Project Title: Protein Function Prediction from Sequential Data

CE7412: Computational and Systems Biology by Professor Jagath Rajapakse

Keywords: Protein Function Prediction, Sequence Model, Multi-label Classification

1 INTRODUCTION

Protein function prediction is a key area of research in bioinformatics that involves the use of computational methods to predict the biological functions of proteins. It is an important field because understanding the function of proteins is essential for understanding cellular processes and disease mechanisms, as well as for the development of new drugs and therapies. There are multiple approaches to predict protein functions, such as using sequence-based features, protein-protein interaction networks, protein structures, and biomedical literature. Among them, the protein's amino acid sequence is the most available information for most proteins. Therefore, methods that can precisely forecast protein functions solely based on sequence could be the most comprehensive and widely applicable to proteins that lack extensive research. Conventional techniques aim to identify comparable sequences that possess known functional annotations and distinct sequence motifs linked with specific functions. Recent machine learning methods demonstrate more advantages in scalability for handling large-scale problems, automated feature extraction without the need for human knowledge, robustness for noisy data, and adaptability for different datasets. DeepGO [1] is one of the first deep learning models which can predict protein functions using the protein amino acid sequence and interaction networks. However, DeepGO has several restrictions on the sequence length, missing features and number of predicted classes. DeepGOPlus [2] has been proposed to overcome these issues, which is based on Convolutional Neural Network (CNN) using multiple 1D convolution layers. However, based on what has been found in the field of Natural Language Processing (NLP), CNNs are not the most effective model handling sequential data. Models like Recurrent Neural Network (RNN), Long-Term-Short-Memory (LSTM), and transformer demonstrate more promising performance than CNNs for sequential data. Therefore, this project aims to benchmark advanced sequence learning models as mentioned and their hybrid architectures against DeepGOPlus. Performance metrics like F-score, semantic distance, and precision-recall area are compared among various models using CAFA3 [3] and SwissProt [4] datasets. Their advantages and disadvantages are analyzed at the end of the report.

2 DATASETS

CAFA3 (Critical Assessment of Function Annotation) is a dataset used for the third edition of the CAFA challenge, a biennial competition that evaluates computational methods for predicting protein function. In this project, we used annotations published in September 2016 and tested benchmark published on November 15, 2017.

The CAFA3 dataset consists of protein sequences from 18 model organisms, including humans, mice, yeast, and fruit flies. For each protein, a set of annotations representing the known molecular functions, biological processes, and cellular components are provided. The dataset also includes experimental data, such as protein-protein interactions and gene expression profiles, which can be used to improve the accuracy of function prediction methods.

Following the preprocessing steps of DeepGOPLus, we propagate the annotations using the hierarchical structure of the Gene Ontology, June 1, 2016 version, which consists of 10,693 molecular functions, 4,034 cellular components, and 29,264 biological processes. We consider all types of relations between classes, after which, we count the number of annoated proteins for each GO class and filter classes with less than 50 annotations.

For SwissProt, experimental annotations before Janurary 2016 are used as training set and those between Janurary 2016 and October 2016 were used for testing. To prevent data leakage during training, we remove from the testing set 23 target classes that are used in CAFA3 evaluation set.

The final training data consists of 75,495 proteins. This is further split into training and validation sets using a ratio of 9:1. The testing set consists of 3,974 data points. There are 5,828 classes in total. Top 20 classes are shown in Figure 1.

