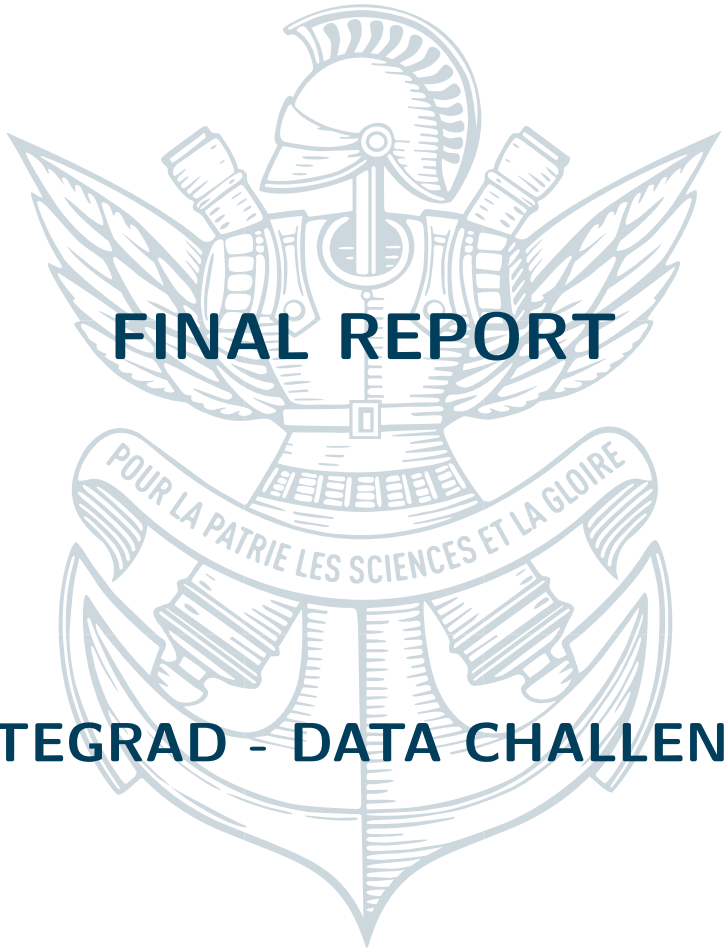


# FINAL REPORT

## ALTEGRAD - DATA CHALLENGE

January 22, 2023

Alexis Bouley
Matias Etcheverry

# CONTENTS

# 1 INTRODUCTION

Machine learning has many applications. First widely applied to images, it is gradually being extended to biology. In this approach, it is possible to predict properties on molecules before testing them in the laboratory. The challenge is twofold: to limit research costs and to enrich scientific advances. With this advance, it is notably possible to predict a type of interaction between two molecules or to modify a molecule virtually so that it respects certain properties [1]. Despite recent scientific advances in the application of deep learning to biology, the functioning of proteins is still not fully understood and research in this field is still ongoing.

This report aims to study more than 6000 proteins in the context of the *Advanced Learning for TExt and GRAph Data* course. The objective is to determine a precise property of these proteins. Our work consists of proposing an embedding approach, which allowed us to achieve good predictions. We will first review the specified dataset. We will then present our 2 approaches and how to combine them to obtain good results.

# 2 THE DATASET

The dataset consists in the representations of 6111 proteins. Proteins are large molecules composed of amino acids. There are 20 types of amino acids. Each of these acids provides the molecule with different chemical properties, such as hydrogen bounds, or physical properties, such as specific spatial folding.

## 2.1 Provided data

We have data on 6111 proteins, separated into two categories:

- On the one hand, we have the amino acid sequence for each of the proteins. This can be interpreted as a sentence where 1 word represents 1 amino acid. For instance, `RWEHDKFRE` is the smallest amino acid sequence in the dataset. Figure 1 shows the distribution of the

