Bacteria Taxonomic Classification using Graph Neural Networks

EAIS 24
May 23-24, 2024, Madrid, Spain

D. Amato [1] **S. Calderaro** [1] **G. Lo Bosco** [1]
R. Rizzo [2] **F. Vella** [2]

[1]University of Palermo - Department of Mathematics and Computer Science

[2]National Research Council of Italy - Institute for High Performance Computing and Networking

salvatore.calderaro01@unipa.it
github.com/salvatorecalderaro

Outline

introduction

the used dataset

graph neural networks

graph construction

experimental results

conclusions

Introduction

DNA sequence categorization represents one of the most interesting problems of bioinformatics since sequences with similar structures have similar properties.

In this paper, we presented a new deep-learning approach based on the combination of graph-based encoding and the use of a Graph Neural Network.

We tested our proposal for the bacteria taxonomic classification by conducting experiments on a dataset of 3000 16S sequences, achieving results that were comparable to state-of-the-art ones.

The used dataset

The 16S rRNA sequences were obtained from the RDP Ribosomal Database Project II repository (version 10.27).

3,000 sequences were selected and categorized into five hierarchical taxonomic levels: **PHYLUM** (the broadest), **CLASS**, **ORDER**, **FAMILY**, and **GENUS** (the most specific).

The dataset contains 1,000 sequences for each of the three most common bacterial phyla: *Proteobacteria*, *Actinobacteria*, and *Firmicutes*.

The used dataset

Phyla	Number of categories for each taxa				
	PHYLUM	CLASS	ORDER	FAMILY	GENUS
Actinobacteria	1	1	4	12	79
Firmicutes	1	3	5	19	110
Proteobacteria	1	2	13	34	204
Total number of classes	3	6	21	65	393

Table: 16S bacteria dataset composition.

The used dataset

The dataset is well-balanced at the **Phylum** level,but it becomes significantly imbalanced for the remaining taxonomic categories.

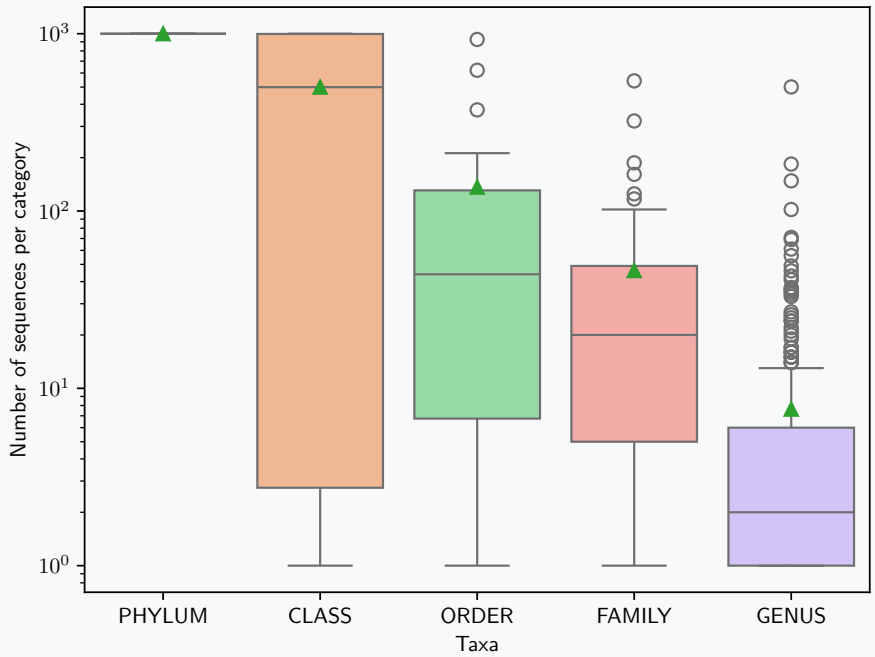

Figure: 16S bacteria dataset samples distribution.

