Paper Review “Towards Sparse Hierarchical Graph Classifiers”

Cangea et al. [1]

Suresh Kumar Choudhary / 20.12.2021

Course: Geometric Data Analysis
Course Instructor: Professor Christoph von Tycowicz

Table of Contents

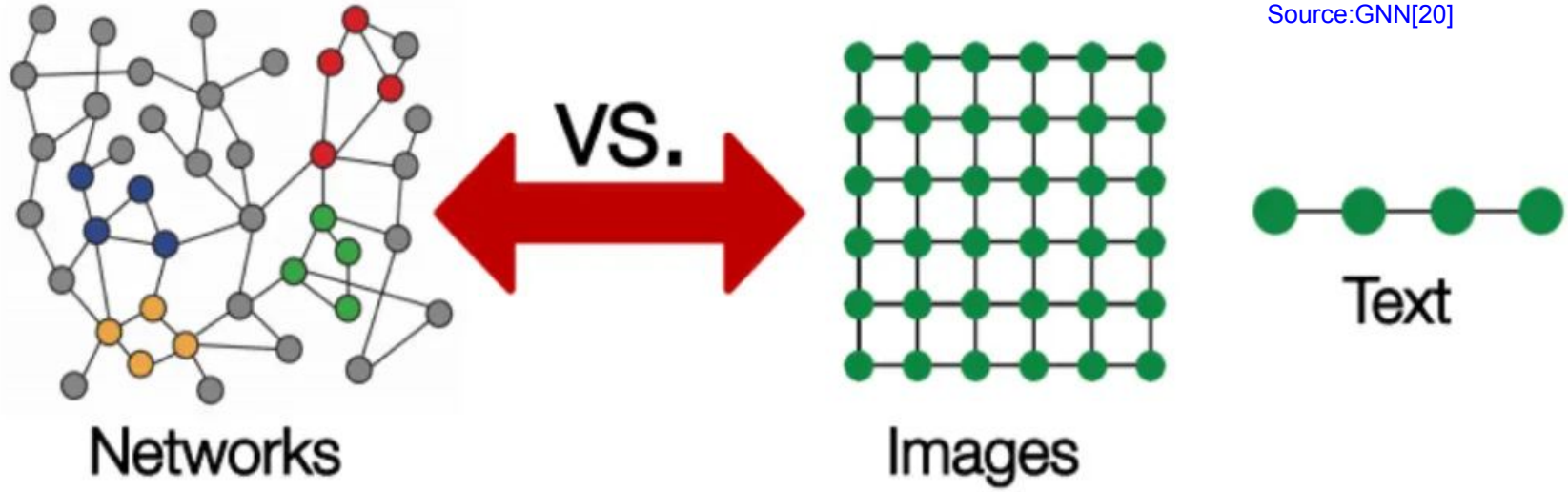
Goal of the study

- Sparse Hierarchical Graph Classification: Predicting a single label for the entire graph

Motivation

- The previous prominent approach to pooling has quadratic memory requirements during training, therefore not scalable to large graphs

Introduction: Graph vs Images



Source:GNN[20]

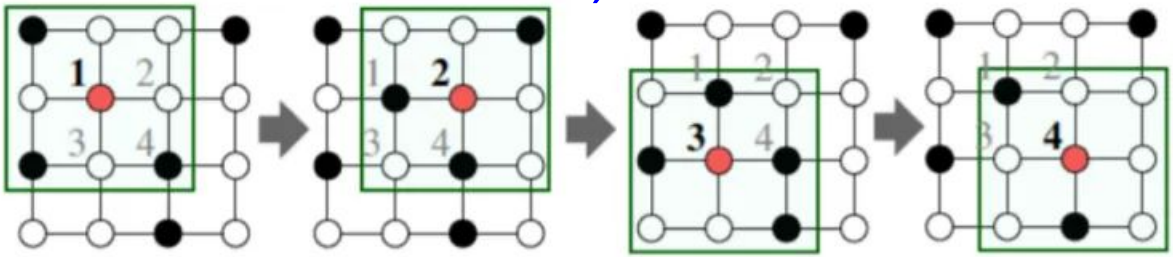
- Arbitrary in size
- Complex topology(no spatial locality)
- Non euclidean
- No fixed node ordering

