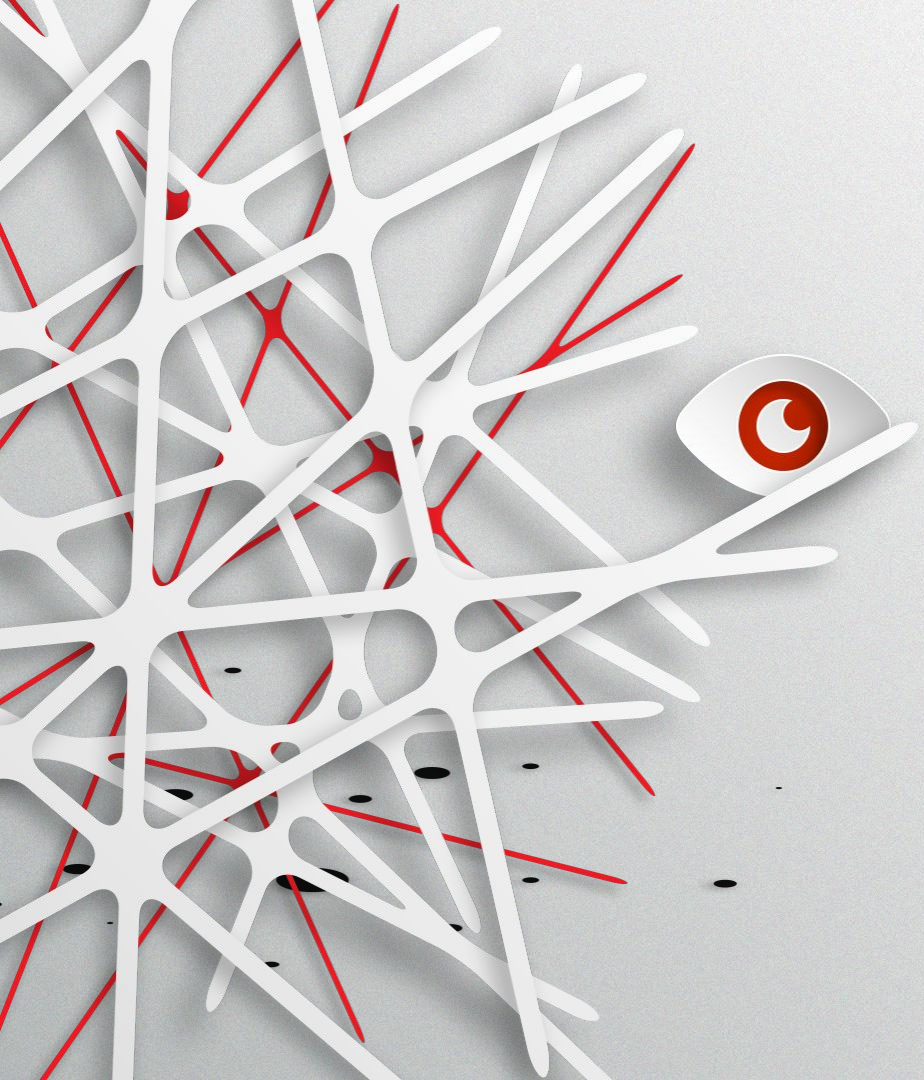


An End-to-End Deep Learning Architecture for Graph Classification

—— Muhan Zhang et al. Washington University in St. Louis

DGCNN Model

Jie Tang, Menghong Han, Xiaofan Sun

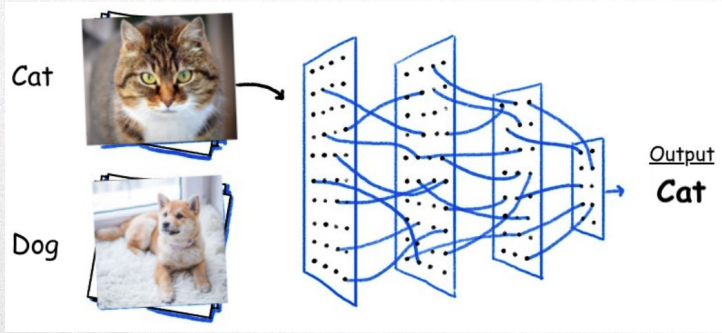
Outline

- Background & Introduction
- Challenges
- Problem Definition
- DGCNN Model Explanation
- Experiments
- Comments

