Graph Neural Networks

John Shi and Shreyas Chaudhari

Intro to Deep Learning - Lecture

April 21, 2025

Our Research Team

Undergraduate Interns:
Yao (Lavender) Jiang
Xujin (Chris) Liu
Wendy Summer
Austin Lin
Farida Abdelmoneum

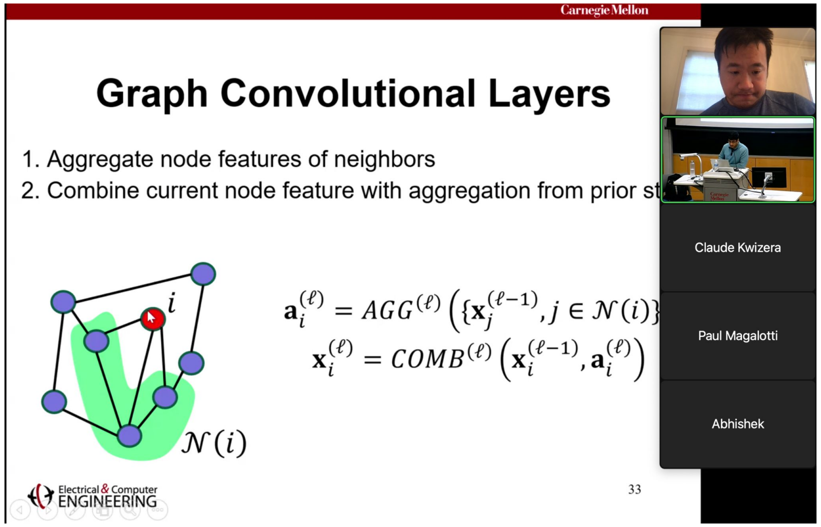
Masters/Ph.D. Students:
Mark Cheung
Oren Wright (SEI)
Mayur Gowda

Postdoc:
Jian Du

Advisor:
José Moura (ECE)

Graph Signal Processing and Deep Learning: Convolution, Pooling, and Topology:
https://arxiv.org/abs/2008.01247

Our Lecture Last Year



Graph Convolutional Layers

1. Aggregate node features of neighbors
2. Combine current node feature with aggregation from prior st

$$\mathbf{a}_i^{(\ell)} = AGG^{(\ell)}\left(\{\mathbf{x}_j^{(\ell-1)}, j \in \mathcal{N}(i)\}\right)$$
$$\mathbf{x}_i^{(\ell)} = COMB^{(\ell)}\left(\mathbf{x}_i^{(\ell-1)}, \mathbf{a}_i^{(\ell)}\right)$$

Day after ICASSP 2024…

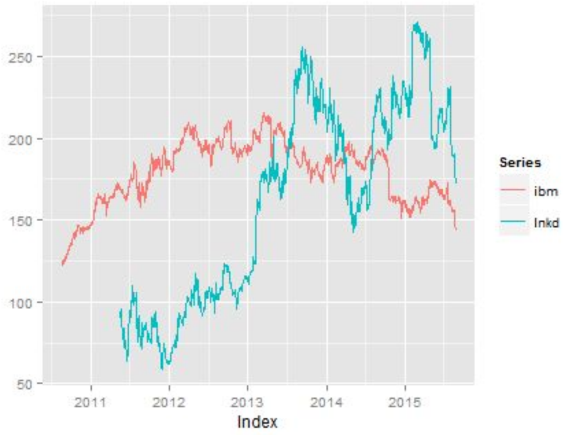
Got back from Korea Sunday night and got up Monday morning for lecture…

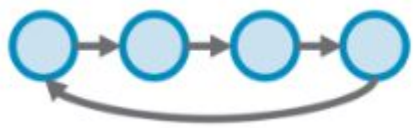
Luckily this year, we had a week to recover :P
(Though we were offered the day after ICASSP again…)

Grid-Structured Data

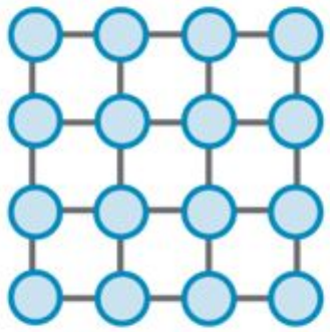
1-D and 2-D data

Time Series

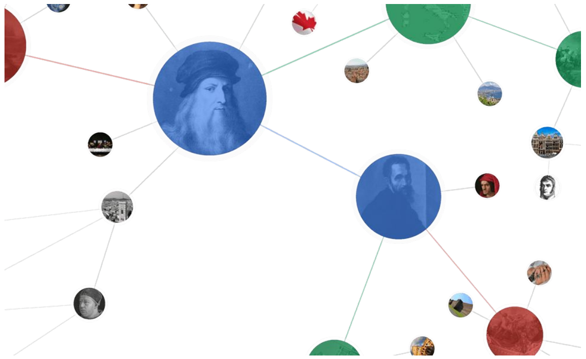


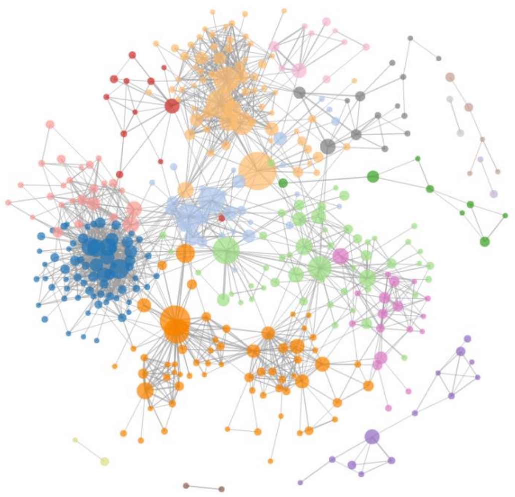


Images

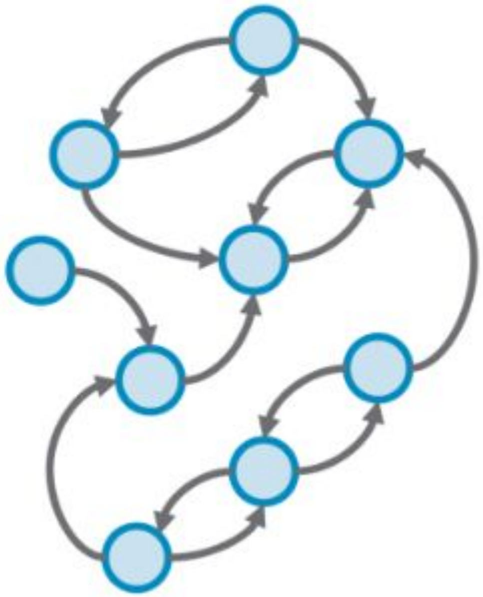


Data defined on an Irregular Graph





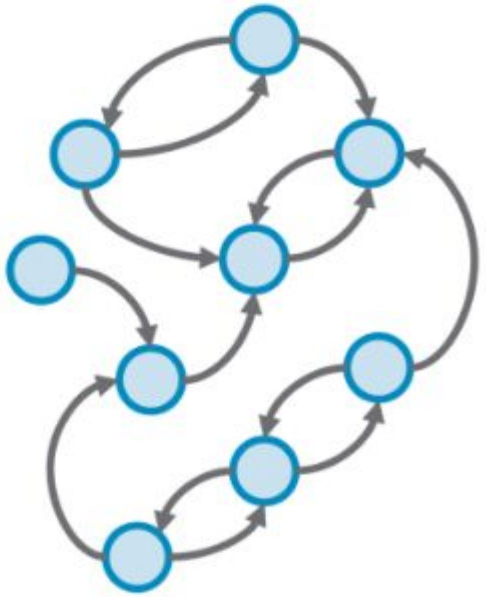
Social Networks
Sensor Feeds
Web Traffic
Supply Chains
Biological Systems
...



Challenges in Graph Data

Has a Fixed, Intuitive Ordering

Social Networks
Sensor Feeds
Web Traffic
Supply Chains
Biological Systems
...

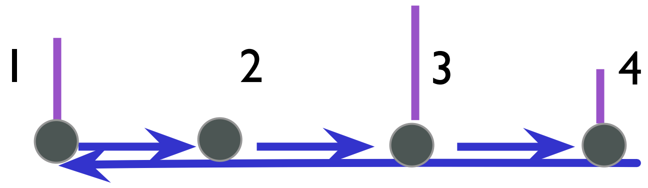


No Ordering (Which node is node 1 here?)

How do we incorporate the graph information?

