.4977±0.0004	0.4958±0.0003	0.4959±0.0003
		MHME-CNN	0.7849±0.0003	0.7998±0.0002	**0.7752±0.0003**	**0.7900±0.0002**
	MF	simple DCNN	0.5282±0.0002	0.5208±0.0003	0.5069±0.0004	0.5101±0.0003
		HMC-LMLP	0.4958±0.0003	0.5001±0.0002	0.4979±0.0005	0.4980±0.0001
		MHME-CNN	0.8007±0.0002	0.8449±0.0003	**0.8104±0.0002**	**0.7939±0.0005**
Cerevisiae	BP	simple DCNN	0.5401±0.0003	0.5343±0.0006	0.5321±0.0002	0.5181±0.0005
		HMC-LMLP	0.4973±0.0003	0.4989±0.0001	0.4980±0.0002	0.4981±0.0002
		MHME-CNN	0.8419±0.0004	0.8900±0.0002	**0.8508±0.0003**	**0.8449±0.0003**
	CC	simple DCNN	0.5277±0.0001	0.5345±0.0005	0.5223±0.0003	0.5403±0.0007
		HMC-LMLP	0.4940±0.0009	0.4976±0.0004	0.4957±0.0005	0.4956±0.0003
		MHME-CNN	0.8020±0.0003	0.7797±0.0001	**0.7745±0.0004**	**0.7648±0.0002**
	MF	simple DCNN	0.5559±0.0002	0.5087±0.0003	0.5335±0.0003	0.5253±0.0002
		HMC-LMLP	0.4962±0.0003	0.5000 ±0.0002	0.4981±0.0003	0.4981±0.0005
		MHME-CNN	0.8209±0.0003	0.7915±0.0007	**0.8005±0.0003**	**0.8160±0.0002**

2.6 Summary

This chapter introduces the proposed deep CNN model with multi-head and multi-end (MHME) for GO annotation to improve the performance of complex hierarchical multilabel classification. The deep CNN MHME model shares one deep CNN model with a set of linear classifiers, in which autoencoders are applied to fuse the features and reduce the dimension of feature vectors. Furthermore, 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 to capture the two-way relationship of labels.

The simulation results show that our proposed MHME CNN model works well and performs better than the state-of-the-art traditional methods. Moreover, our pre-built MHME CNN model also performs well on other GO annotation examples based on transfer learning.

Chapter 3

Localization Prediction with Transfer Learning of GO Annotation

3.1 Background

[1] This chapter investigates the protein function indirectly by the subcellular localization, which significantly relates to the protein function. The prediction of eukaryotic protein subcellular localization plays an important role in many biological processes since the location information of protein reveals how a cell is working as a basic unit of life [101]. One of the applications is that it is utilized in studying targeted drugs. It has been proved that one protein may appear in multi-locations which makes the prediction more complex. In this situation, automatic prediction is required. The subcellular localization prediction as a multilabel classification task, has been realized by using many machine learning methods like support vector machine (SVM) [102, 103],autoencoder [104], decision trees [105], etc.

When a one-hot or n-gram encoding method is used to vectorize protein sequences, a feature extractor is usually needed to extract and map the feature to space where it is more linearly separable. A neural network, especially deep CNN, can be used as such a

[1]This chapter mainly extends the Journal paper: X.Yuan, E.Pang, K.Lin and J.Hu. "Deep Protein Subcellular Localization Predictor Enhanced with Transfer Learning of GO Annotation", *IEEJ Trans. on Electrical and Electronics Engineering*, Vol. 15, No. 4, pp.559-567, Apr. 2021.

feature extractor. However, it is inevitable to use a large dataset for the training, which is difficult for many protein subcellular localization prediction cases. On the other hand, gene ontology (GO) terms reveal that individual genes contribute to the biology of an organism at the molecular, cellular, and organism levels. GO annotation, which includes the Cellular Component terms describe parts of cells and structures associated with cells throughout the taxonomy range, if available, is useful genetic information for predicting protein subcellular localization. X. Cheng et al. (2018) [67] and K.C. Chou et al.(2001) [68] investigated the cases like the proteins of human beings, fungus where GO annotation is available by applying both sequence feature and annotation feature for localization prediction and found that it is better than using only sequence feature. The works of Refs.[106, 107] encoded feature vectors by GO correlation information instead of using the presence or frequency of GO terms. They exploit the hidden correlation between the annotation features of proteins. However, experimentally annotated proteins are not always available for many species. Fortunately, for some species such as human, mouse, Arabidopsis thaliana, etc., experimentally annotated proteins are available [76]. It, therefore, is highly motivated to enhance predicting protein subcellular localization by using a transfer learning of these available experimental GO annotations from various related species.

A natural way may be considered to build a sequence-feature based prediction model for GO annotation, then apply the predicted GO annotation to enhance predicting subcellular protein localization, since there are available many machine learning methods for GO annotations using protein sequence [58, 108]. The incorrect GO annotation prediction, especially in the case of transfer learning, however, may result in poor performance of protein subcellular localization prediction. The similarity of GO annotations plays an essential role in many protein studies [109, 110, 111, 112]. Considering the similarity between the predictions of protein sequence to GO annotation and subcellular localization, another way to consider is to use the multitask learning method. The GO annotation predictor and the subcellular localization predictor share a common feature extractor, which improves the accuracy of each predictor by extracting the common features. As the GO annotation information can improve localization prediction, the multitask learning method is undoubtedly very suitable for this problem. However, the

prediction task of GO annotation has more than 3000 labels, while the subcellular localization task has only less than 20 labels, which results in an imbalance problem. The imbalance problem will decrease the performance of the task with fewer labels, which is our target task.

This chapter proposes a new method of predicting subcellular localization by using a transfer learning of GO annotation. It is structurally similar to the multitask learning method. The GO annotation predictor and the subcellular localization predictor share a deep CNN feature extractor, but they work in a pretraining and fine-tuning way. The GO annotation predictor is a hierarchical multilabel model [57, 80, 90], consisting of a deep CNN feature extractor and a set of linear multilabel classifiers [113]. It is trained by a large amount of available experimental GO annotations from various related species. The pretrained deep CNN feature extractor is then transferred to the subcellular localization predictor. It will be fine-tuned by using a limited-sized subcellular localization dataset. In the subcellular localization prediction model, the deep CNN feature extractor plays two roles: extracting features from the vectorized protein sequence and mapping the feature onto a feature space where it is more linearly separable. Therefore, the transfer learning of GO annotation, especially on closely related species, is expected to improve subcellular localization prediction significantly. The proposed method is applied to Swiss-Prot datasets and the results are compared with the state-of-the-art methods. Experiment results demonstrate the effectiveness of the proposed method.

The rest of this chapter is organized as follows. Section 3.2 introduces the problem formulation. Section 3.3 introduces two parallel predictors of the subcellular localization and the GO annotation, sharing a deep CNN feature extractor. Section 3.4 describes the training processes in a pretraining and fine-tuning manner. Section 3.5 carries out experiments on a set of Swiss-Prot datasets, and transfer learning experiments compared with different methods. Finally, Section 3.6 has a summary showing the conclusions of this chapter.

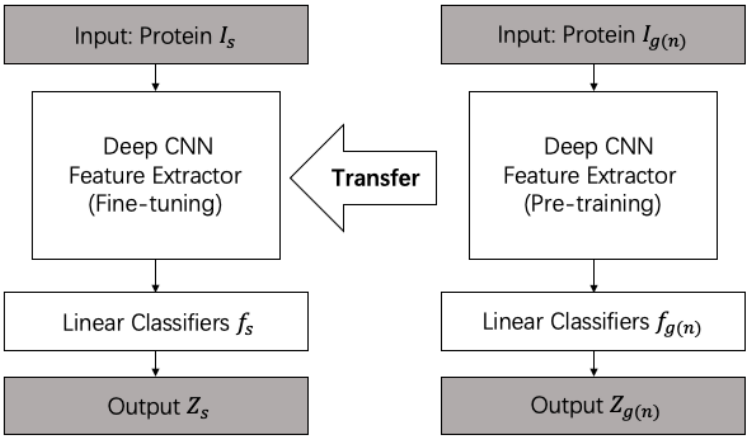


FIGURE 3.1: An image of two parallel predictors of GO annotation and subcellular localization, sharing a deep feature extractor.

3.2 Problem Formulation

3.2.1 Problem and notation

Suppose we are given a dataset of protein subcellular localization from a specific target species, $\{x_s(k), y_s(k)|k = N_s\}$, where N_s is the size of dataset, $x_s(k)$ is vectorized protein k, and $y_s(k)$ is the subcellular location vector, $y_s(k) = [s_1, ..., s_L]^T$, which L is the number of localizations. The problem is to build a multilabel prediction model for predicting protein subcellular localization.

For some species such as human, mouse and Arabidopsis thaliana, etc, there are available experimentally annotated proteins. Suppose we collect a large dataset of protein GO annotation from those species, $\{x_g(k), y_g(k)|k = N_g\}$, where N_g is the size of dataset, $x_g(k)$ is vectorized protein k, and $y_g(k)$ is the hierarchical GO annotation vector, $y_g(k) = [g^{(1)}(k), ..., g^{(q)}(k)]^T$, $g^{(i)}(k) = [g_1^{(i)}(k), ..., g_{n_i}^{(i)}(k)]$ $(i = 1, ..., q)$ where q is the number of levels, and n_i is the number of label in i level. The GO annotation dataset will be used to help building a subcellular localization prediction model by using transfer learning. A '1 gram + 2 gram' encoding method (Section 1.5) encodes the protein sequence of amino acids into appearance frequency vector which is a column vector with a length of 420.

3.2.2 Enhanced subcellular localization predictor

When designing protein subcellular localization predictor, we consider two parallel predictors of GO annotation predictor and subcellular localization predictor sharing a powerful deep CNN feature extractor, as shown in Figure 3.1, described by

$$Z_g = f_g\Big(\phi(I_g)\Big) \tag{3.1}$$

$$Z_s = f_s\left(\phi(I_s)\right) \tag{3.2}$$

where I_g, Z_g and I_g, Z_g are the input-output vectors of GO annotation predictor (right side of Figure 3.1) and subcellular localization predictor (left side of Figure 3.1), respectively. Efforts are made to design a powerful deep CNN feature extractor, $\phi(\cdot)$, to extract and map to a feature space where it is more linearly separable. The deep CNN feature extractor is first trained by using a large dataset of protein GO annotation from various species, then it is transferred to subcellular localization predictor where being fine-tuned by using the protein subcellular localization dataset. We will introduce the details of the two predictors in the following sections.

3.3 Deep CNN Feature Extractor

With a powerful feature extractor, the subcellular localization predictor can be simply designed as a linear multilabel classifier. We consider an ordinal 1-d deep CNN model with fixed input and output size as the feature extractor, defined by

$$\phi = \phi(\Omega^{(2)}, I) \tag{3.3}$$

In our experiments, $N_o = \dim(I) = 512$. Since the deep CNN model will first be trained as feature extractor in the GO annotation predictor using a large dataset from various related species, and then transferred to the subcellular localization predictor with fine-tuning, it therefore is key point to avoid overfitting and to make it sure only extracting common feature.

3.3.1 The GO annotation predictor

As described in Chapter 2, labels in the GO annotation problem are organized in q levels. An efficient way to capture the label relations at different levels is to design a hierarchical multilabel classifier consisting of hierarchically organized $q+1$ nonlinear classifiers. By sharing a deep CNN with the $q+1$ classifiers, we design a deep CNN model with multiple heads and multiple ends (MHME) to implement the $q+1$ classifiers 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. We briefly summarize as follows.

The MHME CNN model has three parts: the body part, the multi-end part and the multi-head part, which realizes the $q+1$ classifiers with sharing the deep CNN feature extractor as the body part, and represented by:

$$Z_{gn} = f_{gn}\left(\Omega_n^{(3)},\ \phi\left(\Omega^{(2)},\ a_n\left(\Omega_n^{(1)},\ I_{gn}\right)\right)\right), \tag{3.4}$$

where $n = 1, 2, ..., q+1$, f_n, ϕ and a_n denote the body part, the multi-end part and the multi-head part, and $\Omega_n^{(3)}$, $\Omega^{(2)}$, $\Omega_n^{(1)}$ are the parameters, respectively.

3.3.2 The subcellular localization predictor

As described in Section 3.2, labels in the subcellular localization problem are organized in only one level. By using the deep CNN feature extractor transferred from the deep GO annotation predictor, a hierarchical multilabel classifier can be designed by using a linear network with logistic sigmoid outputs:

$$Z_{s1} = f_{s1}\left(\Omega_{q+2}^{(3)},\ \phi\left(\Omega^{(2)},\ x_s\right)\right) \tag{3.5}$$

$$Z_s = f_s\left(\Omega_{q+3}^{(3)},\ \phi\left(\Omega^{(2)},\ a_{q+3}\left(\Omega_{q+3}^{(1)},\ I_{s1}\right)\right)\right) \tag{3.6}$$

where $I_{s1} = [x_s^T, \hat{Z}_{s1}^T]^T$, $Z_{s1} = y_s$, $Z_s = y_s$, and $\hat{Z}_{s1}$ denotes the prediction value of Z_{s1}.

