ALTEGRAD 2022-2023 Data Challenge: Proteins classification

Kaggle Team: Ruan (Sicheng MAO and **Yang ZHANG** and **Yunhao CHEN)**
{sicheng.mao,yang.zhang,yunhao.chen}@polytechnique.edu

1 Introduction

Proteins are large bio-molecules and composed of single or multiple chains of amino acids. When amino acids bind together, they form a long chain called a polypeptide that can be represented by the amino sequence. The sequence of amino acids then begin to fold, creating the 3D shape of the protein. This structure determines its specific chemical functionality, but the exact details of this process are not yet fully understood. The goal of this project is to classify the proteins using both the sequential information of polypetide and the structure information of the protein.

2 Dataset

The dataset contains 6111 proteins information which is composed of two parts: **sequence.txt** provides the amino acid chains of these proteins and the other five files provide structure information of these **6,111** proteins in form of undirected graphs, where nodes model the amino acids and edges model the chemical bonds. There are in total **1,572,264** nodes, each attributed by an 86-dim vector encoding spatial and bio-chemical properties of the amino acid. There are in total **15,213,222** edges , each attributed by a 5-dim vector encoding geometrical and chemical properties of the bond.

Within a brief data exploration, it is worth noting that the each node of the graph are added with a self-loop and there is a very small proportion of edges which doesn't belong to any of the four types.

3 Metric and Task

The task is to use the sequential and structural information of those proteins provided above to classify them into **18** different classes. The performance is evaluated by the **multi-class cross entropy** loss.

4 Pipeline

We divide our project's pipeline into three steps, namely the feature extraction and the classification. We mainly focus on first separately extract sequential embeddings and structural embeddings perform classification algorithms on them and finally try to combine them together.

As to the classification part, we mainly depend on the automl tool **AutoGluon** to take care of all the model selection and evaluation issues. In the following paragraphs, we will introduce carefully the feature extraction methods we used and give a brief introduction to AutoGluon.

5 Sequential feature extraction

5.1 TF-IDF

As the baseline indicates, we can simply view the protein sequence as plain text and apply traditional NLP techniques, e.g. we can create TF-IDF embedding for each protein by the corresponding sklearn function.

5.2 Protvec

We can use more advanced pretrained NLP model with domain specific knowledge. For example. The paper [1] proposes an approach to compute dense distributed representation of protein sequence. A skip-gram model has been trained on database *Swiss-Prot*. It achieves an average family classification accuracy of 94% on two disordered protein databases: *DisProt* as well as a database of disordered regions of phenylalanine-glycine nucleoporins (FG-Nups). By extracting the vector embedding of protein in *Protvec* and with the help of classifiers chosen by Autogluon, we obtain **1.48** as score. It is not really satisfactory. But we should notice that the paper [1] only considers the disordered proteins. We guess the drop of performance may result from the lack of secondary and ordered three-dimensional structure.

5.3 ProteinBERT

To moving on, we have tried another interesting paper which introduces a self-supervised language model specifically designed for proteins [3]. Inspired by language modeling, especially by BERT architecture, this paper combines language modeling with a novel task of Gene Ontology (GO) annotation prediction as pretraining scheme. It aims at capturing local and global representations of proteins. The local representation of a protein encodes every amino acids while the global representation encodes the whole protein. Due to GPU memory limit, here we only use the global representation as our extracted

features. A detailed introduction to ProteinBERT is attached in the appendix.

We took use of the protein embedding obtained in the pretraining process of ProteinBERT. Then the classification via Autogluon was performed. Generally speaking, the performance has been obviously improved. We have tested different embedding dimensions and we arrived at a best score **0.89**.

Compared to Protvec, one advantage of ProteinBERT is the use of GO annotations, which contains much information about protein function and their relationship. There are mutual information flows between local and global representations. Those flows allow the protein embedding to provide more information about functions and potentially structure of protein.

5.4 AlphaFold2

Another proposition is model AlphaFold2 which aims at predicting a proteins 3D structure from its amino acid sequence [4]. It was proposed in 2021 by DeepMind and published on Nature. AlphaFold incorporates physical and biological knowledge about protein structure, leveraging multi-sequence alignments, into the design of the deep learning algorithm. It achieves historical improvement, with atomic accuracy for prediction of protein 3D structures.