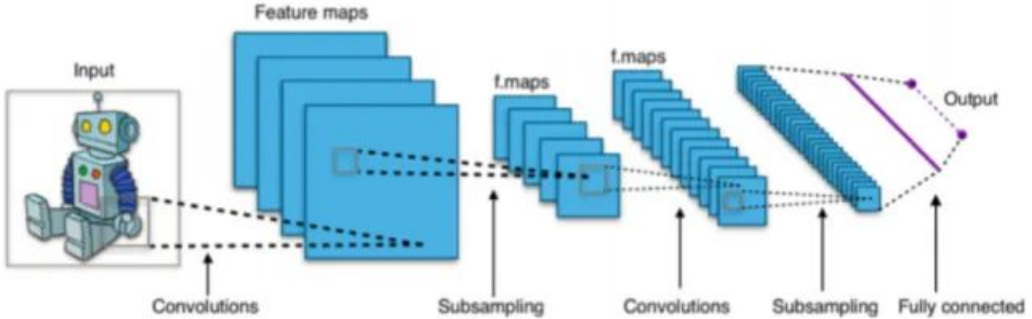
- Grid (Spatial locality)
- Structured data
- Deep learning designed for grid and sequences

Introduction: Why CNN fails on Graph?

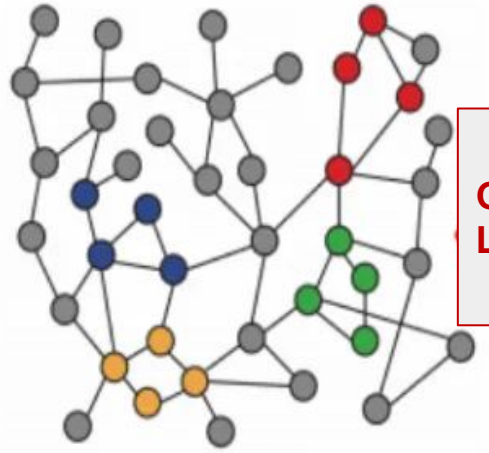
- Locality
- Aggregation
- Composition(function of a function)

Source:GNN[20]



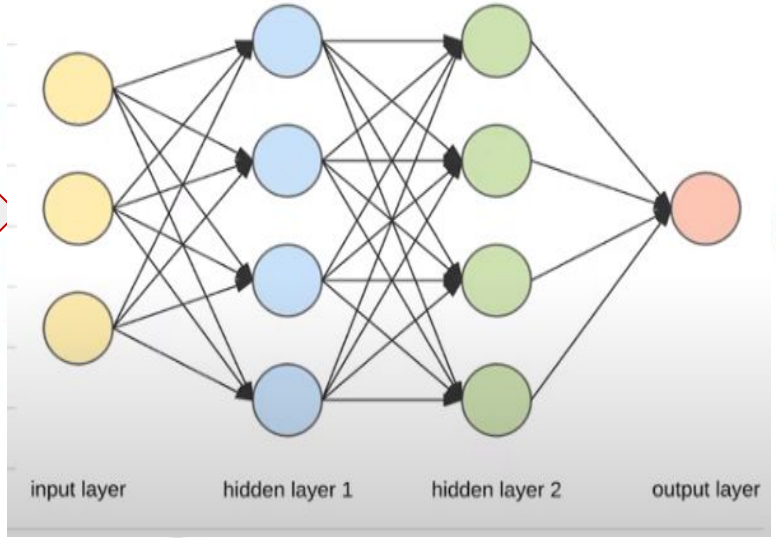


Introduction: Geometric deep learning



Networks

Geometric Deep Learning

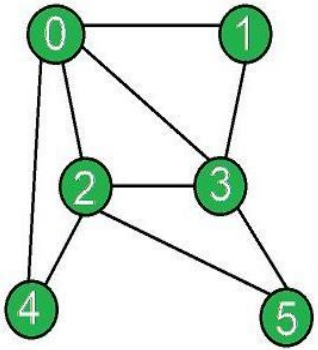


How Train Graphs using Neural Networks?

- ?

How Train Graphs using Neural Networks?

- Converting Graphs into Adjacency Matrix

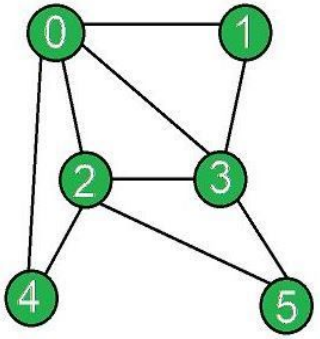


	0	1	2	3	4	5
0	0	1	1	1	1	0
1	1	0	0	1	0	0
2	1	0	0	1	1	1
3	1	1	1	0	0	1
4	1	0	1	0	0	0
5	0	0	1	1	0	0

How Train Graphs using Neural Networks?