3.4 Training Process

The training process consists of two parts: the pretraining and the fine-tuning.

3.4.1 Pretraining the deep CNN feature extractor

The GO annotation predictor which is a hierarchical multilabel model is represented by:

$$Z_{g1}(k) = f_{g1}\left(\omega_1,\ \phi\left(W,\ I_{g1}(k)\right)\right),\ \ I_{g1}(k) = x_g(k) \tag{3.7}$$

$$\begin{aligned} Z_{g2}(k) &= f_{g2}\left(\omega_2,\ \phi\left(W,\ A_2\left(\lambda_2,\ I_{g2}(k)\right)\right)\right), \\ & I_{g2}(k) = a_{g1}(k) \end{aligned} \tag{3.8}$$

$$\cdots$$

$$\begin{aligned} Z_{gq}(k) &= f_{gq}\left(\omega_q, \phi\left(W, A_q\left(\lambda_q,\ I_{gq}(k)\right)\right)\right), \\ & I_{gq}(k) = a_{g(q-1)}(k) \end{aligned} \tag{3.9}$$

$$\begin{aligned} Z_{g(q+1)}(k) &= f_{g(q+1)}\left(\omega_{q+1}, \phi\left(W, A_{q+1}\left(\lambda_{q+1}, I_{g(q+1)}(k)\right)\right)\right), \\ & I_{g(q+1)}(k) = a_{g(q)}(k) \end{aligned} \tag{3.10}$$

The deep CNN feature extractor is pretrained during the training of the GO annotation predictor, referred to Section2.4) for more details.

- *Step* 1 The training of the parameters $\Omega_1^{(3)}$, W from Eq.(3.7): $\Omega_1^{(3)}$ is the parameter of classifier f_{g1} and $\Omega^{(2)}$ is the parameter of the deep CNN feature extractor. They are trained in a supervised manner with logistic output by optimizing the sigmoid cross-entropy (SCE) loss function $E = \sum_{i=1}^{n} \varepsilon_{SCE}(\hat{Z}_{g(n)}, Z_{g(n)})$. In this formulation, ε_{SCE} is defined by

$$\begin{aligned} \varepsilon_{SCE}(\hat{z}, z) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C_n}[z_{ij} \times log(\hat{z}_{ij}) \\ + (1 - z_{ij}) \times log(1 - \hat{z}_{ij})] \end{aligned} \tag{3.11}$$

  where N is the number of samples and $C_n = \dim(Z_n)$.

Set the input $I_{g1} = x_g$, and after training we get the predictions $\hat{Z}_{g1}$ by Eq.(3.7).

- *Step* 2 The training of the parameters $\Omega_2^{(1)}$ from Eq.(3.8): $\Omega_2^{(1)}$ is the parameter of the autoencoder a_2 which is trained in an unsupervised manner by optimizing the mean squared error (MSE) loss function $E_n = \varepsilon_{MSE}(\hat{I}_{g(n)}, I_{g(n)})$. Let z be the input of encoder and $\hat{z}$ be the output of decoder. Then ε_{MSE} is defined by

$$\varepsilon_{MSE}(\hat{z}, z) = \frac{1}{N}\sum_{i=1}^{N}(\hat{z}_i - z_i)^2 \tag{3.12}$$

  where N is the number of samples.

  Set $I_{g2} = [x_g^T, \hat{Z}_{g1}^T]^T$, and after training the autoencoder we get the output A_2.

- *Step* 3 The training of the parameters $\Omega^{(2)}$, $\Omega_n^{(3)}$,$\Omega_n^{(1)}$,$(n = 2, 3, ...q+1)$: $\Omega_n^{(3)}$, $\Omega_n^{(1)}$ are the parameters of the classifier f_{gn}, the autencoder a_n respectively. Set the input $I_{gn} = a_{gn}(k)$, and train the parameters $\Omega_n^{(3)}$, $\Omega_n^{(1)}$, $\Omega^{(2)}$ based on Eqs.(3.9)-(3.12), the training process is as the same as the *Step* 1 and *Step* 2. Then repeat *Step* 2 and 3 until an appropriate stop condition is satisfied.

3.4.2 Fine-tuning the deep CNN feature extractor

With the pretrained deep CNN feature extractor $\phi(\Omega^{(2)}, \cdot)$, we build the localization predictor represent by Eq.(3.5) and Eq.(3.6). The deep CNN feature extractor is fine-tuned during the training of subcellular localization predictor.

- *Step* 1 The training of the parameters $\Omega_{q+2}^{(3)}$, $\Omega^{(2)}$ from Eq.(3.5): $\Omega_{q+2}^{(3)}$ is the parameter of classifier f_{s1} and they are trained in a supervised manner with logistic output by optimizing the Sigmoid Cross-Entropy (SCE) loss function $E = \sum_{i=1}^{n} \varepsilon_{SCE}(\hat{Z}_{s1}, Z_{s1})$. Set the input as x_s, and train the parameters based on Eq.(3.5) and Eq.(3.11). After training we get the predictions $\hat{Z}_{s1}$.

- *Step* 2 The training of the parameters $\Omega_{q+3}^{(3)}$, $\Omega_{q+3}^{(1)}$, W: $\Omega_{q+3}^{(1)}$ is the parameter of the autoencoder a_{q+2} which is trained in an unsupervised manner by optimizing the Mean Squared Error (MSE) loss function $E_n = \varepsilon_{MSE}(\hat{I}_{s1}, I_{s1})$. Ω is the parameter

TABLE 3.1: Swiss-Prot datasets

Dataset	Sequences	GO-terms	locations
DataSet1	52774	3281	-
DataSet2	34300	-	14
DataSet3-z	2631	-	14
DataSet3-d	1078	-	14
DataSet3-c	2181	-	14
DataSet3-ch	643	-	14
DeepLoc	14004	-	10
HumT	379	-	12
Höglund	5959	-	11

of the classifier f_s and the training method is the same as the *Step* 1. Same to *Step* 1, set the input $I_{s1} = [x_s^T, \hat{Z}_{s1}^T]^T$, and train the parameters based based on Eq.(3.6), Eq.(3.11) and Eq.(3.12) until an appropriate stop condition is satisfied.

3.5 Experiment Results

In this section, the proposed subcellular localization predictor is applied to Swiss-Prot datasets. Experiment results are compared with different methods.

3.5.1 Datasets

The proteins on the database Uniprot [100] included reviewed entries named Swiss-Prot. In the experiments, we create several Swiss-Prot datasets as shown in Table 3.1. DataSet1 consists of 52774 protein sequences from the eight type species[2] with 3281 GO annotations [16] organized in 6 levels ($q = 6$). DataSet1 is used for pretraining the deep CNN feature extractor by a transfer learning of a deep GO annotation predictor. DataSet2 consists of 34300 protein sequence samples also from these 8 species with 14 subcellular locations (Names of locations shows in Section 1.2.1.2).DataSet2 will be divided into 80% of training set, 10% of testing set and 10% of validation set, when building the deep subcellular localization predictor. DataSet3's are a set of datasets consisting of protein sequences from different species: DataSet3-z from 'zebrafish',

[2]The eight different type species including human, mouse, rat, Arabidopsis thaliana, cerevisiae, rice, fruit fly, and fungus.

DataSet3-d from 'dog', DataSet3-c from 'cat', DataSet3-ch from 'chimpanzee', which will be used for testing the performance of transfer learning. Datasets of DeepLoc, HumT and Höglund are three benchmark datasets used in Refs. [53, 107, 114], which will be used to compare the proposed method with the state-of-the-art methods.

3.5.2 Evaluation metrics

The evaluation metrics for multilabel classification include the multilabel precision (MPre), the multilabel recall (MRe), the multilabel F1-score (MF1) [95], the multilabel G-means (MG) and the multilabel Accuracy (MAcc). Considering that the multilabel classification method in our work is a combination of a multi-binary classifier for each class, MPre and MRe are the averages for the Precision 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 $(x_i, y_i), i = 1...|D|, y_i \in \{0,1\}^m$ is the set of labels. Suppose $z_i \in \{0,1\}^m$ be the labels predicted by the multilabel classifier for sample x_i. The definitions of the evaluation metrics are given by

$$MPre = \frac{1}{m}\sum_{i=1}^{|D|}\frac{|y_i \cap z_i|}{|z_i|} \tag{3.13}$$

$$MRe = \frac{1}{m}\sum_{i=1}^{|D|}\frac{|y_i \cap z_i|}{|y_i|} \tag{3.14}$$

$$MF1 = \frac{2 * MPre * MRe}{MPre + MRe} \tag{3.15}$$

$$MG = \sqrt{MPre * MRe} \tag{3.16}$$

$$MAcc = \frac{1}{m}\sum_{i=1}^{|D|}\frac{|y_i \cap z_i|}{|y_i \cup z_i|} \tag{3.17}$$

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

3.5.3 Experiment settings

For the modeling environment, this training algorithm is implemented in Python. The deep CNN model is built via Chainer, developed by Community, Preferred Networks, Inc. [99]. Each dataset is divided into three sets in the experiments: training set, validation set, and test set. We introduced the details of the deep CNN feature extractor in Section 2.5.2.1.

Since the deep CNN feature extractor is designed by referring to the VGG16 model, in our experiments, the default values of hyper-parameters suggested by the VGG16 model are used. But some hyper-parameters such as learning rate and stop epochs are determined by using trial and error method. The training of the supervised learning model: $f_{gn}\left(\Omega_n^{(3)}, \phi\left(\Omega^{(2)}, \cdot\right)\right)$ of the GO annotation predictor and $f_{s1}\left(\Omega_{n+1}^{(3)}\, \phi\left(\Omega_{n+2}^{(3)}, \cdot\right)\right)$, $f_s\left(\Omega_{n+2}^{(3)}, \phi\left(\Omega^{(2)}, \cdot\right)\right)$ of the subcellular localization predictor, 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 : $a_n(\Omega_n^{(1)}, \cdot)$ of the GO annotation predictor and $a_{n+1}(\Omega_{n+1}^{(1)}, \cdot)$ of the subcellular localization predictor, the initial learning rate used is 0.05 and the learning was stopped after 30 epochs where the performance for validation set stops improving.

3.5.4 Comparisons with the state-of-the-art methods

In this subsection, the proposed enhanced subcellular localization predictor (enhanced-SL) is applied to three benchmark datasets of subcellular localization, to compare with the state-of-the-art methods. The results are also compared with a deep subcellular localization predictor (SL-seq), which is used as a baseline. The deep subcellular localization predictor has the same structure of the proposed model which is trained using only the protein sequences of subcellular localization.

When building the proposed enhanced-SL model, the deep feature extractor is pre-trained using DataSet1, then fine-tuned by using each benchmark dataset. When building the baseline (SL-seq) model, the deep feature extractor is trained only using the

benchmark dataset. Table 3.2 shows the results of the three benchmark datasets. Two state-of-the-art methods are considered for the comparisons. One is the CONV A-BLSTM model which used multiply deep learning methods (CNN, RNN, LSTM) without GO annotations features [53]. Row 1 in Table 3.2 shows the results of the CONV A-BLSTM copied from Refs.[53] and [114]. The other state-of-the-art method is the Hum-mPLoc3.0 model which is trained using more GO annotations also from swiss-prot (GO: 14737) [107]. Row 2 in Table 3.2 shows the results of the Hum-mPLoc3.0 model copied from Refs. [107] and [114].

For each model, these were calculated from the 30 resampled estimates produced by repeated cross-validation, and Table 3.2 shows the results: mean ± standard deviation. Besides the five evaluation metrics, a statistical hypothesis test (Hypothesis test in Table 3.2) is conducted to see if the new models are statistically significant based on the baseline model. And in this case, the statistical significance is represented by the p-value.

Table 3.2 shows the results of the baseline (SL-seq) and the proposed enhanced deep subcellular prediction model (enhanced-SL) for the three benchmark datasets: DeepLoc, HumT and Höglund. From the results in Table 3.2, we can see that the proposed enhanced deep subcellular prediction model not only has better performance than the baseline, but also outperforms the two state-of-the-art methods. According to the hypothesis test of the p-values, the proposed enhanced-SL model outperforms the two state-of-the-art methods significantly. In the hypothesis test, the F1-score of SL-seq model is the baseline and the enhanced-SL model performed significantly better than the other two models, which we can find out by the p-values.