Challenges in Graph Data

Discrete Signal Processing (DSP) works well on time and images

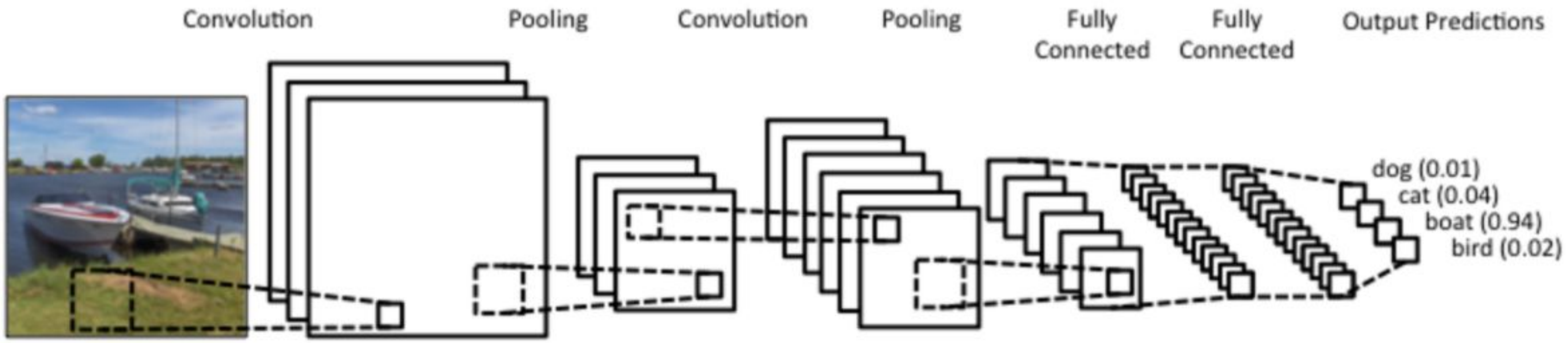


Time Series

But fail for graph data
- No fixed ordering
- No way to incorporate graph data.

Challenges in Graph Data

Classical CNNs perform well on Grid-Structured (Euclidean) Data

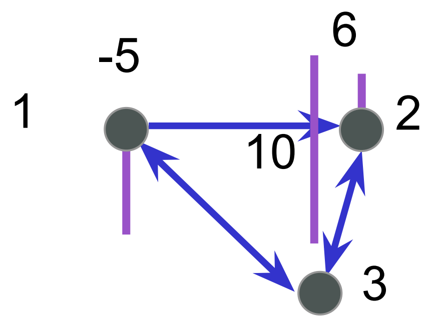


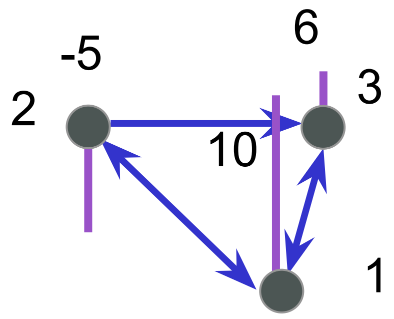
But fail for graph data
- No fixed ordering
- No way to incorporate graph data.

Challenges in Graph Data - Ordering

Ordering – Operations need to be ***permutation invariant***

In other words, we want the operation to produce the same (or permuted) numerical results *regardless of ordering*





s = [-5, 6, 10]

Sum: -5+6+10 = 11

First Difference: -5 – 6 – 10 = -21

ReLU: [0, 6, 10]

s = [10, -5, 6]

Sum: 10 - 5 + 6= 11

Permutation Invariant

First Difference: 10 – (–5) – 6 = 9

NOT Permutation Invariant

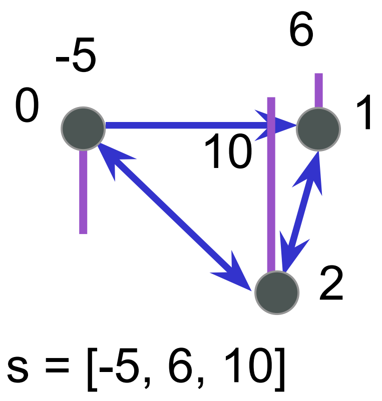
ReLU: [10, 0, 6]

Permutation Invariant

Electrical & Computer ENGINEERING

Challenges in Graph Data – Incorporating Graph Data

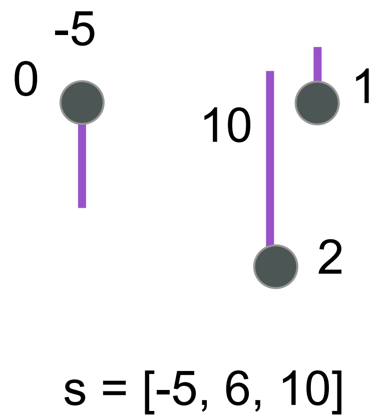
In the previous example, none of the operations actually used the graph!



Sum: -5+6+10 = 11

First Difference: -5 – 6 – 10 = -21

ReLU: [0, 6, 10]



Same Results

Main Question

How can we design deep learning techniques for graphs?

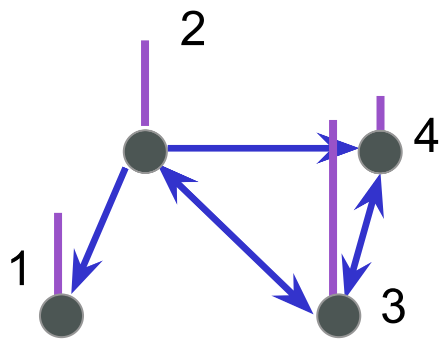
- Must incorporate graph information
- All operations must be permutation invariant

This field is called Geometric Deep Learning

Graph Signal Processing

Discrete Signal Processing works well for grid-structured data

Graph Signal Processing tries to do everything you can do in DSP, but for graph data.



$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

Adjacency Matrix Ordering is

Node I -> Node j means

$$A_{ji} = 1$$

In GSP!

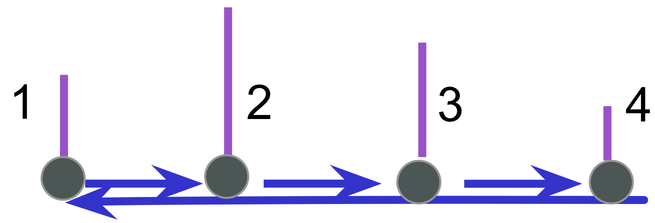
Different from Comp Sci

CMU Course: 18-898D: Special Topics in Signal Processing: Graph Signal Processing and Geometric Learning (offered most Fall semesters)

DSP to GSP – Structure, Data

DSP:

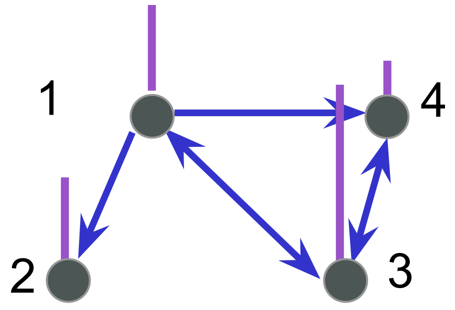
Assume the signal is periodic with period T



time: path + boundary condition

$$A_c = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

GSP:



general graph G (fixed & given)

$$\mathrm{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

Graph signal: $s : v_n \to x_n$

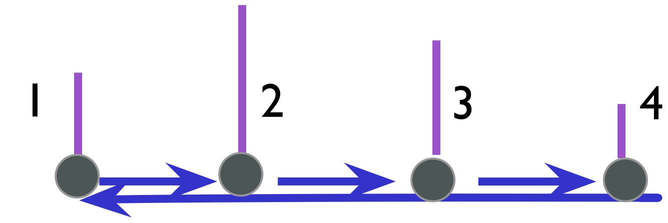
$$\mathcal{V} \to \mathbb{C}$$

- attributes on knowledge graph
- measurements on sensor network
- voltages on power grid
- …

Electrical & Computer ENGINEERING

DSP to GSP – Graph Shift

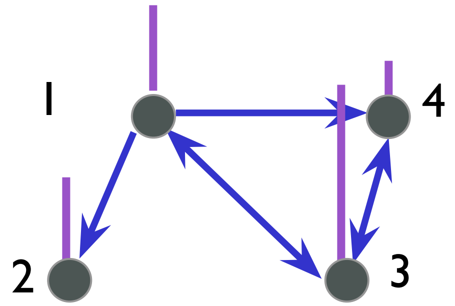
DSP:



time: line graph

$$A_c = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

GSP:



general graph G (fixed & given)

$$\mathrm{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

Graph Shift: As

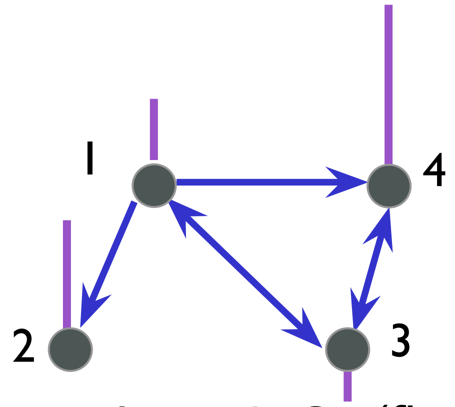
$$s = [2, 5, -1, 10]^T$$
$$As = [10, 2, 5, -1]^T$$

$$s = [2, 5, -1, 10]^T$$
$$As = [-1, 2, 2 + 10, 2 - 1]^T = [-1, 2, 12, 1]^T$$

Incorporates Graph Structure! What about Permutation Invariance?