Graph Neural Networks

Here, we recall some basic notions regarding the graph concept. A **graph** G is a tuple $G = (V, E, X)$ where:

- V is the set of nodes;
- E is the set of edges;
- $X \in \mathbb{R}^{|V| \times D}$ is the node feature matrix;
- $A \in \mathbb{R}^{|V| \times |V|}$ is the so-called **adjacency matrix**;
- $D \in \mathbb{R}^{|V| \times |V|}$ is the so-called **degree matrix**.

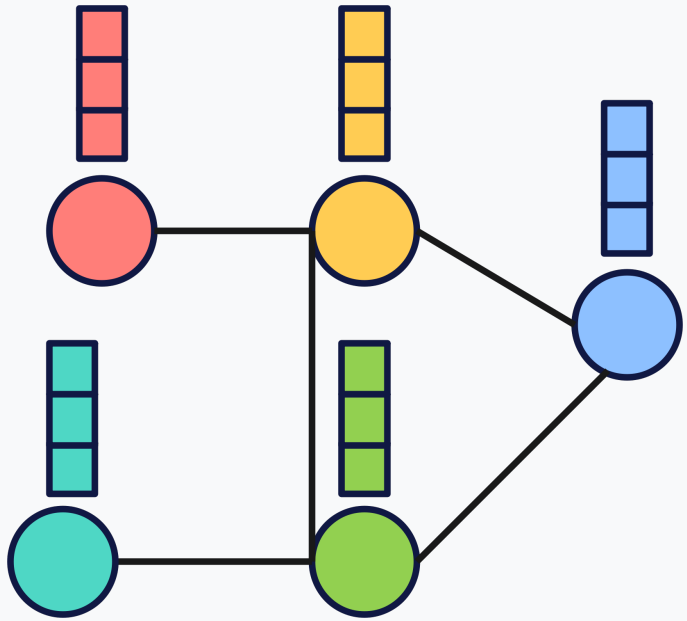

Figure: A simple graph.

Graph Neural Networks

Graph Neural Networks (GNN) are a particular kind of deep neural network that processes data with a graph structure.

These networks have the capability to execute various tasks:

1. **Node prediction**: we have a graph with a set of labelled nodes, and the goal is to assign a label to unlabelled ones.
2. **Link prediction**: we have a graph and we want to predict future or missing links in the graph.
3. **Graph classification**: the goal is to predict the graph class starting from a set of labelled graphs used for the neural network training.

We tackle the issue of classifying bacteria by framing it as a graph classification problem.

Graph Neural Networks

A GNN is a series of stacked layers. Each one performs the following operations:

- **AGGREGATE**: aggregates the information from the neighbours of each node;
- **COMBINE**: updates the current node representation by combining the aggregated information.