TABLE 3.2: Comparison results with the state-of-the-art methods

Datasets	Methods	Evaluation Metrics					Hypothesis test (MF_1)		
		MPrecision	**MRecall**	**MF1-score**	**MG-mean**	**MAcc**	**Z-statistic**	**p-value**	**Sig**[1]
DeepLoc	CONV A-BLSTM	-	-	-	-	0.7511	baseline(MAcc)		
	SL-seq	0.4669±0.0008	0.5037±0.0036	0.4815 ±0.0027	0.4844± 0.0023	0.8151±0.0109	2.3031	0.014319	**
	Enhanced-SL	0.5012±0.0016	0.5030±0.0071	**0.5020±0.0041**	**0.4902±0.0034**	**0.8724± 0.0207**	4.5401	0.000045	***
HumT	Hum-mPLoc3.0	-	-	0.6300	-	0.6500	baseline (MF1-score)		
	SL-seq	0.5187±0.0015	0.6306± 0.0038	0.6214± 0.0017	0.6237± 0.0034	0.9046± 0.0055		-	
	Enhanced-SL	0.6717 ±0.0017	0.6589 ± 0.0054	**0.6589± 0.0032**	**0.6605±0.0039**	**0.9095±0.0141**	2.7511	0.005063	***
Höglund	CONV A-BLSTM	-	-	-	-	**0.9138**		-	
	Hum-mPLoc3.0	-	-	0.5900	-	0.6400	baseline (MF1-score)		
	SL-seq	0.4755±0.0015	0.5093± 0.0052	0.4739± 0.0031	0.4905 ±0.0049	0.8268± 0.0145		-	
	Enhanced-SL	0.6170± 0.0007	0.6325 ± 0.0054	**0.6207± 0.0029**	**0.6237±0.0041**	0.8903±0.0176	3.0700	0.002307	***

[1] Significance: * p-value ≤ 0.1, ** p-value ≤ 0.05, *** p-value ≤ 0.01

3.5.5 Comparisons with different transfer learning methods

In this subsection, we perform experiments on DataSet1 and DataSet2 described in Table 3.1. The target task is to build a multilabel subcellular localization classifier using DataSet2. Same to the way building the proposed enhanced-SL model in Subsection 3.5.4, the deep feature extractor is first pretrained using DataSet1, then fine-tuned by using DataSet2. The proposed method is compared with the other two ways to transfer the learning of GO annotation. One is building a GO annotation predictor, then using the prediction of GO annotation as additional input features (SL-GoP). Another is the multitask learning method (Multitask), parallelly building a GO annotation and a subcellular localization predictor that sharing a deep feature extractor. Same to Subsection 3.5.4, 30 resampled estimates produced by repeated cross-validation, a statistical hypothesis test (Hypothesis test in Table 3.3) is conducted to see if the new models are statistically significant based on the baseline model (the state-of-the-art methods). And the statistical significance is represented by the p-value.

Table 3.3 shows a comparison of different ways test on the testing set of DataSet2. The first row of Table 3.3 shows the results of the baseline without the transfer learning of GO annotation. We can see that both the multitask method and the transfer learning methods are better than the baseline (SL-seq) which only uses protein sequence of subcellular localization, without transfer learning of GO annotation. The multitask method works better than the SL-GoP method but it could only apply to the problem in which the protein samples have experimental GO annotations. The SL-GoP method has poor performance maybe due to the overfitting of predicting GO annotations. The proposed enhanced deep subcellular localization prediction model works the best and it could transfer the feature extractor which carries the knowledge of predicting GO annotation from different proteins to the subcellular localization predictor.

TABLE 3.3: Results of different transfer learning methods

Methods	**Evaluation Metrics**					**Hypothesis test**		
	MPrecision	**MRecall**	**MF1-score**	**MG-mean**	**MAcc**	**Z-statistic**	**p-value**	**Sig**
SL-seq	0.6188±0.0023	0.6276±0.0031	0.6158±0.0028	0.6214±0.0028	0.8847±0.0024	baseline (MF1-score)		
SL-GoP	0.6283±0.0010	0.6330±0.0021	0.6272 ±0.0019	0.6298±0.0041	0.8800±0.0071	1.4084	0.084822	*
Multitask	0.6595±0.0083	0.6892±0.0042	0.6727±0.0085	0.6737±0.0021	0.9051±0.0102	3.3235	0.001208	***
Enhanced-SL	**0.6973± 0.0057**	**0.7275±0.0014**	**0.7114±0.0256**	**0.7119±0.0113**	**0.9395±0.0316**	3.5869	0.000606	***

3.5.6 Performance of transfer learning within or out species

As shown in Table 3.3, the experiments in Section 3.5.5 have proved that the learning of GO annotation on some species can be transferred to significantly enhance the subcellular localization predictors of the same species. This section shows the results of transfer learning out species. That is, the learning of GO annotation on some species is transferred to enhance the subcellular localization predictors of different species. The three prediction models were trained in Section 3.5.5 using DataSet1 and DataSet2 are directly applied to test on four other datasets from different species without any further training. From the results shown in Table 3.4, we can see that the enhanced-SL model has better performance than the SL-seq model and the SL-GoP model.

TABLE 3.4: Performance of transfer learning from different species

Datasets	Methods	Evaluation Metrics				
		MPre	MRe	MF1	MG	MAcc
DataSet3-z	SL-seq	0.4674±0.0017	0.5212±0.0051	**0.4914± 0.0013**	0.4931±0.0021	0.8774±0.0013
	SL-GoP	0.4688±0.0005	0.5089±0.0013	0.4850± 0.0037	0.4877± 0.0029	0.8725±0.0013
	Enhanced-SL	0.4752±0.0015	0.5133±0.0022	0.4906± 0.0016	**0.4933±0.0014**	**0.8807±0.0022**
DataSet3-d	SL-seq	0.5168±0.0011	0.5598±0.0027	0.5362±0.0015	0.5374±0.0029	0.8902±0.0012
	SL-GoP	0.5007±0.0008	0.5059±0.0016	0.4858±0.0023	0.5032± 0.0015	0.8846±0.0013
	Enhanced-SL	0.5116±0.0015	0.5665±0.0026	**0.5363±0.0027**	**0.5378±0.0017**	**0.9052±0.0021**
DataSet3-c	SL-seq	0.4974±0.0004	0.4992±0.0033	0.4874±0.0015	0.4980±0.0018	0.8774±0.0019
	SL-GoP	0.4834±0.0001	0.5321±0.0015	0.5061 ± 0.0021	0.5070±0.0014	0.8871± 0.0027
	Enhanced-SL	0.5033±0.0007	0.5390±0.0023	**0.5199±0.0019**	**0.5206±0.0023**	**0.9214±0.0016**
DataSet3-ch	SL-seq	0.5091±0.0009	0.5016±0.0012	0.4726±0.0014	0.5049±0.0011	0.8807±0.0015
	SL-GoP	0.5345±0.0013	0.5909±0.0018	0.5597±0.0016	0.5612±0.0015	0.8824±0.0013
	Enhanced-SL	0.6113±0.0009	0.5788±0.0016	**0.5615±0.0015**	**0.5933±0.0021**	**0.9046±0.0017**

As mentioned before, the deep feature extractor plays two roles in the subcellular localization prediction. The first role is to extract features from the vectorized protein sequence; the second role is to map the features onto a feature space where it is more linearly separable. Therefore, when enhancing the deep feature extractor by using transfer learning, for the first role, any protein sequence-based prediction may be helpful, while for the second role, a related function prediction is required in order to transfer useful knowledge. Since the subcellular localization is related to GO annotation, we employ the transfer learning of GO annotation on 8 different species to improve the prediction of subcellular localization. In such a case, the species that are more closely related to the 8 species are expected to get more improvement, while the species that are distantly

related to the 8 species are expected to get less improvement. From the results shown in Table 3.4, the three different datasets show different improvements demonstrating that the enhanced feature extractor by transfer learning is effective:

- *Distantly related species*: From Table 3.4, we find that the DataSet-z is improving very less. It is the data of zebrafish which is distantly related to the species of the training data(DataSet1).
- *Closely related species* : The Dataset3-ch is improving more which is the data of Chimpanzee. Considering that the similarity of genes in chimpanzee and human (in the DataSet1), which is proved that the proposed method is more effective to the samples of closely related species.
- *Within species* : From Table 3.3, the DataSet2 is the within species data which has an obvious improvement by the proposed method.

3.6 Summary

In this chapter, we introduce the proposed subcellular localization predictor. It has a powerful deep CNN feature extractor which is enhanced using a transfer learning of GO annotation. The proposed method consists of two main steps: pretraining the deep CNN feature extractor by building a deep GO annotation predictor, and fine-tuning the deep CNN feature extractor when building the subcellular localization predictor. The deep CNN GO annotation predictor consists of a set of hierarchical multilabel classifiers sharing one common deep CNN feature extractor. The deep CNN subcellular localization predictor consists of a multilabel classifier and the deep CNN feature extractor which is transferred from the learning of GO annotation predictor. A large set of experimentally annotated proteins from various species are used to pretrain the deep CNN feature extractor, which then transfers the common feature mapping of annotations to improve the performance of subcellular localization prediction. The proposed method has good performance on the Swiss-Port datasets when transfer learning using

the protein samples both within and out species. Furthermore, it outperforms the state-of-the-art methods on all three benchmark datasets. Especially on dataset DeepLoc our model is 12% enhanced of Accuracy than the CONV A-BLSTM method, and the dataset Höglund 3% enhanced of F1-score than the Hum-mPLoc3.0 method.

Chapter 4

PPI Prediction Based on Transfer Learning for Complex Detection

4.1 Background

[1] In this chapter, we predict protein functions by the interaction of proteins: protein-protein interaction (PPI) and protein complex. Protein complexes are molecular groups of more than one functionally related protein, which are essential to biological processes in a cell. The complexes exist in many biological processes and they perform a vast amount of functions like cell cycle control, signaling, protein folding, etc. Since the wrong protein complexes are clinical manifestations of the disease, protein complex detection becomes an important indicator of the biological process. However, experimental detections of protein complexes are expensive and inefficient. Therefore, computational approaches, such as clustering-based methods, are widely adopted to detect the complexes from protein-protein interaction networks.

Many previous works are clustering-based methods. MCODE detects complexes by selecting an initial vertex of high local weight and iteratively adding neighbor vertices with similar weights [115]. CFinder is a clique-finding algorithm that forms clusters

[1] This chapter mainly extends the Journal paper: X. Yuan, H. Deng and J. Hu, "Constructing a PPI Network Based on Deep Transfer Learning for Protein Complex Detection",*IEEJ Trans. on Electrical and Electronics Engineering*, Vol.17, No.3, pp.436-444, March, 2022.

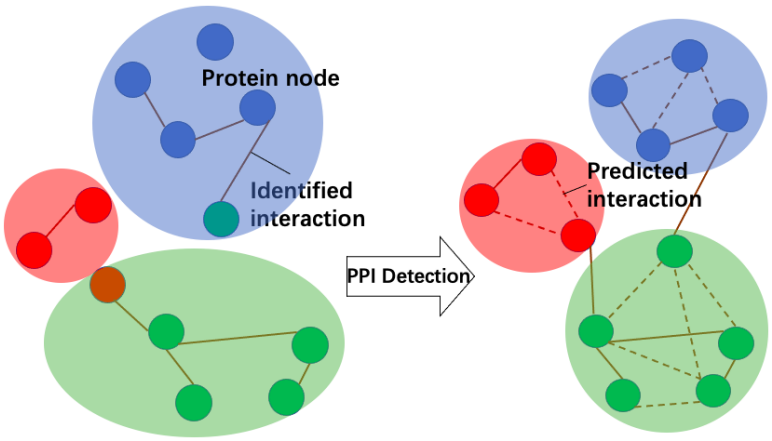


FIGURE 4.1: Clustering images of the experimentally identified PPI network and the reconstructed complete PPI network.

of fully connected subgraphs of different minimum clique sizes [116]. CDIP identifies complexes by exploiting biological and topological properties for the complexes [117]. Ref. [55] defines a new condition for clustering proteins in density, diameter and cosine similarity by selecting specific nodes as key nodes. These conventional clustering-based methods for protein complexes have high requirements for the sample characteristics, which are not conducive to large-scale detection. Therefore, spectral clustering may be more suitable for protein complexes detection with finite samples during the modeling compared with different clustering methods. One remarkable advantage of spectral clustering is its ability to cluster nodes that are not necessarily vectors, and to use for this a similarity, which is less restrictive than a distance [118].

A comprehensive and accurate PPI network plays an essential role in increasing the accuracy of spectral clustering detection for protein complexes. However, since the experimentally identified PPIs are usually very limited for most species, many unknown PPIs exist in a given PPI network. As an example shown in the left of Figure 4.1, if those unknown interactions are just set to be negative (also called protein-protein no-interaction, PPNI), the incomplete PPI network may mislead the clustering algorithm to give a wrong result. Therefore, it is highly motivated to develop a deep PPI predictor to estimate those unknown PPIs and detect protein complexes from a reconstructed PPI network.