- Converting Graphs into Adjacency Matrix

- Not invariant to
  Node ordering



	0	1	2	3	4	5
0	0	1	1	1	1	0
1	1	0	0	1	0	0
2	1	0	0	1	1	1
3	1	1	1	0	0	1
4	1	0	1	0	0	0
5	0	0	1	1	0	0

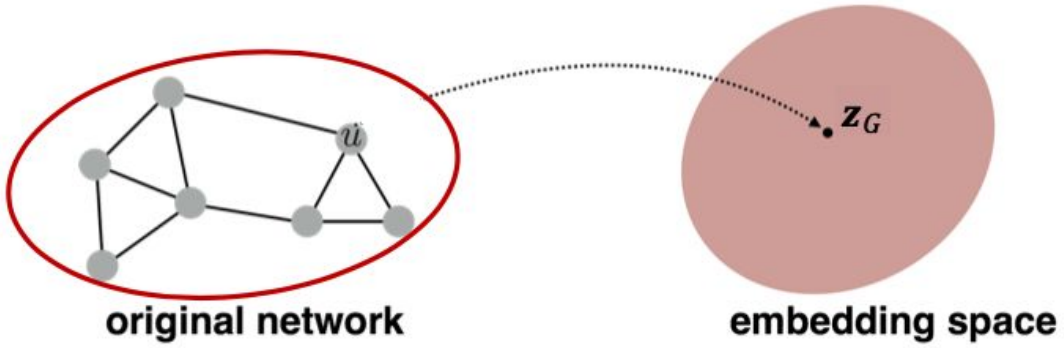
- Not applicable to different graph sizes

How Train Graphs using Neural Networks?

- 2nd approach: use same approach as in CNN
  - Locality
  - Aggregation
  - Stacking layers (composition)
- How?

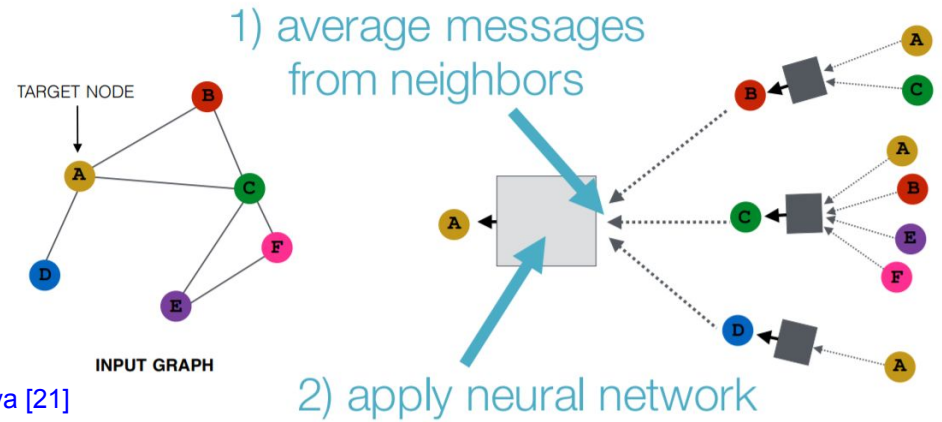
How Train Graphs using Neural Networks?

- 2nd approach: use same approach as in CNN
  - Locality
  - Aggregation
  - Stacking layers (composition)
- How: using node embedding or graph embedding



How Train Graphs using Neural Networks?

- 2nd approach: use same approach as in CNN
  - Locality
  - Aggregation
  - Stacking layers (composition)
- How: using node embedding or graph embedding



Source: AnalyticsVidya [21]

Related Work

Paper	Study Focus
Krizhevsky et al. [2]	ImageNet Classification with Deep Convolutional Neural Networks
[4,6,14, 16,17]	Graph convolutional layers
[12-15]	Aggregation of node representations: Global pooling layer
[3-11]	Clusters which coarsen the graph in a hierarchical manner through a fixed and pre-defined cluster assignment
Ying et al. [3]	Soft clustering assignments, learned in a differentiable way (DiffPool)

