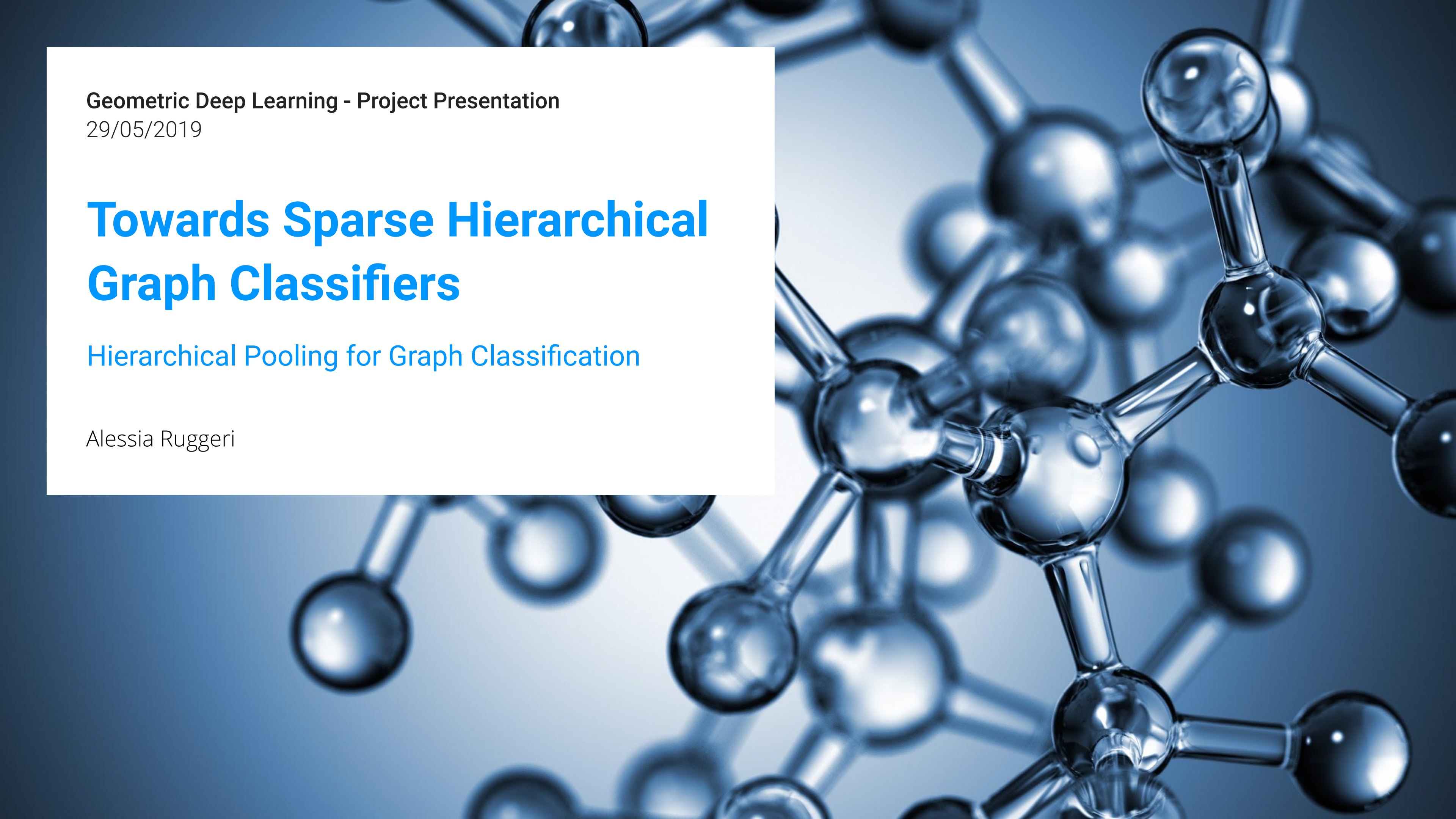


Geometric Deep Learning - Project Presentation
29/05/2019

Towards Sparse Hierarchical Graph Classifiers

Hierarchical Pooling for Graph Classification

Alessia Ruggeri

Introduction

The focus of this project is on the paper *Towards Sparse Hierarchical Graph Classifiers*, which present an interesting architecture for **hierarchical graph classification**.