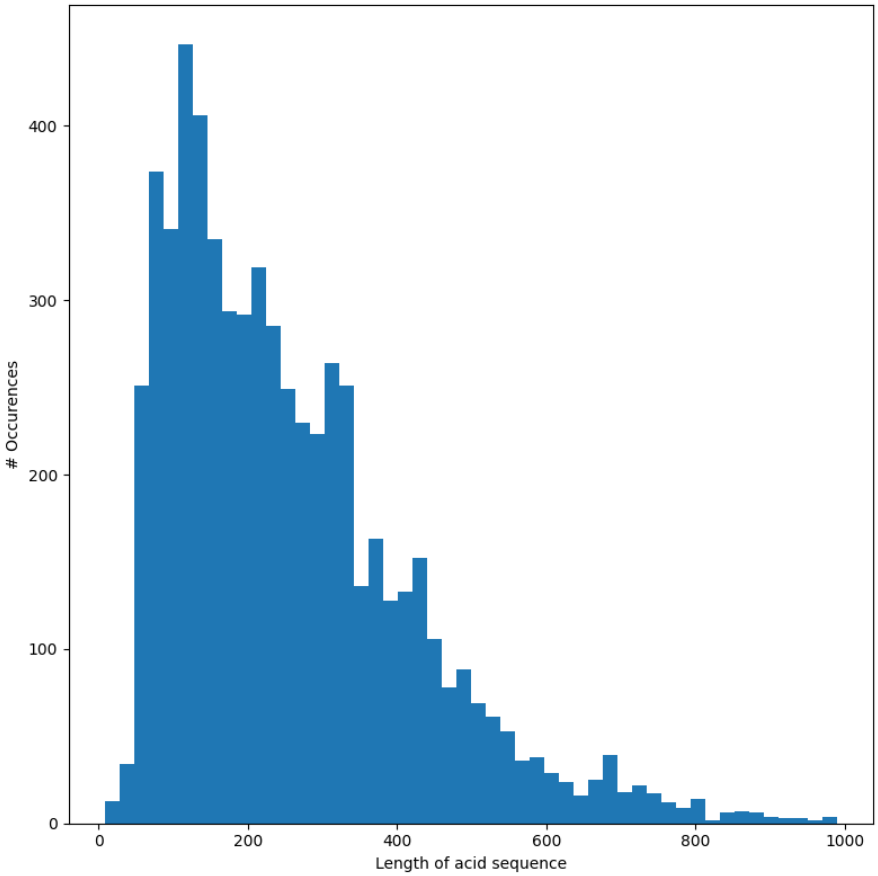


Figure 1: Histogram of the length of amino acid sequences

length of the sequences in the dataset. While most of the proteins have a 100-300 long amino acid sequences, the distribution has a heavy tail.

- On the other hand, we have access to the graph structure of proteins. In this structure, the amino acids are seen as nodes. A list of edges is also provided. There are 3 ways to define an edge between 2 amino acids[1]:
  - the distance between 2 acids is below a threshold value
  - the 2 acids have a peptide bond,
  - the 2 acids have a hydrogen bond

  In addition, we are given for each node its amino acid type, its hydrogen bond acceptor and donor status as well as features derived from the *EXPASY* protein scale dataset [3].

[1] A fourth way is also possible, but is mostly always null: a $k$-NN edge.

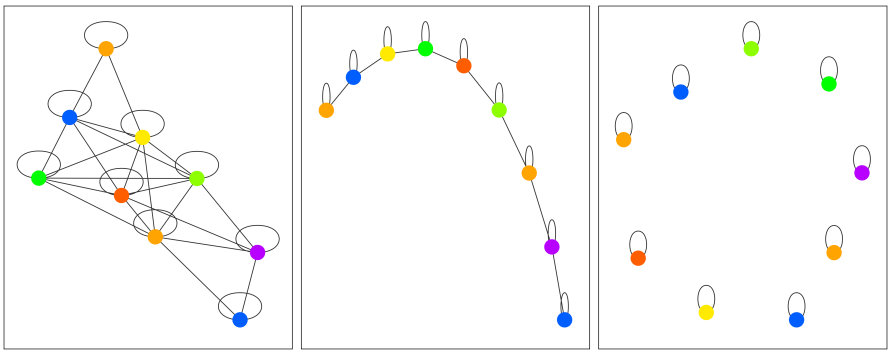

Figure 2: Graphs of the `RWEHDKFRE` protein. From left to right: distance, peptide bond and hydrogen bound egdes. Each color of the nodes represents a type of amino acid.

We specify that these features are entirely defined by the type of amino acid that a node is. Figure 2 shows the 3 possible graphs for the smallest amino acid sequence. The most informative graph is obtained with distance edges.