Vectorized protein sequences are often applied as the input feature of a PPI prediction

model. Yu et al. [55] used amino acid background frequency (AABF). And Rehman et al. [54] used frequency compositions of amino acids (monopeptides), dipeptides & tripeptides, which results in a huge feature set with the possibility of unimportant frequency compositions. Considering that the frequency of tripeptides is too sparse to train as the features, we will use a '1-gram + 2-gram' encoding (monopeptides & dipeptides) method to extract an appearance frequency as the input feature. After vectorizing the protein sequences, we design a deep feature extractor based on a convolutional neural network (CNN) to extract and map the feature to a space that is easier to deal with. On the other hand, it usually needs a large dataset to train a deep CNN feature extractor. For most species, there are available only limited experimentally identified PPIs. We consider a transfer learning of the deep CNN feature extractor. Although deep transfer learning is popular in deep learning based image processing, it is not easy to implement deep transfer learning in the PPI prediction. We need to make sure that the deep CNN model only extracts common features to avoid overfitting. For this purpose, we try to share the deep CNN model 1) with some closely related prediction models; 2) in as many prediction models as possible.

It has been known that the proteins with similar gene ontology (GO) annotations are highly related to the interactions [71, 119]. And the proteins with different subcellular localization (SL) are often negatively interacted [120]. Therefore, it is natural to consider that GO annotation and subcellular localization predictions are closely related to the PPI predictions. Experimentally annotated proteins are not always available for most species, however, they are fully available for some species such as human, mouse, *Arabidopsis thaliana*, etc., which are called the type species. Moreover, GO annotation and subcellular localization are multilabel prediction problems. An efficient way to capture the label relations is to design a group of multilabel classifiers consisting of hierarchically organized nonlinear classifiers. Therefore, by taking advantage of the experimentally annotated proteins from type species, we will pretrain the deep CNN feature extractor in the group of deep GO annotation and subcellular localization predictors and transfer it to the deep PPI predictor.

In summary, this chapter proposes a deep learning based PPI predictor to construct a

PPI network, from which protein complexes are detected by using a spectral clustering method. The deep PPI predictor consists of a semi-supervised SVM classifier and a deep CNN feature extractor. The deep CNN feature extractor is first pretrained in a group of GO annotation and subcellular localization predictors implemented by a multi-head and multi-end (MHME) deep CNN model [113, 121, 122], using datasets from the type species, then fine-tuned in a binary PPI detector using experimentally identified PPI samples. In this way, the unknown PPIs are predicted by a deep PPI predictor enhanced with the transfer learning of GO and SL annotations. Finally, a spectral clustering method detects protein complexes from a complete reconstructed PPI network. The experiments of the proposed method on the two benchmark datasets show better performance than the state-of-the-art methods.

The rest of this chapter is organized as follows. Section 4.2 is the problem description. Section 4.3 introduces the transfer learning of the deep CNN model. Section 4.4 is the PPI detection and the protein complex detection. Section 4.5 shows the training process of the whole framework. Section 4.6 is the experiments on benchmark datasets. Finally, Section 4.7 has a summary showing the conclusions of this chapter.

4.2 Problem Description

4.2.1 Protein Complex Detection

Suppose we collect a PPI network with N protein nodes and L interactions, an undirected graph $G = (V, E)$ is used to model it. E is an edge set which are the experimentally identified interactions among these proteins by $E = \{e_n(x_i, x_j)|e_n(x_i, x_j) \in (0,1), n = 1,2,...,L\}$ and V is a protein node set by $V = \{x_i, x_j|i, j = 1,2,\ldots,N\}$ that x_i and x_j are vectorized protein sequences. There is a complex set $C = \{C_1, C_2, ..., C_i|i = k\}$, $y_i \in C$, where k is the number of the clusters. The problem is to determine which protein complexes y_i, the proteins are belonging to. Graph G is the input of the spectral clustering algorithm for detecting the protein complexes. As shown in Figure 4.1, a PPI network is usually constructed by taking individual proteins as vertices and pairwise interactions

between them as edges. Same to the previous chapters, a '1 gram + 2 gram' encoding (Section 1.5) method encodes the protein sequence of amino acids into an appearance frequency vector.

4.2.2 PPI Prediction

A complete PPI network of N protein nodes contains C_N^2 edges, each corresponding to a PPI. However, the collected experimentally identified PPIs are usually very limited, resulting in an incomplete PPI network. Therefore, we propose a deep PPI detector to predict the unknown PPIs to rebuild a complete PPI network so as to improve the performance of protein complexes clustering.

The deep PPI predictor consists of a deep CNN model as feature extractor $\phi(\cdot)$ and an SVM classifier $f_{svm}(\cdot)$.

The predictor is represented by the following formulas:

$$\Phi(x_i, x_j) = g(\phi(x_i), \phi(x_j)) \tag{4.1}$$

$$Z(x_i, x_j) = f_{svm}(\Phi(x_i, x_j)) \tag{4.2}$$

where $g(\cdot,\cdot)$ is a 2-Norm based feature combination method that will be discussed in Section 4.4.1, x_i and x_j are the vectorized inputs of protein pairs (x_i, x_j) and $Z(x_i, x_j)$ is the output of PPI $e(x_i, x_j)$. By applying a deep CNN feature extractor, the features of two inputs are first extracted and mapped to a space where it is easier to deal with, and then are combined by a 2-Norm method to get the output $\Phi(x_i, x_j)$. Finally, an SVM classifier is applied to obtain the interaction prediction.

4.2.3 Feature Extraction and Transformation

Training a deep CNN model needs a large number of samples. Since the experimentally identified PPI samples are insufficient, the deep CNN feature extractor is trained in a transfer learning manner. That is, we first pretrain the deep CNN feature extractor

by an MHME model that implements a group of nonlinear predictors of GO and SL annotations, using experimentally annotated proteins which from the eight type species as mentioned in Section3.5.1, then fine-tune it with the experimentally identified PPI samples.

Let us denote the group of predictors of GO annotation and subcellular localization by:

$$Z_n = f_n(\phi(I_n)), \quad n = 1, 2, ... \tag{4.3}$$

where I_n, Z_n are the input vectors and output vectors of the predictors, the details of which will be discussed in the next section. And $\phi(\cdot)$ represents the CNN feature extractor. After pretraining in the deep CNN MHME models, the deep feature extractor is fine-tuned in a binary classifier with the identified PPI samples.

In the following sections, we will introduce more details of the deep PPI detector and spectral clustering.

4.3 Deep CNN Feature Extractor

A powerful deep feature extractor with a sample binary classifier is designed for PPI prediction. Let us consider an ordinal 1-d deep CNN model with fixed input and output size, defined by

$$\Phi_i = \phi(W, x_i), \tag{4.4}$$

where W is the weight and the input vector $x_i \in R^{420}$, the output vector $\Phi_i \in R^{N_0}$. In our experiments, $N_0 = 512$. Figure 2.4 in Section 2.5.2.1 shows the details of the model structure. When pretraining the deep CNN model in a class of GO annotation predictors and subcellular localization predictors using a large dataset from the type species, it is important to avoid overfitting and to ensure it only extracting common feature. After pretraining, the deep CNN model is then transferred to the PPI detector for further fine-tuning.

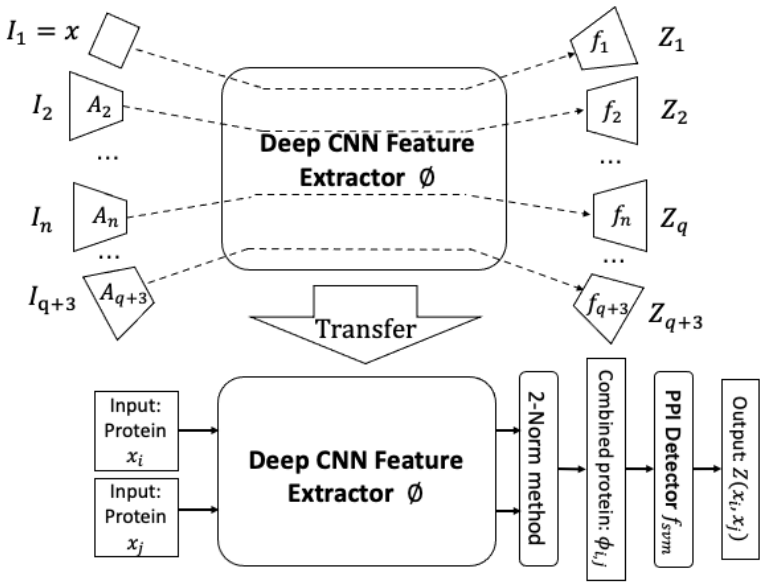


FIGURE 4.2: Deep transfer learning of the PPI detector.

4.3.1 Pretraining

The GO and SL annotations are multilabel classification problems. Suppose the labels in the GO annotation problem are organized in q levels described by the vectors v_l, $l = 1, 2, ...q$, and the labels of SL annotation are described by the vector v_s. To capture efficiently the label relations, we may design a hierarchical multilabel classifier, consisting of hierarchical organized $q+1$ nonlinear classifiers for GO annotation predictor, and 2 nonlinear classifiers for SL annotation predictor. In order to share the deep CNN feature extractor in these $q+3$ classifiers, we design a deep CNN model with multiple heads and multiple ends (MHME) to implement the $q+3$ classifiers 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. We briefly summarize as follows. Section 2.3 for more details.

Figure 4.2 shows the deep transfer learning of the PPI detector in a pretraining and fine-tuning way. The upper part of Figure 4.2 is a deep MHME CNN model which introduce in Section 2.3. The model composes of three parts, represented by:

$$Z_n = f_n(\omega_n, \phi(W, A_n(\lambda_n, I_n))), n = 1, 2, ..., q+3 \tag{4.5}$$

where ϕ denotes the body part, A_n the multi-end part, and f_n the multi-head part, respectively, and ω_n, W, λ_n are the parameter sets of the three parts. I_n and Z_n are the input vectors and output vectors for the $q+3$ classifiers.

In the pretraining, we share the deep CNN feature extractor in the $q+3$ classifiers of the GO and SL annotations so as to prevent overfitting and to ensure to extract only common features. In our experiments, $q = 6$.

4.3.2 Fine-tuning

After pretraining the deep CNN feature extractor by the GO and SL annotation predictors, and before concatenating it with the PPI detector, the deep feature extractor is fine-tuned with the experimentally identified known PPI samples. Firstly, the protein-pairs are extracted feature by the deep CNN model and then combined by the 2-Norm method according to Eq. (4.8). Then, a binary classifier is designed by using a linear network with logistic sigmoid output:

$$\begin{aligned} Z(x_i, x_j) &= f_{ppi}\Big(\Phi(x_i, x_j)\Big), \\ \Phi(x_i, x_j) &= g(\phi(W, x_i), \phi(W, x_j)) \end{aligned} \tag{4.6}$$

where the output $Z(x_i, x_j)$ is the label vector of the interaction.

4.4 PPI Prediction and PPI Network Clustering

After the transfer learning for the feature extractor, an RBF kernel SVM classifier is constructed in a semi-supervised manner. A 2-Norm method, to combine the pair of protein sequences, also introduce in this section. And at last, the spectral clustering to detect the protein complexes.

4.4.1 2-Norm Method

Assuming there is an interaction between protein seq_i and seq_j. A combining feature calculated by 2-Norm method is defined by[51]

$$g(x_i, x_j) = \sqrt{x_i^2 + x_j^2}, \quad i, j = 1, ..., N \tag{4.7}$$

$$\Phi(x_i, x_j) = \sqrt{\Phi_i^2 + \Phi_j^2} \tag{4.8}$$

where x_i and x_j are the encoding vectors of protein seq_i and seq_j by the '1 gram + 2 + gram' encoding method. The Eq. (4.8) expresses the combining of the output feature of the deep CNN model, where the output $\Phi(x_i, x_j)$ is the feature matrix of the protein pairs.

4.4.2 SVM Classifier for PPI Detection

With the combined feature of protein pairs as input, the PPI detection is performed using a binary classified of SVM with RBF kernel. The SVM classifier is trained in an inductive semi-supervised learning way to take advantage of unlabeled data, which improves classification performance. The bottom part of Figure 4.2 shows the structure of the deep PPI detector. It is composed of two parts, the deep CNN feature extractor and the RBF kernel SVM.

The following formula expresses the SVM classifier with the labeled feature from the deep CNN model:

$$f_{svm} = K^T(\phi_{i,j})\alpha + b, \tag{4.9}$$

where $K(\cdot)$ is the kernel matrix of RBF kernel function, while α and b are the weight and bias of the SVM, and $\phi_{i,j} = \Phi(x_i, x_j)$ is the combined input vector of the classifier.

There are two parts of training data for semi-supervised training: the known PPI combined feature $\phi_{i,j}$ with the label y_t and the unknown PPI combined feature with the

pseudo-label y_u, which are defined as:

$$\begin{cases} (\phi_{i,j}, y_t), & e(x_i, x_j) \in E \\ (\phi_{i,j}, y_u), & e(x_i, x_j) \notin E \end{cases} \tag{4.10}$$

The inductive semi-supervised SVM improves the accurate prediction of interactions, ensuring that the reconstructed PPI network is complete. Adding with the predicted PPIs by the SVM classifier, the last step is the spectral clustering from the reconstructed PPI network.

