ALTEGRAD Project Presentation: Cellular Component Ontology Prediction

CECCHINI Alessandro
CHAUVIN Paul
LOISEAU Thibaut

January 26th 2023

1. Introduction

- Proteins are essential biomolecules that play important roles in living organisms, including **acting as enzymes and providing structural support**.

- The **structure of proteins is determined by the sequence of amino acids**, which fold to form a 3D structure that determines their specific chemical functionality, but the exact details of this process are not yet fully understood.

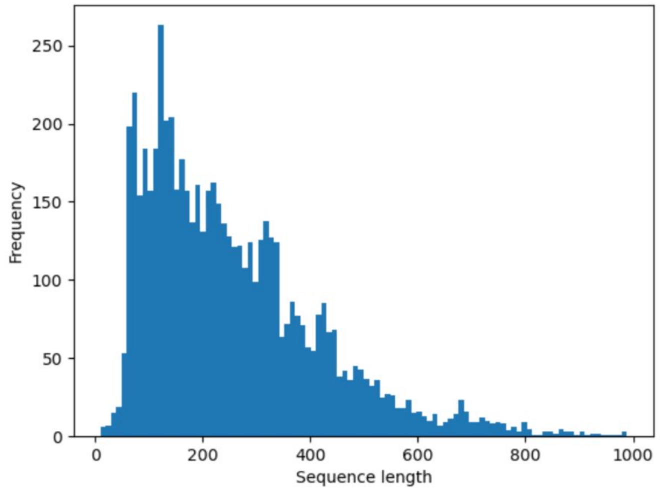
- We propose to study and apply machine learning/deep learning techniques to **classify 6k+ proteins into 18 different classes**, each representing a characteristic of a protein

- Two different types of models where implemented: **sequence based and structure based models** will be used for the classification task

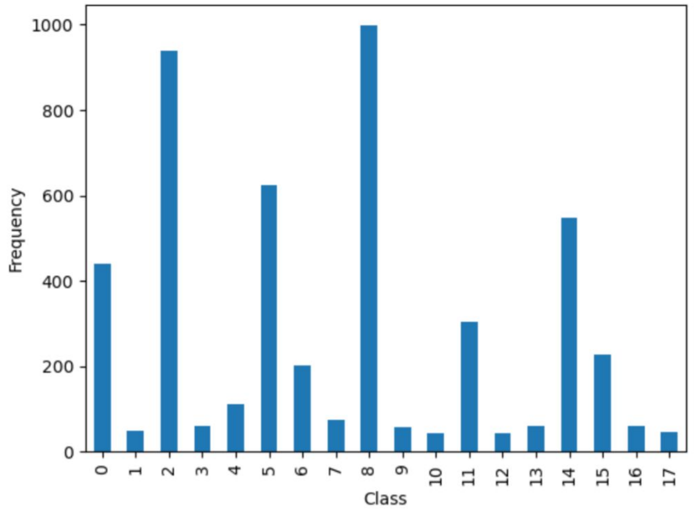
2.1. Methodology - Data Used

- This project aims to **classify a dataset of 6,111 proteins into 18 different classes**, each representing a characteristic of the protein.
- Some of the proteins are **labeled (4888)** while others are **not (1223)**.
- For **sequence based models**, we used the sequence of each protein and the labels associated.
- For **structure based models**, we used edges, attributes of the edges, attributes of the nodes, graph indicators and labels for the proteins
- The goal of the project is to **predict the class label of proteins that have not been labeled**.

2.2. Methodology - Data Analysis



Distribution of the sequence length over training and testing set



Distribution of the number of examples per class over training set

- For **sequence base** models, we will use a sequence length of 512.
- We will also change the sampling method during training by taking a **weighted random sampler** to get an evenly distributed dataset.