Each protein is associated with a class. The class represents a characteristic of the location where the protein performs its function obtained from the Cellular Component ontology. This class is rather difficult to understand. Thus, it will push us to cover all possible data to achieve good performance. Figure 3 shows the smallest and longest protein for each class. A novice human eye cannot distinguish the class using only graph and amino acids data.

## 2.2 Splitting the dataset

First, we notice that there is a significant class imbalance in the dataset, as shown in Figure 4. We also looked at the class imbalance through the dataset. To do so, we sliced the dataset every 1000 samples and then looked at the histogram of the classes on each of these slices. These histograms had more or less the same profile on each slice, which suggests that the dataset is simlarly imbalanced everywhere, especially on the test set.

We then split the dataset in a 90-10% ratio to define a training and validation set. All our models are then trained on the training set then validated on the validation set.

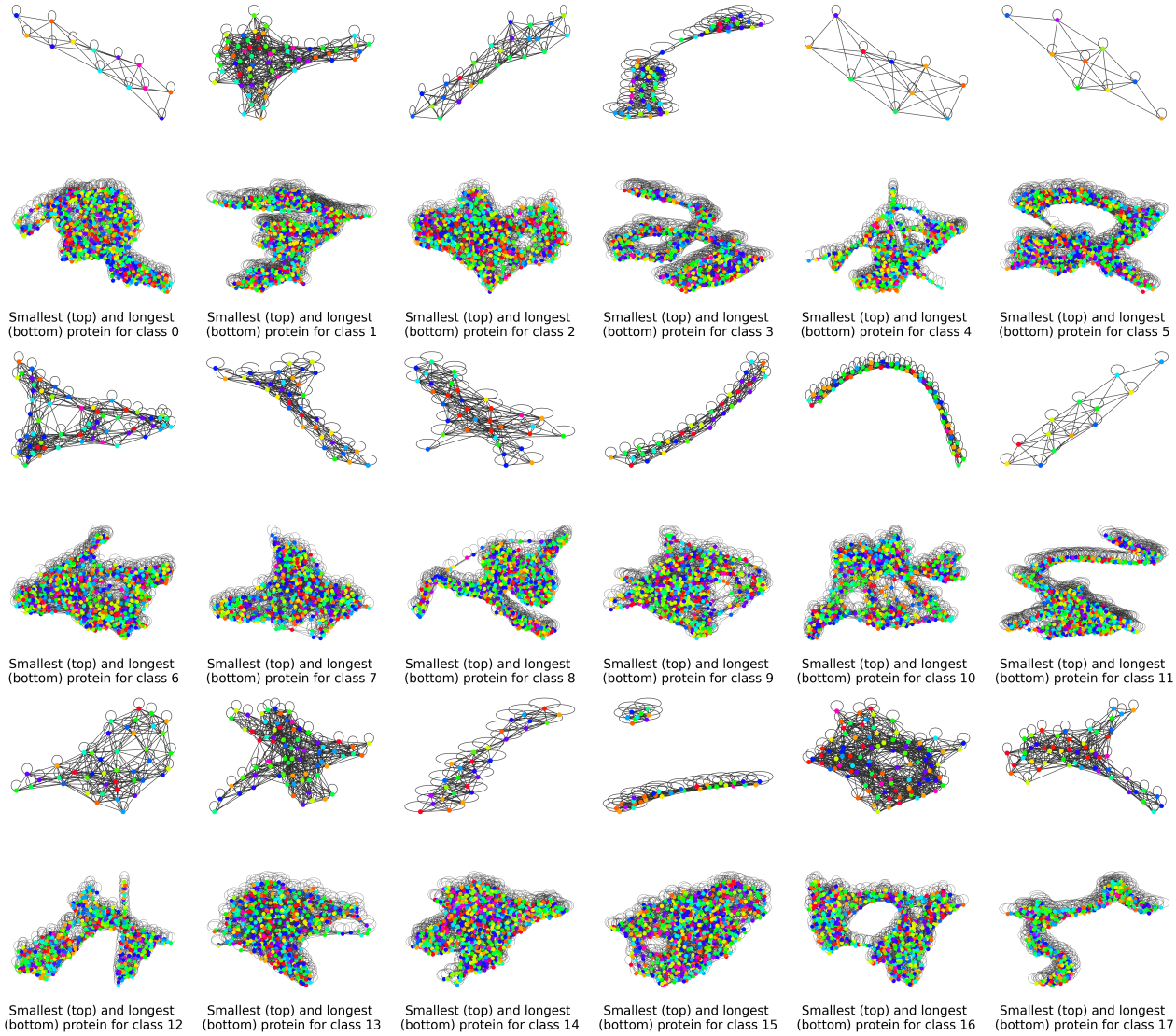


Figure 3: Smallest and longest protein for each label. Each color of the nodes represents a type of amino acid.