Graph Shift – Permutation Invariance

GSP:



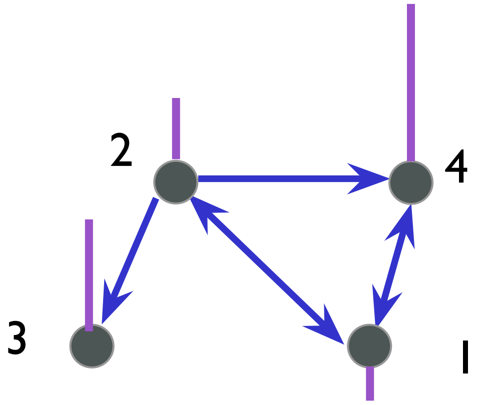
general graph G (fixed & given)

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

$$s = [2, 5, -1, 10]^T$$

$$As = [-1, 2, 2+10, 2-1]^T = [-1, 2, 12, 1]^T$$

Graph Shift: As



$$s_2 = [-1, 2, 5, 10]^T$$

$$A_2 = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

$$A_2 s_2 = [2+10, -1, 2, -1+2]^T$$
$$= [12, -1, 2, 1]^T$$

Permuted *As*!

Graph Shift – Permutation Invariance Proof

Permutation Matrix Π: Obtained by permuting the rows of the identity matrix

Each row has 1 one and N-1 zeros. Each row and column only has one 1

Every Permutation matrix is orthogonal: $\Pi^{-1} = \Pi^T$

$$\Pi = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad s = [2, 5, -1, 10]^T \qquad \Pi s = [-1, 2, 5, 10]^T = s_2$$

Graph Shift: As

$$A_2 = \Pi^T A_1 \Pi \qquad\qquad s_2 = \Pi s_1$$

$$A_2 s_2 = \Pi A_1 \Pi^T \Pi s_1 = \Pi A_1 s_1$$

GSP – Filtering and Convolution

- In GSP, a Linear, Shift-Invariant Graph Filter

(max degree: N-1, why?)

$$P(A) = \sum_{i=0}^{N-1} p_i A^i$$

- Convolution: Matrix-Vector product of a polynomial of the Adjacency Matrix and the signal s

$$P(A)s$$

- Graph Fourier Transform: Found using eigendecomposition of A

$$A = \mathrm{GFT}^{-1}\Lambda\mathrm{GFT}$$
$$\widehat{s} = \mathrm{GFT}s$$

GSP – Permutation Invariant Proofs

$$A_2 = \Pi A_1 \Pi^T \qquad s_2 = \Pi s_1$$

Convolution is Permutation Invariant

$$\begin{aligned} P(A_2)s_2 &= P(\Pi A_1 \Pi^T)\,(\Pi s_1) \\ &= \Pi P(A_1)\Pi^T \Pi(s_1) \\ &= \Pi\,(P(A_1)s_1) \end{aligned}$$

GFT is Permutation Invariant

$$A_1 = \mathrm{GFT}_1^{-1}\Lambda_1 \mathrm{GFT}_1$$

$$A_2 = \Pi A_1 \Pi^T = \Pi \mathrm{GFT}_1^{-1}\Lambda_1 \mathrm{GFT}_1 \Pi^T = \underbrace{\Pi \mathrm{GFT}_1^{-1}}_{\mathrm{GFT}_2^{-1}}\Lambda_1 \underbrace{\mathrm{GFT}_1 \Pi^T}_{\mathrm{GFT}_2}$$

$$\widehat{s}_2 = \mathrm{GFT}_2 s_2 = \mathrm{GFT}_1 \Pi^T \Pi s_1 = \mathrm{GFT}_1 s_1 = \widehat{s}_1$$

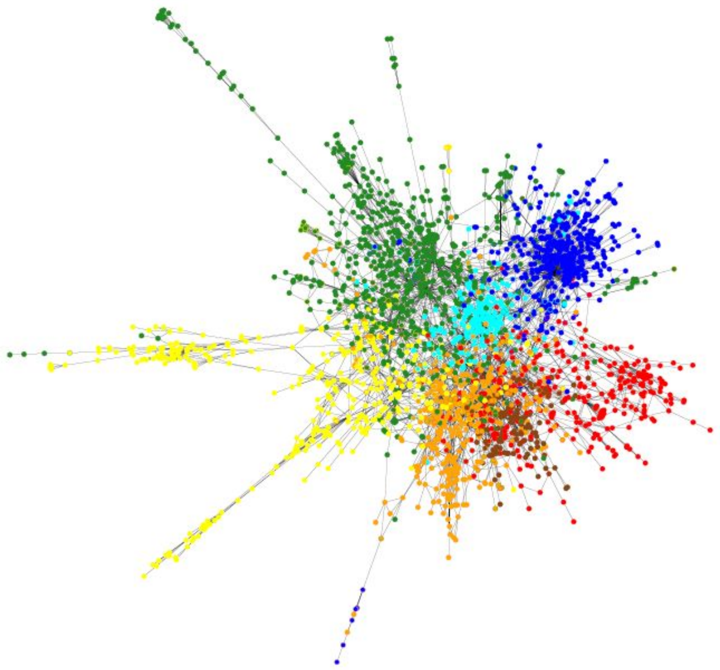
Graph Neural Networks Tasks

Types of Experiments

1. Node Classification
   - Classify each node in the graph
   - Graph shows connections between each data point
   - No pooling
   - GCNNs vs. MLPs
2. Graph Signal Classification
   - Classify the entire graph signal
   - Graph shows connections between the **features**
   - Graph can be different for each data point
   - Can do pooling

Node Classification: Citation Network

Classify the type of paper using the citation network (**graph**) and a bag of words (**data**) for each paper (**node**)



Color represents classes

Dataset	Nodes	Labelled Nodes	Label Rate	Edges	Connectivity	Features	Classes
Citeseer	3327	120	0.036	4732	4e-4	3703	6
Cora	2708	141	0.052	5429	7e-4	1433	7
Pubmed	**19717**	**59**	0.003	44338	**1e-4**	500	3

Node Classification

Why do we need a graph? Why not just use an MLP?

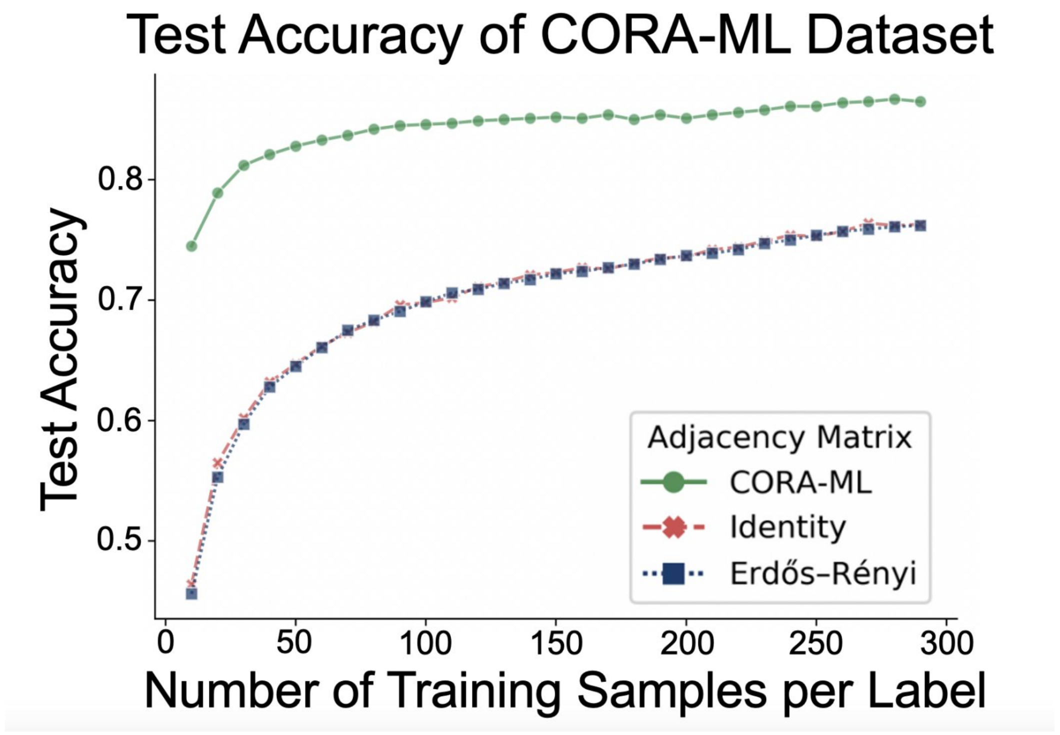
The graph allows for smaller training set.