2.3. Methodology - Sequence based models

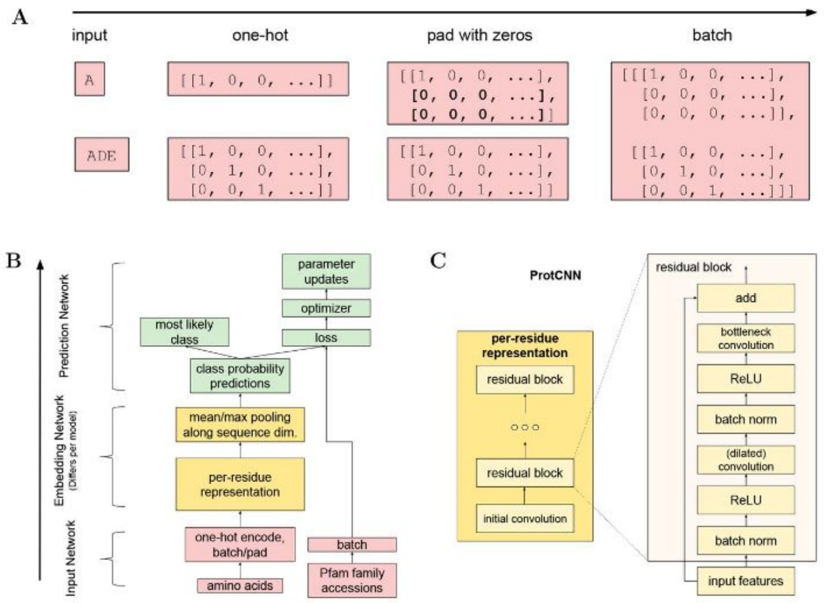
1. Modifications made on the sequence baseline

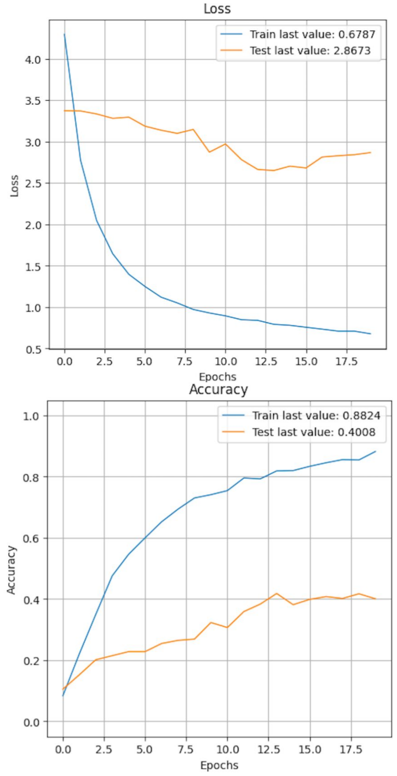
- Modify the number of n-grams:
    - Trying with 3 to 7 n-grams taken into account => decrease of the performances

- Modify the classifier used in the sequence baseline
    - Initially: Logistic regression
    - Change using Support Vector Machine
    - Change using Decision Tree

2.3. Methodology - Sequence based models

2. Protein Computational Neural Network (ProtCNN)

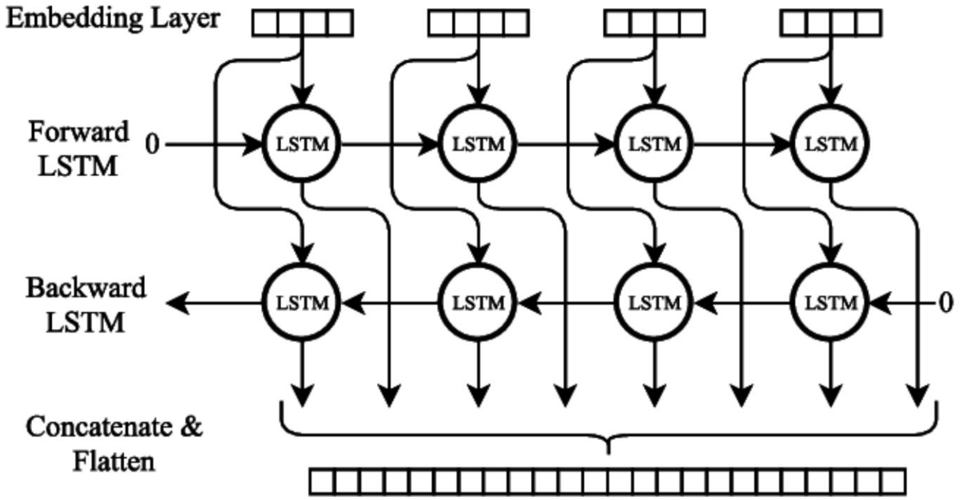




Problem: overfitting

- Added cross validation (Kfold =5)
- Added class weight
- Intended up/downsampling
- Added more regularization
- Early Stopping