Background & Introduction

Growing prevalence of Neural Networks on application domains:

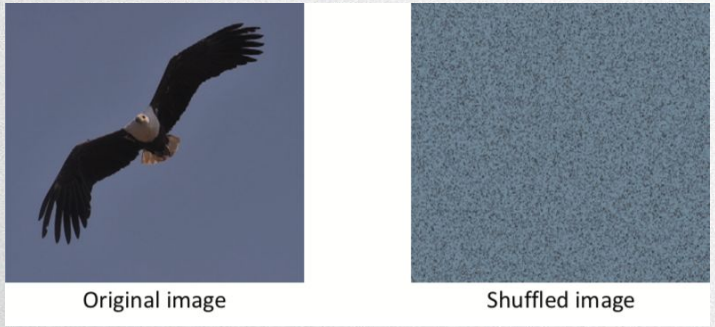
- ➔ Image classification (Alex, Sutskever, and Hinton 2012)
- ➔ Natural language processing (Mikolov et al. 2013)
- ➔ Reinforcement learning (Mnih et al. 2013)
- ➔ Time series analysis (Cui, Chen, and Chen 2016)



Cornerstone of existing methods:

Using fixed input order to extract higher-level features

example

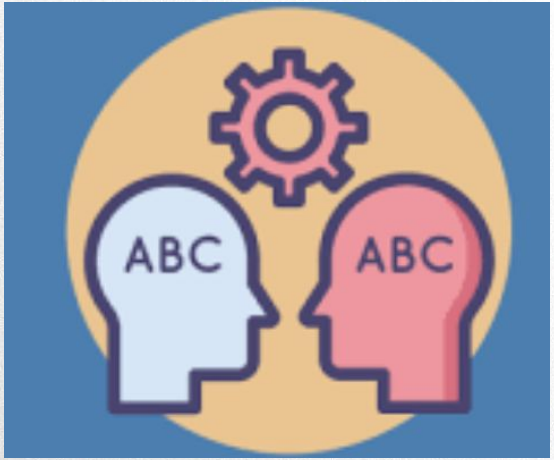


Challenges & Problem Definition

Problems of past Neural Network applications:

1) The lack of ordered tensor representations limits the applicability of neural networks on graph

2) After extracting localized vertex features, these features are directly summed up as a graph-level feature→ performance on graph data is not satisfactory



Challenges: Given a dataset containing graphs (G, y)

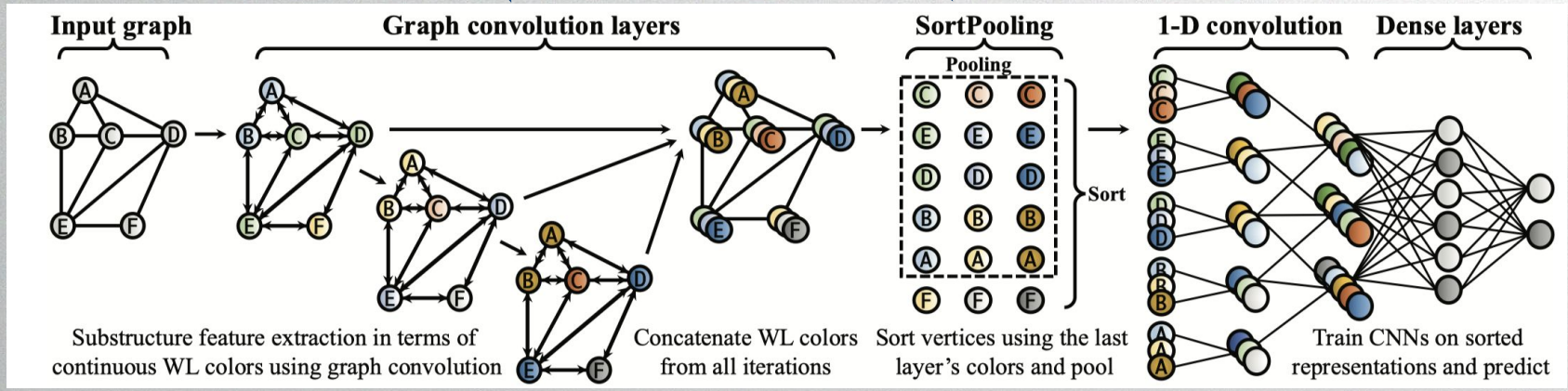
1) How to extract useful features characterizing the rich information encoded in a graph for classification purpose → **Design a localized graph convolution model**