Graph is like an “answer key” - Helps with classification.

Dataset	Nodes	Labelled Nodes	Label Rate	Edges	Connectivity	Features	Classes
Citeseer	3327	120	0.036	4732	4e-4	3703	6
Cora	2708	141	0.052	5429	7e-4	1433	7
Pubmed	**19717**	**59**	0.003	44338	**1e-4**	500	3

How good is my answer key?

- Graph provides “answer key” for classification problem, but how good is it?
- Graph greatly affects accuracy.



How good is my answer key?

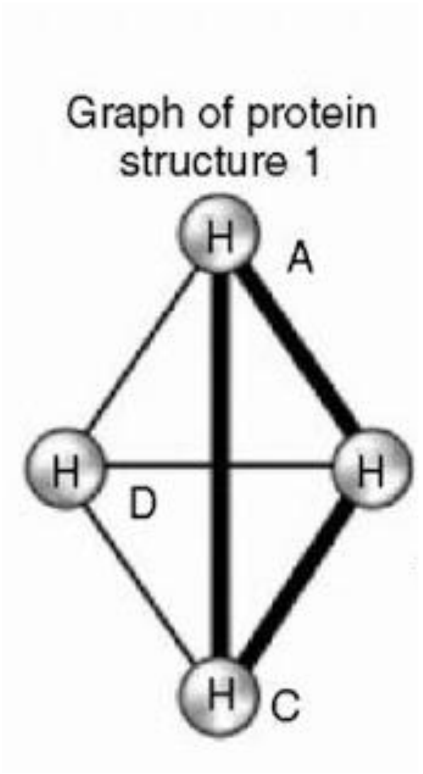
- How to form a graph:
  - Use some intuitive heuristic believed to be related to the classification
  - Use cross-correlation of the features.

- Many factors about the graph affect accuracy:
  - Homophilly/Heterophilly: Are nodes with the same classes connected? -> High homophilly
  - Features: Are nodes with similar features connected?
  - Clustering: Are nodes with the same class clustered together? Same Features?

Graph Classification: Biological Network

- Each protein, chemical and enzyme structure is a **different** graph
- Data is the elements of each node in the structure
- Predict properties based on the chemical structure

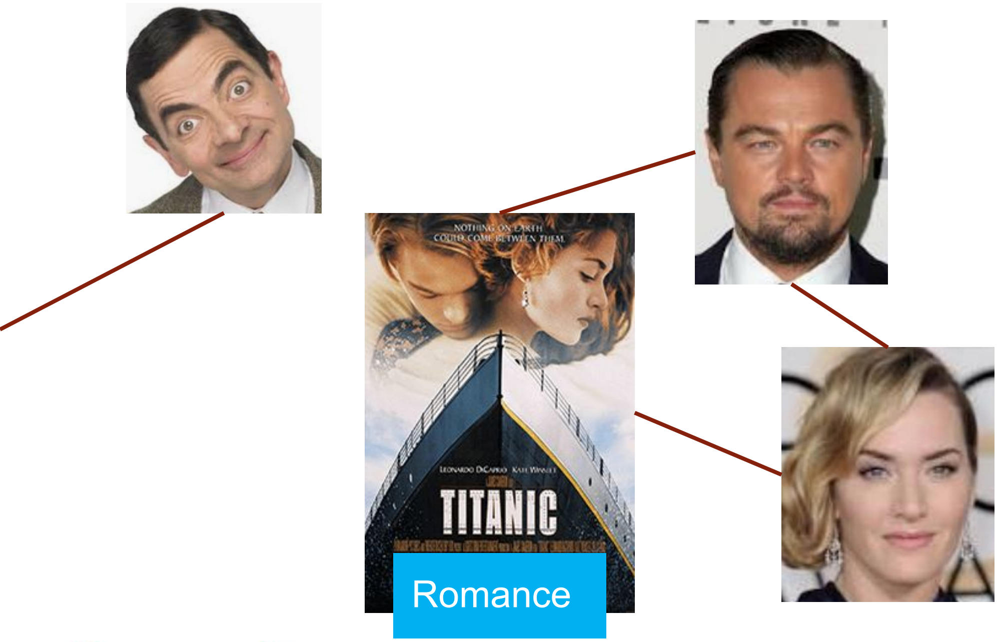
Dataset	#Graphs	Classes	Avg. #nodes	Avg. # edges
MUTAG	188	2	17.7	38.9
PTC	344	2	26.7	50.7
ENZYME	600	6	32.6	124.3
D&D	1178	2	284.4	1921.6



Graph Classification: More Examples

- **IMDB-binary:**
  ego-networks of actors who have appeared together in movies.
  classify network into action/romance movie

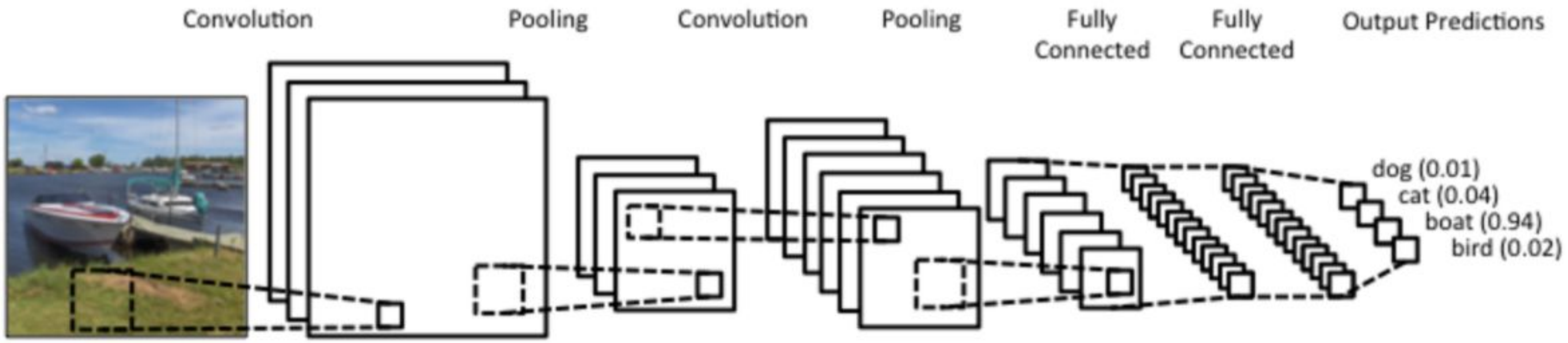
- **Reddit-binary:**
  does the user come from Q&A forum or discussion forum?


NOTHING ON EARTH COULD COME BETWEEN THEM
LEONARDO DiCAPRIO KATE WINSLET
TITANIC
Romance

CNNs to GNNs

DSP -> GSP

How do we change the CNN to Graph CNN?