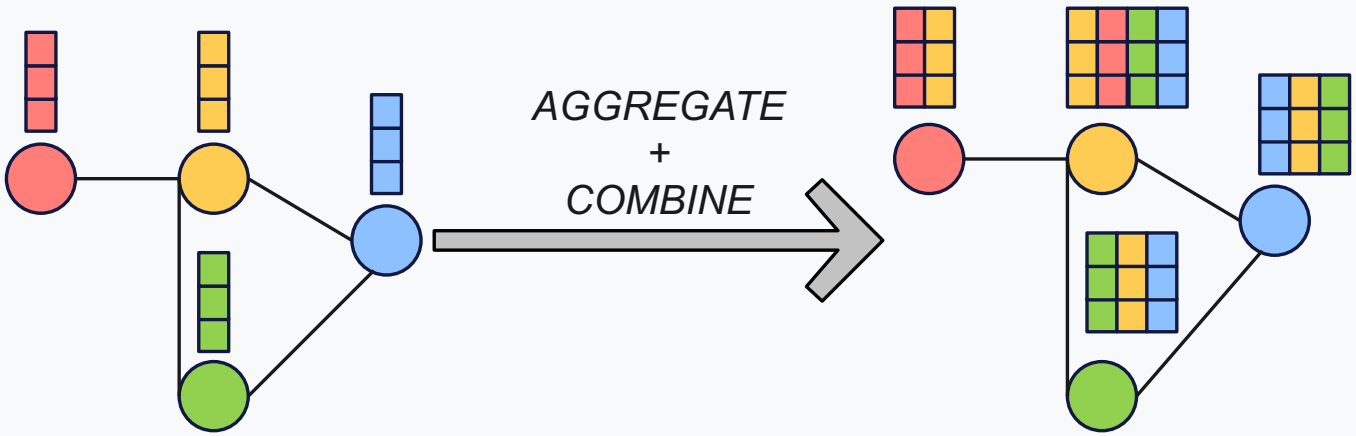

Figure: Overall mechanism of a GNN layer.

Graph Neural Networks

For the graph classification task to obtain a feature vector representing the whole graph, **READOUT** must be performed. For example, given a graph G and the final node representation H, the **AVERAGE READOUT** operation is defined as follows:

$$h_G = \frac{1}{|V|} \sum_{v \in V} h_v.$$

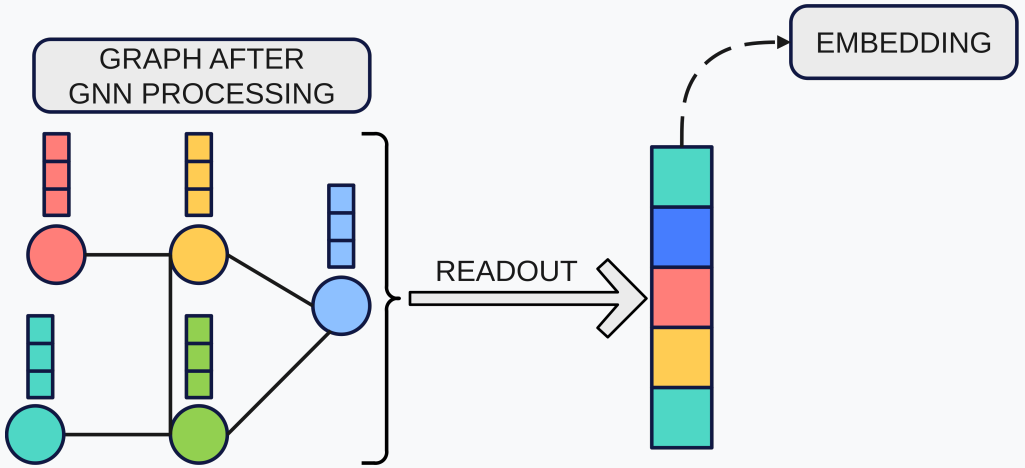

Figure: READOUT mechanism.

Graph Neural Networks

Graph Convolutional Networks (GCN) are among the most used GNN. A GCN layer is described by the following equation:

$$H^{k+1} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{k}W^{k}\right).$$

A GCN layer transforms the node embeddings, employing a non-linear transformation on the new embeddings. Each new embedding is computed as the mean of the neighbours of all nodes.

Graph construction

A **De Bruijn graph** is a representation of a string of symbols, in our case, a DNA sequence.

A sequence is represented as a composition of their subparts, the so-called k-mers.

To build the graph, given a sequence S on the alphabet $\Sigma = \{A, C, G, T\}$ and an integer $k \geqslant 2$:

1. idenity all the k-mers of the sequence S (the number of possible k-mers of S is $L - k + 1$ where $L = |S|$)
2. assign $k - 1$-mers to the nodes;
3. connect the nodes u and v if u overlaps v.