2.3. Methodology - Sequence based models
3. Bidirectional Long Short Term Memory (BiLSTM)

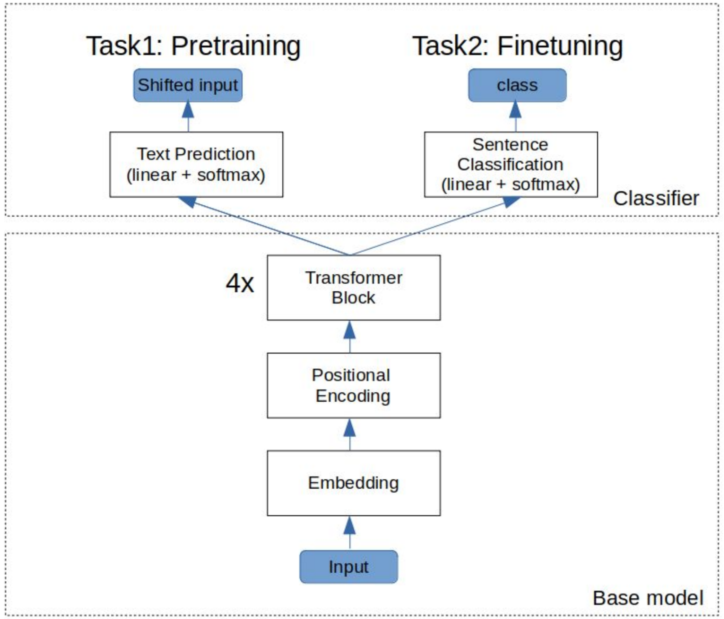


Problem: overfitting

- Added cross validation (Kfold =5)
- Added class weight
- Intended up/downsampling
- Added more regularization
- Early Stopping
- Added dropout

2.3. Methodology - Sequence based models

4. Fine-tuning pretrained models (Transfer Learning)



- Use a **ProtBERT** pretrained model on **BFD dataset** (2.1 billion sequences).
- Adding a classifier head automatically with Hugging Face AutoModelForClassification.
- Freezing the pretrained model.
- Using a starting learning rate of 10^{-3} with a decay of 10 every 5 epochs. We train for 25 epochs.

https://huggingface.co/Rostlab/prot_bert_bfd

2.4. Methodology - Structure based models

Model

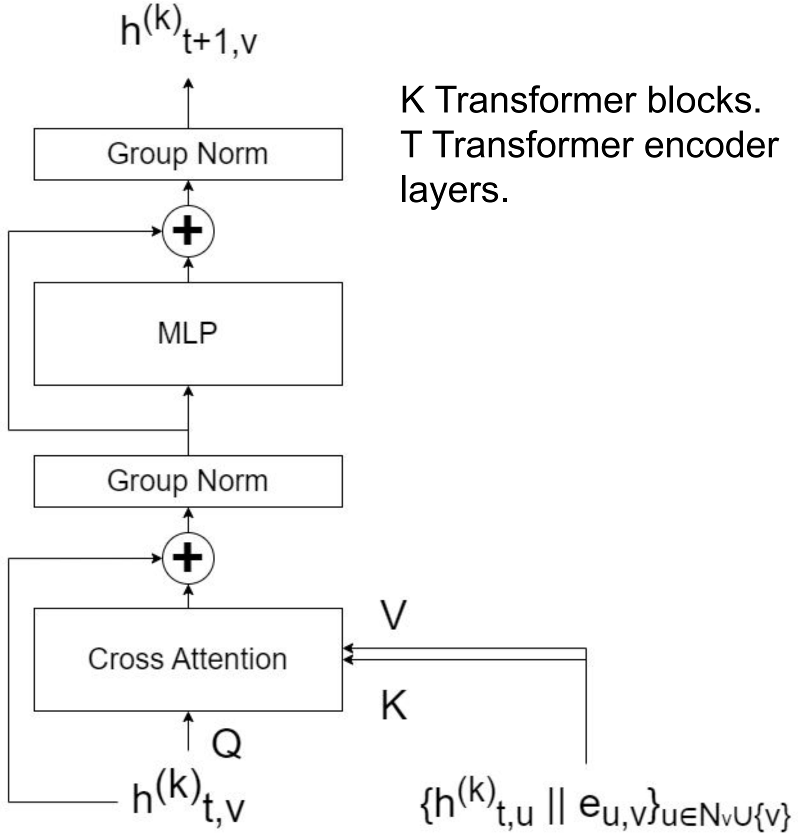
$$\mathcal{G} = (V, E, \{h_v^0\}_{v \in V}, \{q_e^0\}_{e \in E})$$