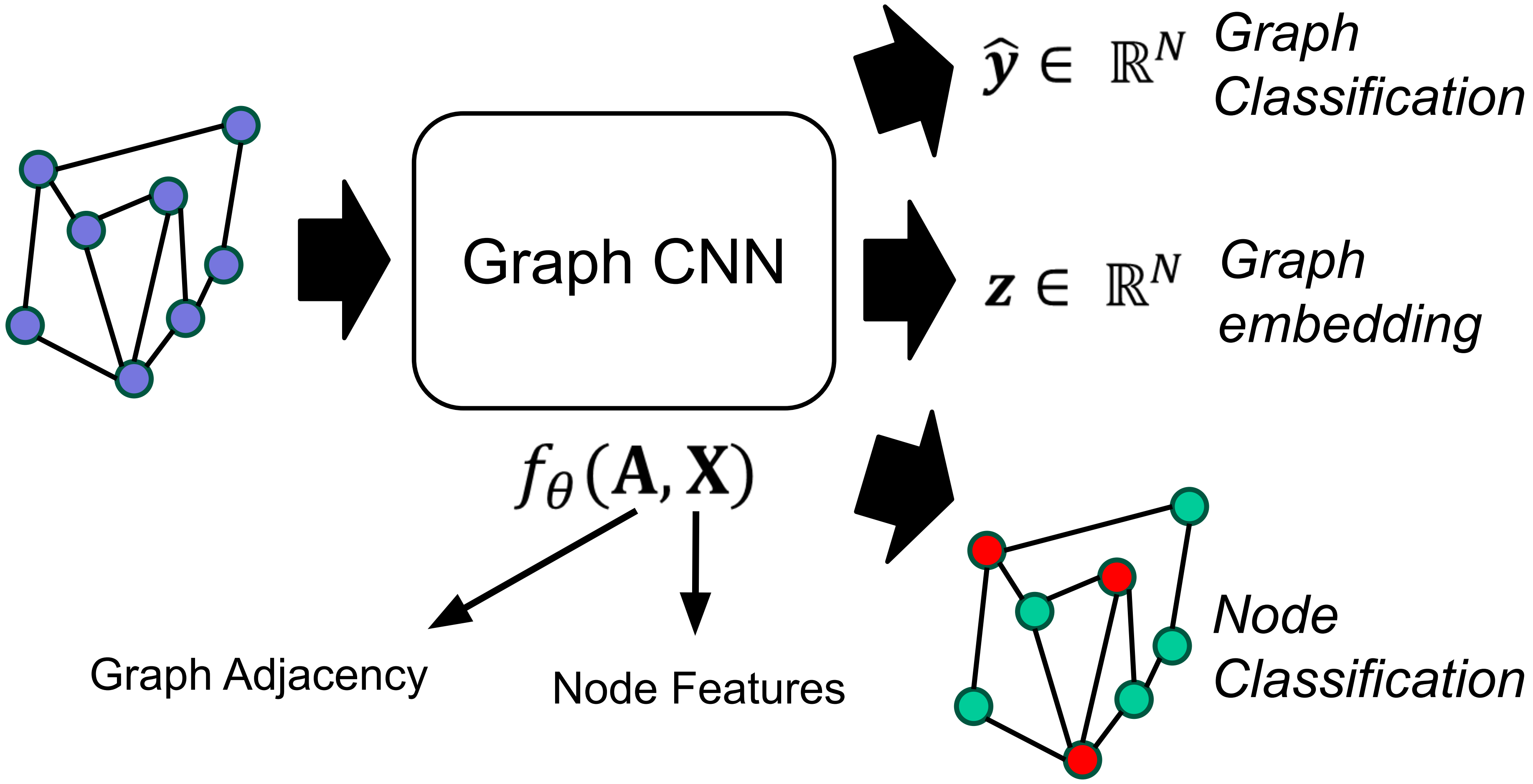
Challenges of Graph CNN

Traditional CNNs do not perform well on Graph data.

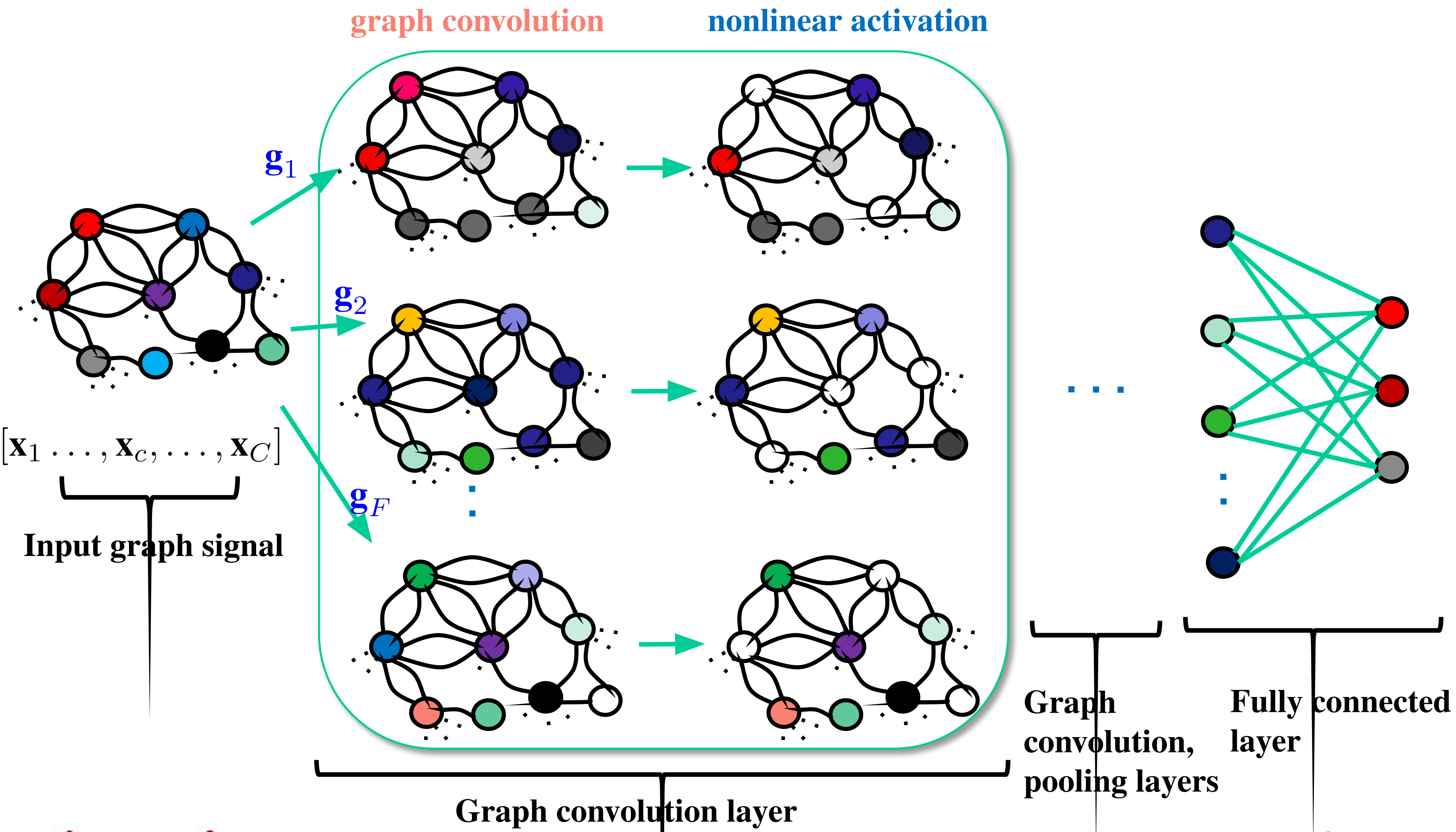
- What *graph* problems can be solved?
- What are *graph* convolutional layers?
- What is *graph* pooling?
- Which *graph* architecture should be used?

Graph Neural Network Architectures

Graph CNNs


Graph CNN
$f_\theta(\mathbf{A}, \mathbf{X})$
Graph Adjacency
Node Features
$\hat{\boldsymbol{y}} \in \mathbb{R}^N$
Graph Classification
$\boldsymbol{z} \in \mathbb{R}^N$
Graph embedding
Node Classification

Architecture of Graph CNN



Why do we need Graph CNNs?

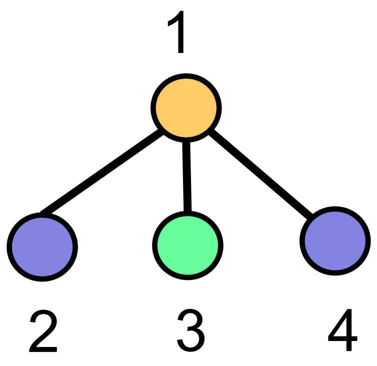
- Varying input size
- Permutation invariance – graph representation not unique

$$f_\theta(\mathbf{PAP}^\top, \mathbf{PX}) = f_\theta(\mathbf{A}, \mathbf{X})$$ Permutation Invariance (Graph Classification)

$$f_\theta(\mathbf{PAP}^\top, \mathbf{PX}) = \mathbf{P}f_\theta(\mathbf{A}, \mathbf{X})$$ Permutation Invariance (Node Classification)

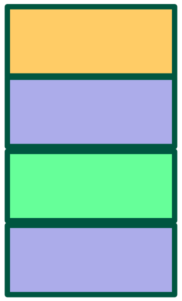
(Some works refer to this as Permutation Equivariance)

