Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Quantum future
The shift in the communication paradigm

Biological Data Project Report

Protein Function Prediction

Authors: Chelsie Romain, Donatas Vaiciukevicius
Professor: Damiano Piovesan

02/23/2024

1 Objectives

Proteins play crucial roles in nearly every biological process. Deciphering their functions is paramount for understanding cellular mechanisms, disease pathways, and drug design. Traditional experimental methods for annotating protein functions are labor-intensive and time-consuming, prompting the need for efficient computational approaches. In recent years, deep learning has emerged as a powerful tool for this purpose. By leveraging protein sequence data, deep learning models can potentially capture patterns that can enable the accurate and scalable prediction of protein functions.

The aim of this report is to explore many deep learning architectures for extracting features from protein sequences to determine their function annotation. A comparison on the performance of these models for protein function prediction was done and the final best model was chosen for testing.

2 Materials and Methods

The method used to perform protein function prediction was supervised learning, where both the protein IDs and it's associated GO terms were provided during training. Since each protein can have multiple non-exclusive GO terms associated with it, this is a multi-label classification problem.

There were a total of 4,277,047 entries in the training set, however only 3004 unique GO term IDs. Visualizing the 100 most frequent GO terms and their percentiles, we see that the most frequent term (purple bar) is represented in 25% of the samples, and the next 99 terms all fall within 50-th percentile.

The project specification requires that one target cannot be associated with more than 1500 terms for MF, BP and CC sub-ontologies combined, to limit prediction file sizes. Thus, the first 1500 most frequent 'GO_terms' were extracted and saved into a list for use as labels. In doing so, however, there was not a large loss of data as the cumulative distribution has a strong left skew and the first 1500 GO terms cover 93.59% of the data.

2.1 Dataset and Data Representation

The training examples were proteins in their ProtT5 embedding form, i.e. a data frame where each row represented a unique protein in a numerical vector of length 1024. The training labels were created using a Python script that matched each protein to its associated GO terms, i.e. a data frame where each row is a protein and its value is a 1500-long indicator vector of 0's and 1's where a position is 1 if the GO term for that position is in the set of GO terms for that protein and 0 otherwise. Thus, the task of our model architectures is to predict the 1500-long label vectors given the 1024-long numerical embedding vector.

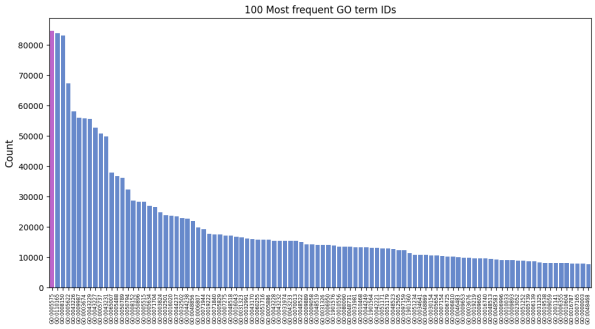

Figure 1: 100 Most frequent GO term IDs

2.2 Training

In total, 16 different model architectures, split across 4 Jupyter Notebooks, were tried on each of the three gene ontologies. To remain within usage limits for RAM and GPU, one Notebook per architectural type of the network was created: simple multi-layer perceptron, traditional feed-forward neural networks, convolutional neural networks, and a residual convolutional neural network. The "best" performing model in each category was saved and then loaded into the final notebook used for testing. A model was considered to have the best performance if it had the highest F1 score over our protein dataset.

For neural networks to learn, the error must be backpropagated to update the weights of the hidden nodes. The error used for the models tested was binary cross-entropy. This is because it allows for loss to be computed on every vector component independently so that every label that is above the threshold is assigned a 1, consistent with a multi-label problem. The softmax loss was considered i.e. to find the most likely term, assign a 1, and then propagate 1's to all

the parent terms. However in this instance, the parent terms *are* the most likely terms in any case, and such an approach may not have produced accurate results for leaf nodes and more specialized functions.

To choose the best model, 10-fold cross-validation was used while training each architecture. In this procedure, the data is randomly shuffled and divided into 10 folds. The model is retrained 10 times with each fold acting as a hold-out or validation set once, i.e. each sample is used in the hold-out set 1 time and used to train the model 9 times. Due to computational constraints during cross-validation each model was trained for 5 epochs only.

The models picked based on cross-validation were evaluated against each other and the best-performing model for each ontology was picked. Afterward, new training-validation splits were created, and the models were trained for longer, using learning rate scheduling to ensure that the most possible performance could be extracted from the architectures. Accuracy and F1 score were computed on the validation set afterwards, as an indicator of how the model could perform on unseen data in the future.