The aim of the project is to analyze, implement and try to reproduce the results presented by the paper.

1.

Problem formulation & Background

Problem formulation

The paper studies the problem of **graph classification**, related to the generalization of the pooling layer to graphs.

In order to preserve the topological information, a differentiable graph pooling layer is needed, which can generate **hierarchical representations** of graphs.

Background

Previous approaches to the **generalization of the pooling layer to graphs** through the aggregation of node representations:

> Global pooling layer
>
> Clusters which coarsen the graph in a hierarchical manner through a fixed and pre-defined cluster assignment
>
> Soft clustering assignments, learned in a differentiable way (DiffPool)

Background

The soft clustering assignment performed by **DiffPool** leads to state-of-the-art results on several graph classification benchmarks, but it requires a quadratic $O(kV^2)$ storage complexity to be executed.

This paper presents a **Hierarchical Pooling** included in an architecture able to replicate almost the same performance, requiring only $O(V + E)$ storage complexity.

2.

Model

Model - Input graphs

The input graph is represented by:

> Matrix of node features, $\mathbf{X} \in \mathbb{R}^{N \times F}$
>
> Adjacency matrix, $\mathbf{A} \in \mathbb{R}^{N \times N}$

where *N* is the number of nodes in the graph and *F* the number of features for each node. *A* is binary and symmetric.

Model - Layers

In order to build the presented model architecture for graph classification, we need to define:

> Convolutional layer
>
> Pooling layer
>
> Readout layer

Model - Convolutional Layer

The architecture makes use of a **mean-pooling propagation rule**:

$$\mathrm{MP}(\mathbf{X}, \mathbf{A}) = \sigma\left(\hat{\mathbf{D}}^{-1}\hat{\mathbf{A}}\mathbf{X}\boldsymbol{\Theta} + \underbrace{\mathbf{X}\boldsymbol{\Theta}'}_{\text{skip-connection}}\right)$$

where:

> $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ is the adjacency matrix with inserted self-loops
>
> $\hat{\mathbf{D}}$ is the corresponding degree matrix, i.e. $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$
>
> σ is the rectified linear (ReLU) activation
>
> $\boldsymbol{\Theta}, \boldsymbol{\Theta}' \in \mathbb{R}^{F \times F'}$ are learnable linear transformations applied to every node

Model - Pooling Layer

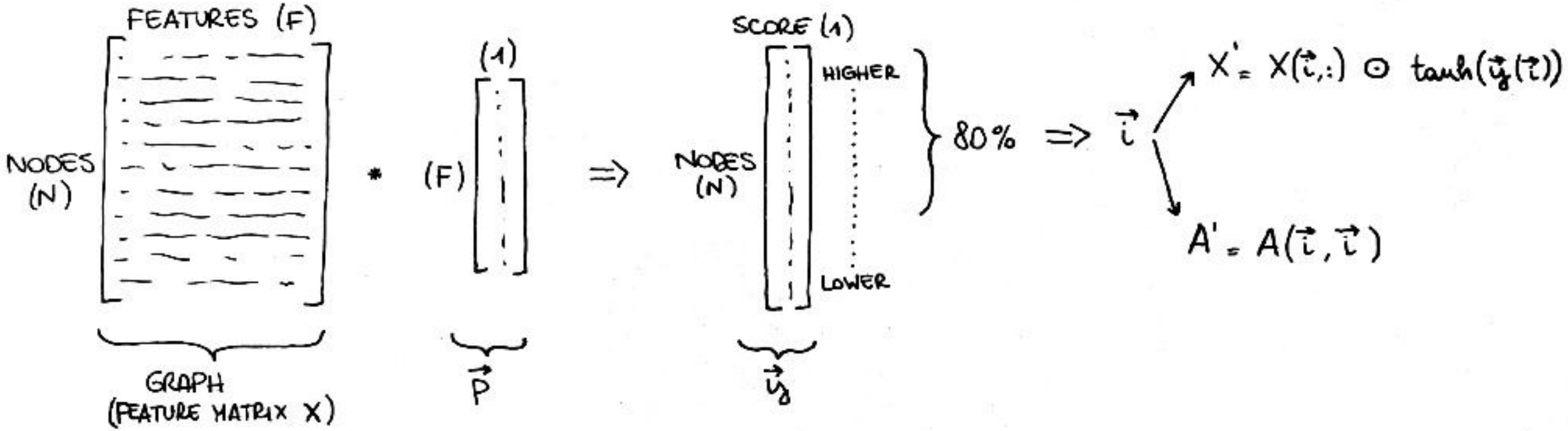
The hierarchical pooling layer reduces the graph with a **pooling ratio**, $k \in (0,1]$, decreasing the number of nodes in a graph from N to $\lceil kN \rceil$. The choice of the nodes to drop is done based on a **projection score** against a learnable vector $\vec{p}$, which is also used as **gating value**.