Model

- Input graphs
- Layers
- Architecture

Model: Input Graphs

- The input graph is represented by:
  - Matrix of node features:

$$X \in R^{N \times F}$$

  - Adjacency matrix:

$$A \in R^{N \times N}$$

  - 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

- Convolutional layer
- Pooling layer
- Readout layer

Model: Convolutional Layer

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

$$MP(X, A) = \sigma(\hat{D}^{-1}\hat{A}X\Theta + X\Theta')$$

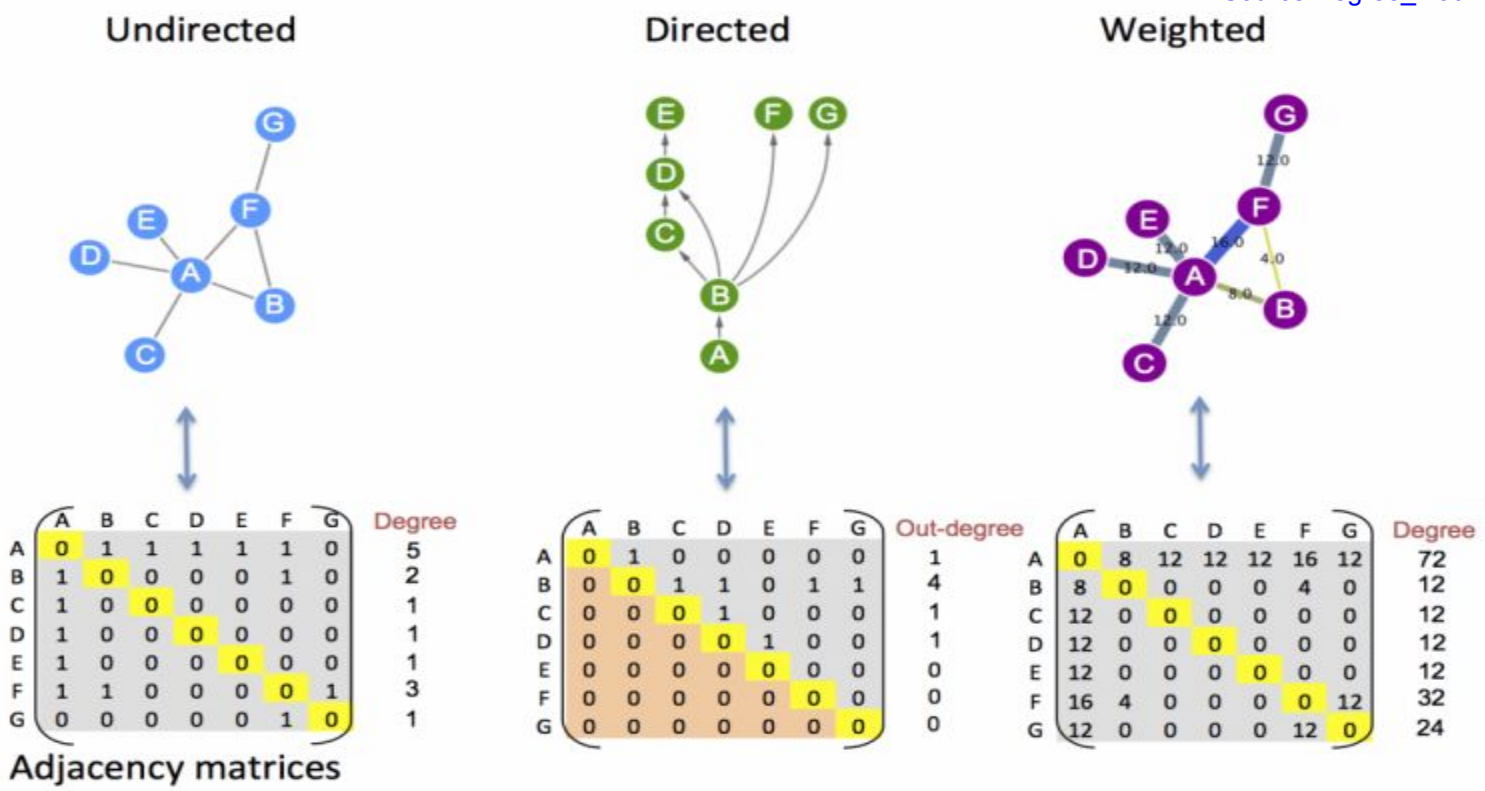
Skip - Connection

Where:

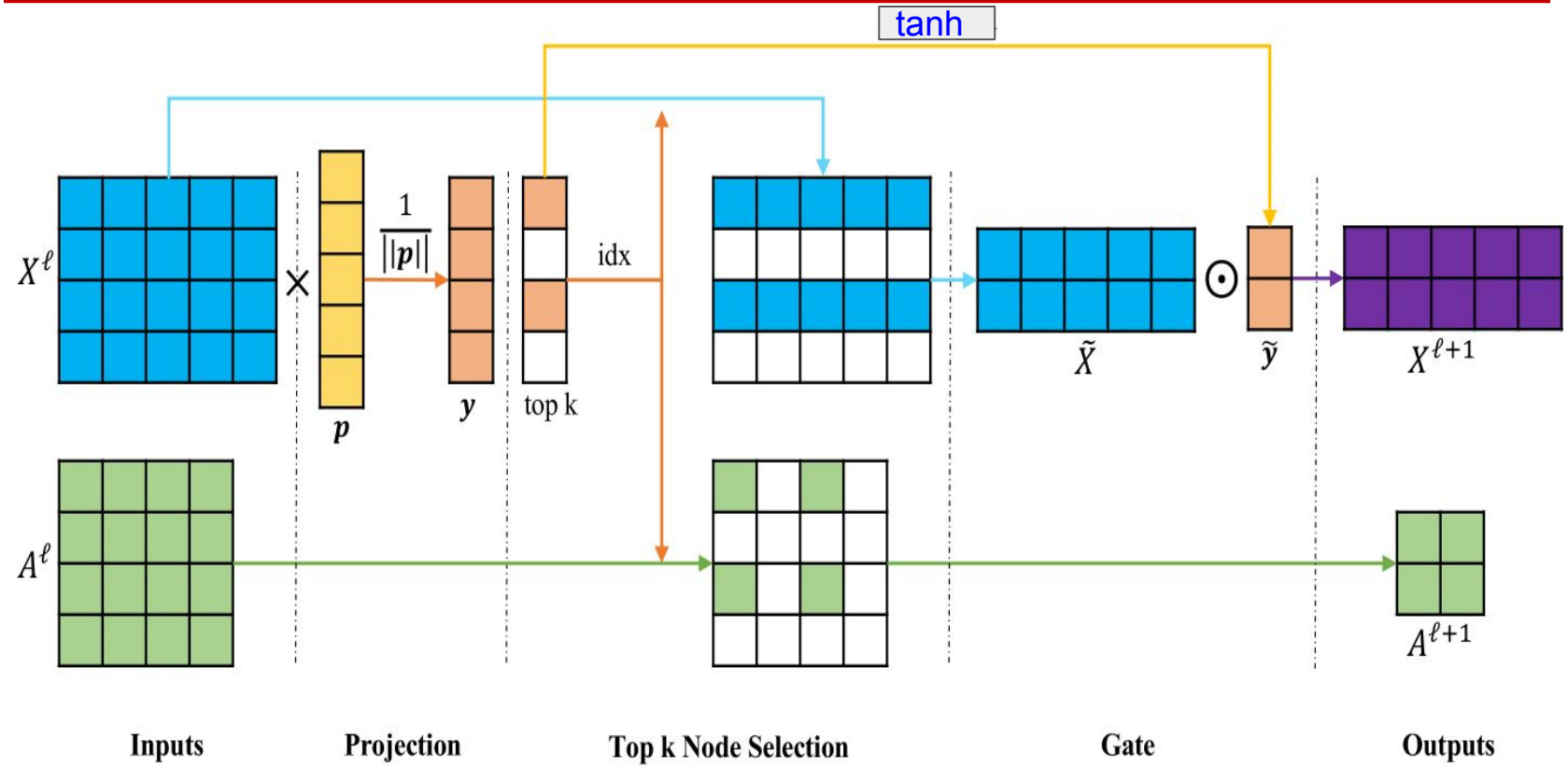
  - σ is rectified linear (ReLU) activation function
  - $\hat{A} = A + I_N$ Adjacency matrix with inserted self-loops
  - $\hat{D}$ is its corresponding degree matrix $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$
  - $\Theta, \Theta' \in R^{F \times F'}$ are learnable linear transformations

Model: Convolution Layers

Source:Degree_matrix [22]



Model: Pooling Layer



Model: Pooling Layer

- computing a pooled graph, (X' , A'), from an input graph (X, A)) can be expressed as follows:

$$\vec{y} = \frac{X\vec{p}}{||\vec{p}||} \quad \vec{i} = top - k(\vec{y}, k) \quad X' = (X \odot tanh(\vec{y}))_{\vec{i}} \quad A' = A_{\vec{i},\vec{i}}$$

- Here, ||.|| is the L2 norm, top-k selects the top-k indices from a given input vector,
- broadcasted element wise multiplication
- Vector i is an indexing operation which takes slices at indices specified by vector i.

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 , which is also used as gating value.