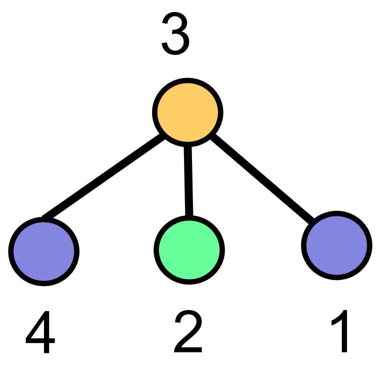
Compositions of permutation equivariant functions are also permutation equivariant



$$A = \begin{matrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{matrix}$$

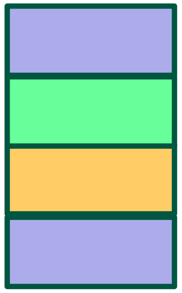
X =





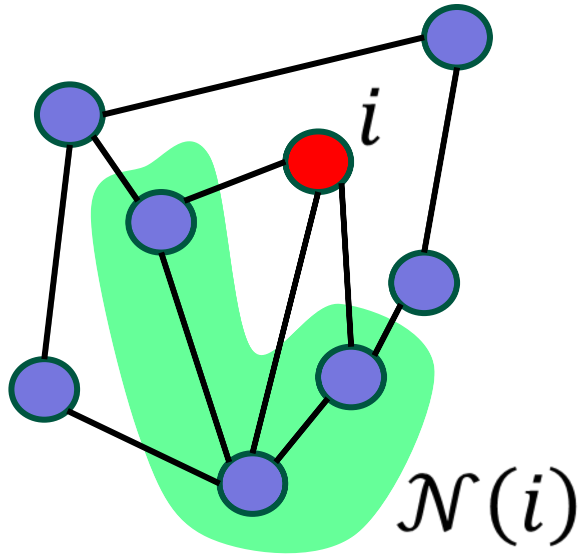
$$A = \begin{matrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{matrix}$$

X =



Graph Convolutional Layers

1. Aggregate node features of neighbors
2. Combine current node feature with aggregation from prior step



$$\mathbf{a}_i^{(\ell)} = AGG^{(\ell)}\left(\{\mathbf{x}_j^{(\ell-1)}, j \in \mathcal{N}(i)\}\right)$$
$$\mathbf{x}_i^{(\ell)} = COMB^{(\ell)}\left(\{\mathbf{x}_i^{(\ell-1)}, \mathbf{a}_i^{(\ell)}\right)$$

Simple Graph Convolution Layer

Learned at each layer

$$\mathbf{x}_i^{(\ell+1)} = \mathbf{W}_1^\top \mathbf{x}_i^{(\ell)} + \mathbf{W}_2^\top \sum_{j \in \mathcal{N}(i)} e_{j,i} \mathbf{x}_j^{(\ell)}$$

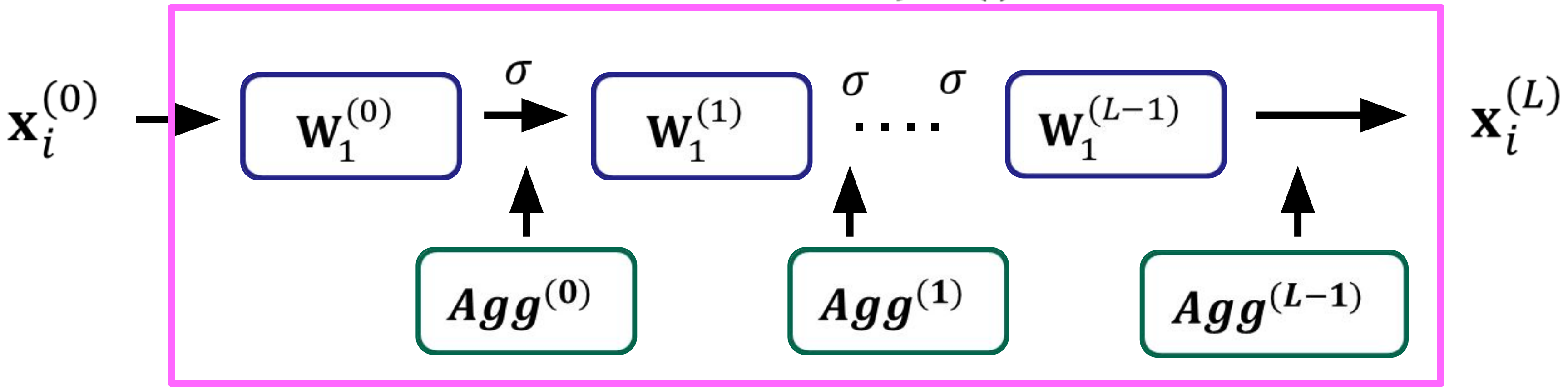
Vectorized Representation:

$$\mathbf{X}^{(\ell+1)} = \mathbf{X}^{(\ell)} \mathbf{W}_1 + \mathbf{A}\mathbf{X}^{(\ell)} \mathbf{W}_2$$

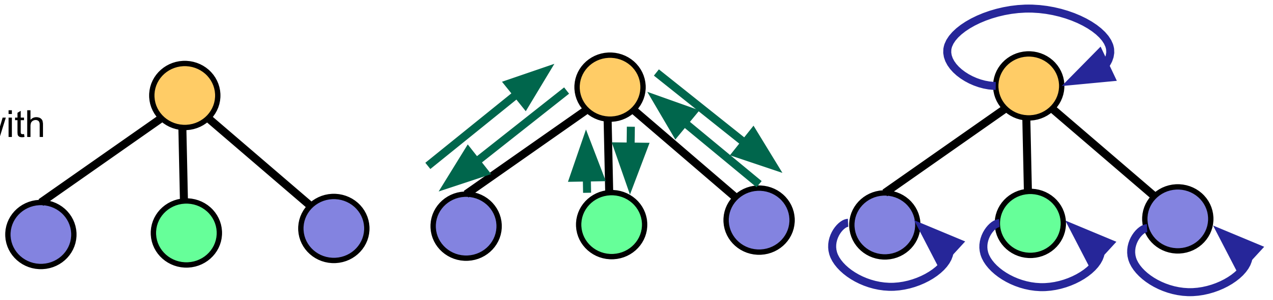
Morris et. Al., “Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks”, 2019.

Graph CNNs as Distributed MLPs

$$\mathbf{x}_i^{(\ell+1)} = \mathbf{W}_1^\top \mathbf{x}_i^{(\ell)} + \mathbf{W}_2^\top \sum_{j \in \mathcal{N}(i)} e_{j,i} \mathbf{x}_j^{(\ell)}$$



1. Aggregate
2. Single Linear layer with aggregation as bias
3. Nonlinearity
4. Repeat



Aggregate

Combine (Linear layer with aggregation as bias)

Graph Convolutional Layers

$$\mathbf{x}^{(\ell+1)} = \mathbf{W}^{\top} \sum_{j \in \mathcal{N}(i)} \frac{e_{j,i}}{\sqrt{\hat{d}_j \hat{d}_i}} \mathbf{x}^{(\ell)}$$

GCN: $\mathbf{X}^{(\ell+1)} = \widehat{\mathbf{D}}^{-\frac{1}{2}} \widehat{\mathbf{A}} \widehat{\mathbf{D}}^{-\frac{1}{2}} \mathbf{X}\mathbf{W}$ Kipf, Welling, 2017.

TAGCN: $\mathbf{X}^{(\ell+1)} = \sum_{k=0}^{K} \left(\mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}\right)^{k} \mathbf{X}\mathbf{W}_k$ Du, Zhang, Wu, Moura, Kar, 2018.

$\bar{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$ Normalized Adjacency matrix

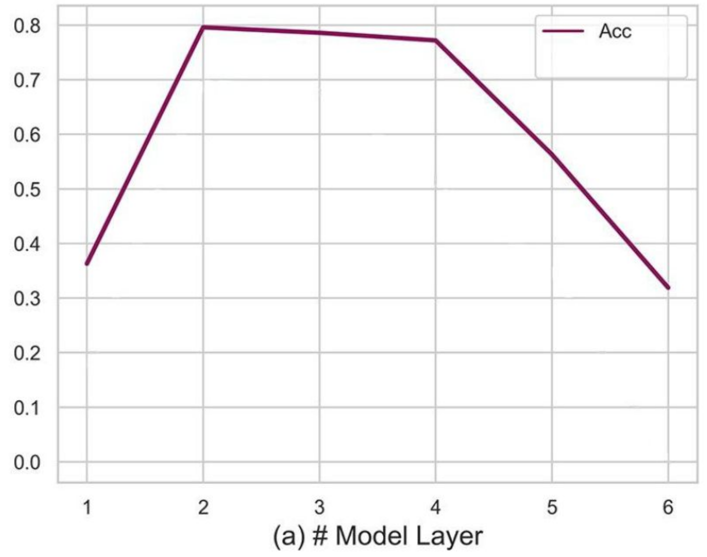
Why normalize?

$$\mathbf{X}^{(L)} = \mathbf{A}^{L} \mathbf{X}^{(0)} \mathbf{W}_1 \mathbf{W}_2 \ldots \mathbf{W}_L = \underbrace{(\mathrm{GFT}^{-1}) \boldsymbol{\Lambda}^{L} (\mathrm{GFT})}_{\text{Blows up if largest eigenvalue is greater than 1}} \mathbf{X}^{(0)} \mathbf{W}_1 \mathbf{W}_2 \ldots \mathbf{W}_L$$

Over-Smoothing Problem

Chen et. al., “Measuring and recliving over smoothing…” AAAI 2020

Kei Ishikawa, “GNN Oversmoothing”, ETH Zurich Course Slides.