2.3 Model architectures

2.3.1 Feed-forward Neural Network (FNN)

A feed-forward neural network (FNN) is characterized by the unidirectional forward flow of information from the input nodes to the hidden layer and then output nodes. The idea here, is to take information from the entire sequence and form a representation within the hidden layers by fully connecting each layer. For the FNN, two main types of architecture were tested of which there were six subtypes for the first and only one for the second and third, for a total of eight tested architectures. The first type of feed-forward architecture (FNN1) consisted of the input layer, three dense layers of all the same size with a non-linear activation function, and finally the output layer. The six subtypes were obtained by adjusting the number of hidden neurons in each layer and the activation function. Each of the following were tried:

- 512 neurons in each layer; relu activation function
- 256 neurons in each layer; relu activation function
- 128 neurons in each layer; relu activation function
- 512 neurons in each layer; tanh activation function
- 256 neurons in each layer; tanh activation function
- 128 neurons in each layer; tanh activation function

The second type of feed-forward architecture (FNN2) had an hourglass or bottleneck shape. It consisted of the input layer, then five successive dense layers of decreasing then increasing size ($512 \rightarrow 256 \rightarrow 128 \rightarrow 256 \rightarrow 512$), and finally the output layer. The activation function in this case was relu.

Finally, the third type (MLP) consisted of the input layer, a dense layer of 1024 units with a linear activation function on each layer, and finally the output layer. This was tested as a more simplistic approach.

The best-performing architecture was the same across all 3 ontologies; three dense layers of 512 units each with the relu activation function.

2.3.2 Convolutional Neural Network (CNN)

Convolutional neural networks, incorporate regularization into the network by using fewer connections from kernel windows and pooled layers, compared to their dense fully connected counterparts. This architecture was tested to determine if the information within short windows of the protein sequence is predictive of its function and how neighboring proteins in the same window may affect each other. For the CNN, one architecture was tested of which there were six subtypes. It consisted of the input layer followed by one convolutional layer, a max pooling layer, a flatten layer, a dense layer and finally the output layer. The six subtypes were obtained by adjusting the number of filters and the kernel size. Each of the following were tried:

- 16 filters; kernel size of 64
- 16 filters; kernel size of 128
- 16 filters; kernel size of 256
- 32 filters; kernel size of 64
- 32 filters; kernel size of 128
- 32 filters; kernel size of 256

The best performing architecture for both Biological Processes and Molecular Function consisted of 32 filters with a kernel size of 64, while the best architecture for Cellular Component had 32 filters and a kernel size of 256.

2.3.3 Residual Neural Network (ResNet)

When working with naive CNNs, network's depth quickly becomes a limiting factor. Adding residual connections was shown in prior work to improve the network's ability to learn, as gradients could propagate more easily during backpropagation. We wanted to test if making a deeper level convolutional network that could possibly focus on both lower-lever and higher-level features would increase the function prediction performance.

Inspired by ResNet used for imaging, we defined a ResNet1D architecture, that consisted of multiple residual blocks. Each block is made of two convolutional layers with a kernel size of 3, separated with a relu activation function and batch normalization layers. The initial input to the block is downsampled if necessary and added to the output of the prior layers. Then it is fed through another relu activation. The model starts with a convolutional block, batch normalization, relu and max-pooling, and afterward, 8 residual blocks are added, doubling the number of channels in the convolutional layers every two blocks ($64 \rightarrow 128 \rightarrow 256 \rightarrow 512$).

2.4 Testing

After training on all the different architectures and choosing the best one we performed re-training for each ontology using an 80/20 train/validation split on all the data. Next, we used these final models to perform predictions on the test set according to ontology. The models ended up reaching their peak performance after around 15 to 18 epochs, after which the validation loss would plateau, and the models would start overfitting. This produced 3 data frames, each with a label

	Biological Processes		Molecular Function		Cellular Component	
	Accuracy	F1Score	Accuracy	F1Score	Accuracy	F1Score
CNN1	0.9697	0.4469	0.9751	0.4748	0.9799	0.4578
ResNet1D	0.9723	0.1985	0.9699	0.2366	0.9773	0.3032
FNN1	0.9770	0.4644	0.9770	0.5102	0.9812	0.4973
FNN2	0.9762	0.4426	0.9765	0.4962	0.9806	0.4962
MLP	0.9752	0.3663	0.9729	0.2845	0.9800	0.4150