Model: Readout Layer

- The readout layer flatten the information about the input graph in a fixed-size representation.
- Concatenate the results of a global average pooling and a global max pooling at the end of each conv-pool block.
- Apply a global sum pooling before submitting the information to an MLP to obtain final predictions.

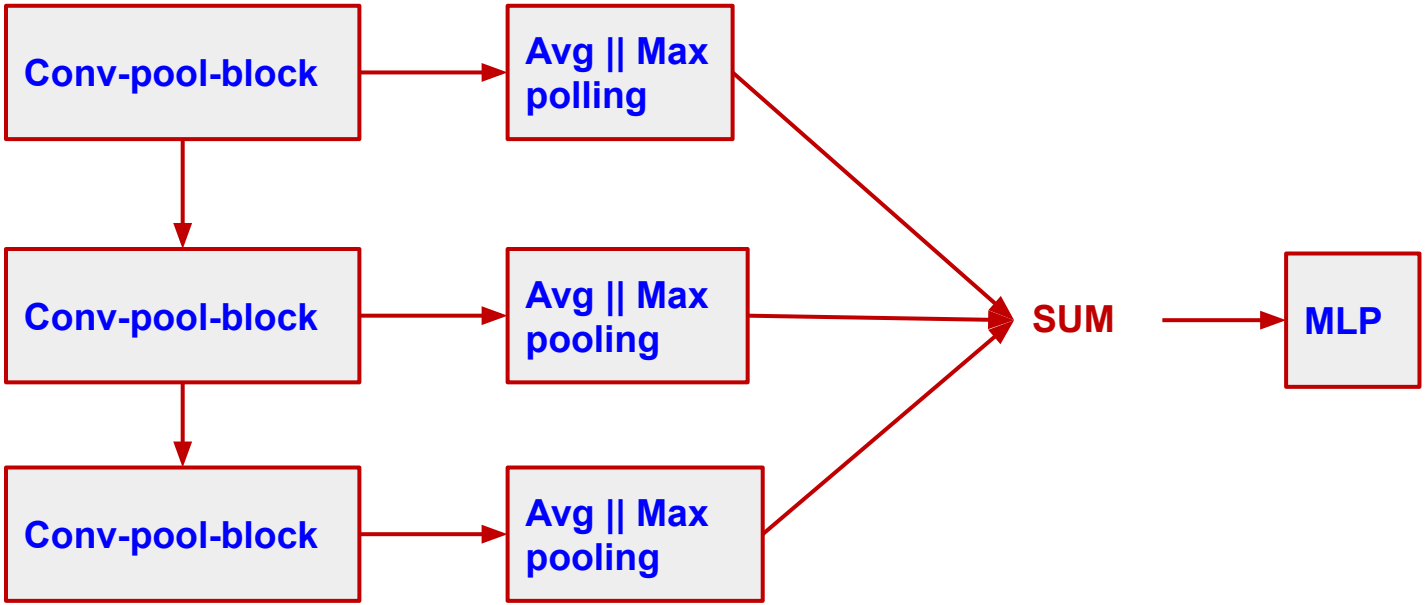
Model: Readout Layer

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

Predictions = MLP ($\vec{s}$)

- where N is the number of nodes of the graph at layer l
- Vector x_i are the i-th node's feature vector.
- || denotes concatenation