Graph construction

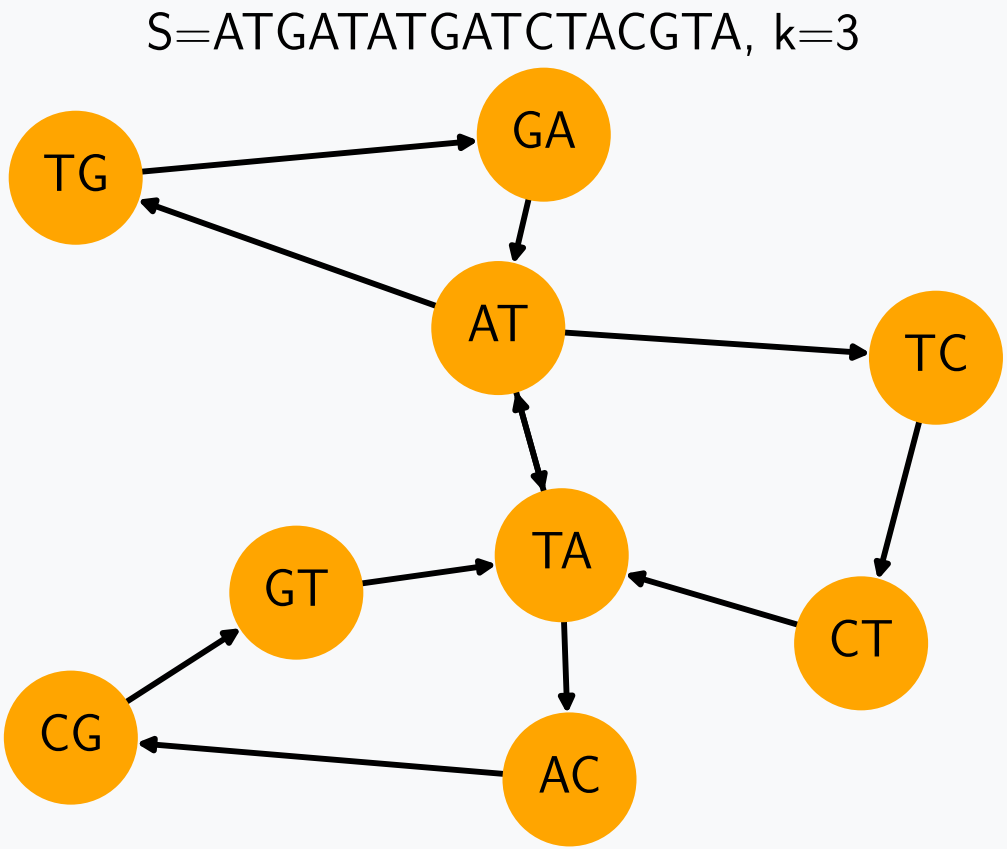

Figure: Example of a De Bruijn graph for a short sequence.

Graph construction

For the GNN computation, assigning the initial features to the node is essential. The matrix $X \in \mathbb{R}^{|V| \times D}$ stores this information.

To encode graph nodes we use one-hot encoding:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^\top$$
$$C = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}^\top$$
$$G = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}^\top$$
$$T = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^\top$$

Graph construction

Using tis econding - for each node - we obtain a matrix of size $|\Sigma| \times k - 1$. For instance for the sequence $S = AATTG$ the obtained matrix will be:

$$X_S = \begin{bmatrix} \mathbf{1} & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \\ 0 & 0 & \mathbf{1} & \mathbf{1} & 0 \end{bmatrix}$$

Finally, to get the node features, we concatenate each matrix column obtaining a vector with $D = |\Sigma| \times k - 1$ components.

Experimental results

For our experimental activity, we perform two kinds of experiments:

1. we used the full-length sequences;
2. we extracted 500 random consecutive nucleotides from the sequences, obtaining subsequences of a length of 500 bp.

Considering that there is a pre-defined split into training and testing, we perform 10-fold cross-validation for both setups.

Experimental results

The experiment workflow starts with building the Bruijn graph with $k = 5$.