4.4.3 Spectral Clustering of PPI Network

We construct an undirected graph via the reconstructed PPI network to detect protein complexes by the spectral clustering. The graph G represents the PPI network, V represents the set of protein nodes vectors, and E represents the set of edges which are the interactions:

$$G = G_u(V, E, W_1), \tag{4.11}$$

W_1 is the weight matrix of the edges which is composed of the weights between protein nodes, and is decided by the protein interaction:

$$w_{i,j} = \begin{cases} 1, & y_{i,j} = y_t = 1 \\ 0.8, & y_{i,j} = y_u = 1 \\ 0, & y_{i,j} = y_t = 0, y_{i,j} = y_u = 0 \end{cases} \tag{4.12}$$

where the weights of the identified positive interactions y_t are 1, the predicted positive interactions y_u are set to be 0.8, and the negative interactions are 0. From the weight matrix, we could get the degree d_i for each node $v_i \in V$:

$$d_i = \sum_{j}^{n} w_{i,j}, \tag{4.13}$$

With the degree of each node d_i we could get the degree matrix D which is a diagonal matrix:

$$D = \begin{bmatrix} d_1 & \cdots & \cdots \\ \cdots & d_2 & \cdots \\ \vdots & \vdots & \ddots \\ \cdots & \cdots & d_n \end{bmatrix} \tag{4.14}$$

From the matrix W_1 and D we get the main tools for spectral clustering, the Laplacian matrix Q. We represent the weight matrix W_1 of graph G as an adjacency matrix; then we also can get the degree matrix D which is how many edges connect to a node, which is

$$Q = D - W_1. \tag{4.15}$$

The spectral clustering model with the input of three special matrices W_1, D, Q, the label clusters C_i, and the output of A_i which are the prediction labels of protein complexes.

4.5 Training Process

This section is the training process of the proposed method.

4.5.1 Supervised Learning of Deep CNN Feature Extractor

The GO and SL annotation predictor, which is a hierarchical multilabel model, is represented by:

$$
\begin{aligned}
Z_1 &= f_1(\omega_1,\ \phi(W,\ I_1)),\ \ I_1 = x && (4.16)\\
Z_2 &= f_2(\omega_2,\ \phi(W,\ A_2(\lambda_2,\ I_2))),\\
&\qquad I_2 = a_1 && (4.17)\\
&\cdots\\
Z_{q+1} &= f_{q+1}\left(\omega_{q+1}, \phi\left(W, A_{q+1}\left(\lambda_{q+1}, I_{q+1}\right)\right)\right),\\
&\qquad I_{q+1} = a_{q+1} && (4.18)\\
Z_{q+2} &= f_{q+2}\left(\omega_{q+2}, \phi\left(W, A_{q+2}\left(\lambda_{q+2},\ I_{q+2}\right)\right)\right),\\
&\qquad I_{q+2} = a_{q+2} && (4.19)\\
Z_{q+3} &= f_{q+3}\left(\omega_{q+3}, \phi\left(W, A_{q+3}\left(\lambda_{q+3},\ I_{q+3}\right)\right)\right),\\
&\qquad I_{q+3} = a_{q+3} && (4.20)
\end{aligned}
$$

(Eqs. (4.16)-(4.18)) represent the GO annotation predictors and (Eqs. (4.19), (4.20)) represent the subcellular localization predictor. We briefly describe the training steps as follows:

- *Step* 1: Setting $I_n = x,\ n = 1, 2, ..., q+3$ in Eqs.(4.16)-(4.20), pretrain the parameters W, ω_n, which is realized in a supervised manner with logistic output to optimize a sigmoid cross-entropy (SCE) loss function $E = \sum_{i=1}^{n} \varepsilon_{SCE}(\hat{Z}_n, Z_n)$, where ε_{SCE} is defined by

$$
\begin{aligned}
\varepsilon_{SCE}(\hat{z}, z) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}[z_{ij} \times log(\hat{z}_{ij})\\
+ (1 - z_{ij}) \times log(1 - \hat{z}_{ij})],
\end{aligned}
\qquad (4.21)
$$

  where N is the number of samples and $M = \dim(Z_n)$.

Set the inputs I_n $(n = 1,2,...,q+3)$ to x, from Eqs. (4.16)-(4.20), we obtain the estimates of Z_n, $\hat{Z}_n$, which are used to train the autoencoders $A_n(\cdot,\cdot)$.

- *Step* 2: Train the parameters λ_n $(n = 1,2,...q+3)$, by setting $I_n = [x^T, \hat{Z}_n^T]^T$. The loss function for optimizing the autoencoder $A_n(\cdot,\cdot)$ is the mean squared error (MSE) $E_n = \varepsilon_{MSE}(\hat{I}_n, I_n)$. Denoting z the input of encoder and $\hat{z}$ be the output of decoder, ε_{MSE} is defined by

$$\varepsilon_{MSE}(\hat{z}, z) = \frac{1}{N}\sum_{i=1}^{N}(\hat{z}_i - z_i)^2, \tag{4.22}$$

  where N is the number of samples.

- *Step* 3: Train the parameters W, ω_n $(n = 1,2,3,...q+3)$ in Eqs.(4.16)-(4.20). In this stage the inputs are set as $I_n = a_n$ and the training process is the same as *Step* 1.

- *Step* 4: Repeat *Step* 2 and 3 until an appropriate stop condition is satisfied.

4.5.2 Semi-supervised Learning of PPI Predictor

To train the PPI predictor, we optimize the semi-supervised SVM by the following objective function:

$$\min_{\alpha,b} \frac{1}{2}\sum_{i,j=1}^{N} \max(1 - y_{i,j}(K^T(\phi_{i,j})\alpha + b), 0)^2 + \frac{1}{2}\alpha^T K \alpha, \tag{4.23}$$

where $y_{i,j}$ and $\phi_{i,j}$ are the output label and the input combined protein vector of the training data, N is the number of instances.

- *Step* 1: Pretraining the f_{svm} with known PPI samples $(\phi_{i,j}, y_t)$ by function Eq. (4.23). Then labeled the unknown PPI samples with pseudo-label y_u by Eq. (4.9).

- *Step* 2: Training the parameters α, b with both known samples$(\phi_{i,j}, y_t)$ and unknown samples$(\phi_{i,j}, y_u)$ by Eq. (4.23). After that, we update the pseudo labels by Eq. (4.9).

- *Step* 3: Repeat *Step* 2 until an appropriate stop condition is satisfied.

4.5.3 Unsupervised Learning of Protein Complex Detector

The steps of spectral clustering for protein clustering show in the following:

- **Input**:Similarity matrix of graph G, number k of clusters.
- Construct the weighted adjacency matrix W_1 by Eq. (4.12).
- Compute the graph Laplacian L by Eq. (4.15), calculate the standard Laplacian which is $D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$.
- Compute the first k eigenvectors $u_1, u_2, ..., u_k$ of $D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$.
- Let $U \in \mathbf{R}^{n\times k}$ be the matrix containing the vectors $u_1, u_2, ..., u_k$ as columns.
- For $i = 1, ..., n$, let $z_i \in \mathbf{R}^n$ be the vector corresponding to the i-th row of U.
- Cluster the points z_i in $\mathbf{R}^n$ with the k-means algorithm into clusters $C_1, ...C_k$
- **Output**: Clusters $A_1, ..., A_k$ with $A_i = \{j \mid z_j \in C_i\}$

4.6 Experiment Results and Discussions

4.6.1 Datasets

In our experiments, there are three groups of datasets, as shown in Table 4.1. The first group is the datasets of protein complex detection. Two benchmark datasets of protein complexes are applied to prove the experimental performance. CYC2008 [123] is a catalogue of manually curated heteromeric protein complexes from Yeasts; MIPS [124] provided the expert curation of genomes and interaction maps in plants, fungi, and mammals by regular annotation of protein sequences. The second group is the experimentally identified PPI datasets that are applied to train the PPI predictor by semi-supervised learning. This PPI dataset comprises 12890 protein pairs, in which 6445

positive interactions from the DIP dataset [125] and 6445 negative interactions from the Negatome dataset (Negatome is a database of proteins and protein domains that are unlikely to engage in physical interactions)[126]. The third group is the GO&SL dataset for pretraining the deep feature extractor. It consists of 34000 protein sequences from the eight type species with 3281 GO annotations organized in 6 levels (q = 6), and 14 subcellular localizations.

TABLE 4.1: Details of the three groups of datasets

Dataset	proteins	Pairs	Interaction	Clusters	GO	SL
CYC2008	387	-	-	14	-	-
MIPS	601	-	-	22	-	-
DIP	5742	6445	P	-	-	-
Negatome	3297	6445	N	-	-	
GO&SL	34000	-	-	-	3281	14

4.6.2 Evaluation Metrics

Three classical evaluation metrics, Precision, Recall and F-score, evaluate the efficiency of the proposed method. Precision is the proportion of correct positive predictions; Recall is the proportion of positive samples that are correctly predicted positive; F-score gives the overall performance:

$$Precision = \frac{TP}{TP+FP} \tag{4.24}$$

$$Recall = \frac{TP}{TP+FN} \tag{4.25}$$

$$F\text{-}score = \frac{2*Precision*Recall}{Precision+Recall} \tag{4.26}$$

where the true(T) or false(F) samples are protein pairs in a cluster. A true positive (TP) decision assigns two interacted protein nodes to the same cluster, and a true negative (TN) decision assigns two non-interacted nodes to different clusters. A false positive (FP) decision assigns two non-interacted nodes to the same cluster, and a false negative (FN) decision assigns two similar nodes to different clusters.

4.6.3 Experimental Setting

For the modeling environment, this training algorithm is implemented in Python. The deep CNN model is built via Chainer, developed by Community, Preferred Networks, Inc. [99]. Classification model, SVM, invokes from the Scikit-learn library [127]. And the spectral clustering is based on the Networkx Framework [128].

When applying the algorithm described in Subsection 4.5.2, the training algorithm starts with a learning rate of 0.0075. The algorithm ends after 120 epochs when the performance stops improving on the validation set. On the other hand, when implementing the unsupervised learning of the autoencoders, A_n, the training algorithm starts with a learning rate of 0.05. It ends after 30 epochs when the performance on the validation set stops improving. The semi-supervised SVM is first pretrained by the training set of the PPI dataset. Then, this semi-supervised learning uses a mini-batch manner of 5000 input by Eqs. (4.9) and (4.23) to ensure the accuracy of the SVM classifier.

4.6.4 Comparisons with the State-of-Art Methods

4.6.4.1 Results of PPI prediction

We compare the semi-supervised PPI predictor with four state-of-the-art methods on two benchmark datasets, and Table 4.2 shows the results. Zeng et al.[51] applied n-gram to extract feature and discussed the different feature selection way on predicting PPIs. Wei et al.[52] investigated the prediction of negative interactions by n-gram feature extraction, feature selection and ranking, and k-means clustering acts as a classifier. Sun et al.[60] built a deep predictor which consist a stacked autoencoder with a SVM classifier on sequence-based feature. And in the works of Bandyopadhyay et al.[109], they predicted PPI with GO annotation-based feature on SVM classifier. All results of the four state-of-the-art methods are copied from their papers. We can see that the proposed method has better performance on both of the datasets.

TABLE 4.2: Comparisons of PPI prediction accuracy with different methods

Datasets	**Methods**				
	Zeng et al.[51]	Wei et al.[52]	Sun et al.[60]	Bandyopadhyay et al.[109]	Ours
DIP	0.9594	-	0.9377	0.8700	**0.9762**
Negatome	-	0.8580	-	-	**0.9680**

4.6.4.2 Results of protein complex detection

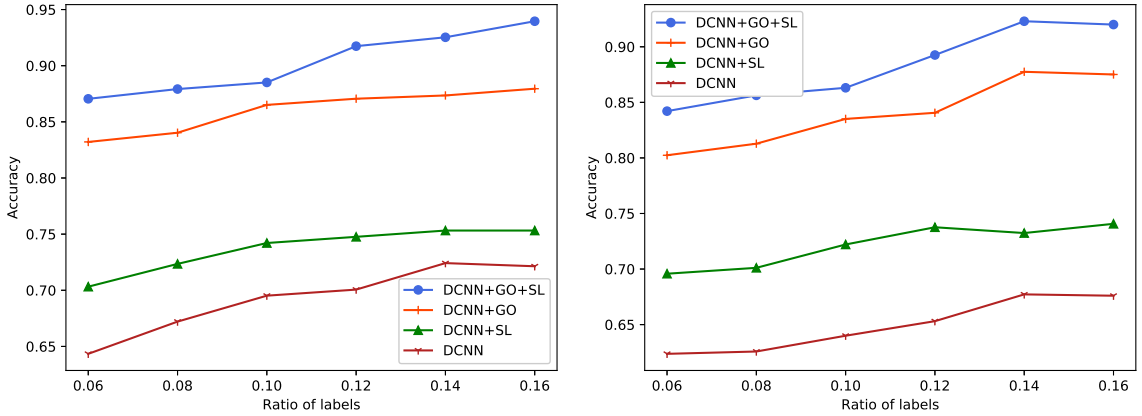
We compare the results of applying three state-of-the-art clustering methods and our method to the reconstructed PPI network, and the results of applying to the raw PPI network. Table 4.3 shows the results. MCODE[115] method is based on vertex weighting by local neighborhood density and outward traversal from a locally dense seed protein. CFinder[116] detecting complex by reading the protein interactions, performs a search for protein dense subgraphs. PCD-BEns[54] employs Multi-Dimensional Scaling for the grouping of known complexes by exploring inter-complex relations. We can see that comparing with the results on the raw PPI network, all the methods achieve a better performance on the reconstructed PPI network.