The node classification accuracy (Acc) of GCNs on the CORA dataset.

Increased depth of Graph CNNs can lead to lower accuracy

$$\mathbf{X}^{(L)} = \bar{\mathbf{A}}^{L}\mathbf{X}^{(0)}\mathbf{W}_1\mathbf{W}_2 \ldots \mathbf{W}_{\mathrm{L}}$$

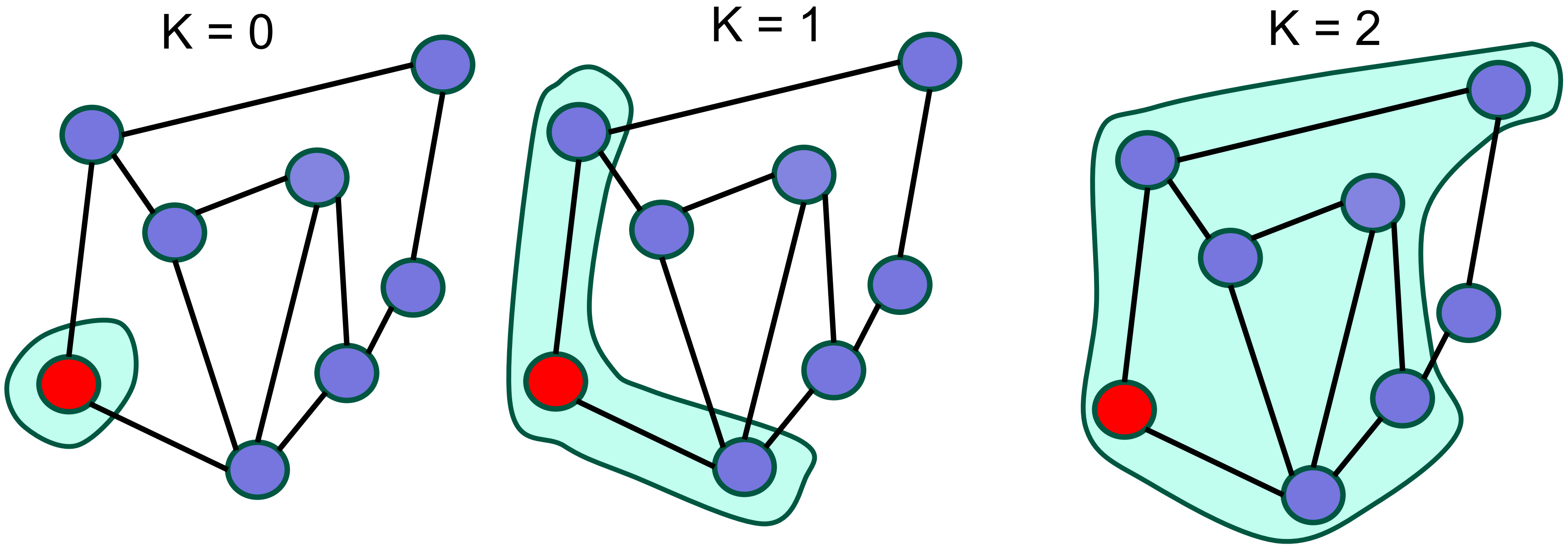
$$= \underbrace{(\mathbf{U}^{-1})\boldsymbol{\Lambda}^{L}(\mathbf{U})\mathbf{X}^{(0)}}_{\text{Converges to dominant eigenvector of } \bar{\mathbf{A}} \text{ by power iteration}}\,\mathbf{W}_1\mathbf{W}_2 \ldots \mathbf{W}_{\mathrm{L}}$$

TAGCN

TAGCN: $\mathbf{X}^{(\ell+1)} = \sum_{k=0}^{K} \overline{\mathbf{A}}^{k}\mathbf{X}\mathbf{W}_{k}$

Du, Zhang, Wu, Moura, Kar, 2018.

Hyperparameter K is the K-hop neighborhood over which information is aggregated



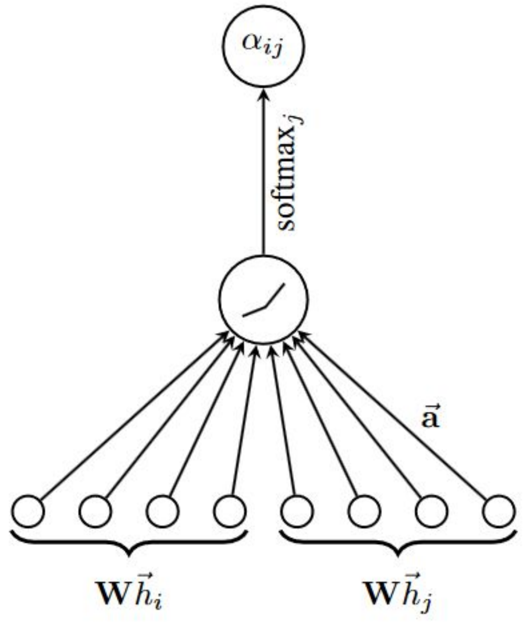
Kind of like how a CNN aggregates its neighbors!

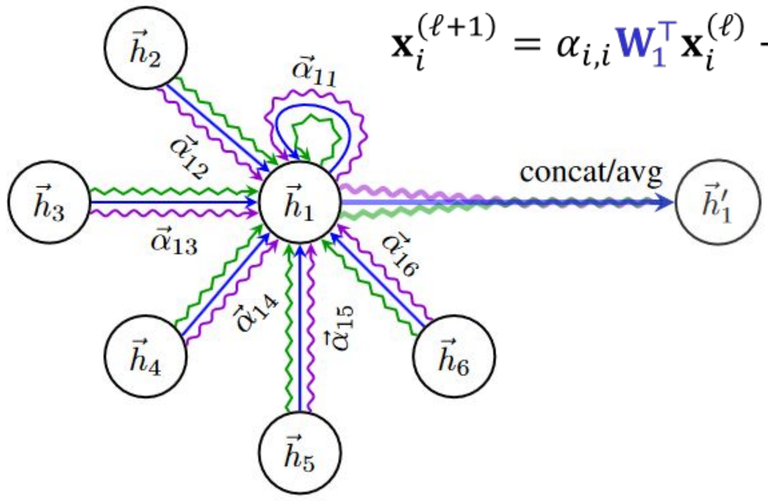
Graph Attention Layers

- Aforementioned graph CNNs use fixed edge weights of $\mathbf{A}$
- Graph attention networks: First learn proper edge weights, then convolve

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

$$\mathbf{x}_i^{(\ell+1)} = \alpha_{i,i}\mathbf{W}_1^\top \mathbf{x}_i^{(\ell)} + \mathbf{W}_2^\top \sum_{j\in\mathcal{N}(i)} \alpha_{i,j}\mathbf{x}_j^{(\ell)}$$

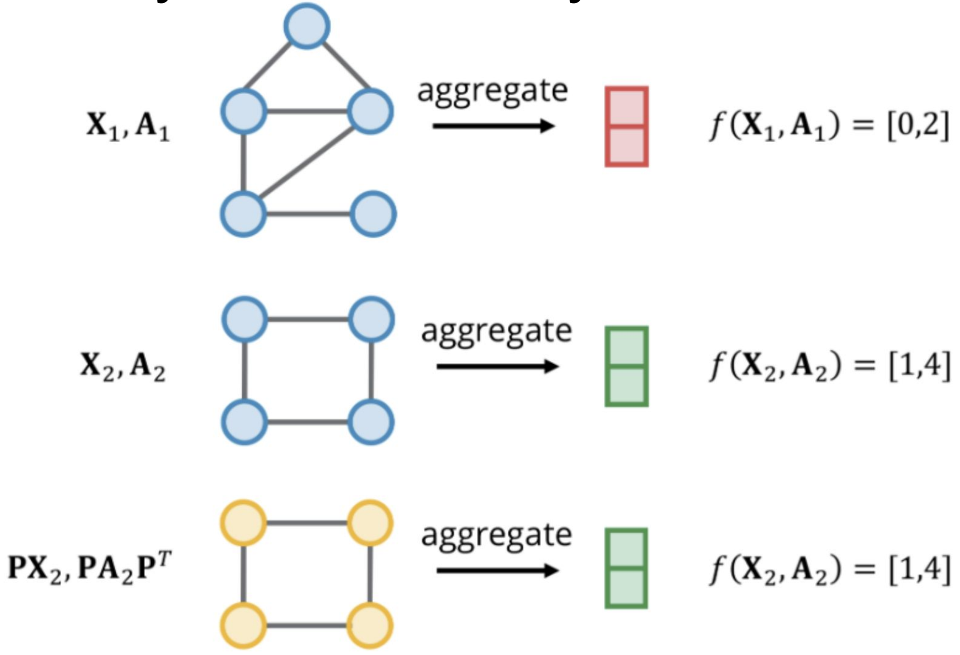




Velickovic, et al. Graph Attention Networks, 2018.