Model - Pooling Layer

$$\vec{y} = \frac{\mathbf{X}\vec{p}}{||\vec{p}||} \qquad \vec{i} = \mathrm{top}-k(\vec{y}, k) \qquad \tilde{y} = \tanh(\vec{y}(\vec{i}))$$

$$\tilde{\mathbf{X}} = \mathbf{X}(\vec{i}, :) \qquad \mathbf{X}' = (\tilde{\mathbf{X}} \odot \tilde{y}) \qquad \mathbf{A}' = \mathbf{A}(\vec{i}, \vec{i})$$

where ||·|| is the L_2 norm and $\odot$ is (broadcasted) element-wise multiplication.



Model - Readout Layer

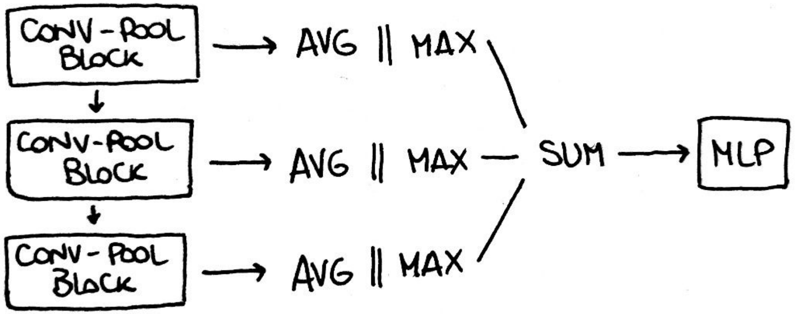
The readout layer flatten the information about the input graph in a **fixed-size representation**. This operation is performed by **concatenating** the results of a global average pooling and a global max pooling at the end of each conv-pool block, and then applying a global sum pooling before submitting the information to an **MLP** to obtain final predictions.

Model - Readout Layer

$$\vec{s}^{(l)} = \frac{1}{N^{(l)}} \sum_{i=1}^{N^{(l)}} \vec{x}_i^{(l)} \,\|\, \max_{i=1}^{N} \vec{x}_i^{(l)} \qquad \vec{s} = \sum_{l=1}^{L} \vec{s}^{(l)}$$