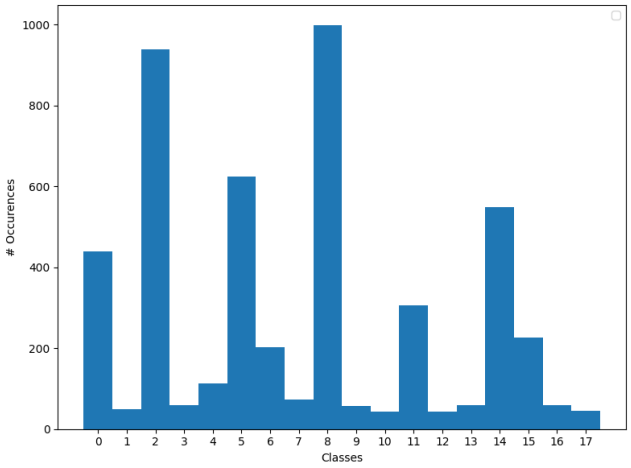


Figure 4: Distribution of classes in the dataset.

# 3
# LOSS & METRIC

In order to train our model, we use the multiclass cross-entropy loss, defined as

$$\text{CrossEntropyLoss} = -\sum_{i}\sum_{c=1}^{18} y_{i,c}\log(\hat{p}_{i,c})$$

where $y_{i,c}$ is the binary label, which equals 1 if observation $i$ is class $c$ and $\hat{p}_{i,c}$ is the predicted probability of observation $i$ belonging to the class $c$. We think this loss is relevant for our problem as it tells how confident a prediction is. Moreover, the Kaggle challenge expects us to submit a list of probabilities for each protein in the test set. Finally, accuracy is used as an additional metric, as it has the merit of being more readable for a human.

# 4
# PROPOSED APPROACHES

The amount of data provided is really small: we have a total of 6111 proteins, including the test set. This is actually very little, especially for applying deep learning methods, which are known to improve with the amount of data. This remark has motivated us to firstly look for

lightweight methods before exploring pretrained embeddings.

## 4.1 Sequence algorithms

A first approach is to focus only on the amino acid sequence. Indeed, a protein is simply a sequence of amino acids. Its folding will cause particular physico-chemical properties to the protein. Thus, if we apply a prediction model to the sequence, i.e. before folding, then the model will be able to compute features simpler than 3D folding graphs.

**Term Frequency-Inverse Document Frequency** Each amino acid sequence is interpreted as a sentence. The Term Frequency-Inverse Document Frequency (TF-IDF) algorithm is a very good solution for deriving features on text documents. Since we have very little data, it is probably more efficient to use an algorithm that does not require learning to derive features while limiting overfitting. We implement the TF-IDF embedding so that it focuses on block of amino acids of lengths 1, 2 and 3.

In addition to the features calculated by the TF-IDF, we add 11 global graph features such as the average degree of amino acids or the number of related components. This approach is mostly motivated by the incomplete and partial representation of the sequence of a protein. Indeed, a sequence can't capture the number of connected components, which may influence the class of a protein. A logistic regression algorithm is then applied on those features for class prediction. A hyperparameter optimisation is then conducted. The regularisation parameter is reduced, the solver is changed and a balanced distribution between classes is imposed.

**Trying to improve TF-IDF** Another idea stems from the observation that the absolute value of the logistic regressor coefficients were highly correlated with the sum of the features over the dataset induced by the TF-IDF. This makes us think that TF-IDF is a good method of calculating the features that will be relevant for classification. With this in mind we tried to increase the range of the n-gram to 10 in order to take into account more distant amino acid relationships. A PCA then allows only the most important 3000 features to be selected once the TF-IDF algorithm has been run, before classifying. Unfortunately this idea does not improve our performance.