2) How to sequentially read a graph in a meaningful and consistent order → **Design a SortPooling layer which sorts graph vertices in a consistent order so that traditional neural networks can be trained on the graphs**

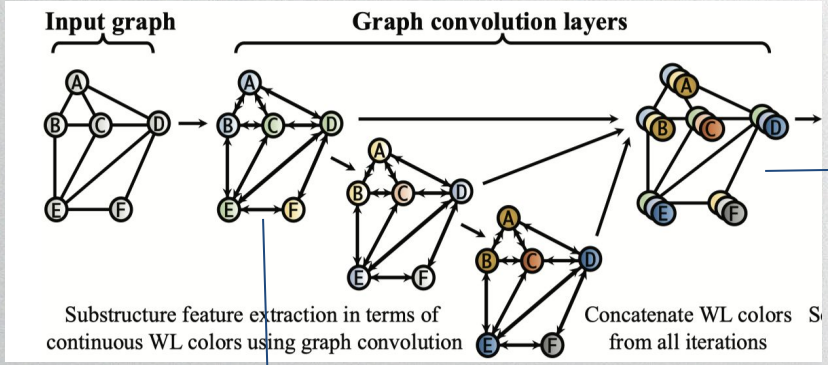
DGCNN Model Basic Idea

Creating three stages:

- ➔ **Graph convolution layers** extract vertices' local substructure features and define a consistent vertex ordering
- ➔ **A SortPooling layer** sorts the vertex features under the previously defined order and unifies input sizes
- ➔ **Traditional convolutional and dense layers** read the sorted graph representations and make predictions



DGCNN Model -- Graph Convolution Layers



Output:

$$\mathbf{Z}^{1:h} := [\mathbf{Z}^1, \ldots, \mathbf{Z}^h]$$

Each row can be regarded as a "feature descriptor" of a vertex

Multi-scale Substructure Features

Local Substructure Information

Proposed form Given a graph $\mathbf{A}$ and its node information matrix $\mathbf{X} \in \mathbb{R}^{n \times c}$, our graph convolution layer takes the following form:

$$\mathbf{Z} = f(\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}), \quad (1)$$

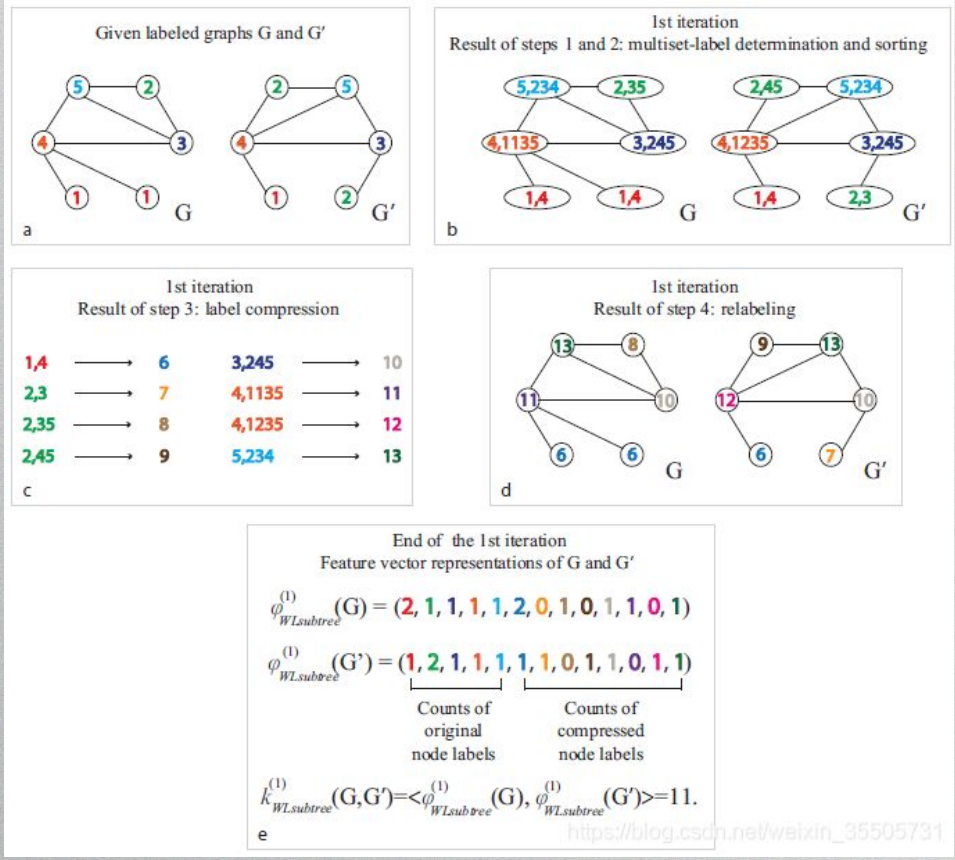
where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix of the graph with added self-loops, $\tilde{\mathbf{D}}$ is its diagonal degree matrix with $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$, $\mathbf{W} \in \mathbb{R}^{c \times c'}$ is a matrix of trainable graph convolution parameters, f is a nonlinear activation function, and $\mathbf{Z} \in \mathbb{R}^{n \times c'}$ is the output activation matrix.