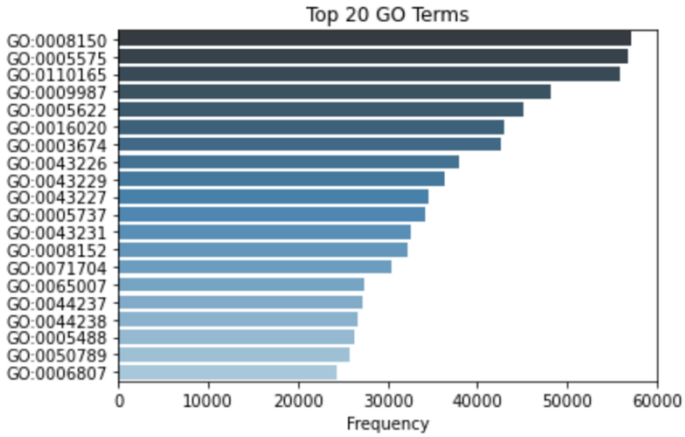

Figure 1: Top 20 most frequent GO terms

Each protein is associated with multiple GO terms. Hence, this is a multi-label classification problem. We obtain the number of classes for each protein and plot the histogram in Figure 2.

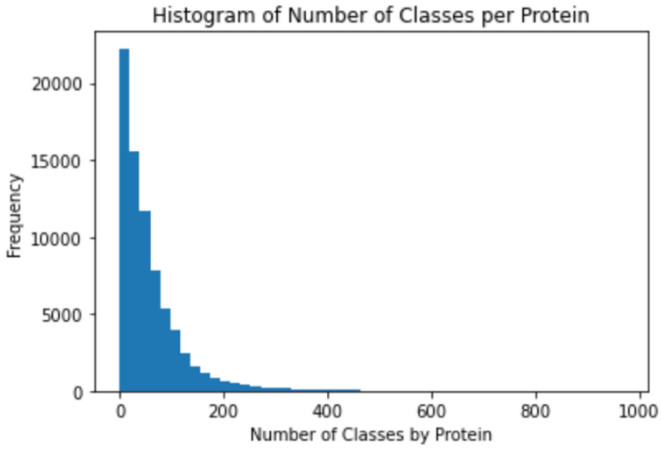

Figure 2: Distribution of number of classes per protein

3 METHODS

3.1 Benchmark Models

In function prediction from protein sequence, each protein is associated with multiple GO terms simultaneously. The goal is to predict the set of labels, i.e., GO terms, that best describes each protein. Traditional methods search for similar sequences with known functional annotations and specific sequence motifs associated with some function. Recent machine learning methods demonstrate more advantages in scalability for handling large-scale problems, automated feature extraction without the need for human knowledge, robustness for noisy data, and adaptability for different datasets. Therefore, we are interested in investigating the performance of various advanced machine learning models and their hybrid architectures in the protein function prediction problem. The benchmarking models are as follows:

1. **Convolutional Neural network (CNN)**: CNN is well-known for solving tasks involving spatial information, such as image classification and object detection. It can automatically learn local feature representations and hierarchies of features, which can capture complex patterns in the input data. DeepGOPlus is based on CNN models using 1D convolution with 512 filters and 16 kinds of lengths for each filter as feature extraction.

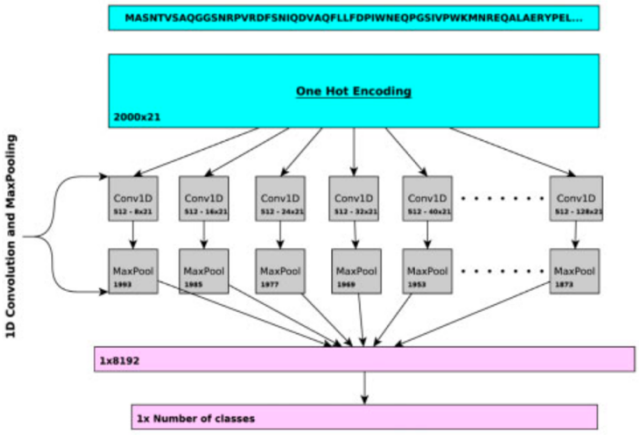

Figure 3: DeepGOPlus model architecture [2]

2. **Recurrent neural network (RNN):** RNN is more popular than CNN in natural language processing as it can capture sequential information from data, process inputs with varying lengths, and generate variable-length outputs. The findings might also apply to protein sequences.