TABLE 1: Architectures performance comparison

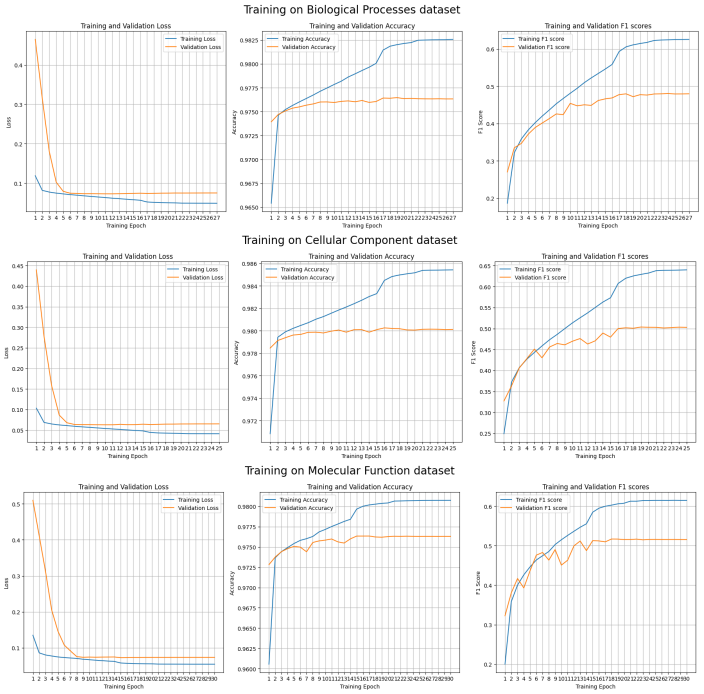

FIGURE 2: Training process of the final models

probability according to that ontology's model (since we do not know to which ontology the test proteins belong). For our final prediction, the maximum probability across all 3 models was taken as the final value ("any" ensemble voting).

3 Results

Overall the feed-forward neural network with dense blocks of equal size was the best-performing architecture. It obtained F1 scores of 0.4644, 0.5102 and 0.4973 for BP, MF and CC respectively. The next best architecture for BP was the CNN with an F1 score of 0.4469. Meanwhile for MF and CC, the next best was the hourglass FNN both with an F1 score of 0.4962. The full set of results is reported in Table 1.

The plot makes it clear that the CC ontology has the highest performance in terms of both F1 score and accuracy regardless of the model architecture. Additionally, the models of the MF ontology tend to outperform those of the BP ontology. These differences in the datasets are likely due to the differing encodings of information within the sequences which varies based on their ontology.

Generally, all of our models suffer from a common issue - they have high accuracy, but relatively low F1 scores. To be precise, while training we noticed that while precision was relatively high, our models struggled with recal. Further investigation shown that the models tend to be biased towards the lack of GO terms, rather than the existence of them. We have also observed that per-label F1 scores were correlated with the number of labels in the training data - the more often the label occurs, the higher its corresponding F1 score. We have attempted oversampling less common labels to account for the imbalance, but as it is a multi-label classification problem this turned out more difficult than expected - proteins would have both common and uncommon GO terms, therefore the common labels would get oversampled as well. We have also attempted using weighted cross-entropy loss, using inverse label frequency as their weight for loss computation, unfortunately to no avail.

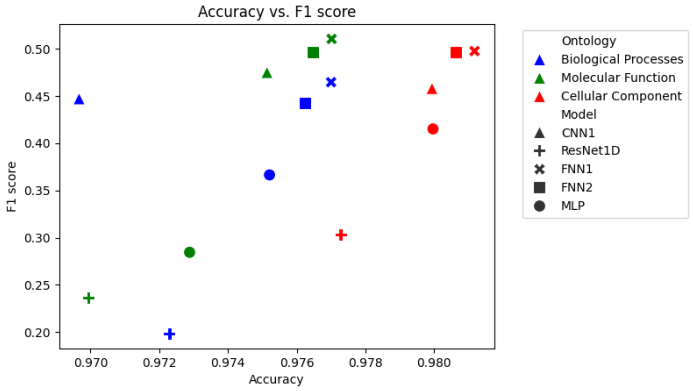

FIGURE 3: Accuracy vs F1 Score

4 Conclusions

Of all the model architectures, the feed-forward neural network with dense blocks of equal size performed the best on all of the GO sub-ontologies. This suggests that even though residual neural networks are typically understood as being more appropriate for handling sequence data, here they do not offer more expressivity than vanilla FNNs and perform significantly worse. This may be an artifact of the parameters used in the ResNet here or a feature of the architecture itself.

The biggest hurdle to improving the performance of the models is finding a solution to compensate for less common labels. We believe that having the issue fixed would improve performance of most of the models tested, as they were capable of predicting the more common GO terms.

References

Elnaggar, A., Heinzinger, M., Dallago, C., Rehawi, G., Wang, Y., Jones, L., Gibbs, T., Feher, T., Angerer, C., Steinegger, M., Bhowmik, D., & Rost, B. (2021). ProtTrans:

Towards Cracking the Language of Life's Code Through Self-Supervised Learning. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(8), 1.

Gligorijević, V., Renfrew, P.D., Kosciolek, T. et al. Structure-based protein function prediction using graph convolutional networks. Nat Commun 12, 3168 (2021). https://doi.org/10.1038/s41467-021-23303-9

Kulmanov, M., Khan, M. A., & Hoehndorf, R. (2018). DeepGO: Predicting protein functions from sequence and interactions using a deep ontology-aware classifier. Bioinformatics, 34(4), 660–668. https://doi.org/10.1093/bioinformatics/btx624