COM 599100 Deep Learning Final Project Report – Protein Family (Group 4)

Po-Yu Chou (team leader), Yi-Yu Zheng, Ya-Ting Yang, Yu-Chia Huang and Yu-Hsiu Huang

***Abstract*—In the field of bioinformatics, identifying protein function from amino acid sequence is a fundamental problem. With a thorough understanding of protein structures, the progress of drug design and genetic engineering will be significantly accelerated. Investigating protein functional often involves structural studies (crystallography) or biochemical studies, which require time consuming efforts. In this project, we explore how well we can represent biological function through examination of raw sequence alone. With the emerging study of deep neural networks, various fields have groundbreaking progress by incorporating the novel methods of DNN such as computer vision and natural language processing. Using a large corpus of protein sequences and their annotated protein families, many works have succeeded in classifying the structure of protein for several datasets. In this work, we experiment two deep neural network architectures—GRU and 1D-CNN to train classifiers for protein family identification for the Research Collaborators for Structural Bioinformatics (RCSB) Protein Data Bank (PDB) dataset.**

***Index Terms*—classification, deep learning, protein family.**

I. Introduction

With the development of advanced measuring techniques and instruments, we are able to retrieve a myriad of important information about the structure of biological macromolecules using X-ray crystallography, Nuclear Magnetic Resonance (NMR) spectroscopy and cryo-electron microscopy. Accurate identification of protein functions has applications in a wide variety of areas, such as understanding diseases, drug design and genetic engineering for agriculture. Nevertheless, high throughput experiments like the next generation sequencing technologies are resulting in a large number of new protein sequences uncharacterized [1].

Sequenced-based methods for protein fold recognition can be summarized into two categories: the sequence alignment methods and machine learning/deep learning based methods. The former one determines the unknown structure of sequences by calculating the alignment scores between sequences. Despite the success, the sequence alignment methods are essentially an indirect means of nearest neighbor methods, which cannot give an insightful explanation about the sequence-structure relationship. Consequently, we are motivated to propose a deep learning-based end-to-end protein structure classifier. We can expect our model not only have a decent performance in terms of classification accuracy but also obtain meaningful features extracted automatically from the neural networks without the bioinformatics expertise.

Po-Yu Chou (105061110) [†] Yi-Yu Zheng (105061108) [†]
Ya-Ting Yang (105061210) [†] Yu-Chia Huang (105061236) [†]
Yu-Hsiu Huang (104061249) [†]
[†]Department of Electrical Engineering, National Tsing Hua University

II. Material and Method

For the data preprocessing phase, we first merge 'pd_data_no_dups.csv' and 'pd_data_seq.csv' on 'structureId'. However, after inspecting the dataset, we found that there are a plenty of structures only having few instances, therefore, we decide to use the top-10 most frequently appeared classification types at first. Second, as the figure 1 shows, the instances in each class are quite imbalanced. In order to eliminate the bias toward any class, we reduce the instance on the top-3 most frequently appeared classification types and use it as the first dataset (10 classes). After that, we also try to discard some lengthy sequences (sequence length above 2000) and only keep classification classes which have more than 1000 instances as the second dataset (34 classes).

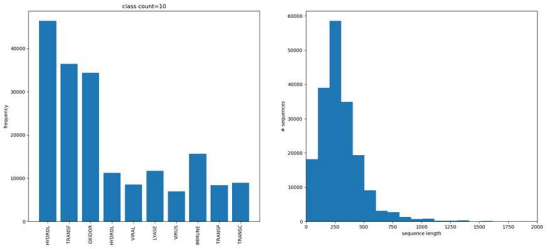

Figure 1. Raw top 10 class instance distribution

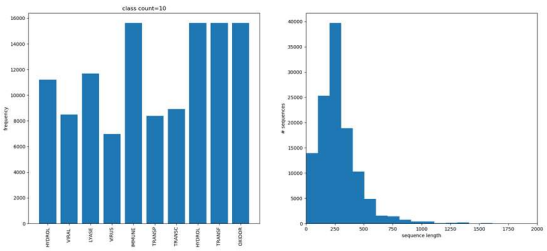

Figure 2. Processed top 10 class instance distribution

A. 1d-CNN Approach without n-gram

number of classes: 10, 34

hyper-parameter	**value**
maximum length	500

learning rate	0.001
embedding dimension	11
batch size	128
number of epochs	16 (early stop)
optimizer	Adam
loss function	categorical cross entropy

operation
Word Embedding
Conv1D(filters=256, kernel_size=6, activation='relu')
MaxPooling1D(pool_size=2)
Conv1D(filters=128, kernel_size=3, activation='relu')
MaxPooling1D(pool_size=2)
Flatten
Dense(256, activation='relu')
Dense(10 or 34, activation='softmax')

B. 1d-CNN Approach with 4-gram

number of classes: 10

hyper-parameter	**value**
maximum length	1024
learning rate	0.001
embedding dimension	22
batch size	256
number of epochs	12
optimizer	Adam
loss function	categorical cross entropy

operation
Word Embedding
Conv1D(filters=128, kernel_size=6, activation='relu')
MaxPooling1D(pool_size=2)
Conv1D(filters=64, kernel_size=3, activation='relu')
MaxPooling1D(pool_size=2)
Flatten
Dense(256, activation='relu')
Dense(10, activation='softmax')