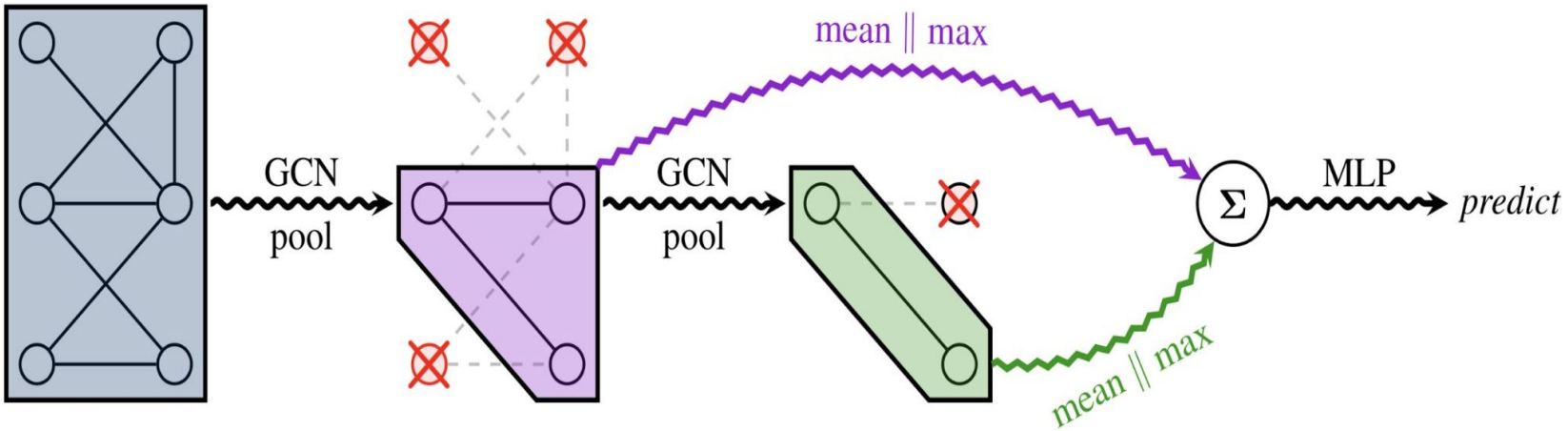
Model: Readout Layer



Model Architecture:

- The presented architecture comprises three conv-pool blocks (in the image they are only two)

Source: Cangea et al. [1]



Experiments: Datasets & Model Parameters

Type	Dataset	Size	No. of Classes	No. of Feature Layers	Learning Rate	Epochs
Biological	Enzymes	597	6	128	0.0005	100
	Proteins	1113	2	64	0.005	40
	D & D	1178	2	64	0.005	20
Scientific collaboration	Collab	5000	3	128	0.005	30

- Pooling ratio k = 0.8, and used Adam optimizer
- 10-fold cross validation approach is used for evaluation

My implementation:

- Used Pytorch and colab
- Implemented all the layers from scratch
- Replicated the same architecture
- Used the same parameters described in paper
- Tested on enzymes dataset only
- Code can be found at Github [23]

Result: Classification Accuracy in %

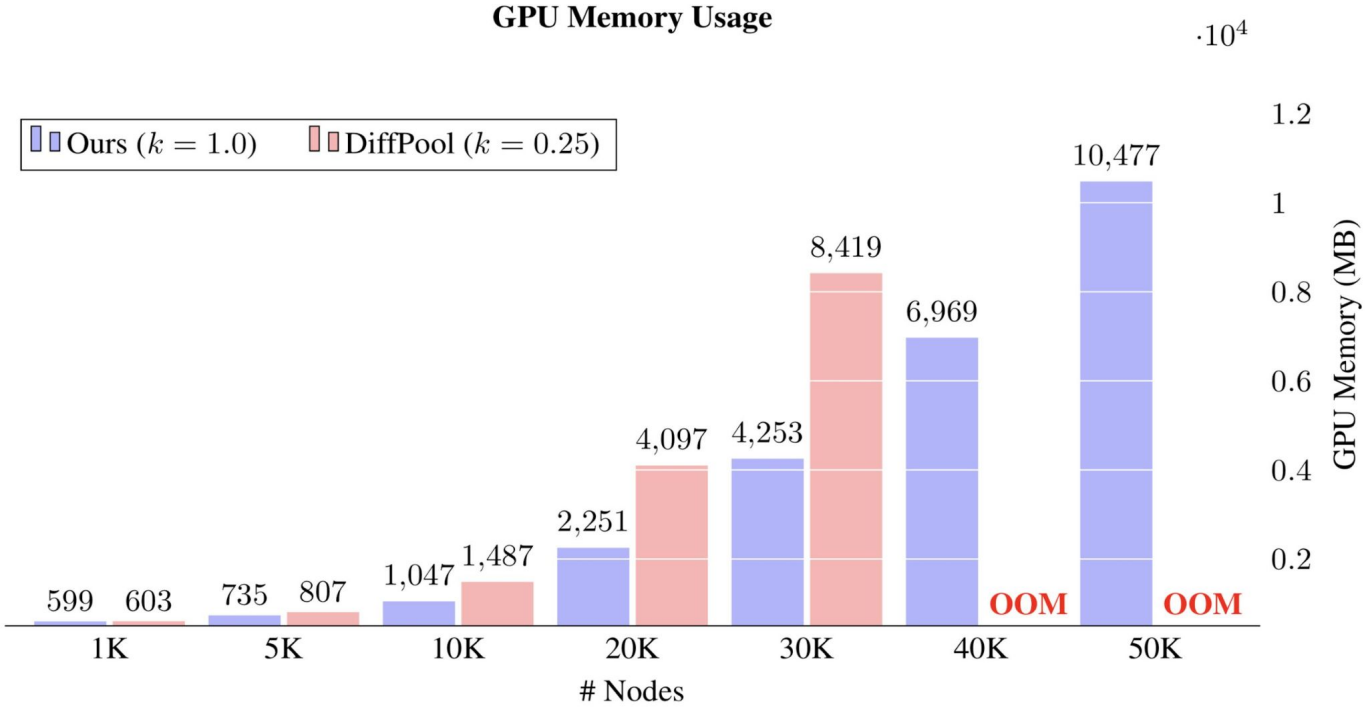
Source:Cangea et al.[1]