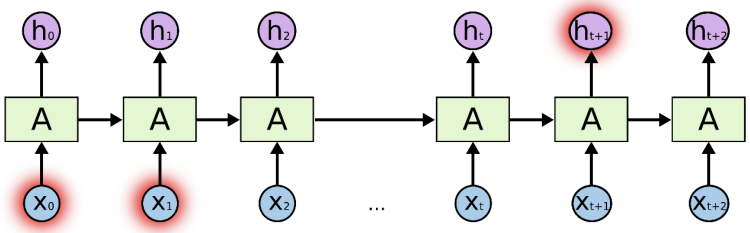

Figure 4: RNN model architecture

3. **Long short-term memory (LSTM):** LSTM solves the gradient vanishing and explosion problem in modelling long sequences. It selectively retains or discards information using gating mechanisms, which allows them to learn from both short-term and long-term patterns in the input data.

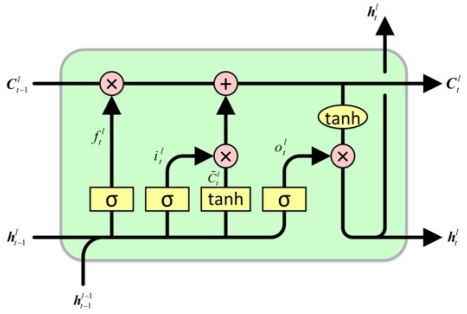

Figure 5: LSTM cell architecture

4. **Transformer:** Transformer models have shown state-of-the-art performance on various sequence modeling tasks. They can capture global dependencies in the input data using self-attention mechanisms, which allows them to process sequences of varying lengths in parallel and capture long-term dependencies more effectively. Transformer models are recently invented deemed as a successor of RNNs. The promising computation efficiency and performance make it a dominant technique in many fields like natural language processing and computer vision.

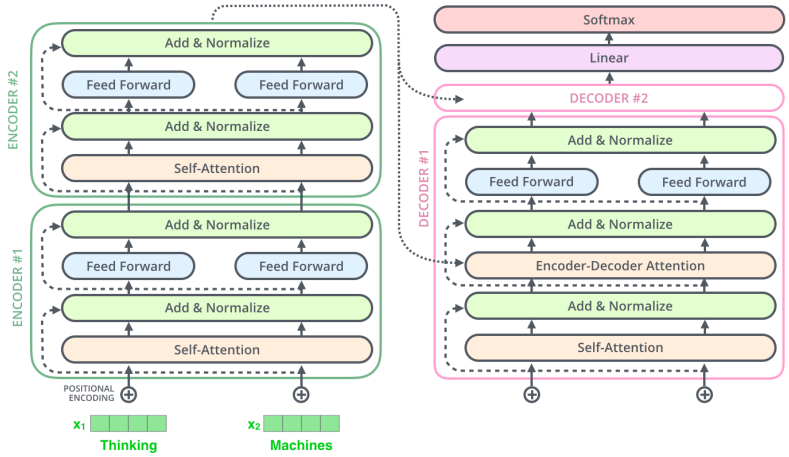

Figure 6: Transformer architecture

5. **Hybrid models:** One way to leverage different advantages of different machine learning models is to hybrid the model architecture. In this study, we combine CNN and RNN layers in sequence as one mother and in parallel as another model. We aim to see if such hybrid model could inherit the capability of extracting spatial information from CNN and extracting temporal information from RNN.

