Project #20 : protein to pathway classification

Marco Creta
University of Modena and Reggio Emilia
{319988}@studenti.unimore.it

Abstract

The primary goal of this project is to determine whether a given protein sequence originates from a defined pathway in Homo Sapiens. The approach consist of extracting robust, context-aware representations from plain protein sequences and use such information to perform the classification task. Different popular architectures will be explored, highlighting the potential of using pre-trained protein language models for targeted biological tasks.

1. Introduction

1.1. Availble models

In this work, we test several popular transformer-based models that have been pretrained on large protein sequence datasets. These models has been trained on extensive protein databases (UniRef90, UniRef50, and BFD), providing diverse and complementary feature representations that can be utilized for downstream protein classification tasks.

1.2. proteinBERT

ProteinBERT [1] is a specialized language model designed for protein (amino acid) sequences. It is based of BERT architecture but is trained on 106M UniRef90 proteins, it's highly adaptable for diverse protein analysis tasks. It predicts both whole-sequence and per-position properties—supporting tasks like classification, multilabel, and regression. The model outputs a concatenated embedding of all hidden states from a given protein sequence, which can serve as features for other ML applications.

1.3. proteinGPT2

ProtGPT2 [3] is a GPT2-based, decoder-only transformer model designed for protein design. It features 36 layers, a 1280-dimensional embedding space, and 738 million parameters. Trained on raw UniRef50 protein sequences using a causal next-token prediction objective, it generates protein sequences that retain key natural features—such as amino acid propensities, secondary structure, and globularity. Since there is no classification token the output embeddings can be extracted as a fusion of per-token representaations.

1.4. proteint5

ProtT5-XL-BFD [2] is an encoder/decoder masked language model built on the T5-3B architecture and trained on 2.1 billion protein sequences. It uses a Bart-like MLM denoising objective—randomly masking 15% of uppercase amino acid tokens to learn biophysical properties of proteins. The model, which processes inputs as space-separated tokens (with a vocabulary of 21, mapping rare amino acids to X) up to 512 tokens.

2. Datasets

2.1. UP000005640: Homo sapiens Proteome

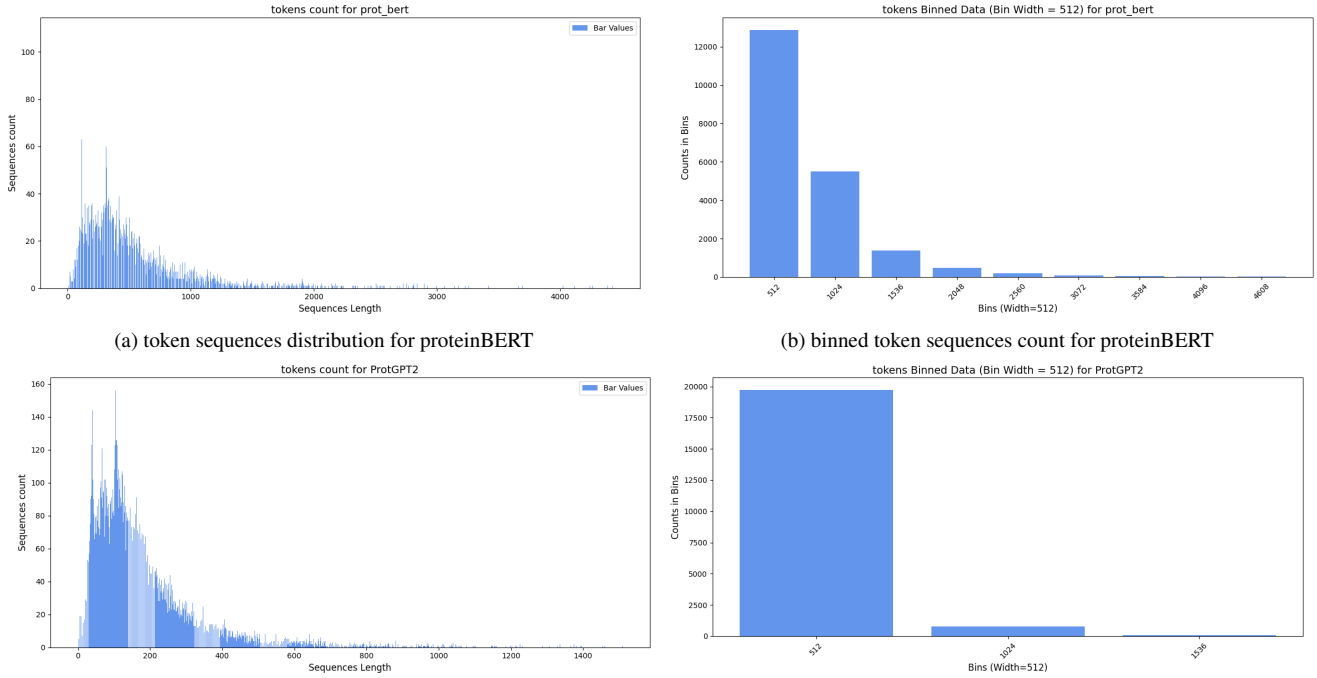
UP000005640 represents the comprehensive proteome of *Homo sapiens* (human) curated by Universal Protein Resource (UniProt).

