[[1]](#footnote-1)

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.

# INTRODUCTION

W

ith 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.

# Material and Method

For data preprocessing, we first select ‘proteins’, choose ‘sequence’ and ‘classification’, then combine two datasets using ‘structureId’. However, there are lots of classification types only have few instances, therefore, we decide to use the top-10 most frequently appeared classification types at first. Then, during the process of preprocessing, we find that the dataset is somehow imbalanced, as a result, we perform under sampling 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 too-long sequences and only keep classification types which have more than 1000 instances as the second dataset (34 classes) we use.

In order to create a distributed representation of our protein sequences, we represent each sequence as a series of overlapping n-grams (a block of 3 or 4 amino acids) and create a distributed representation of each n-gram using vectors trained from the model. The data was later split into training/validation folds at an 80/20 ratio preserving class stratification.

For the methods, from the reference paper [2], Protein Family Classification with Neural Networks, Gated Recurrent Neural Networks perform pretty well on the protein family classification, and thus we would apply this method to complete our task.

Gated Recurrent Neural Networks extend recurrent neural networks (RNNs) by using gated recurrent units (GRUs). GRUs consist of two additional gates, an update gate and a reset gate. The reset gate determines how much of the previous hidden state is used before the nonlinear activation. The update gate determines how much of the new memory content is used with the previous hidden state to determine the new hidden state. Together, these two gates allow long or short term dependencies to be expressed.

When training the model, we would apply maxpooling over all hidden layers’ outputs of the forward net and backward net, respectively, stack the maxpooled outputs before feeding into the softmax layer, and experiment with a number of learning rates and method of dropout.

From the reference paper [3], Protein Family Classification with one dimensional Convolutional Neural Networks also get pretty well performance, therefore, we would apply 1d-CNN on this task and compare it with other methods.

CNNs usually share the same characteristics and follow the same approach, no matter of their dimension. The key difference is the dimensionality of the input data and how the feature detector (or filter) slides across the input data. A 1d-CNN is very effective when expecting to derive interesting features from fixed-length segments of the overall data set, often being applied to time sequence of sensor data, signal data over fixed-length period, and NLP.

We then feed them into several models constructed by different methods as follows:

## 1d-CNN Approach without Gram

number of classes: 10

|  |  |
| --- | --- |
| hyper-parameter | value |
| maximum length | 500 |
| learning rate | 0.001 |
| embedding size | 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, activation='softmax') |

## 1d-CNN Approach with 4-Gram

number of classes: 34

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

|  |
| --- |
| operation |
| word embedding (embedding dim=50) |
| 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') |

## 1d-CNN Approach with 4-Gram

number of classes: 10

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

|  |
| --- |
| operation |
| word embedding (embedding dim=22) |
| 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') |

## GRU Approach with 3-Gram

number of classes: 34, 10

|  |  |
| --- | --- |
| hyper-parameter | value |
| maximum length | 512 |
| learning rate | 0.001 |
| embedding size | 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(34, activation='softmax') |

## GRU+CNN Approach with 3-Gram

number of classes: 10, 34

|  |  |
| --- | --- |
| hyper-parameter | value |
| maximum length | 256 |
| learning rate | 0.0005 |
| embedding size | 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(34), activation='sigmoid') |

# Evaluation

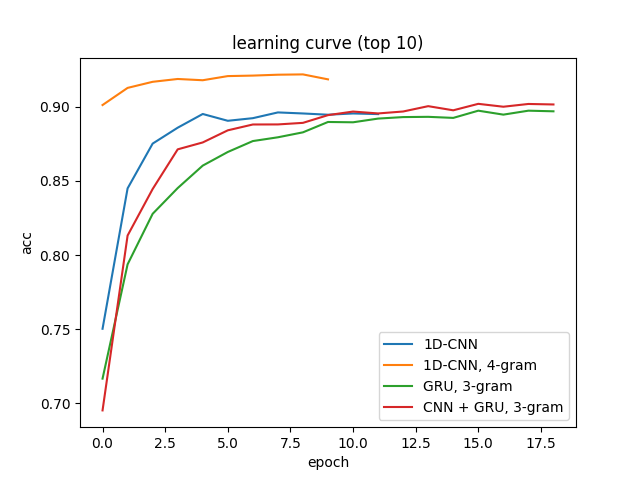
1. AUCROC
2. ACCURACY

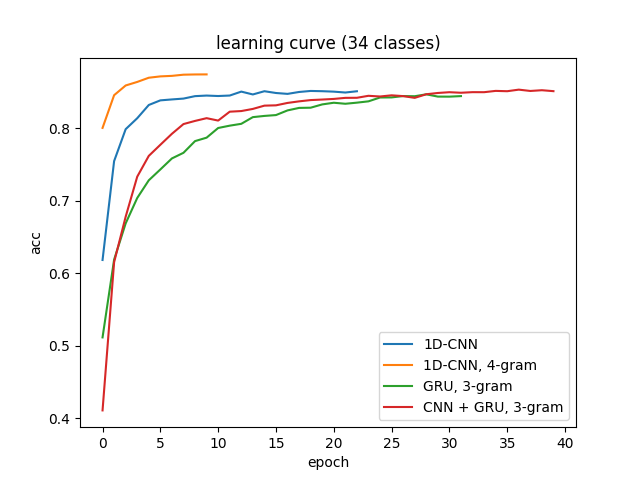
For evaluation of our model, we calculate several performance measures, such as: precision, recall, accuracy, and F1 score defined as:

Where tp are the number of true positives, fp are the number of false positives, tn are the number of true negatives and fn are the number of false negatives

According to “Protein Family Collection with Neural Network, the F1-score of this project can reach 0.948452 (using GRU); according to “DeepSF: deep convolutional neural network for mapping protein sequences to folds”, for some specific folds, the accuracy can reach 97.5%. However the aforementioned results are evaluated on different datasets, so the actual performance of our work could potentially be better after delicate fine tuning with respect to the PDB dataset.

# Results





1DCNN\_4gram\_34

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

1DCNN\_4gram\_top10

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

CNN\_GRU\_3gram\_34

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

CNN\_GRU\_3gram\_top10

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

GRU\_3gram\_top10

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

GRU\_3gram\_34

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

naïve\_bayes\_ngram\_34class (ngram=4-5)

|  |  |  |  |
| --- | --- | --- | --- |
|  | 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 |

naïve\_bayes\_ngram\_top10 (ngram=4-5)

|  |  |  |  |
| --- | --- | --- | --- |
|  | 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 |

# Discussion

The first difficulty we may encounter is sequence encoding (choosing between different embedding methods) because different numeric representation of the amino acid sequence may influence our model performance. The second difficulty is that there may be a lot of information and classes of the protein families, so we need to decide which to keep and which to delete. Last but not least, no matter which model we use (GRU, 1dCNN,……), we do require a lot of computation resources to finish the training part.

# References

1. Nauman, M., Rehman, H. U., Politano, G., & Benso, A. (2018). Beyond Homology Transfer: Deep Learning for Automated Annotation of Proteins. Journal of Grid Computing. doi:10.1007/s10723-018-9450-6
2. Timothy K. Lee, Tuan Nguyen, "Protein Family Classification with Neural Networks", 2016.
3. Hou J, Adhikari B, Cheng J. DeepSF: deep convolutional neural network for mapping protein sequences to folds. Bioinformatics. 2017;34(8):1295-1303.

1. 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 [↑](#footnote-ref-1)