	Model	Datasets			
		Enzymes	*D&D*	*Collab*	*Proteins*
Kernel based	Graphlet	41.03	74.85	64.66	72.91
	Shortest-path	42.32	78.86	59.10	76.43
	1-WL	53.43	74.02	78.61	73.76
	WL-QA	60.13	79.04	80.74	75.26
GNN	PatchySAN	–	76.27	72.60	75.00
	GraphSAGE	54.25	75.42	68.25	70.48
	ECC	53.50	74.10	67.79	72.65
	Set2Set	60.15	78.12	71.75	74.29
	SortPool	57.12	79.37	73.76	75.54
	DiffPool-Det	58.33	75.47	**82.13**	75.62
	DiffPool-NoLP	62.67	79.98	75.63	77.42
	DiffPool	**64.23**	**81.15**	75.50	**78.10**
	Ours	64.17	78.59	74.54	75.46

Result: GPU Memory Usage

- Paper's method and DiffPool during training

Source:Cangea et al.[1]



Conclusion and Discussion

- Accuracy at most with in range of 1% as state of art algorithm (diffpool)
- State of art (diffpool) requires a quadratic $O(kV^2)$ storage complexity to be executed.
- Proposed approach requires only $O(V + E)$ storage complexity.

Conclusion and Discussion

- Accuracy at most with in range of 1% as state of art algorithm (diffpool)
- State of art (diffpool) requires a quadratic $O(kV^2)$ storage complexity to be executed.
- Proposed approach requires only $O(V + E)$ storage complexity.
- Hierarchical Graph Pooling with Structure Learning?
- Bridging the Gap Between Spectral and Spatial Domains in Graph Neural Networks?

References

1. Towards sparse hierarchical graph classifiers

   C Cangea, P Veličković, N Jovanović, T Kipf, P Liò - arXiv preprint arXiv:1811.01287, 2018.

2. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.
3. Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, Jure Leskovec, Joseph M Antognini, Jascha Sohl-Dickstein, Nima Roohi, Ramneet Kaur, et al. Hierarchical graph representation learning with differentiable pooling. CoRR, 2018

References (continued)

4. Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203, 2013

5. Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In International conference on machine learning, pages 2014–2023, 2016

6. Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In Advances in Neural Information Processing Systems, pages 3844–3852, 2016

References (continued)

8. Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 869–877, 2018

9. Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In Proc. CVPR, volume 1, page 3, 2017.

References (continued)

10. Martin Simonovsky and Nikos Komodakis. Dynamic edgeconditioned filters in convolutional neural networks on graphs. In Proc. CVPR, 2017

11. Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li Fei-Fei, Joshua B Tenenbaum, and Daniel LK Yamins. Flexible neural representation for physics prediction. arXiv preprint arXiv:1806.08047, 2018.

12. David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In Advances in neural information processing systems, pages 2224– 2232, 2015.

References (continued)

13. Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In International Conference on Machine Learning, pages 2702–2711, 2016.

14. Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. arXiv preprint arXiv:1704.01212, 2017.

15. Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li Fei-Fei, Joshua B Tenenbaum,and Daniel LK Yamins. Flexible neural representation for physics prediction. arXiv preprint arXiv:1806.08047, 2018

References (continued)

16. Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907, 2016.

17. Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. International Conference on Learning Representations, 2018.

18. H. Gao, S. Ji Graph u-nets Proceedings of ICML (2019), pp. 2083-2092

19.Datasetshttps://ls11-www.cs.tu-dortmund.de/staff/morris/graphkerneldatasets

References (continued)

20.GNN https://neptune.ai/blog/graph-neural-network-and-some-of-gnn-applications

21. Node computation graph https://www.analyticsvidhya.com/blog/2021/09/getting-started-with-graph-neural-networks/

22. degree_matrix https://www.tekportal.net/adjacency/

23. Github Account https://github.com/sureshkuc/Freie-Universitat-Berlin/tree/main/GEOMETRIC-DATA-ANALYSIS

Questions?

Thank You