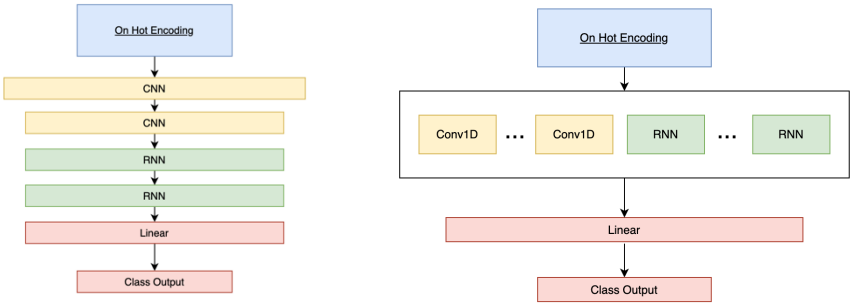

Figure 7: Hybrid CNN+RNN architecture (left: in sequnce, right: in parallel)

3.2 Evaluation

As mentioned, protein function prediction in this study is a multi-label classification problem. Each sample, i.e., protein sequence, has multiple ontology labels, including MFO, BPO, and CCO. To evaluate the classification performance, three evaluation metrics have been proposed: F_{max}, S_{min}, [3] and AUPR [2].

F_{max} is the maximum F score over all prediction thresholds t, which is the harmonic mean of the average precision $AvgPr(t)$ and average recall $AvgRc(t)$ among proteins. The average precision and recall are defined as followings:

$$AvgPr(t) = \frac{1}{m(t)} \sum_{i=1}^{m(t)} \frac{\sum_f I(f \in P_i(t) \wedge f \in T_i)}{\sum_f I(f \in P_i(t))}$$

$$AvgRc(t) = \frac{1}{n} \sum_{i=1}^{n} \frac{\sum_f I(f \in P_i(t) \wedge f \in T_i)}{\sum_f I(f \in T_i)}$$

where f is a GO class, T_i is the set of ground truth labels, $P_i(t)$ is the predicted classes for the protein i at the threshold t, $m(t)$ is the number of proteins that have no less than one class, n denotes the total protein number, $I(\cdot)$ is the Boolean function that returns 1 when the condition is satisfied. Prediction thresholds are set with 0.01 intervals, $t \in [0, 1]$.

S_{min} is the minimum semantic distance between ground truth and predictions over all prediction thresholds t based on the class information content $IC(c)$. It is the Euclidean norm of the average uncertainty $ru(t)$ and the average misinformation $mi(t)$. They are defined as followings:

$$IC(c) = -\log\left(\Pr\left(c|P(c)\right)\right)$$

$$ru(t) = \frac{1}{n} \sum_{i=1}^{n} \sum_{c \in T_i - P_i(t)} IC(c)$$

$$mi(t) = \frac{1}{n} \sum_{i=1}^{n} \sum_{c \in P_i(t) - T_i} IC(c)$$

where c is a GO class, $P(c)$ is the set of parent classes of c, $Pr(\cdot)$ function returns the conditional probability.

AUPR is the area under the precision-recall curve, which presents the overall model performance under different thresholds unlike the previous metrics focus on the threshold with the best performance.

In the study, the complete GO ontology is considered when computing parent child classes. As GO classes in one sub-ontology (MFO, BOP or CCO) may have relations with classes in other sub-ontology, the GO classes are defined first then separated into three sub-ontologies. For example, the MFO acyl carrier activity (GO: 0000036) has a 'part-of' relation with the BPO fatty acid biosynthetic process (GO: 0006633). This kind of situation has been taken into account when computing evaluation metrics while CAFA3 does not.

4 EXPERIMENT AND RESULTS

4.1 Experiment setup

For the CNN (Baseline) method, we used a simple architecture following the paper.

For the RNN method, we used 4 parallel 2-layer GRU architectures with (256,256) hidden units. Next, all outputs from GRU modules will be concatenated before being fed into the Dense layer to get the prediction.

For the RNN+CNN method, we used a hybrid architecture consisting of 2 convolutional layers followed by 2-layer GRU modules.

For the RNN+CNN parallel method, we used a similar architecture as CNN (Baseline) and RNN but much smaller. The output from CNN and RNN will then be concatenated before being fed into the Dense layer to obtain the prediction.