TABLE 4.3: Comparison results of protein complex detection on the raw and reconstructed PPI Networks

Datasets	**Methods**	**Raw PPIN**			**Reconstructed PPIN**		
		Precision	**Recall**	**F-score**	**Precision**	**Recall**	**F-score**
MIPS	MCODE[115]	0.3930	0.2143	0.2664	0.4531	0.2656	**0.3221**
	CFinder[116]	0.2730	0.2470	0.2507	0.3652	0.2917	**0.3021**
	PCD-BEns[54]	0.4097	0.4782	0.4425	0.5321	0.4752	**0.5211**
	DCNN+GO+SL	0.5487	0.3169	0.4018	0.7333	0.5432	**0.6041**
CYC2008	MCODE[115]	0.3561	0.2458	0.3167	0.3972	0.3741	**0.3772**
	CFinder[116]	0.3367	0.3056	0.3180	0.3656	0.2917	**0.3314**
	PCD-BEns[54]	0.5125	0.4980	0.4997	0.5123	0.5523	**0.5414**
	DCNN+GO+SL	0.5714	0.3945	0.4667	0.8125	0.5909	**0.6842**

4.6.5 Ablation Studies

In this part, we carry out an ablation study under different features of transfer learning. The baseline is the deep CNN feature extractor of the random initial value parameter with no transfer learning (DCNN). Then we also compare to the deep CNN feature



(a) Results of CYC2008 (b) Results of MIPS

FIGURE 4.3: Ablation studies of the inductive semi-supervised PPI prediction

extractor transfer learning from GO prediction (DCNN+GO), and from SL prediction (DCNN+SL).

4.6.5.1 Results of PPI prediction

Section 4.6.4.1 shows the comparison results of PPI prediction with state-of-the-art methods. This section is the results of the semi-supervised PPI prediction in different ratios of labels. Figures 4.3(a) and 4.3(b) show the results of dataset CYC2008 and MIPS, respectively. The DCNN+GO model transfers only from the GO annotation predictors and the DCNN+SL model and transfers only from the SL predictors, which we can see perform better than the baseline model. And transfer learning from both of them outperforms in all the methods obviously. The accuracy of detecting by the proposed PPI predictor can respectively reach 93.96% for the CYC2008 dataset and 92.30% for the MIPS dataset, indicating that this method can be employed to predict the unknown PPI.

4.6.5.2 Results of protein complex detection

Table 4.4 shows the results of ablation studies of the proposed method with transfer learning based on both GO and SL. DCNN+GO denotes the transfer learning based

TABLE 4.4: Results of ablation studies

Datasets	Methods	Evaluation Metrics			Hypothesis test (F-score)		
		Precision	Recall	F-score	Z-statistic	p-value	Sig
MIPS	Raw PPIN	0.5487±0.0049	0.3169 ± 0.0102	0.4018±0.0095	-		
	DCNN	0.2916±0.0323	0.6190±0.0214	0.4023± 0.0320	baseline (F-score)		
	DCNN+SL	0.4864±0.0131	0.4090±0.0305	0.4444±0.0371	3.5884	0.0024	***
	DCNN+GO	0.4169±0.0426	0.5325±0.0310	0.4993±0.0519	4.8689	0.000326	***
	DCNN+GO+SL	0.7333±0.0656	0.5432±0.0428	**0.6041± 0.0973**	6.5293	0.000033	***
CYC2008	Raw PPIN	0.5714±0.0071	0.3945±0.0065	0.4667±0.0077	-		
	DCNN	0.3608 ±0.0417	0.5000±0.0651	0.4580±0.0324	baseline (F-score)		
	DCNN+SL	0.4706±0.0276	0.5254±0.0043	0.4974±0.0419	2.9735	0.0069	***
	DCNN+GO	0.6014± 0.0451	0.4243±0.0518	0.5667±0.0729	4.7152	0.00041	***
	DCNN+GO+SL	0.8125±0.0712	0.5909±0.0530	**0.6842±0.0900**	7.9478	0.000006	***

only on GO, while DCNN+SL based only on SL. DCNN denotes the results without transfer learning, and Raw PPIN denotes the results of the raw PPI network. We show the results by averaging over 10 resampled estimates of cross-validation. As shown in Table 4.4, the results are given by mean and standard deviation. In addition to the three evaluation metrics, the p-value, a statistical hypothesis test (Hypothesis test in Table 4.4) is also shown to prove the statistical significance of the new models.

From Table 4.4, we find that the performance of clustering is affected when PPI detectors are not accurate enough (for CYC2008, the results of the raw PPI network are even better than the results of baseline). With increasing the accuracy of the PPI prediction, clustering performances from the corresponding reconstructed PPI network is increasing. And the proposed method shows the best result.

4.7 Summary

This chapter proposes a deep learning based PPI predictor to construct a PPI network from which protein complexes are detected using spectral clustering. The deep PPI detector consists of a transfer learning based CNN feature extractor and a semi-supervised SVM classifier. The CNN feature extractor is pretrained by hierarchical GO annotations and subcellular localization predictors and fine-tuned by a binary PPI classifier. The proposed method transfers a deep CNN feature extractor to the PPI predictor, which

helps the accuracy of the PPI prediction because the GO and SL annotations contribute to the interactions of proteins. The spectral clustering finally clusters the protein complexes. Experimental results on two benchmark datasets show that the proposed method outperforms the state-of-the-art methods.

Chapter 5

Conclusions

5.1 Summaries

In this dissertation, we have proposed deep transfer learning methods for predicting protein GO annotation, protein subcellular localization, and protein-protein interaction. By sharing a deep CNN feature extractor with as many related prediction tasks as possible, we have constructed powerful deep neural network models for prediction of protein GO annotation, localization and interaction based on a deep transfer learning from different tasks and different species.

- GO annotation is first formulated as a very complicated hierarchical multilabel classification tasks consisting of a set of related local classifiers. A deep CNN model with multiple heads and multiple ends (MHME) is proposed to implement the whole set of hierarchically organized local classifiers. In this way, by sharing a deep CNN with multiple local classifiers, we can extract common feature and construct more powerful local classifiers for each level with limited training samples and realize a deep transfer learning from different species to achieve better classification performance.

- A deep protein subcellular localization predictor is constructed, consisting of a linear classifier and a deep CNN feature extractor. By using a deep MHME CNN

model, the deep CNN feature extractor is first shared and pretrained in a deep GO annotation prediction task based on a large dataset from type species, and then is transferred to the subcellular localization prediction task with fine-tuning using protein localization samples. In this way, we realize a deep protein subcellular localization prediction enhanced with a transfer learning of GO annotation, from different species and different tasks.

- A novel deep transfer learning based PPI detector is developed to reconstruct a PPI network for protein complex detection. Considering the facts that the similarities of GO annotations contribute to protein interactions, and the differences of subcellular localizations contribute to negative interactions, a deep MHME CNN model is used to pretrain a deep CNN feature extractor in a class of deep GO annotation and subcellular localization prediction tasks using datasets from the type species, then transfer it to the PPI prediction task for fine-tuning, so as to have a deep PPI detector enhanced with a transfer learning of GO annotation and subcellular localization prediction, from different species and different tasks.

Experimental results on benchmark datasets show that the proposed methods outperform the state-of-the-art methods and confirm the effectiveness of the proposed methods.

5.2 Future Research Topics

It still has a huge amount of works to continue of deep modeling for protein bioinformatic analysis:

- *For Protein bioinformatics* This research treats the protein sequence as a heat map and extracts feature by the deep CNN model. However, other deep learning models could improve the feature extraction work for protein sequence or the other sequence data. Except for these three issues, many other function prediction problems are looking forward to solving by developing new algorithms.

Furthermore, we will investigate other deep learning applications of bioinformatics prediction with the sequence-based and annotation-based features.

- *For Bioinformatics* The way we treat the protein sequence could also investigate the other sequence data in bioinformatics. And the transfer learning of features between species is also the problem of the other bioinformatics, as the various researches in genomics. We are going to apply this model to this kind of research.

Bibliography

[1] T. Shafee. (2016) File:protein structure (full).png. [Online]. Available: https://en.wikipedia.org/wiki/File:Protein_structure_(full).png#metadata

[2] Gene Ontology Consortium. (1999) Gene ontology. [Online]. Available: http://geneontology.org/

[3] M. Hung and W. Link, "Protein localization in disease and therapy," *Journal of Cell Science*, vol. 124, no. 20, pp. 3381–3392, 2011.

[4] R. Kent, M. Schlesinger, and B. Wybourne, "On algebraic approaches to the genetic code," *Canad. J. Phys*, vol. 76, pp. 445–452, 1998.

[5] G. B. Fogel and D. W. Corne, *Evolutionary Computation in Bioinformatics*. Elsevier, 2002.

[6] C. A. Orengo, A. D. Michie, S. Jones, D. T. Jones, M. B. Swindells, and J. M. Thornton, "Cath–a hierarchic classification of protein domain structures," *Structure*, vol. 5, no. 8, pp. 1093–1109, 1997.

[7] M. Margulies, M. Egholm, W. E. Altman, S. Attiya, J. S. Bader, L. A. Bemben, J. Berka, M. S. Braverman, Y.-J. Chen, Z. Chen *et al.*, "Genome sequencing in microfabricated high-density picolitre reactors," *Nature*, vol. 437, no. 7057, pp. 376–380, 2005.

[8] P. Bork, T. Dandekar, Y. Diaz-Lazcoz, F. Eisenhaber, M. Huynen, and Y. Yuan, "Predicting function: from genes to genomes and back," *Journal of Molecular Biology*, vol. 283, no. 4, pp. 707–725, 1998.

[9] G. Pandey, V. Kumar, and M. Steinbach, "Computational approaches for protein function prediction: A survey," *Computer Science & Engineering (CS&E) Technical Reports*, 2006.

[10] E. Buxbaum, *Fundamentals of Protein Structure and Function*. Springer, 2007, vol. 31.

[11] N. Lan, R. Jansen, and M. Gerstein, "Toward a systematic definition of protein function that scales to the genome level: Defining function in terms of interactions," *Proceedings of the IEEE*, vol. 90, no. 12, pp. 1848–1858, 2002.

[12] C. A. Ouzounis, R. M. Coulson, A. J. Enright, V. Kunin, and J. B. Pereira-Leal, "Classification schemes for protein structure and function," *Nature Reviews Genetics*, vol. 4, no. 7, pp. 508–519, 2003.

[13] S. C. Rison, T. C. Hodgman, and J. M. Thornton, "Comparison of functional annotation schemes for genomes," *Functional & Integrative Genomics*, vol. 1, no. 1, pp. 56–69, 2000.

[14] A. Ruepp, A. Zollner, D. Maier, K. Albermann, J. Hani, M. Mokrejs, I. Tetko, U. Güldener, G. Mannhaupt, M. Münsterkötter *et al.*, "The funcat, a functional annotation scheme for systematic classification of proteins from whole genomes," *Nucleic Acids Research*, vol. 32, no. 18, pp. 5539–5545, 2004.

[15] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig *et al.*, "Gene ontology: tool for the unification of biology," *Nature Genetics*, vol. 25, no. 1, pp. 25–29, 2000.

[16] Gene Ontology Consortium, "The gene ontology (go) database and informatics resource," *Nucleic Acids Research*, vol. 32, no. suppl_1, pp. D258–D261, 2004.

[17] A. H. Millar, C. Carrie, B. Pogson, and J. Whelan, "Exploring the function-location nexus: using multiple lines of evidence in defining the subcellular location of plant proteins," *The Plant Cell*, vol. 21, no. 6, pp. 1625–1631, 2009.

[18] National Institutes of Health. (1988) National center for biotechnology information. [Online]. Available: https://www.ncbi.nlm.nih.gov/

[19] I. M. Keseler, J. Collado-Vides, S. Gama-Castro, J. Ingraham, S. Paley, I. T. Paulsen, M. Peralta-Gil, and P. D. Karp, "Ecocyc: a comprehensive database resource for escherichia coli," *Nucleic Acids Research*, vol. 33, no. suppl_1, pp. D334–D337, 2005.

[20] M. S. Scott, S. J. Calafell, D. Y. Thomas, and M. T. Hallett, "Refining protein subcellular localization," *PLoS Computational Biology*, vol. 1, no. 6, p. e66, 2005.