C. 1d-CNN Approach with 4-gram

number of classes: 34

hyper-parameter	**value**
maximum length	2000
learning rate	0.001
embedding dimension	50
batch size	256
number of epochs	12
optimizer	Adam
loss function	categorical cross entropy

operation
Word Embedding
Conv1D(filters=128, kernel_size=6, activation='relu')
MaxPooling1D(pool_size=2)
Conv1D(filters=64, kernel_size=3, activation='relu')
MaxPooling1D(pool_size=2)
Flatten
Dense(256, activation='relu')
Dense(34, activation='softmax')

D. GRU Approach with 3-gram

number of classes: 10, 34

hyper-parameter	**value**
maximum length	512
learning rate	0.001
embedding dimension	100
number of hidden units	100
dropout rate	0.5
batch size	512
number of epochs	50 (early stop)
optimizer	Adam
loss function	categorical cross entropy

operation
Word Embedding
GRU
Dropout
Dense(256, activation='relu')
Dense(10 or 34, activation='softmax')

E. CNN+GRU Approach with 3-gram

number of classes: 10, 34

hyper-parameter	**value**
maximum length	256
learning rate	0.0005
embedding dimension	100
dropout rate	0.5
batch size	256
number of epochs	40 (early stop)
optimizer	Adam
loss function	categorical cross entropy

operation
Word Embedding
Conv1D(filters=128, kernel_size=6, activation='relu')
MaxPooling1D(pool_size=2)
Conv1D(filters=64, kernel_size=3, activation='relu')
MaxPooling1D(pool_size=2)
GRU
Dropout
Dense(256, activation='relu')
Dense(10 or 34, activation='sigmoid')

III. EVALUATION

A. *Simulation Environment*

OS: Windows 10
Framework: Keras using Tensorflow backend
CPU: Intel Core i5-4460 3.20Ghz
GPU: Nvidia GTX970

B. *Results*

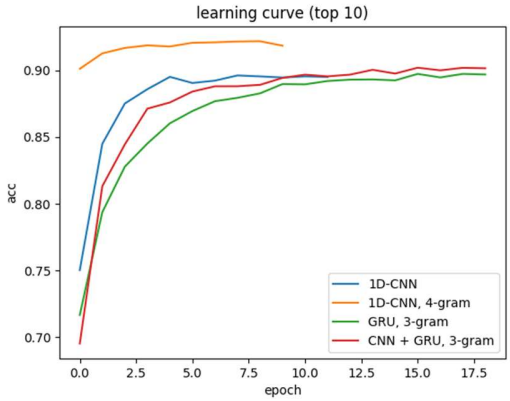

Figure 3. Learning curve (10 classes)

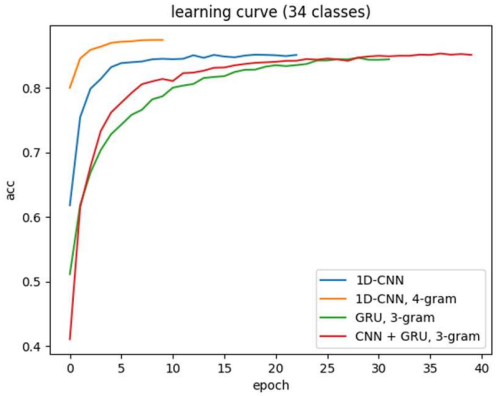

Figure 4. Learning curve (34 classes)

1d CNN without n-gram (10 classes)

	Precision	Recall	F1-score
Macro Average	0.90	0.90	0.90
Weighted Average	0.90	0.90	0.90
Accuracy		0.90	

1d CNN without n-gram (34 classes)

	Precision	Recall	F1-score
Macro Average	0.83	0.80	0.81
Weighted Average	0.85	0.85	0.85
Accuracy		0.85	

1d CNN with 4-gram (10 classes)

	Precision	Recall	F1-score
Macro Average	0.94	0.93	0.93
Weighted Average	0.94	0.94	0.94
Accuracy		0.94	

1d CNN with 4-gram (34 classes)

	Precision	Recall	F1-score
Macro Average	0.85	0.83	0.84
Weighted Average	0.88	0.88	0.88
Accuracy		0.88	

GRU+CNN with 3-gram (10 classes)

	Precision	Recall	F1-score
Macro Average	0.90	0.91	0.90
Weighted Average	0.90	0.90	0.90
Accuracy		0.90	

GRU +CNN with 3-gram (34 classes)

	Precision	Recall	F1-score
Macro Average	0.81	0.79	0.80
Weighted Average	0.85	0.85	0.85
Accuracy		0.85	

GRU with 3-gram (10 classes)

	Precision	Recall	F1-score
Macro Average	0.90	0.90	0.90
Weighted Average	0.90	0.90	0.90
Accuracy		0.90	