For the LSTM method, we used 8 parallel 3-layer LSTM architectures with (512,256,128) hidden units.

For the Transformer method, we used the multi-head self-attention mechanism. The model had 4 attention heads and a feed-forward dimension of 64.

We trained these models for 12 epochs with a batch size of 32 and used the Adam optimizer with a learning rate of 0.0003. We used a total of 4 GPUs NVIDIA GeForce RTX 3090 for this project.

4.2 Results

Table 1: Comparison of performance using various model architectures

Method	$F_{max}\uparrow$			$S_{min}\downarrow$			AUPR ↑		
	MFO	BPO	CCO	MFO	BPO	CCO	MFO	BPO	CCO
CNN (Baseline) # params (59.5m)	0.544	0.469	0.623	8.724	22.573	7.823	0.487	0.404	0.627
RNN # params (11,6m)	0.557	0.476	0.624	8.626	22.692	7.837	0.480	0.408	0.628
CNN+RNN parallel # params (19,1m)	0.558	0.488	0.625	8.610	22.433	7.782	0.491	0.419	0.627
CNN+RNN # params (15,9m)	0.552	0.472	0.625	8.606	22.469	7.764	0.489	0.422	0.629
LSTM # params (22,6m)	**0.560**	**0.485**	**0.625**	**8.592**	**22.343**	**7.716**	**0.511**	**0.428**	**0.631**
Transformer # params (244,8m)	0.497	0.436	0.617	8.966	23.259	7.933	0.443	0.371	0.622
Tuning									
LSTM 1 # params (45,1m)	**0.565**	**0.487**	**0.626**	**8.588**	**22.250**	**7.708**	**0.515**	**0.432**	**0.635**
LSTM 2 # params (26,8m)	0.557	0.475	0.617	8..600	22.457	7.779	0.526	0.406	0.628

Method	$F_{max}\uparrow$			$S_{min}\downarrow$			AUPR ↑		
	MFO	BPO	CCO	MFO	BPO	CCO	MFO	BPO	CCO
Transformer 1 # params (244.8m)	0.502	0.442	0.620	8.823	23.152	7.876	0.443	0.376	0.627
Transformer 2 # params (120.2m)	0.492	0.426	0.598	9.063	24.216	8.144	0.437	0.359	0.609
Transformer 3 # params (120.2m)	0.485	0.427	0.598	9.323	23.907	8.024	0.442	0.367	0.612

Table 1 reports the experimental results obtained from various model architectures. Best performances are in bold.

We find that RNN outperformed CNN in F_{max} and achieved comparable performance in S_{min} and AUPR.

With the hybrid architecture of CNN+RNN, the model was able to perform better than the CNN baseline. As for the 2 variants, CNN+RNN parallel, with more parameters than CNN+RNN, achieved higher F_{max}, comparable S_{min} and AUPR.

LSTM obtained the best performance among all methods. This shows that LSTM's ability to selectively remember or forget information makes it more effective at learning longer sequences.

Our transformer did not perform as well as the baseline and other methods. We speculate it caused by the large memory requirements for storing positional information of long protein sequences.

5 DISCUSSION AND CONCLUSION

5.1 Time complexity

Table 2: Epoch training time and number of parameters of models

Method	Training time (second per epoch)	Number of params
CNN (Baseline)	666s	59.5m
RNN	937s	11.6m
CNN+RNN parallel	1004s	19.1m
CNN+RNN	148s	15.9m
LSTM	6187s	22.6m
Transformer	223s	244.8m

The table provides information about the time required to train various machine learning models, including their method, training time per epoch, and the number of parameters. Based on this information, we can draw several observations and insights.