**Heavier sequence features** Another idea was to construct a model which directly learns the features of amino acids, before feeding them to a classifier. This constructed model would

be able to capture long and short range interaction between amino acids in a sequence. Thus, we apply an embedding layer on the amino acid sequence. Then we tried 2 types of models that could take into account the information induced by the sequence of a protein:

- A bidirectional LSTM and a transformer to take into account the order of amino acids
- A convolutional residual network to take into account the proximity between different types of amino acids.

Unfortunately these two types of models use too many parameters and lead to overfitting, in addition to giving disappointing performances on the training set.

## 4.2 Structure algorithms

A second approach was to study the graph structure of proteins. This approach is motivated by the fact that the 3D arrangement of proteins influences their activity, and therefore possibly their class. We decide to apply a Graph Convolutional Network (GCN) to the graphs of the proteins [5].

The nodes of each graph correspond to the amino acids of the sequence. Furthermore, we have easy access to the features of each amino acid in the dataset. These features contain the type of amino acid, but 66 other very technical features. It is hoped that these other features proposed in the *EXPASY* protein scale dataset will accelerate the convergence of our model and improve our performance.

3 different adjacency matrices are created for each graph: each matrix corresponds to an edge type. It is important to note that the adjacency matrix from the distance edges is binary. We wish to modify this matrix to take into account the information induced by the distance between 2 acids. Our intuition is that the closer the distance between 2 nodes, the closer their adjacency weights should be. The distance is a function defined in $\mathbb{R}$, whereas we want the adjacency weight between 2 acids to be between 0 and 1. A good idea would be to use the sigmoid function but its starting set is $\mathbb{R}$ which does not coincide with the distance function. That is why we normalise and center the distance between the nodes, then multiply by $-1$ so that the closest nodes have the highest adjacency coefficients. Finally, we apply the sigmoid function to obtain our new adjacency weights in $[0, 1]$.

We build a GCN that takes into account the 3 types of edges at each iteration. For this, we use 3 message passing layers, one per adjacency matrix. The outputs of these 3 message

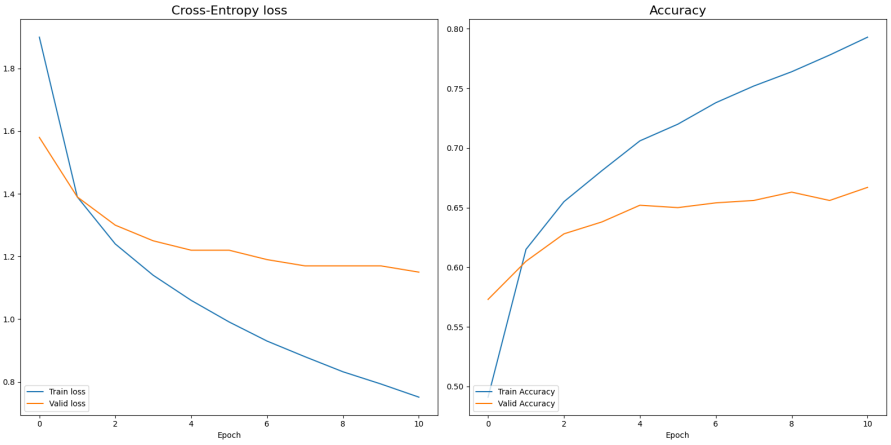


Figure 5: Learning curves of the improved version of the GCN, using embedded sequence features.

passing layers are then concatenated, to form a set of features. This set of features is then fed again to 3 message passing layers, one per adjacency matrix, The outputs of those second 3 message passing layers are concatenated and fed to fully connected layers which outputs a probability to belong to one of the 18 classes.

**Improving GCN** Instead of using the given features, we looked for pretrained solutions. With this approach, we decided to use ProtBert [2]. For instance, [4] used a similar approach to classify protein-protein interaction. ProtBert is based on Bert model which is pretrained on a large corpus of protein sequences in a self-supervised fashion. We used the public implementation. The ProtBert model was pretrained on Uniref100, a dataset consisting of 217 million protein sequences.