The dataset contains the full genomic repertoire of human protein-coding genes and includes 20.644 entries, each entry corresponding to a unique gene and representing its canonical protein product in a one-to-one mapping to ensure consistency in protein identification and downstream analyses.

Along the primary amino acid sequences, other informations are present like structure, post-translational modifications, and subcellular localization.

2.2. R-HSA-*: list of genes per pathway

There is no straigh dataset that express the correlation between a given pathway ans its genes, so each dataset in costructed manually. First, we can extract the list of proteins for a given pathway from the Reactome pathway browser [] and download the list of proteins associated with such pathway in notation UniProtKB AC/ID. Then we can translate the list to standard UniProtKB notation by using the UniProt ID-mapping service.



(a) token sequences distribution for proteinBERT

(b) binned token sequences count for proteinBERT

(c) token sequences distribution for proteinGPT2

(d) binned token sequences count for proteinGPT2

Figure 1. distribution of tokenized sequences. Right 0.03% outliers have been removed for better visualization. Note that Proteint5 and proteinBERT share the same token lengths

3. Methods

3.1. Multiclass multilabel task

Our task is, given a protein sequence, classify it with respect to a list of arbitrary chosen pathways (our classes). The main issue is that there is no one-to-one association proteins and pathways, so a given protein can be found in many different pathways.

We solve this issue by setting up a multiclass multilabel classification tak, where each protein can be assigned up to n different labels, where $n - 1$ correspond to the number of considered pathways and must be chosen in advance and the remaining class is the default one, not associated o any other pathway.

The classification process is broken down to n binary classifications, the resulting label will be a multi-hot vector which indicate the pathways associated to the single protein.

3.2. Chosen pathways

We selected two pathways for analysis. The first is the broad macro-family of metabolism, created by merging multiple sub-pathways to achieve wide representation; it comprises 2,022 proteins. The second is the much smaller, specific pathway R-HSA-109581 (apoptosis), which contains only 168 proteins. These pathways share an overlap of 42 proteins. This setup—with its highly unbalanced representations and significant overlap—allows us to test the feasibility of our approach under difficult conditions.

3.3. Unbalanced data

Protein distribution across the classes in very unbalanced, the average number of proteins associated to a single pathway goes for 50 to 300, meanwhile the total number of instances is 20.644, in the worst case scenario we can end up with an imbalance of 3 orders of magnitude.

To overcome this issue not only we construct a custom weighting procedure to ensure that in each batch the same number of instances will be seen in the training, but also each class will have a different weight in loss computation.

3.4. Data processing

The data have been preprocessed and the resulting embeddings stored to speed up training and reduce computational cost.

The context window of all the models is fixed on a 512 token length. Even when a model allow a longer sequence, the models are explicitly trained on a fixed 512 sequences.

Since many protein have a tokenized length which exceeds the window, we adopt a sliding window chunking approach, extracting from every chunk of length 512 (special tokens included) the corresponding embeddings and translating the window of a stride of half the window (special

tokens excluded), to have a 1/2 overlap between chunks.

For ProteinBERT each amynoacid correspond to one token, so the sequence is split into 510 characters, spaced by a whitespace (as requested by the tokenizer) and then passed to the tokenizer. The special tokens $[CLS]$ and SEP are handled automatically by the tokenizer in this case.

For ProterinGPT2 instead, one token can correspond to many characters, since there is no 1-to-1 correspondence the sequence is first tokenized entirely, and then split into 511 chunks. The special token $< |endoftext| >$ is inserted manually at the beginning of the sequence and the mask window is extended accordingly.

For proterint5 the tokenization prodecure is the same as ProteinBert with the difference that only the $[EOS]$ token must be inserted as termination and that rare amynoacids (U, Z, O, B) are mapped to a X.