As illustrated in Fig.1, the architecture of AlphaFold2 is extremely complex. Generally speaking, the whole model consists of three parts: feature extraction, encoder and decoder. Given a human protein sequence, in the first part, it searches the similar sequences (of human or other species) in a genetic database, which is called multiple sequence alignments (MSA). In parallel, the interaction relationship between pairs of amino acids is considered, in order to capture the structure information. The encoder contains 48 transformer blocks and outputs the vector embedding of given protein as well as the embedding of pairing relationship between amino acids. In the last part, the decoder makes use of the above vector embeddings to predict the 3D protein structure.

Ideally, we'd like to extract the vector embedding of the input protein sequence, produced by the encoder. However, the whole model and required database is so large that exceeds our available compute resources. For instance, AlphaFold2 needs multiple genetic databases to do MSA. The total download size for the full unzipped databases is 2.62 TB, not to mention the huge model parameters. Due to the limitations of storage and compute resources, we gave up the use of AlphaFold2. But we can reasonably expect a promising result from this approach.

5.5 ESMFold

Developed by Meta AI, ESMFold is another competitive transformer-based model to predict protein fold problem. Comparing to Alphafold2, it gets rid of multiple sequence alignments and structural information, can extract features directly from single protein sequence. We use the minium pretrained ESMfold encoder from Hugging Face to extract the feature for every amino acids in a protein, then we simply sum them up to get a global representation for the protein. However, features extracted in this way doesn't get better performance than ProteinBERT. We also try to use the amino acid embedding as node attribute for our structural feature extraction step but we find the shape of the embeddings is always two dimension longer than the total length of the protein sequence and we don't know how to make alignment.

6 structural feature extraction

6.1 GCN

Now we are interested in the structural information of the data, we first use a GCN(Graph Convolutional Network)[5] model, which consists of two convolutional layers for message passing, two linear layers for classification, and a global add pool as a readout layer. The GCN aims to learn node representations via aggregating node features of the neighborhood through graph convolution, the one-layer message passing can be written as:

$$H^{(k+1)} = \sigma(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(k)}W^{(k)})$$

where $\widetilde{A} = A + I$ and $\widetilde{D}is$ a diagonal matrix such that $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$. The performance of this model is 1.9 and is very close to the baseline, the reason for this low performance could be that the protein is a very complex structure, simply apply add aggregation is not enough, and small layer GCN can't have large-scale message passing and deep GCN will have smooth problem which means all features will end up the same.

6.2 GAT

Then we move to GAT(Graph Attention Network)[7] model, the previous GCN model equally treats every node, and GAT uses an attention mechanism to give importance to every node.

$$x_i' = \alpha_{i,i}\theta x_i + \sum \alpha_{i,j}\theta x_j$$

where the attention coefficient is :

$$\alpha_{i,j} = \frac{exp(LeakyReLU(a^T[\theta x_i || \theta x_j]))}{\sum_{k\in N(i)\cup(i)} exp(LeakyReLU(a^T[\theta x_i || \theta x_k]))}$$

In our model, we use only one GAT layer and a MLP layer for classification, and a global add pool for the readout function. The score we get is not very satisfying too, and very similar to the GCN model: 1.8, this might again be because of the readout function is not efficient, and they both use only the connectivity of the graph but the edge feature is not considered.

6.3 GIN

Then we tried a very recent network: GIN(Graph Isomorphism Network)[8], this model is generalized from