ProtBert outputs 1084 embedded features for each node in the sequence. Due to resource restrictions, we limit the embedding to the first 200 amino acids. Intuitively, we think that the class of a protein can entirely be contained in the first 200 acids of the sequence. We also only use the 256 first embedded features of each node. These assumptions simplify the training, both in terms of computational resources and overfitting. Figure 5 shows the training and validation metrics during the training phase. We notice that the model is overfitting. To reduce this phenomenum, we restrict the model to only 1 message passing layer. We also add a weight decay and a dropout layer of probability $p = 0.5$. We also apply a PCA on the 256 embedded features of the sequence.

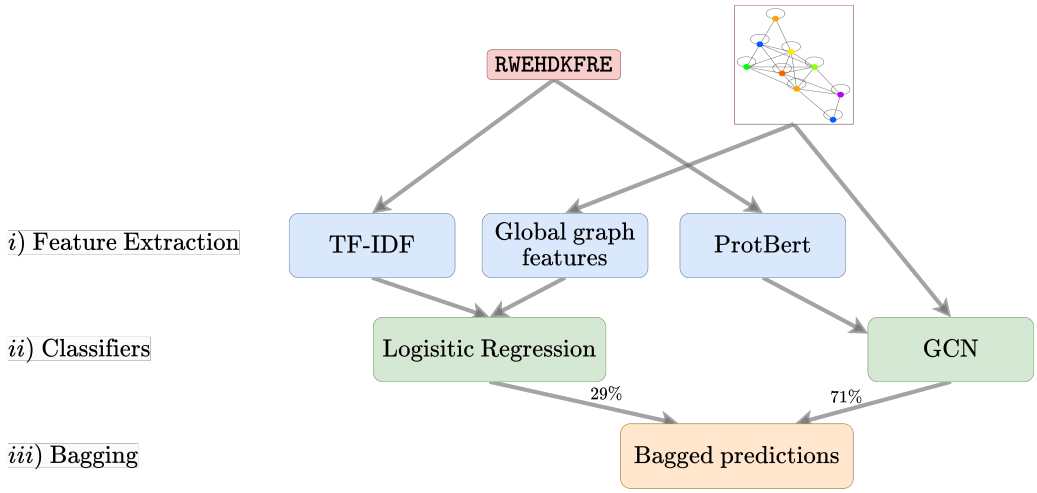


Figure 6: Full pipeline using sequence and structure algorithms.

## 4.3 Combining best of the approaches

Finally, we combine the best of the sequence and structure algorithms. On the one hand, we use a TF-IDF embedding ad well as global graph features fed to a logistic regression. On the other hand, we have a GCN applied to the protein graphs, whose features are outputs of the ProtBert model applied on the sequence on the amino acids. Those are the 2 algorithms which gave the best results. In order to combine them, we use a simple bagging method on the validation set, where the final predictions $\hat{y}$ are:

$$\hat{y} = \alpha\ \hat{y}_{\text{TF-IDF}} + (1 - \alpha)\ \hat{y}_{\text{GCN}}$$

Where $\hat{y}_{\text{TF-IDF}}$ and $\hat{y}_{\text{GCN}}$ are respectively the predictions of the sequence and structure algorithms, and $\alpha \in [0, 1]$ is the ratio of prediction importance. In our study, we find that $\alpha = 0.29$ gave the best results, with respect to the performance metrics. This shows, that the pretrained GCN model is giving the best results. Figure 6 shows the full pipeline to obtain predictions from the 2 models.

Table 1 shows the performance metrics, both on the validation set and on the public test set on Kaggle. The TF-IDF with global graph features is consistently better than the TF-IDF with sequence features only. We also notice that the pretrained GCN is performing better than both the TF-IDF methods. Moreover, we see that the two algorithms, sequence and structure, do not use the same information for their predictions because the bagging method greatly improves their performance. Most of our performance comes from the structure algorithm.

| | Val Loss | Val Accuracy | Kaggle Public Score |
|---|---|---|---|
| TF-IDF | 1.41 | 57.1% | 1.29 |
| TF-IDF with global graph features | 1.33 | 59.5% | 1.24 |
| Pretrained GCN | 1.15 | 66.7% | 1.07 |
| Bagged combination | **1.07** | **70.3%** | **0.993** |