[21] H. N. Chua, W.-K. Sung, and L. Wong, "Exploiting indirect neighbours and topological weight to predict protein function from protein–protein interactions," *Bioinformatics*, vol. 22, no. 13, pp. 1623–1630, 2006.

[22] L. Jacob and J.-P. Vert, "Protein-ligand interaction prediction: an improved chemogenomics approach," *Bioinformatics*, vol. 24, no. 19, pp. 2149–2156, 2008.

[23] I. Friedberg, "Automated protein function prediction-the genomic challenge," *Briefings in Bioinformatics*, vol. 7, no. 3, pp. 225–242, 2006.

[24] D. N. Itzhak, S. Tyanova, J. Cox, and G. H. Borner, "Global, quantitative and dynamic mapping of protein subcellular localization," *Elife*, vol. 5, p. e16950, 2016.

[25] N. Zheng, H. N. Tsai, X. Zhang, and G. R. Rosania, "The subcellular distribution of small molecules: from pharmacokinetics to synthetic biology," *Molecular Pharmaceutics*, vol. 8, no. 5, pp. 1619–1628, 2011.

[26] J. Zhang, R. Cruz-Cosme, W. Zhuang, Meng, D. Liu, Y. Liu, S. Teng, H. Wang, Pei, and Q. Tang, "A systemic and molecular study of subcellular localization of sars-cov-2 proteins," *Signal Transduction and Targeted Therapy*, vol. 5, no. 1, pp. 1–3, 2020.

[27] P. Legrain, "Protein–protein interactions: Protein interactions contribute to protein function," *Trends in Genetics*, vol. 18, no. 8, p. 432, 2002.

[28] O. Klementieva, N. Benseny-Cases, A. Gella, D. Appelhans, B. Voit, and J. Cladera, "Dense shell glycodendrimers as potential nontoxic anti-amyloidogenic agents in alzheimer's disease. amyloid-dendrimer aggregates morphology and cell toxicity," *Biomacromolecules*, vol. 12, no. 11, pp. 3903–3909, 2011.

[29] P. Myler, R. Stacy, L. Stewart, B. Staker, W. Van Voorhis, G. Varani, and G. Buchko, "The seattle structural genomics center for infectious disease (ssgcid)," *Infectious Disorders-Drug Targets*, vol. 9, no. 5, pp. 493–506, 2009.

[30] R. F. Weaver, *Molecular Biology (6th ed.)*. New York: McGraw-Hill, 2012.

[31] S. Oliver, "A network approach to the systematic analysis of yeast gene function," *Trends in Genetics (Regular ed.)*, vol. 12, no. 7, pp. 241–242, 1996.

[32] S. Yokoyama, H. Hirota, T. Kigawa, T. Yabuki, M. Shirouzu, T. Terada, Y. Ito, Y. Matsuo, Y. Kuroda, Y. Nishimura *et al.*, "Structural genomics projects in japan," *Nature Structural Biology*, vol. 7, no. 11, pp. 943–945, 2000.

[33] UniProt Consortium. (2002) The universal protein resource. [Online]. Available: https://www.uniprot.org/

[34] M. Gardiner-Garden and T. Littlejohn, "A comparison of microarray databases," *Briefings in Bioinformatics*, vol. 2, no. 2, pp. 143–158, 2001.

[35] D. H. Haft, B. J. Loftus, D. L. Richardson, F. Yang, J. A. Eisen, I. T. Paulsen, and O. White, "Tigrfams: a protein family resource for the functional identification of proteins," *Nucleic Acids Research*, vol. 29, no. 1, pp. 41–43, 2001.

[36] W. T. Clark and P. Radivojac, "Analysis of protein function and its prediction from amino acid sequence," *Proteins: Structure, Function, and Bioinformatics*, vol. 79, no. 7, pp. 2086–2096, 2011.

[37] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, vol. 215, no. 3, pp. 403–410, 1990.

[38] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped blast and psi-blast: a new generation of protein database search programs," *Nucleic Acids Research*, vol. 25, no. 17, pp. 3389–3402, 1997.

[39] W. R. Pearson and D. J. Lipman, "Improved tools for biological sequence comparison," *Proceedings of the National Academy of Sciences*, vol. 85, no. 8, pp. 2444–2448, 1988.

[40] S. Mei, W. Fei, and S. Zhou, "Gene ontology based transfer learning for protein subcellular localization," *BMC Bioinformatics*, vol. 12, no. 1, pp. 1–12, 2011.

[41] O. Emanuelsson, H. Nielsen, S. Brunak, and G. Von Heijne, "Predicting subcellular localization of proteins based on their n-terminal amino acid sequence," *Journal of Molecular Biology*, vol. 300, no. 4, pp. 1005–1016, 2000.

[42] S. Yu, Chin, J. Lin, Chih, and K. Hwang, Jenn, "Predicting subcellular localization of proteins for gram-negative bacteria by support vector machines based on n-peptide compositions," *Protein Science*, vol. 13, no. 5, pp. 1402–1406, 2004.

[43] S. S. Hannenhalli and R. B. Russell, "Analysis and prediction of functional subtypes from protein sequence alignments," *Journal of Molecular Biology*, vol. 303, no. 1, pp. 61–76, 2000.

[44] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, A. Hung Byers *et al.*, *Big data: The Next Frontier for Innovation, Competition, and Productivity*. McKinsey Global Institute, 2011.

[45] A. Vinayagam, R. König, J. Moormann, F. Schubert, R. Eils, K.-H. Glatting, and S. Suhai, "Applying support vector machines for gene ontology based gene function prediction," *BMC Bioinformatics*, vol. 5, no. 1, pp. 1–14, 2004.

[46] C. Yu, Y. Chen, C. Lu, and J. Hwang, "Prediction of protein subcellular localization," *Proteins: Structure, Function, and Bioinformatics*, vol. 64, no. 3, pp. 643–651, 2006.

[47] E. Kretschmann, W. Fleischmann, and R. Apweiler, "Automatic rule generation for protein annotation with the c4. 5 data mining algorithm applied on swiss-prot," *Bioinformatics*, vol. 17, no. 10, pp. 920–926, 2001.

[48] A. Bulashevska and R. Eils, "Predicting protein subcellular locations using hierarchical ensemble of bayesian classifiers based on markov chains," *Bmc Bioinformatics*, vol. 7, no. 1, pp. 1–13, 2006.

[49] A. Saini and J. Hou, "Progressive clustering based method for protein function prediction," *Bulletin of Mathematical Biology*, vol. 75, no. 2, pp. 331–350, 2013.

[50] R. Cao, C. Freitas, L. Chan, M. Sun, H. Jiang, and Z. Chen, "Prolango: protein function prediction using neural machine translation based on a recurrent neural network," *Molecules*, vol. 22, no. 10, p. 1732, 2017.

[51] J. Zeng, D. Li, Y. Wu, Q. Zou, and X. Liu, "An empirical study of features fusion techniques for protein-protein interaction prediction," *Current Bioinformatics*, vol. 11, no. 1, pp. 4–12, 2016.

[52] L. Wei, P. Xing, J. Zeng, J. Chen, R. Su, and F. Guo, "Improved prediction of protein–protein interactions using novel negative samples, features, and an ensemble classifier," *Artificial Intelligence in Medicine*, vol. 83, pp. 67–74, 2017.

[53] J. J. Almagro Armenteros, C. K. Sønderby, S. K. Sønderby, H. Nielsen, and O. Winther, "DeepLoc: prediction of protein subcellular localization using deep learning," *Bioinformatics*, vol. 33, no. 21, pp. 3387–3395, 2017.

[54] Z. ur Rehman, A. Idris, and A. Khan, "Multi-dimensional scaling based grouping of known complexes and intelligent protein complex detection," *Computational Biology and Chemistry*, vol. 74, pp. 149–156, 2018.

[55] Y. Yu and Z. Zheng, "Protein complex identification based on weighted ppi network with multi-source information," *Journal of Theoretical Biology*, vol. 477, pp. 77–83, 2019.

[56] R. Cerri, R. C. Barros, and A. C. de Carvalho, "Hierarchical classification of gene ontology-based protein functions with neural networks," in *Proc. of the*

2015 international joint conference on neural networks (IJCNN 2015) (Killarney). IEEE, July 2015, pp. 1–8.

[57] R. Cerri, R. C. Barros, A. C. 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.

[58] M. Kulmanov, M. A. Khan, and R. Hoehndorf, "DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier," *Bioinformatics*, vol. 34, no. 4, pp. 660–668, 2017.

[59] M. Kulmanov and R. Hoehndorf, "Deepgoplus: improved protein function prediction from sequence," *Bioinformatics*, vol. 36, no. 2, pp. 422–429, 2020.

[60] T. Sun, B. Zhou, L. Lai, and J. Pei, "Sequence-based prediction of protein protein interaction using a deep-learning algorithm," *BMC Bioinformatics*, vol. 18, no. 1, pp. 1–8, 2017.

[61] K. Chou and H. Shen, "Hum-ploc: a novel ensemble classifier for predicting human protein subcellular localization," *Biochemical and Biophysical Research Communications*, vol. 347, no. 1, pp. 150–157, 2006.

[62] H. Shen and K. Chou, "Gneg-mploc: a top-down strategy to enhance the quality of predicting subcellular localization of gram-negative bacterial proteins," *Journal of Theoretical Biology*, vol. 264, no. 2, pp. 326–333, 2010.

[63] S. Briesemeister, J. Rahnenführer, and O. Kohlbacher, "Yloc-an interpretable web server for predicting subcellular localization," *Nucleic Acids Research*, vol. 38, no. suppl_2, pp. W497–W502, 2010.

[64] S. Chi and D. Nam, "Wegoloc: accurate prediction of protein subcellular localization using weighted gene ontology terms," *Bioinformatics*, vol. 28, no. 7, pp. 1028–1030, 2012.

[65] T. Blum, S. Briesemeister, and O. Kohlbacher, "Multiloc2: integrating phylogeny and gene ontology terms improves subcellular protein localization prediction," *BMC Bioinformatics*, vol. 10, no. 1, pp. 1–11, 2009.

[66] S. Ray, M. De, and A. Mukhopadhyay, "A multiobjective go based approach to protein complex detection," *Procedia Technology*, vol. 4, pp. 555–560, 2012.

[67] X. Cheng, X. Xiao, and K.-C. Chou, "pLoc-mEuk: Predict subcellular localization of multi-label eukaryotic proteins by extracting the key GO information into general pseaac," *Genomics*, vol. 110, no. 1, pp. 50–58, 2018.

[68] K. Chou, "Prediction of protein cellular attributes using pseudo-amino acid composition," *Proteins: Structure, Function, and Bioinformatics*, vol. 43, no. 3, pp. 246–255, 2001.

[69] W. Huang, C. Tung, H. Huang, and S. Ho, "Predicting protein subnuclear localization using go-amino-acid composition features," *Biosystems*, vol. 98, no. 2, pp. 73–79, 2009.

[70] Z. Lei and Y. Dai, "Assessing protein similarity with gene ontology and its use in subnuclear localization prediction," *BMC Bioinformatics*, vol. 7, no. 1, pp. 1–10, 2006.

[71] T. Price, F. I. Peña, and Y. Cho, "Survey: Enhancing protein complex prediction in ppi networks with go similarity weighting," *Interdisciplinary Sciences: Computational Life Sciences*, vol. 5, no. 3, pp. 196–210, 2013.

[72] C. Gavin, Anne, P. Aloy, P. Grandi, R. Krause, M. Boesche, M. Marzioch, C. Rau, L. J. Jensen, S. Bastuck, B. Dümpelfeld *et al.*, "Proteome survey reveals modularity of the yeast cell machinery," *Nature*, vol. 440, no. 7084, pp. 631–636, 2006.

[73] V. Spirin and L. A. Mirny, "Protein complexes and functional modules in molecular networks," *Proceedings of the National Academy of Sciences*, vol. 100, no. 21, pp. 12 123–12 128, 2003.

[74] M. Wu, X. Li, C. Kwoh, and S. Ng, "A core-attachment based method to detect protein complexes in ppi networks," *BMC Bioinformatics*, vol. 10, no. 1, pp. 1–16, 2009.

[75] Y. Zhang, H. Lin, Z. Yang, J. Wang, Y. Li, and B. Xu, "Protein complex prediction in large ontology attributed protein-protein interaction networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 10, no. 3, pp. 729–741, 2013.

[76] Gene Ontology Consortium, "Gene ontology consortium: going forward," *Nucleic Acids Research*, vol. 43, no. D1, pp. D1049–D1056, 2015.

[77] H. B. Borges and J. C. Nievola, "Multi-label hierarchical classification using a competitive neural network for protein function prediction," in *Proc. of the 2012 International Joint Conference on Neural Networks (IJCNN 2012) (Brisbane)*. IEEE, June 2012, pp. 1–8.

[78] S. Feng, P. Fu, and W. Zheng, "A hierarchical multi-label classification algorithm for gene function prediction," *Algorithms*, vol. 10, no. 4, pp. 138–152, 2017.