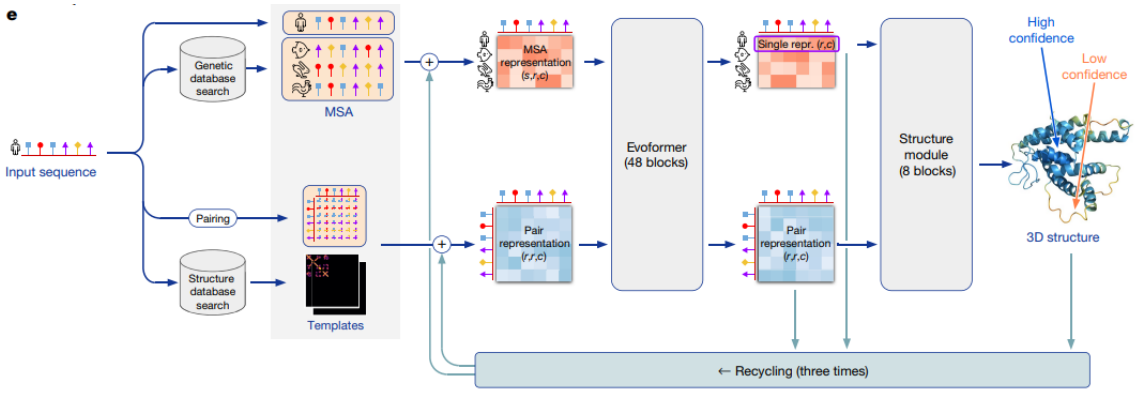

Figure 1: AlphaFold2 architecture

Weisfeiler-Lehman(WL) test resulting in the strongest discrimination among GNNs. The model can be written as follows:

$$h_v^{(k)} = MLP^{(k)}((1+\varepsilon^{(k)}h_v^{(k-1)} + \sum_{u\in N(v)} h_u^{(k-1)})$$

The MLP can approximate any function and could be learned as an injective function and ε is a learnable parameter or scalar. This can ensure that a deep GNN can map any graphs that the WL test of isomorphism decides as non-isomorphic, to different embeddings according to the theorem in paper[8]. And for the read-out function, it takes sum + concat to get the features:

$$h_G = CONCAT(SUM(\{h_v^{(k)}|v \in G)|k = 0,1,...,K)$$

In our model, we use two layers of GIN and two layers of linear layer for classification, we get a slightly better score of 1.7, but still can't be comparable to the sequential model. This problem of the GNN models could be the design of the networks, and the use of global add aggregation.

7 classification and hypertuning

Once we get the feature embeddings, we directly put them into AutoGluon tabular predictor to perform classification task. Developed by AWS team, AutoGluon is a autoML tool aims at facilitating model selection and ensemble. Therefore we don't need to concern too much about the hypertuning and focus on trying different model architectures.

Here is a snapshot of the result of AutoGluon predictor.

8 Evaluation

We summarize all our trial results here in a table.

We use autogluon for all our extracted feature except for the finetuning part. We notice that the best performed model is the ProteinBert with 200 SVD dimension reduction, and the best single classifier reported by autogluon is CatBoost, the final predictor is an ensembled one over all the tried classifiers as illustrated in the previous picture.

	Feature extraction	CE Loss
Sequence	TFIDF + SVD 100	1.19022
	Protvec + SVD 100	1.48465
	ESM + SVD 200	0.99529
	ProteinBert Finetuing	1.10325
	ProteinBert + SVD 100	0.94205
	ProteinBert + SVD 200	0.89582
	ProteinBert + SVD 300	0.90897
Structure	2 layer GCN + mean agg	1.88668
	2 layer GAT + mean agg	1.81989
	2 layer GIN + mean agg	1.84116
Both	ProteinBert + GCN + SVD 300	0.91830

We also find the GNNs don't outperform better than the , this may be due to the fact that the global pooling function is too simple, which takes all node embedding into account while all amino acids don't contribute equally to the protein functionality. We try to use a smarter way of global pooling method as indicated in [9] to reduce noise in global representation but we failed to run the code. We also notice that a combination of sequential and structural features doesn't yield a better result, we believe it's because the extracted structural feature is too bad to drag down the quality of the sequential embedding.

9 Conclusion

We basically use pretrained language based model to extract sequential features of proteins and use the graph neural networks to extract its structural features. We reached a cross entropy loss around **0.89** mainly thanks to ProteinBert and autogluon. However, we find the quality of sequential feature is way better than the structural features. There may be two reasons: 1. the primary structure of protein(the polypetide chain) have a greater importance than the second or higher order of structure to the functionality. 2. the mean readout over all node embedding is too noisy to extract any useful information for protein classification. Possible improvements may be using smarter readout function for GNN to extract structural information and it is also worthwhile to trying more advanced architecture such as graph transformers. [6]