3.5. Number of chunks

The number of overlapped chunks have been set to 3, by doing so we can select up to 0.90% of the total number of tokens. In fact the majority of proteins do not exceeds a total length of 1024 tokens.

	proteinBERT	proteinGPT2	proteint5
%tokens	0.8912	0.9954	0.8912

3.6. classification head

The classification head is composed of 2 main parts, an attentional pooling layer to better learn a feature fusion of the overlapped chunks and a feedforward for the final classification

3.6.1 FeedForward

The feedforward network is designed to transform the input features into output logits for classification. It consists of two main linear layers with interleaved batch normalization, leakyReLU activation, and dropout, followed by a final linear layer that maps the processed features to the desired output size.

During inference since the task is multiclass multilabel classification, the output is passed through a sigmoid layer and classification is made per-output with a global thresholding, thus prdicting multiple classes.

3.6.2 Attentional pooling

The `AttentionPooling` module implements a self-attention pooling mechanism inspired by the paper "Self-Attention Encoding and Pooling for Speaker Recognition" [5]. It is designed to aggregate variable-length sequence representations into a fixed-size vector.

The module takes as input a batch of sequences represented as a tensor of shape (B, T, H), where B is the batch size, T is the sequence length, and H is the hidden dimension. It applies a learned linear transformation (through the weight matrix `W`) to each hidden state, reducing the hidden dimension to 1. This produces a score for every chunk, which is then normalized across the sequence using a softmax function to generate attention weights. These weights highlight the importance of each chunk, and a weighted sum of the original sequence is computed along the sequence dimension. The output is a fixed-size representation of shape (B, H) for each sequence, capturing the most informative features.

3.7. Unbalanced data

3.7.1 weighting

Due to the highly unbalanced data, a weighting approach have been implemented.

Let $Y \in \{0,1\}^{N\times K}$ be the one-hot label matrix, where N is the number of samples and K is the number of classes. For each class j (with $j = 1, 2, \ldots, K$), define the class count as

$$c_j = \sum_{i=1}^{N} Y_{ij}$$

with the convention that if $c_j = 0$, then c_j is set to 1.

The unnormalized class weight for class j is then

$$w_j = \frac{1}{\sqrt{c_j}}.$$

Finally, the normalized class weight is given by

$$\hat{w}_j = \frac{w_j}{\sum_{k=1}^{K} w_k} = \frac{\frac{1}{\sqrt{c_j}}}{\sum_{k=1}^{K} \frac{1}{\sqrt{c_k}}}.$$

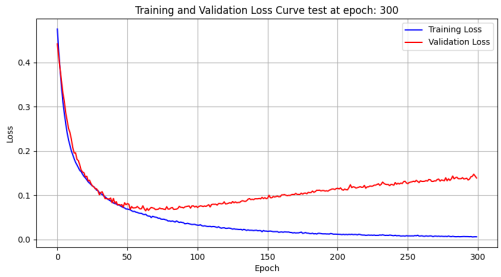
Thus, the final vector of class weights is

$$\hat{w} = (\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_K).$$

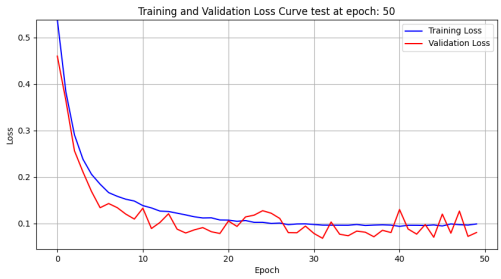
3.7.2 Oversampling

To ensure the presence of at least a few instances of every class for every batch we use an oversampling approach. A weighted batch sampler has been implemented that first compute per-class weight with the previous approach (including the default class). The weights are then combined instance-wise, by summing across all classes of the single instance. We can just use the multi-hot vector as a mask for the weight vector.

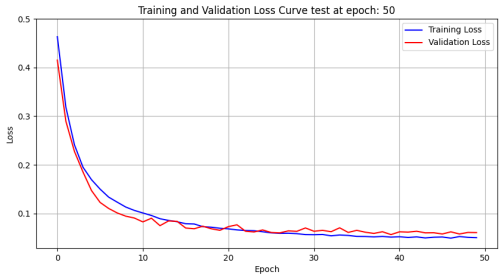
$$\begin{aligned} w'_j &= \hat{w} \times Y_j^T \\ &= (\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_K) \times (0, 1, \ldots, 1)^T \end{aligned}$$



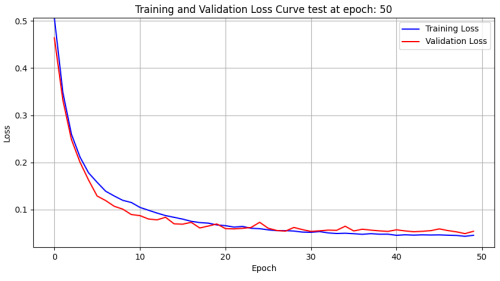
(a) GPT2 before overfitting optimization



(b) Protein BERT



(c) protein GPT2



(d) protein t5

Figure 2. losses with feedforward configuration only and BCE loss

3.8. Losses

3.8.1 BinaryCrossEntopyWithLogits

The loss is computed per-class as a weighted binary cross entopy, then the individual scores are summed. By doing so we can train on our multiclass multilabel task.