Table 1: Performance metrics on validation and test dataset

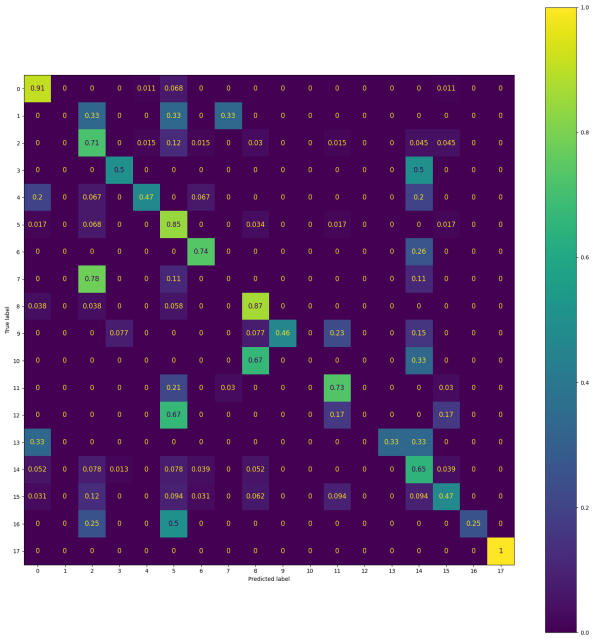


Figure 7: Confusion matrix on the validation set.

Indeed, although this algorithm deals with protein graphs as input, the pre-trained embedding layer gives information about the sequence in the features.

We finally plot the confusion matrix, as show in Figure 7. It display the ratio of true to false predictions for each class. The large class imbalance can be seen on the confusion matrix. Our model will be more likely to predict majority classes, such as 0, 2, 5, 8 and 14. Although this behaviour should be avoided, the similarity between our performance metrics on our validation set and on the public kaggle test set suggests that the public kaggle test set also has a large class imbalance. Some classes are very easily recognised, such as class 17. Finally, we notice that our validation method suffers from the lack of data. Some classes are too poorly represented in our validation dataset, which causes inaccuracies in our confusion matrix.

# 5
# GOING FURTHER & CONCLUSION

Our performance with ProBert embedding gave us the insight that the first layer of embedding before any deep learning solution was crucial for good predictions. Our current solution could be more robust, especially by introducing data augmentation. For example, we could consider the mirror operation that reverses a protein sequence but keeps the same class. Thus, with more computational resources, we would have liked not to limit our embedding to only the first 200 amino acids of a sequence. It would also have been interesting to test more recent embedding methods, or train on more data, such as the Big Fantastic Database (BFD). Better embedding methods will certainly give better results, with our current GCN architecture. Building good features can also be used for traditional machine learning methods, which are still to be explored. Finally, we believe that it is essential to develop an experience in biology to really understand protein labels, to use different models.

# REFERENCES

[1] Haroldas Bagdonas et al. “The case for post-predictional modifications in the AlphaFold Protein Structure Database”. In: *Nature Structural  Molecular Biology* 28 (Oct. 2021). DOI: 10.1038/s41594-021-00680-9.

[2] Ahmed Elnaggar et al. *ProtTrans: Towards Cracking the Language of Life’s Code Through Self-Supervised Deep Learning and High Performance Computing*. 2020. DOI: 10.48550/ARXIV.2007.06225. URL: https://arxiv.org/abs/2007.06225.

[3] Elisabeth Gasteiger et al. “Protein Identification and Analysis Tools on the ExPASy Server"”. In: *The Proteomics Protocols Handbook"*. Ed. by John M." Walker. Totowa, NJ": Humana Press", 2005", 571–607". ISBN: 978-1-59259-890-8". DOI: 10.1385/1-59259-890-0:571". URL: https://doi.org/10.1385/1-59259-890-0:571%22.

[4] Kanchan Jha, Sriparna Saha, and Hiteshi Singh. “Prediction of protein–protein interaction using graph neural networks”. In: *Scientific Reports* 12 (May 2022), p. 8360. DOI: 10.1038/s41598-022-12201-9.

[5] Muhan Zhang et al. "An End-to-End Deep Learning Architecture for Graph Classification". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (Apr. 2018). DOI: 10.1609/aaai.v32i1.11782. URL: https://ojs.aaai.org/index.php/AAAI/article/view/11782.