	model	score_val	pred_time_val	fit_time	pred_time_val_marginal	fit_time_marginal	stack_level	can_infer	fit_order
0	WeightedEnsemble_L3	-0.891835	8.574015	6110.735480	0.001576	4.565410	3	True	26
1	CatBoost_BAG_L2	-0.919700	4.754232	5664.494449	0.105590	1903.196111	2	True	20
2	LightGBMXT_BAG_L2	-0.927140	4.811049	3913.547895	0.162406	152.249557	2	True	16
3	WeightedEnsemble_L2	-0.930269	2.540211	3183.077281	0.001617	5.116302	2	True	14
4	XGBoost_BAG_L2	-0.939213	5.009479	4027.771781	0.360837	266.473444	2	True	23
5	LightGBM_BAG_L2	-0.969090	4.797922	3995.934522	0.149280	234.636185	2	True	17
6	CatBoost_BAG_L1	-0.999276	0.116058	3022.558098	0.116058	3022.558098	1	True	8
7	NeuralNetFastAI_BAG_L2	-1.023223	6.562722	3771.529728	1.914080	10.231391	2	True	15
8	LightGBMLarge_BAG_L2	-1.035384	5.049778	4464.105117	0.401136	702.806779	2	True	25
9	NeuralNetTorch_BAG_L2	-1.058477	6.358984	3772.120343	1.710342	10.822006	2	True	24
10	ExtraTreesGini_BAG_L2	-1.078011	5.019598	3761.968069	0.370956	0.669732	2	True	21
11	RandomForestEntr_BAG_L2	-1.085409	4.936126	3770.630255	0.287484	9.331918	2	True	19
12	RandomForestGini_BAG_L2	-1.091064	5.006528	3763.367219	0.357886	2.068882	2	True	18
13	ExtraTreesEntr_BAG_L2	-1.091783	5.013200	3761.949035	0.364558	0.650698	2	True	22
14	XGBoost_BAG_L1	-1.102782	0.451378	128.769110	0.451378	128.769110	1	True	11
15	LightGBMXT_BAG_L1	-1.137872	0.284004	97.370624	0.284004	97.370624	1	True	4
16	NeuralNetFastAI_BAG_L1	-1.184091	0.891164	7.364564	0.891164	7.364564	1	True	3
17	LightGBM_BAG_L1	-1.209016	0.241845	134.024144	0.241845	134.024144	1	True	5
18	LightGBMLarge_BAG_L1	-1.260780	0.541614	350.762837	0.541614	350.762837	1	True	13
19	NeuralNetTorch_BAG_L1	-1.287691	0.169458	8.474578	0.169458	8.474578	1	True	12
20	RandomForestEntr_BAG_L1	-1.323998	0.223407	9.300573	0.223407	9.300573	1	True	7
21	RandomForestGini_BAG_L1	-1.326605	0.228892	1.465503	0.228892	1.465503	1	True	6
22	ExtraTreesGini_BAG_L1	-1.400114	0.245759	0.556181	0.245759	0.556181	1	True	9
23	ExtraTreesEntr_BAG_L1	-1.409887	0.251137	0.594346	0.251137	0.594346	1	True	10
24	KNeighborsDist_BAG_L1	-2.470747	0.458238	0.028554	0.458238	0.028554	1	True	2
25	KNeighborsUnif_BAG_L1	-2.527403	0.545687	0.029226	0.545687	0.029226	1	True	1

Figure 2: autogluon leaderboard of predictors

10 Appendix

10.1 Gene Ontology

The Gene Ontology (GO) is a major bioinformatics initiative to unify the representation of gene and gene product attributes across all species. The ontology covers three domains:

- cellular component: the parts of a cell or its extracellular environment;
- molecular function: the elemental activities of a gene product at the molecular level, such as binding or catalysis;
- biological process: operations or sets of molecular events with a defined beginning and end, pertinent to the functioning of integrated living units.

The GO ontology is structured as a directed acyclic graph, and each term has defined relationships to one or more other terms in the same domain, and sometimes to other domains. For now, there are three types of relationship between the GO items: *is_a*, *part_of* and *regulates*. For example, Fig.3 illustrates the relationship: A *is a* B; B is *part of* C; we can infer that A is *part of* C

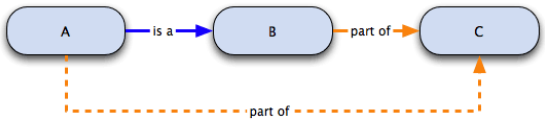

Figure 3: Diagram of relationship between GO items

10.2 ProteinBERT

10.2.1 Datasets

ProteinBERT was pretrained on **106M** proteins derived from *UniProtKB/UniRef90*. For each protein, its amino-acid sequence and associated GO annotations are extracted according to *UniProtKB*. More importantly, only the **8943** most frequent GO annotations that occurred at least 100 times in UniRef90 are considered.