First, the table shows that different machine learning methods have different training times and require different numbers of parameters. For example, the CNN (Baseline) method has the shortest training time per epoch (666 seconds), while the LSTM method has the longest training time per epoch (6187 seconds). Similarly, the Transformer method requires the highest number of parameters (244.8 million), while the RNN method requires the lowest number of

parameters (11.6 million). Notably, although the Transformer has much more trainable parameters than others, its training time is in the low range (223 seconds). It shows the advantage of parallel computing computation in the Transformer model, since 4 GPUs are used in the experiment. In RNN, the output at each time step depends on the output of the previous time step, making it difficult to parallelize the computations across multiple GPUs.

Second, we can observe that combining different machine learning methods can lead to different trade-offs in terms of training time and the number of parameters. For instance, the CNN+RNN parallel method has a relatively long training time per epoch (1004 seconds) compared to the CNN+RNN method (148 seconds). However, the former requires fewer parameters (19.1 million) than the latter (15.9 million).

5.2 Model fine-tuning on LSTM and Transformers

To study the impact of different configurations on the performance of LSTM and Transformer models, we conducted an ablation study, with results summarised in Table 1. Specifically, we varied the number of hidden units and the number of layers in each LSTM model and the different numbers of heads and hidden dims in Transformers.

For the LSTM model, we varied the number of layers from 3 to 4 and the number of LSTM from 8 to 16. We trained and evaluated each configuration on the same dataset. We also used the Adam optimizer with a learning rate of 0.0003 and trained each model for 12 epochs with a batch size of 32.

For the Transformer model, we varied the number of attention heads from 4 to 8 and the hidden dim from 64 to 32. We trained and evaluated each configuration on the same dataset using the same hyperparameters as the LSTM model.

Our results showed that increasing the number of layers and the number of LSTM improved the performance of the LSTM model. However, the Transformer models do not provide any significant performance improvement. Our results showed that decreasing the number of attention heads and layers also decreases the performance of the Transformer model.

5.3 Future work

We have investigated advanced sequence models for protein function prediction including CNN, RNN, LSTM, Transformer, and hybrid variants. It is observed that these advanced model architectures indeed improve the performance (F_{max}, S_{min}, and AUPR) to different extents, and model hyperparameters are critical to obtain the best performance. More sophisticated models could be tested such as bi-directional RNN and hierarchical transformers in the future. To handle longer sequences more efficiently, several modifications of Transformer such as Longformer [5] and Big Bird [6] have been proposed. We could make use of techniques like sparse attention mechanisms and sliding window approaches to reduce the memory requirements of the models. As the performance improvement has been seen in the hybrid architectures, we could further study the ensemble methods considering more models and architectures. Last but not least, transfer learning becomes more and more prevalent, we could explore the effectiveness of pre-training model in other datasets and fine-tuning in the target dataset like BERT. The large language models like GPT [7] are extremely powerful with universal intelligence, which could also provide knowledge or capability for protein function prediction, e.g., through in-context learning.

REFERENCES

[1] Kulmanov M, Khan M A, Hoehndorf R. DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier[J]. Bioinformatics, 2018, 34(4): 660-668.

[2] Kulmanov M, Hoehndorf R. DeepGOPlus: improved protein function prediction from sequence[J]. Bioinformatics, 2020, 36(2): 422-429.

[3] Zhou N, Jiang Y, Bergquist T R, et al. The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens[J]. Genome biology, 2019, 20(1): 1-23.

[4] EMBL-EBI (European Bioinformations Institute). http://www.ebi.ac.uk/swissprot/. Retrieved in April 2023.

[5] Beltagy I, Peters M E, Cohan A. Longformer: The long-document transformer[J]. arXiv preprint arXiv:2004.05150, 2020.

[6] Zaheer M, Guruganesh G, Dubey K A, et al. Big bird: Transformers for longer sequences[J]. Advances in neural information processing systems, 2020, 33: 17283-17297.

[7] Floridi L, Chiriatti M. GPT-3: Its nature, scope, limits, and consequences[J]. Minds and Machines, 2020, 30: 681-694.