$$\mathrm{ATT}(h_{t-1,v}^{(k)}, \{h_{t-1,u}^{(k)} || e_{u,v}\}_{u \in \tilde{N}_v \cup \{v\}}) = \sum_{u \in \tilde{N}_v \cup \{v\}} \alpha_{t,(v,u)}^{(k)} \cdot (V_t^{(k)} [h_{t-1,u}^{(k)} || e_{u,v}])$$

$$\alpha_{t,(v,u)}^{(k)} = \mathrm{SoftMax}\left(\frac{(Q_t^{(k)} h_{t-1,v}^{(k)}) \cdot (K_t^{(k)} [h_{t-1,u}^{(k)} || e_{u,v}])}{\sqrt{d}} \right)$$

$$\mathbf{e}_\mathrm{v} = \left[\mathbf{h}_{0,\mathrm{v}}^{(1)}; \mathbf{h}_{0,\mathrm{v}}^{(2)}; \ldots; \mathbf{h}_{0,\mathrm{v}}^{(K)} \right], \forall \mathrm{v} \in \mathcal{V}$$

$$\hat{\mathbf{y}}_{\mathcal{G}} = \mathrm{softmax}\left(\mathbf{W} \mathbf{e}_{\mathcal{G}} + \mathbf{b} \right)$$



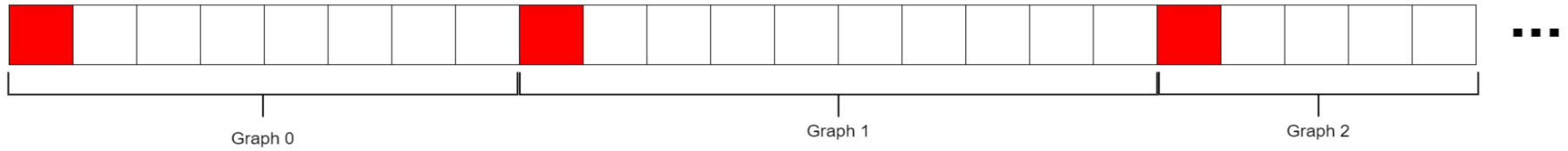
K Transformer blocks.
T Transformer encoder layers.

Input : [B, L+1, NF+NE]

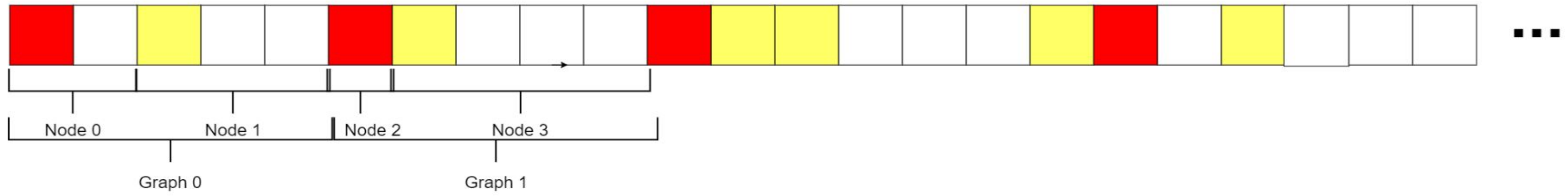
2.4. Methodology - Structure based models

Preprocessing

Node features



Edge features and connections



A

Graph positions in nodes	0	8	19	24	38	57	...
Graphs length in nodes	8	11	5	14	19	7	

C

Node positions in edges	0	8	19	24	38	57	...
Nodes length in edges	8	11	5	14	19	7	

B

Graph positions in edges	0	17	50	63	97	140	...
Graphs length in edges	17	33	13	34	43	21	

2.4. Methodology - Structure based models

Preprocessing

- We construct an array D of size [E,K,|V|,L] = [Epochs, n_blocks, n_nodes, n_neighbors],
  - B[:, :, v, :] is a tensor with entries sampled U{0, …, B[v,1]}, where B[v,1]} represents the number of neighbors of node v.