10.2.2 Sequence and annotation encoding

Protein sequences were encoded as sequences of integer tokens, including 20 standard amino acids, undefined amino-acid (*X*) and 3 additional tokens (*START, END* and *PAD*).

GO annotations of every sequence were encoded as a binary vector of fixed size (8943), where all entries are zeros except those corresponding to GO annotation associated with the protein.

10.2.3 Pre-training task

The idea of using a masked language modeling as pre-training task has been borrowed here. More precisely, the input protein sequences and GO annotations are corrupted (tokens and annotations are randomly replaced). In summary, the pre-training is a dual task, where the model has to recover both the protein sequence and its known GO annotations.

10.2.4 Deep learning architecture

ProteinBERT is a type of a denoising autoencoder. The two inputs (and outputs) of ProteinBERT are (i) protein sequences and (ii) GO annotations.

As illustrated in Fig.4, the model consists of two parallel paths: one for local representations (for protein sequences) and the other for global representations (for GO annotations). The local representations are 3D tensors of shape $B \times L \times d_{local}$ where B is the batch size, L is the mini-batch sequence length, and d_{local} is dimension for the local representations. The global representations are 2D tensors of shape $B \times d_{global}$.

The local and global representations are processed by a series of six transformer-like blocks with skip connections and layer normalizations between their hidden layers. The

information flow between the local and global representations occurs through broadcast fully connected layers (from the global to the local representations) and global attention layers (from the local to the global representations). The global attention is proposed by Bahdanau in [2] to replace the traditional Seq2Seq architecture.

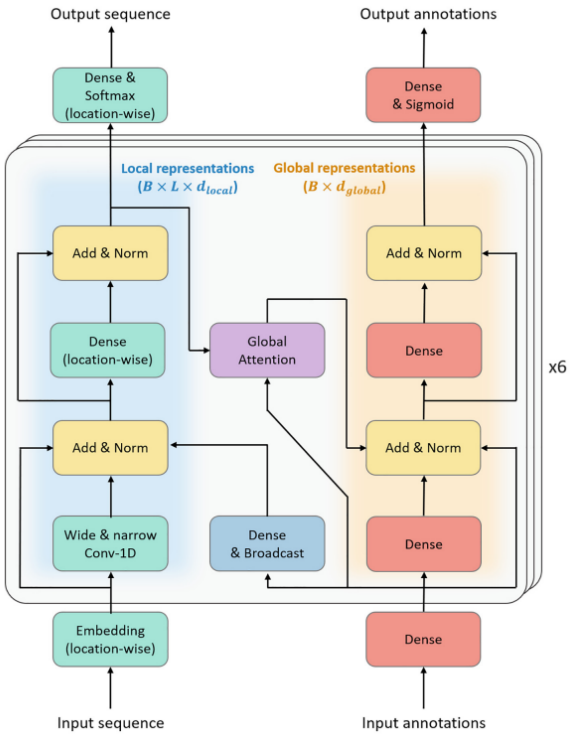

Figure 4: ProteinBERT architecture

References

[1] E. Asgari, M. R. K. Mofrad, Continuous distributed representation of biological sequences for deep proteomics and genomics, PLOS ONE 10 (11) (2015) 1–15.
URL `https://doi.org/10.1371/journal.pone.0141287`

[2] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, arXiv preprint arXiv:1409.0473.

[3] N. Brandes, D. Ofer, Y. Peleg, N. Rappoport, M. Linial, ProteinBERT: a universal deep-learning model of protein sequence and function, Bioinformatics 38 (8) (2022) 2102–2110.
URL `https://doi.org/10.1093/bioinformatics/btac020`

[4] J. M. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Zídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. A. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, D. Hassabis, Highly accurate protein structure prediction with alphafold, Nature 596 (2021) 583 – 589.

[5] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907.

[6] D. Q. Nguyen, T. D. Nguyen, D. Phung, Universal graph transformer self-attention networks, in: Companion Proceedings of the Web Conference 2022 (WWW '22 Companion), 2022.

[7] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, arXiv preprint arXiv:1710.10903.

[8] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks?, arXiv preprint arXiv:1810.00826.

[9] Z. Zhang, J. Bu, M. Ester, J. Zhang, C. Yao, Z. Yu, C. Wang, Hierarchical graph pooling with structure learning, arXiv preprint arXiv:1911.05954.