3.8.2 WightedFocalLoss

Due to the highly imbalanced nature of the task, we tried a custom version of the focal loss [4] where we weight each class independently.

4. Training

All tests have been performed with controlled randomness by fixing random seeds to ensure the same conditions in all tests. Train/val/test splits have been used too to ensure that final results are computed on data never seen, neither in training nor in validation.

Table 1. Per-class statistics

Scores				
Class	**Precision**	**Recall**	**F1**	**Support**
R-HSA-metabolism	0.67	0.60	0.63	435
R-HSA-109581	0.39	0.41	0.40	32
default	0.95	0.97	0.95	3671
Per-class (versus all) counts				
Class	**TP**	**FP**	**FN**	**TN**
R-HSA-metabolism	261	127	174	3568
R-HSA-109581	13	20	19	4078
default	3525	184	146	275
Pre-class Averages (no default)				
Avg Type	**Precision**	**Recall**	**F1**	**Support**
micro avg	0.65	0.59	0.62	467
macro avg	0.53	0.50	0.52	467
weighted avg	0.65	0.59	0.62	467
samples avg	0.07	0.07	0.07	467

4.1. Only FF

we first tried a simple configuration by using only the feedforward module and using as input the concatenation of the chosen feature vectors (3 per protein with an overlap of 1/2 as stated before).

4.1.1 weighted BCE

Because of the oversampling, we stich to a batch size of 64, thus we avoid too many repetitions but still have a batch size big enough to contrast overfitting. We also use a weight decay (L_2 norm) of $3e^{-3}$ in the adam optimizer and a dropout of 0.5 for each layer. Moreover, to contrast overfit we act an early stopping around 50 epochs, where both train and val curves hit a plateau

In addition we noted that the precision despite the penalizations always dropped nearly 0, so we downsized the size

of the classifier network progressively until an equilibrium had been found with an input size of 256.

Table 2. Confusion Matrix with all mock (overlapped) classes

True\Pred	0	1	2	3
0	3525	19	127	0
1	14	10	0	0
2	166	1	260	0
3	4	3	1	0

Label Mappings:

- **0** → binary_label: [0 0], classes: ['R-HSA-109581']
- **1** → binary_label: [0 1], classes: ['R-HSA-109581']
- **2** → binary_label: [1 0], classes: ['R-HSA-metabolism']
- **3** → binary_label: [1 1], classes: ['R-HSA-metabolism','R-HSA-109581']

4.1.2 FocalLoss

We tried to use a custom weighted focal loss (different alphas for different classes) instead of the weighted BCE. We noticed that in the weighted configuration the loss rushes to 0 very rapidly, so we tried to remove the weights (1 for all classes) and it was able to reach comparable performances of the weighted BCE, but it was more prone to overfitting.

4.2. FF + attentional pooling

By fusing across chunks using an attentional pooling layer instead of concatenating the embeddings, the model shows similar but slightly lower performance. The precision is slightly increased, but the recall is lower.

5. Conclusions

Protein-to-pathway classification appears to be a feasible task; however, existing models cannot be used off-the-shelf and require fine-tuning for optimal performance. In our experiments, ProteinGPT2 and ProteinT5 exhibited nearly identical performance, whereas ProteinBERT produced considerably noisier training. This discrepancy is likely due to ProteinBERT's smaller size and older architecture, which may result in the extraction of lower-quality features compared to the other two models.

References

[1] Nadav Brandes, Dan Ofer, Yam Peleg, Nadav Rappoport, and Michal Linial. Proteinbert: a universal deep-learning model of protein sequence and function. *Bioinformatics*, 38(8):2102–2110, 02 2022. 1

[2] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, DEBSINDHU BHOWMIK, and Burkhard Rost. Prottrans: Towards cracking the language of life's code through self-supervised deep learning and high performance computing. *bioRxiv*, 2020. 1

[3] N. Ferruz, S. Schmidt, and B. Höcker. Protgpt2 is a deep unsupervised language model for protein design. *Nature Communications*, 13(1):4348, jul 2022. 1

[4] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2018. 4

[5] Pooyan Safari, Miquel India, and Javier Hernando. Self-attention encoding and pooling for speaker recognition, 2020. 3