- We add D[...] + B[:,0] to obtain the positions of the generated node neighbors in the edge features and connections array. We access the values of the edge connections array for these indices and obtain the neighbors’ node positions in node features.

- To create a batch, we shuffle [A, B, C], and concatenate sequentially graphs until the total number nodes in the batch overpass the max batch node length. We need to update the positions of graphs in sliced A and B arrays and nodes in sliced C array relative to the starting graph and node in the batch.

- A batch is composed of the node and edge features, the updated sliced array A B graphs positions in node and edge features, and the updated sliced array C of nodes positions in edge features and edge connections, the sequence of labels corresponding to the graph found in the batch, the neighbors position list in nodes features and edge features for all nodes in the batch and blocks in the model.

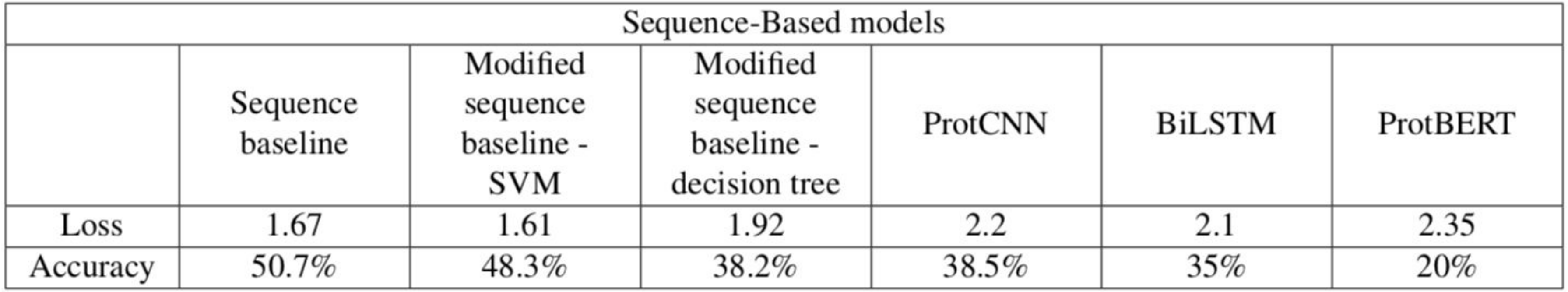
3.1. Results - Metrics used

- Multi-class log: Mainly focused on as it is the one used in kaggle

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(p_{ij})$$

- Accuracy: In order to have a more concrete idea of what we were predicting

3.2. Results - On sequence-based models



Sequence-Based models						
	Sequence baseline	Modified sequence baseline - SVM	Modified sequence baseline - decision tree	ProtCNN	BiLSTM	ProtBERT
Loss	1.67	1.61	1.92	2.2	2.1	2.35
Accuracy	50.7%	48.3%	38.2%	38.5%	35%	20%

4. Conclusion

- We tried several methods for sequence and structure based models.

- We didn’t manage to beat largely the sequence baseline (only a very low increase of performances with SVM classifier)

- We think that our poor results mainly come from the sampling of the data.

5. References

[1] Maxwell L. Bileschi, David Belanger, Drew Bryant, Theo Sanderson, Brandon Carter, D. Sculley, Mark A. DePristo, and Lucy J. Colwell. Using deep learning to annotate the protein universe. May 2019.

[2] Protein sequence classification (2022). https://deepnote.com/@ronakv/protein-sequence-classification- 33797a33-7311-48d3-8794-79a1ee26f406.

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

[4] Dai Quoc Nguyen, Tu Dinh Nguyen, and Dinh Phung. Universal graph transformer self-attention networks, 2019.