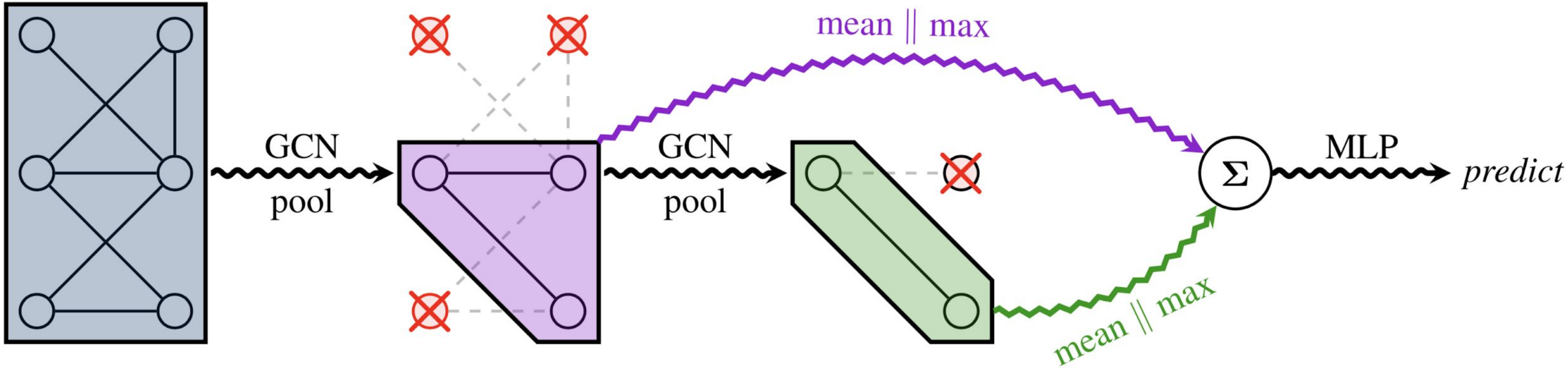
$$\text{predictions} = \mathbf{MLP}(\vec{s})$$

where ‖ denotes the concatenation, $N^{(l)}$e number of nodes of the graph and a$\vec{x}_i^{(l)}$he *i*-th node’s feature vector.



Model - Pipeline

The presented graph neural network architecture comprises **three conv-pool blocks** (in the image they are only two).



3. Experiments

Datasets and Parameters

The graph neural network architecture of the paper has been evaluated on four benchmark tasks, with the following parameters:

Enzymes	128 feature layers, 0.0005 as learning rate, 100 epochs
Proteins	64 feature layers, 0.005 as learning rate, 40 epochs
D&D	64 feature layers, 0.0005 as learning rate, 20 epochs
Collab	128 feature layers, 0.0005 as learning rate, 30 epochs

For all the datasets, they set the pooling ratio k = 0.8 and used Adam optimizer.

For the evaluation of my implementation, I used only *Enzymes* and *Proteins* datasets.

My implementation

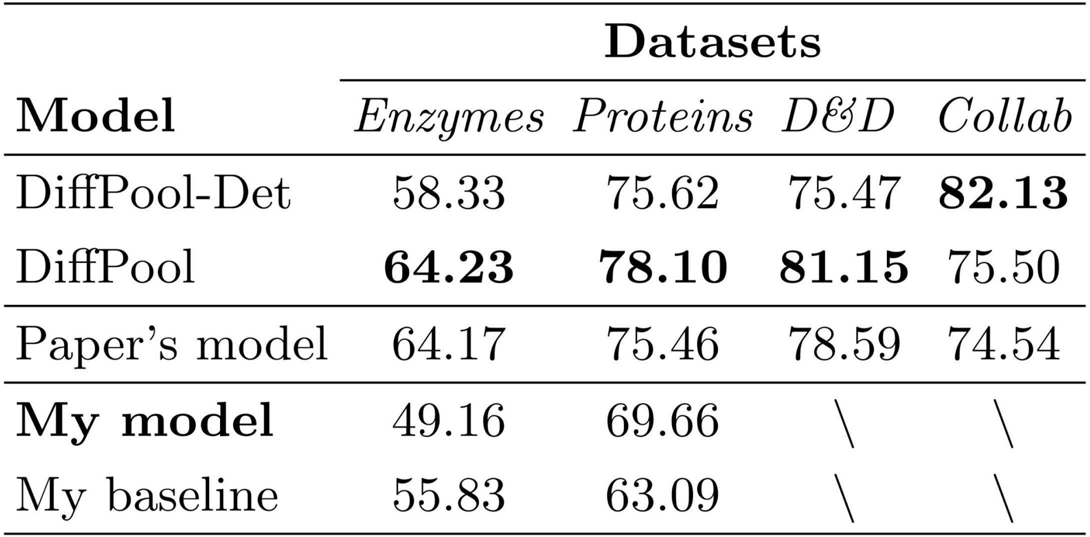
I replicated the same architecture described in the paper using **Tensorflow 2.0** on Google Colab and implementing all the layers from scratch.