Experiment	**#nodes**	**#edges**	**time (s)**
full-length sequences	249.880 ± 3.013	1445.148 ± 50.934	259.034
500 bp sequences	209.600 ± 6.878	753.646 ± 24.186	81.306

Table: Statistics of the builded graphs.

Experimental results

Once we obtained the graphs, we trained a GNN using the Adam algorithm with a mini-batch of 32.

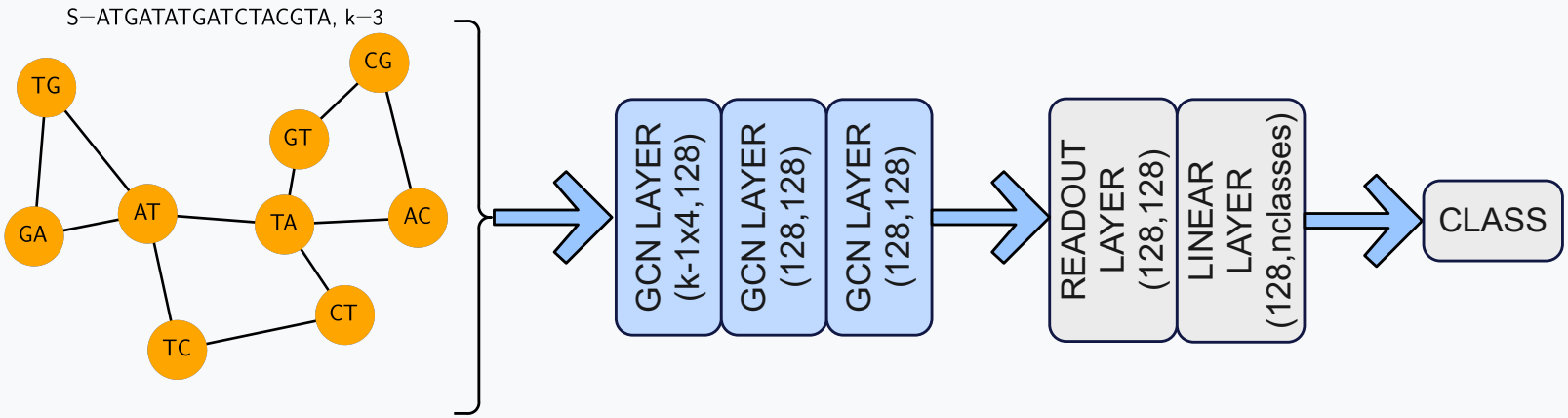

Figure: The proposed GNN.

Experimental results

TAXA	Acc.	Pre.	Re.	F1	Epochs	Train Time (s)	Test Time (ms)
PHYLUM	0.998 ± 0.003	0.998 ± 0.003	0.997 ± 0.004	0.997 ± 0.004	150	45.806 ± 1.077	26.591 ± 1.846
CLASS	0.996 ± 0.002	0.999 ± 0.001	0.996 ± 0.002	0.997 ± 0.001	200	62.848 ± 0.654	27.113 ± 1.556
ORDER	0.960 ± 0.010	0.967 ± 0.007	0.960 ± 0.010	0.961 ± 0.009	600	180.024 ± 3.190	25.002 ± 1.889
FAMILY	0.949 ± 0.012	0.970 ± 0.008	0.950 ± 0.012	0.956 ± 0.010	1500	461.574 ± 8.064	26.141 ± 1.870
GENUS	0.776 ± 0.020	0.848 ± 0.024	0.776 ± 0.020	0.797 ± 0.021	2500	740.929 ± 3.026	23.902 ± 1.764

Table: Results or the full-length sequences.