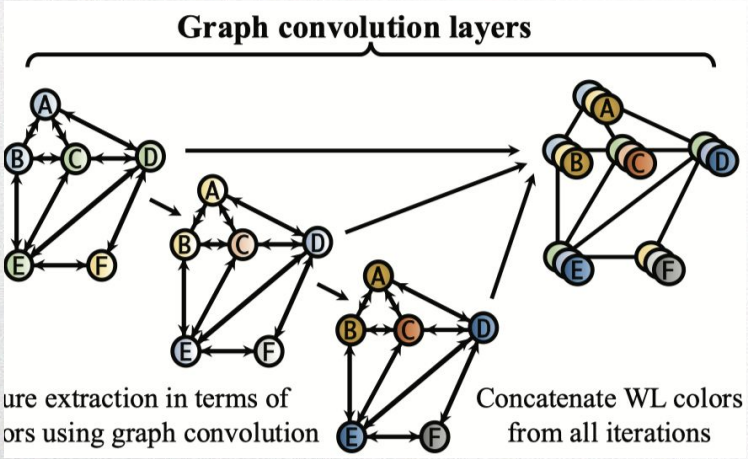
$$\mathbf{Z}^{t+1} = f(\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}\mathbf{Z}^t\mathbf{W}^t)$$

	Input	1st layer	2nd layer	3rd layer (Final)	Output
Shape	n*15	Z_1 (n*5)	Z_2 (n*3)	Z_3 (n*2)	n*(5+2+3)

DGCNN Model -- Graph Convolution Layers

Connection with Weisfeiler-Lehman Graph Kernel





$$\mathbf{Z}_i = f([\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}]_i\mathbf{Y}) = f(\tilde{\mathbf{D}}_{ii}^{-1}(\mathbf{Y}_i + \sum_{j\in\Gamma(i)}\mathbf{Y}_j)).$$

View Y_i as a continuous color of vertex i

“soft” version of the WL

DGCNN Model--The SortPooling Layer

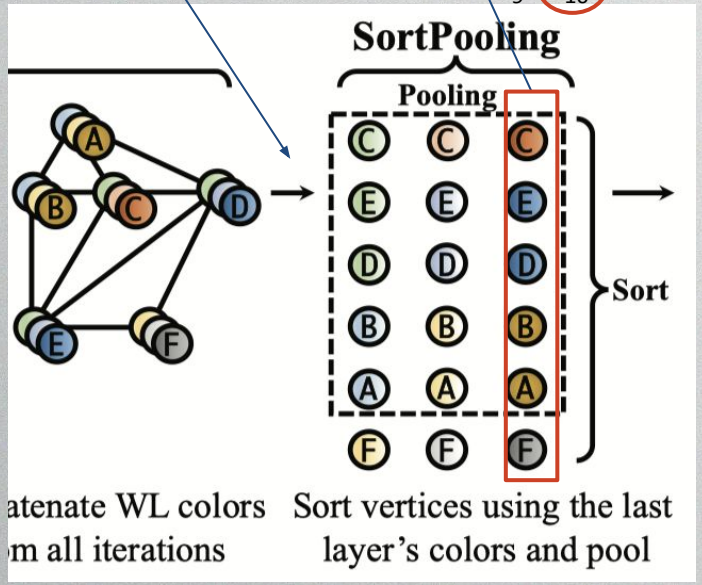
By what order should we sort the vertices?

Use graph's structural role, the continuous **WL colors Zt,** t = 1, . . . , h to sort vertices.