In order to have a **baseline** for my hierarchical model, I also implemented a much simpler GCN model, which consists only of three convolutional layers, followed by a global max pooling layer just before the MLP for the classification.

I used the **same parameters** of the paper both for the hierarchical architecture and for the GCN model.

Results

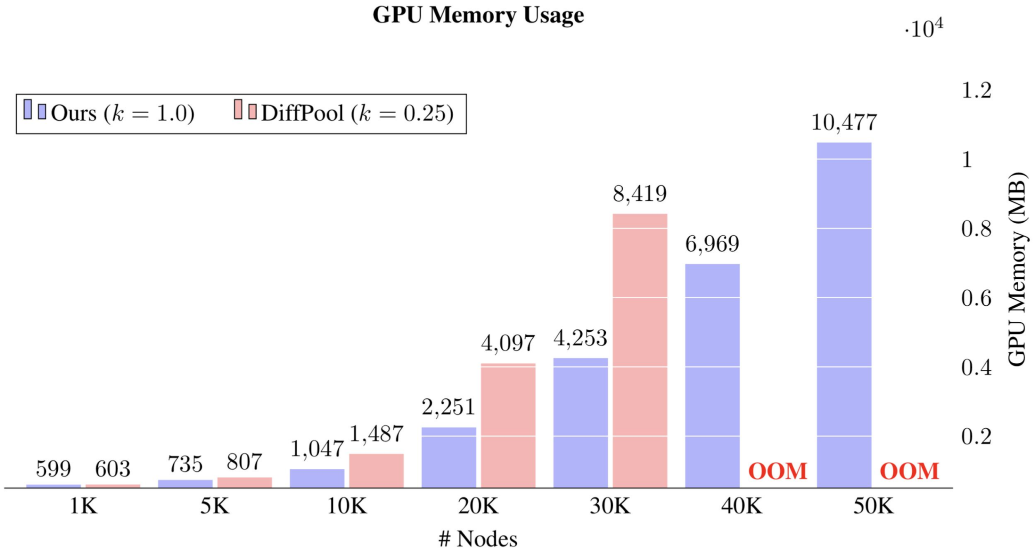
The following table shows the classification accuracy percentages of the models:



	Datasets			
Model	*Enzymes*	*Proteins*	*D&D*	*Collab*
DiffPool-Det	58.33	75.62	75.47	**82.13**
DiffPool	**64.23**	**78.10**	**81.15**	75.50
Paper's model	64.17	75.46	78.59	74.54
My model	49.16	69.66	\	\
My baseline	55.83	63.09	\	\

Results

The following graph shows the GPU memory usage of the paper’s method and DiffPool during training:



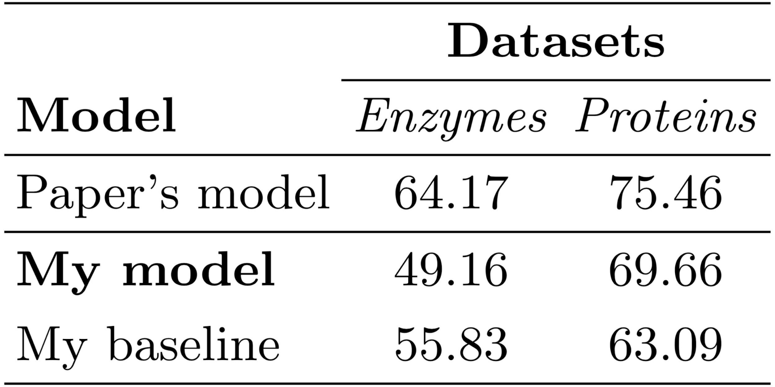
4.
Conclusions

Replicated model

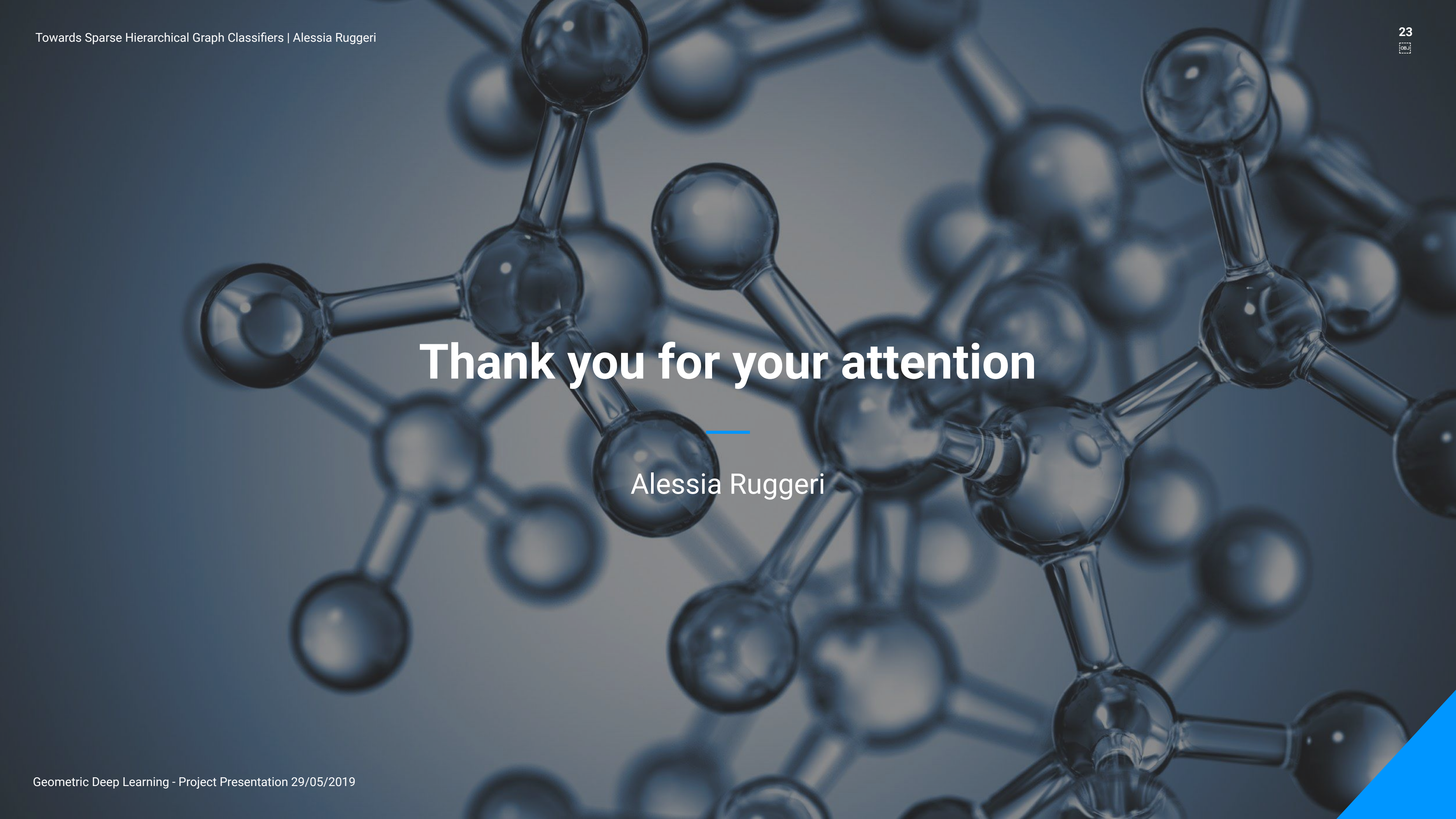
My implementation was **not able to reproduce the results** presented in the paper.

The obtained accuracy is very variable and hugely influenced by the parameters.

Probably the authors omitted crucial details in the paper.



	Datasets	
Model	*Enzymes*	*Proteins*
Paper's model	64.17	75.46
My model	49.16	69.66
My baseline	55.83	63.09



Thank you for your attention

Alessia Ruggeri