TAXA	Acc.	Pre.	Re.	F1	Epochs	Train Time (s)	Test Time (ms)
PHYLUM	0.995 ± 0.007	0.995 ± 0.007	0.995 ± 0.007	0.995 ± 0.007	150	37.324 ± 0.317	21.747 ± 2.335
CLASS	0.986 ± 0.011	0.989 ± 0.012	0.986 ± 0.011	0.987 ± 0.012	200	49.714 ± 0.498	21.409 ± 1.050
ORDER	0.908 ± 0.018	0.923 ± 0.015	0.908 ± 0.018	0.910 ± 0.019	600	148.656 ± 1.061	21.461 ± 1.618
FAMILY	0.847 ± 0.024	0.884 ± 0.029	0.846 ± 0.024	0.857 ± 0.027	1500	373.334 ± 1.273	21.601 ± 1.293
GENUS	0.616 ± 0.018	0.713 ± 0.023	0.616 ± 0.018	0.647 ± 0.019	2500	614.462 ± 15.961	20.779 ± 1.965

Table: Results or the 500 bp subsequences.

Experimental results

We compare our approach with one based on k-mers encoding and using a Recurrent Neural Network.

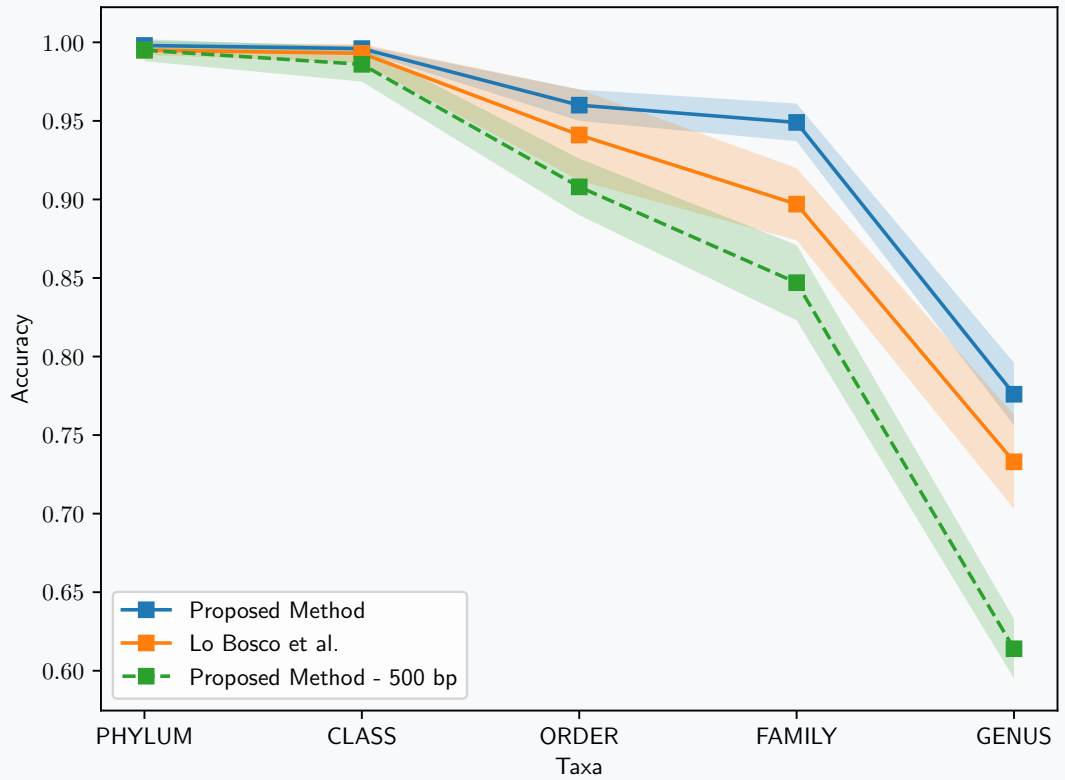

Figure: Comparison results in terms of Accuracy.

Conclusions

This paper presented a method to represent and classify nucleotide sequences.

GNN has shown to be an elegant and effective method to process the data, providing relevant results in the classification task.

Future works include the analysis of the effect on the classification performance of the length for the k-mers and the comparison of this method with short and long sequence datasets.

Thank you for your attention !