$\mathbf{Z}^{1:h} := [\mathbf{Z}^1, \ldots, \mathbf{Z}^h]$

$Z_h = Z_3$ (n*2)

f_9 f_{10}



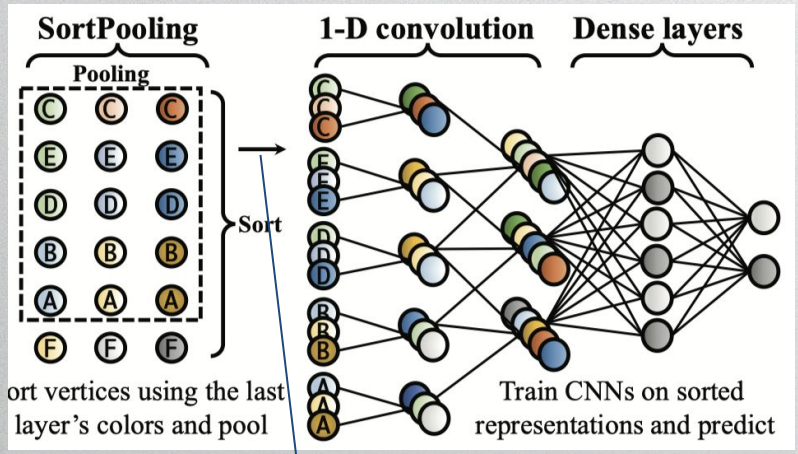
Order:
First sorting vertices using the last channel of Z_h in a descending order

Output: $\mathbf{Z}^{sp}$ of size $k \times \sum_1^h c_t$ (k*10)

k: user-defined integer so that m% of graphs have nodes more than this integer k

Unify: delete the last n − k rows if n > k, or adding k − n zero rows if n < k

DGCNN Model--Remaining layers



$k \times \sum_1^h c_t$ into a $k(\sum_1^h c_t) \times 1$ vector row-wise

(k*10) 1*10k Graph

- ➔ Reshape
- ➔ Add a 1-D convolutional layer
- ➔ Add a MaxPooling layers
- ➔ Add a 1-D convolutional layer
- ➔ Add a dense layer

Reimplementation Result

	MUTAG	PTC	NCI1	PROTEINS	D&D	COLLAB	IMDB-B	IMDB-M
Paper Result	**85.83**±1.66	58.59±2.47	**74.44**±0.47	**75.54**±0.94	79.37±0.94	**73.76**±0.49	**70.03**±0.86	47.83±0.85
Reimplementation	83.89±**0.08**	**58.82±0.08**	73.34±**0.02**	74.20±**0.48**	**81.20±0.53**	72.97±0.96	68.7±**0.03**	**48.2±0.37**

* Bioinformatics datasets: MUTAG, PTC, NCI1, PROTEINS, D&D; social network datasets: COLLAB, IMDB-B, IMDB-M.

Our Ideas

- ➔ Use 2-hop neighbors or BFS and DFS methods to extract vertices' local substructure features in graph convolution layers
- ➔ Deeper CNN or 2-D convolution
- ➔ SortingPooling based on last *m* channels giving higher weights to later channels
- ➔ Hyperparameter tuning on k

Paper's & Our Modeling Process

Paper's Model

Our Model

Our Model Result

	MUTAG	PTC	NCI1	PROTEINS	D&D	COLLAB	IMDB-B	IMDB-M
Paper Result	**85.83**±1.66	58.59±2.47	**74.44**±0.47	**75.54**±0.94	79.37±0.94	**73.76±0.49**	**70.03**±0.86	**47.83**±0.85
Our Model	82.49±**0.07**	**58.90±0.04**	73.34±**0.02**	74.20±**0.48**	**80.70±0.43**	72.97±0.96	67.65±**0.03**	45.2±**0.47**
Best K	0.6	**0.55**	0.6	0.6	**0.65**	0.9	0.9	0.9

* Bioinformatics datasets: MUTAG, PTC, NCI1, PROTEINS, D&D; social network datasets: COLLAB, IMDB-B, IMDB-M.

Comments

➔ **Advantages**

- Standard deviations of experiment results are included
- Clear explanation on math equations
- Accepts graphs as input without the need of any preprocessing

➔ **Disadvantages**

- A visualization on experiment results could be more intuitive
- No detailed codes of model selection
- Running time is not included
- Unclear explanations on connections with WL kernel

Thank you!