GRU with 3-gram (34 classes)

	Precision	Recall	F1-score
Macro Average	0.81	0.79	0.80
Weighted Average	0.85	0.84	0.84
Accuracy		0.84	

sklearn.naive_bayes+n-gram (n=4-5) (10 classes)

	Precision	Recall	F1-score
Micro Accuracy	0.91	0.91	0.91
Macro Accuracy	0.89	0.91	0.90
Weighted Average	0.92	0.91	0.91

sklearn.naive_bayes+n-gram (n=4-5) (34 classes)

	Precision	Recall	F1-score
Micro Accuracy	0.83	0.83	0.83
Macro Accuracy	0.80	0.80	0.80
Weighted Average	0.84	0.83	0.83

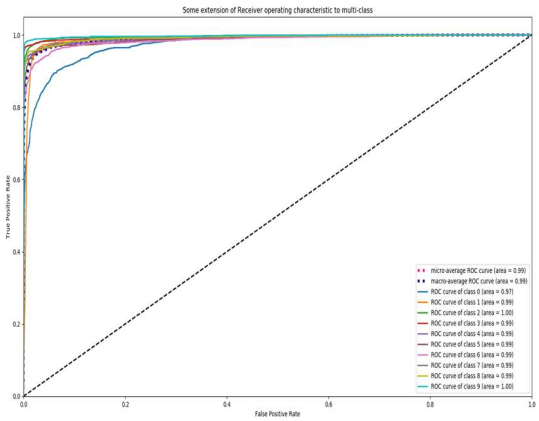

Figure 5. ROCAUC for 1d CNN with 4-gram (10 classes)

C. *Personal Simulation*

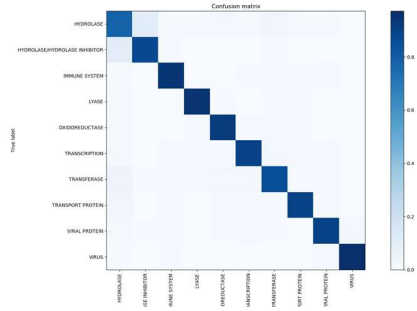

Figure 6. Confusion matrix for 1d CNN without n-gram (10 classes)

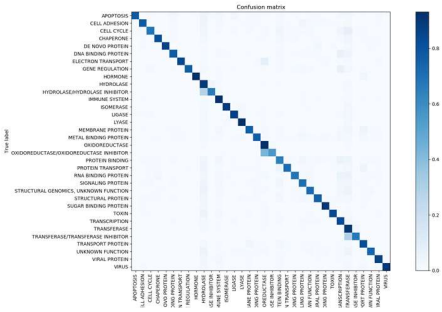

Figure 7. Confusion matrix for 1d CNN without n-gram (34 classes)

	precision	recall	f1-score	support
HYDROLASE	0.78	0.78	0.78	3032
HYDROLASE/HYDROLASE INHIBITOR	0.85	0.87	0.86	2190
IMMUNE SYSTEM	0.95	0.95	0.95	3169
LYASE	0.94	0.95	0.95	2319
OXIDOREDUCTASE	0.92	0.93	0.93	3191
TRANSCRIPTION	0.89	0.90	0.90	1753
TRANSFERASE	0.89	0.86	0.87	3110
TRANSPORT PROTEIN	0.92	0.90	0.91	1642
VIRAL PROTEIN	0.89	0.90	0.90	1753
VIRUS	0.96	0.97	0.97	1465
micro avg	0.90	0.90	0.90	23624
macro avg	0.90	0.90	0.90	23624
weighted avg	0.90	0.90	0.90	23624

Figure 8. Verbose metrics for 1d CNN without n-gram (10 classes)

IV. Discussion

From figure 3 and 4, we can see that pure CNN models dominate the GRU models on both 10 classes and 34 classes datasets. My interpretation is that: secondary or higher structures are determined by local amino acid sequences instead of long and separate parts. This can explain why convolution can have better result, because it only captures local spatial information. Unlike GRU, it can have long-term memory. As for why CNN with 4-gram can outperform that without n-gram, my guess is that it will be more meaningful to interpret amino acids in small group instead of separately. The optimal n for the n-gram may be determined with some domain knowledge.

In this project, we are faced with some difficulty. First, the lack of domain knowledge. Although we can simply treat it as a classification problem for deep learning, I am still convinced that with some bioinformatics background, we may have more tricks to do on the preprocessing of sequences, or , as discussed above, determine the network structure based on prior knowledge. Besides that, we do not have sufficient computing power to support our training process. Especially for the GRU models, which is CPU-intensive, training one epoch will take us around 5 minutes. Therefore, with limiting time, GRU models may not be fine-tuned, causing the performance gap between GRU and CNN models.

V. Work Distribution

Po-Yu Chou: 1d CNN without n-gram
Ya-Ting Yang: 1d CNN with 4-gram, data preprocessing
Yu-Hsiu Huang: GRU +CNN with 3-gram
Yi-Yu Zheng, Yu-Chia Huang:
GRU with 3-gram, data visualization