[79] F. Wu, J. Zhang, and V. Honavar, "Learning classifiers using hierarchically structured class taxonomies," in *Proc. of the 6th International Symposium on Abstraction, Reformulation, and Approximation (SARA 2005) (Airth Castle)*. Springer, July 2005, pp. 313–320.

[80] B. Chen and J. Hu, "Hierarchical multi-label classification based on oversampling and hierarchy constraint for gene function prediction," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 7, no. 2, pp. 183–189, 2012.

[81] X. Zhu and M. Bain, "B-cnn: branch convolutional neural network for hierarchical classification," *ArXiv*, vol. abs/1709.09890, 2017.

[82] C. N. Silla and A. A. Freitas, "Novel top-down approaches for hierarchical classification and their application to automatic music genre classification," in *Proc. of the 2009 IEEE International Conference on Systems, Man and Cybernetics (SMC 2009) (San Antonio)*. IEEE, October 2009, pp. 3499–3504.

[83] L. Zhang, S. K. Shah, and I. A. Kakadiaris, "Hierarchical multi-label classification using fully associative ensemble learning," *Pattern Recognition*, vol. 70, pp. 89–103, 2017.

[84] R. Cerri and A. C. P. de Carvalho, "Hierarchical multilabel classification using top-down label combination and artificial neural networks," in *Proc. of the 11th Brazilian Symposium on Neural Networks(SBRN 2010) (Sao Paulo)*. IEEE, October 2010, pp. 253–258.

[85] X. Wang, H. Zhao, and B. Lu, "Enhance top-down method with meta-classification for very large-scale hierarchical classification," in *Proc. of the 5th International Joint Conference on Natural Language Processing (IJCNLP 2011) (Chiang Mai)*, Novermber 2011, pp. 1089–1097.

[86] A. Mahdi, J. Qin, and G. Crosby, "Deepfeat: A bottom-up and top-down saliency model based on deep features of convolutional neural networks," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 12, no. 1, pp. 54–63, 2019.

[87] C. N. Silla Jr and A. A. Freitas, "A global-model naive bayes approach to the hierarchical prediction of protein functions," in *Proc. of the 9th IEEE International Conference on Data Mining (ICDM 2009) (Miami)*. IEEE, 2009, pp. 992–997.

[88] 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, pp. 31–72, 2011.

[89] 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.

[90] J. Wehrmann, R. Cerri, and R. Barros, "Hierarchical multi-label classification networks," in *Proc. of the 35th International Conference on Machine Learning(ICML 2018) (Stockholm)*, July 2018, pp. 5225–5234.

[91] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT press, 2016.

[92] A. Makhzani and B. J. Frey, "A winner-take-all method for training sparse convolutional autoencoders," *ArXiv*, vol. abs/1409.2752, 2014.

[93] ——, "Winner-take-all autoencoders," *Advances in Neural Information Processing Systems*, vol. 28, pp. 2791–2799, 2015.

[94] X. Pan, P. Rijnbeek, J. Yan, and H.-B. Shen, "Prediction of rna–protein sequence and structure binding preferences using deep convolutional and recurrent neural networks," *BMC Genomics*, vol. 19, no. 1, pp. 1–11, 2018.

[95] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *International Journal of Data Warehousing and Mining*, vol. 3, no. 3, pp. 1–13, 2007.

[96] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *ArXiv*, vol. abs/1409.1556, 2014.

[97] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[98] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *ArXiv*, vol. abs/1502.03167, 2015.

[99] S. Tokui, K. Oono, S. Hido, and J. Clayton, "Chainer: a next-generation open source framework for deep learning," in *Proc. of the 29th Conference on Neural Information Processing Systems (NIPS 2015) (Montreal)*, vol. 5, December 2015, pp. 1–6.

[100] UniProt Consortium, "Uniprot: a hub for protein information," *Nucleic Acids Research*, vol. 43, no. D1, pp. D204–D212, 2014.

[101] P. J. Thul, L. kesson, M. Wiking, D. Mahdessian, A. Geladaki, H. A. Blal, T. Alm, A. Asplund, L. Björk, L. M. Breckels *et al.*, "A subcellular map of the human proteome," *Science*, vol. 356, no. 6340, p. eaal3321, 2017.

[102] S. Hua and Z. Sun, "Support vector machine approach for protein subcellular localization prediction," *Bioinformatics*, vol. 17, no. 8, pp. 721–728, 2001.

[103] J. Shi, S. Zhang, Q. Pan, Y. Cheng, and J. Xie, "Prediction of protein subcellular localization by support vector machines using multi-scale energy and pseudo amino acid composition," *Amino Acids*, vol. 33, no. 1, pp. 69–74, 2007.

[104] L. Wei, Y. Ding, R. Su, J. Tang, and Q. Zou, "Prediction of human protein subcellular localization using deep learning," *Journal of Parallel and Distributed Computing*, vol. 117, pp. 212–217, 2018.

[105] A. C. Lorena and A. C. de Carvalho, "Protein cellular localization prediction with support vector machines and decision trees," *Computers in Biology and Medicine*, vol. 37, no. 2, pp. 115–125, 2007.

[106] H. Zhou, Y. Yang, and H. Shen, "A new subcellular localization predictor for human proteins considering the correlation of annotation features and protein multi-localization," in *Proc. of the 7th Chinese Conference on Pattern Recognition (CCPR 2016) (Chengdu)*. Springer, November 2016, pp. 499–512.

[107] ——, "Hum-mPLoc 3.0: prediction enhancement of human protein subcellular localization through modeling the hidden correlations of gene ontology and functional domain features," *Bioinformatics*, vol. 33, no. 6, pp. 843–853, 2016.

[108] I. M. Armean, K. S. Lilley, M. W. Trotter, N. C. Pilkington, and S. B. Holden, "Co-complex protein membership evaluation using maximum entropy on GO ontology and InterPro annotation," *Bioinformatics*, vol. 34, no. 11, pp. 1884–1892, 2018.

[109] S. Bandyopadhyay and K. Mallick, "A new feature vector based on gene ontology terms for protein-protein interaction prediction," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 14, no. 4, pp. 762–770, 2016.

[110] G. Montañez and Y. Cho, "Predicting false positives of protein-protein interaction data by semantic similarity measures §," *Current Bioinformatics*, vol. 8, no. 3, pp. 339–346, 2013.

[111] X. Lei, J. Zhao, H. Fujita, and A. Zhang, "Predicting essential proteins based on RNA-Seq, subcellular localization and GO annotation datasets," *Knowledge-Based Systems*, vol. 151, pp. 136–148, 2018.

[112] W. Liu, J. Liu, and J. C. Rajapakse, "Gene ontology enrichment improves performances of functional similarity of genes," *Scientific Reports*, vol. 8, no. 1, p. 12100, 2018.

[113] X. Yuan, E. Pang, K. Lin, and J. Hu, "Hierarchical multilabel classifier for gene ontology annotation using multi-head and multi-end deep cnn model," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 15, no. 7, pp. 1057–1064, 2020.

[114] A. Höglund, P. Dönnes, T. Blum, H.-W. Adolph, and O. Kohlbacher, "MultiLoc: prediction of protein subcellular localization using n-terminal targeting sequences, sequence motifs and amino acid composition," *Bioinformatics*, vol. 22, no. 10, pp. 1158–1165, 2006.

[115] G. D. Bader and C. W. Hogue, "An automated method for finding molecular complexes in large protein interaction networks," *BMC Bioinformatics*, vol. 4, no. 1, pp. 1–27, 2003.

[116] B. Adamcsek, G. Palla, I. J. Farkas, I. Derényi, and T. Vicsek, "Cfinder: locating cliques and overlapping modules in biological networks," *Bioinformatics*, vol. 22, no. 8, pp. 1021–1023, 2006.

[117] Y. Yu, L. Lin, C. Sun, X. Wang, and X. Wang, "Complex detection based on integrated properties," in *Proc. of the 18th International Conference on Neural Information Processing (ICONIP 2011) (Shanghai)*. Springer, November 2011, pp. 121–128.

[118] M. Meila, "Spectral clustering: a tutorial for the 2010's," *Handbook of Cluster Analysis*, pp. 1–23, 2016.

[119] T. Nepusz, H. Yu, and A. Paccanaro, "Detecting overlapping protein complexes in protein-protein interaction networks," *Nature Methods*, vol. 9, no. 5, pp. 471–472, 2012.

[120] L. G. Trabuco, M. J. Betts, and R. B. Russell, "Negative protein–protein interaction datasets derived from large-scale two-hybrid experiments," *Methods*, vol. 58, no. 4, pp. 343–348, 2012.

[121] X. Yuan, W. Li, K. Lin, and J. Hu, "A deep neural network based hierarchical multi-label classifier for protein function prediction," in *Proc. of the 2019 International Conference on Computer, Information and Telecommunication Systems (CITS 2019) (Beijing)*. IEEE, August 2019, pp. 1–5.

[122] X. Yuan, E. Pang, K. Lin, and J. Hu, "Deep protein subcellular localization predictor enhanced with transfer learning of go annotation," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 16, no. 4, pp. 559–567, 2021.

[123] S. Pu, J. Wong, B. Turner, E. Cho, and S. J. Wodak, "Up-to-date catalogues of yeast protein complexes," *Nucleic Acids Research*, vol. 37, no. 3, pp. 825–831, 2009.

[124] W. Mewes, Hans, S. Dietmann, D. Frishman, R. Gregory, G. Mannhaupt, K. F. Mayer, M. Münsterkötter, A. Ruepp, M. Spannagl, V. Stümpflen *et al.*, "Mips: analysis and annotation of genome information in 2007," *Nucleic Acids Research*, vol. 36, no. suppl_1, pp. D196–D201, 2008.

[125] I. Xenarios, D. W. Rice, L. Salwinski, M. K. Baron, E. M. Marcotte, and D. Eisenberg, "Dip: the database of interacting proteins," *Nucleic Acids Research*, vol. 28, no. 1, pp. 289–291, 2000.

[126] P. Blohm, G. Frishman, P. Smialowski, F. Goebels, B. Wachinger, A. Ruepp, and D. Frishman, "Negatome 2.0: a database of non-interacting proteins derived by literature mining, manual annotation and protein structure analysis," *Nucleic Acids Research*, vol. 42, no. 1, pp. 396–400, 2014.

[127] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[128] A. Hagberg, P. Swart, and C. S, Daniel, "Exploring network structure, dynamics, and function using networkx," in *Proc. of the 7th Annual Python in Science Conference (SciPy 2008) (Pasadena)*, August 2008, pp. 11–15.

Publication List

Journal Papers

J1. H. Deng, X. Yuan, Y. Tian and J. Hu, "Neural-Augmented Two-Stage Monte Carlo Tree Search with Over-Sampling for Protein Folding in HP Model", *IEEJ Trans. on Electrical and Electronic Engineering*, Vol.17, No.5, May 2022. (9 pages)

J2. X. Yuan, H. Deng and J. Hu, "Constructing a PPI Network Based on Deep Transfer Learning for Protein Complex Detection", *IEEJ Trans. on Electrical and Electronics Engineering*, Vol.17, No.3, pp.436-444, March 2022.

J3. X. Yuan, E. Pang, K. Lin and J. Hu. "Deep Protein Subcellular Localization Predictor Enhanced with Transfer Learning of GO Annotation", *IEEJ Trans. on Electrical and Electronics Engineering*, Vol. 16, No. 4, pp.559-567, April 2021.

J4. X. Yuan, E. Pang, K. Lin and J. Hu. "Hierarchical Multi-label Classification for Gene Ontology Annotation using Multi-head and Multi-end Deep CNN model", *IEEJ Trans. on Electrical and Electronics Engineering*, Vol.15, No.7, pp.1057-1064, July 2020.

Conference Paper with Review

P1. X. Yuan, H. Deng and J. Hu, "Deep Transfer Learning Based PPI Prediction for Protein Complex Detection", in *Proc. of 2021 IEEE International Conference on Systems, Man, and Cybernetics* (SMC 2021), Melbourne, pp.321-326, Oct 2021.

P2. X. Yuan, W. Li, K. Lin and J. Hu, "A Deep Neural Network Based Hierarchical Multi-Label Classifier for Protein Function Prediction", in *Proc. of the 2019 International Conference on Computer, Information and Telecommunication Systems* (CITS 2019), Beijing, pp.1-5, Aug 2019.

P3. P. Liang, X. Yuan, W. Li and J. Hu, "A Segmented Local Offset Method for Imbalanced Data Classification Using Quasi-Linear Support Vector Machine", in *Proc. of 24th International Conference on Pattern Recognition*(ICPR 2018), Beijing, pp.746-751, Aug 2018.

P4. W. Li, P. Liang, X. Yuan and J. Hu, " Non-Local Information for a Mixture of Multiple Linear Classifiers", in *Proc. of 2017 IEEE International Joint Conference on Neural Networks*(IJCNN 2017), Anchorage, pp.3741-3746, May 2017.