Graph Aggregation Layers

After convolutional layers, the output is N x C, which is then put into a fully connected layer.


$\mathbf{X}_1, \mathbf{A}_1$
aggregate
$f(\mathbf{X}_1, \mathbf{A}_1) = [0,2]$
$\mathbf{X}_2, \mathbf{A}_2$
aggregate
$f(\mathbf{X}_2, \mathbf{A}_2) = [1,4]$
$\mathbf{P}\mathbf{X}_2, \mathbf{P}\mathbf{A}_2\mathbf{P}^T$
aggregate
$f(\mathbf{X}_2, \mathbf{A}_2) = [1,4]$

Goal: Get the same dimension for all graphs. Make sure graphs and permuted graphs produce the same result. This is the input to the FC layer.

Graph Aggregation Layers

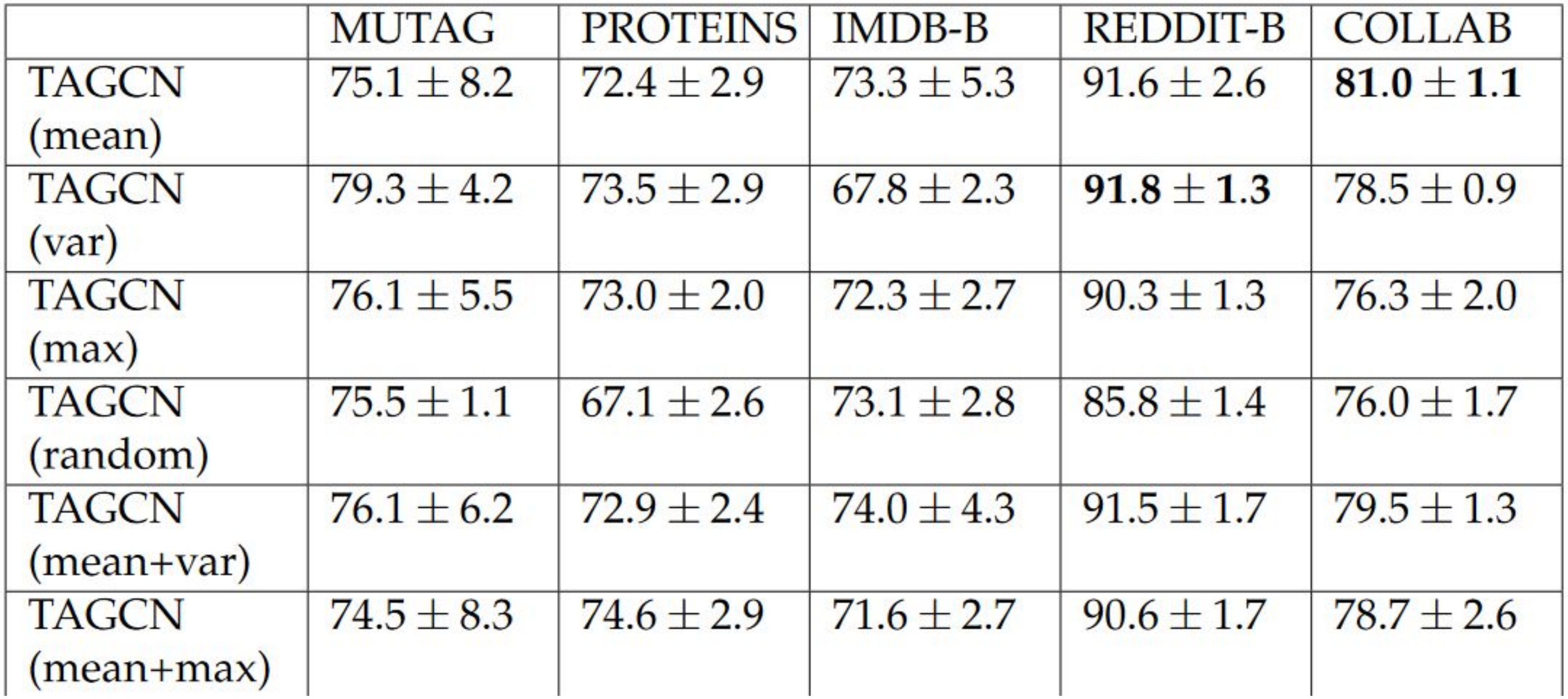
- Architectures thus far yield node embeddings
- Assign each node of the graph a representation $\mathbf{z}_i \in \mathbb{R}^n$
- **How do we obtain a global graph representation?**

Aggregation Layer: $f : \mathbb{R}^{N \times C} \rightarrow \mathbb{R}^{1 \times C}$

- **Global add pooling:** $\mathbf{z}_G = \sum_{i=1}^{N} \mathbf{z}_i$
- **Global mean pooling:** $\mathbf{z}_G = \frac{1}{N}\sum_{i=1}^{N} \mathbf{z}_i$
- **Global max pooling:** $\mathbf{z}_{G,j} = \max\limits_{i=1,\ldots,N} \mathbf{z}_{i,j}$

Comparison of Aggregation Methods

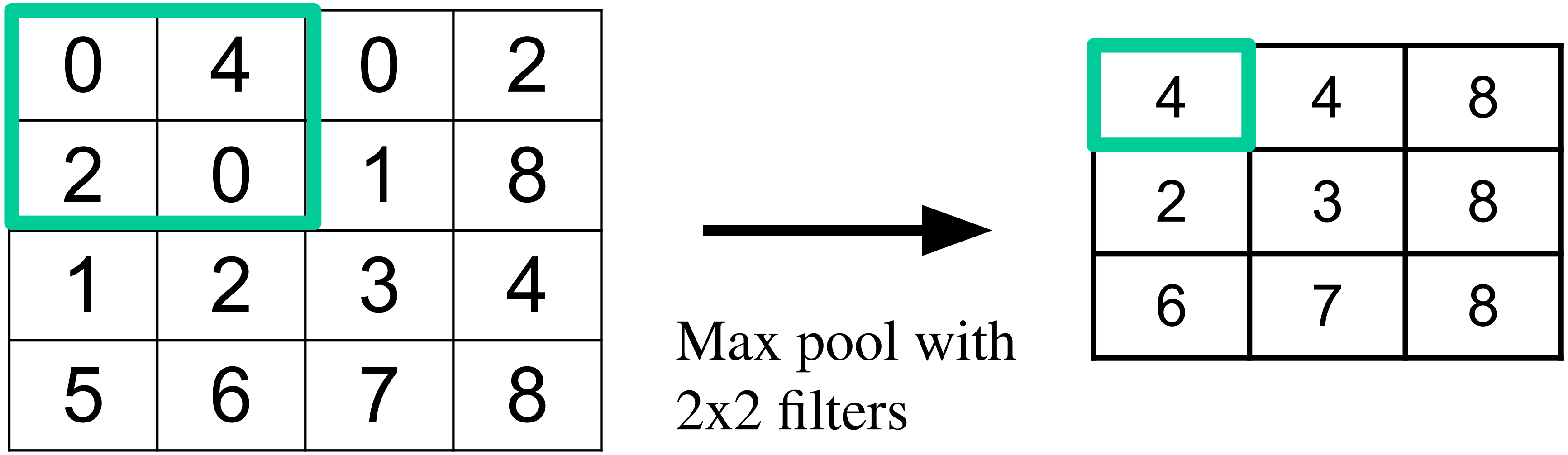
Best aggregation method is data-dependent (Mark Cheung PhD Thesis, 2023)



	MUTAG	PROTEINS	IMDB-B	REDDIT-B	COLLAB
TAGCN (mean)	75.1 ± 8.2	72.4 ± 2.9	73.3 ± 5.3	91.6 ± 2.6	$\mathbf{81.0 \pm 1.1}$
TAGCN (var)	79.3 ± 4.2	73.5 ± 2.9	67.8 ± 2.3	$\mathbf{91.8 \pm 1.3}$	78.5 ± 0.9
TAGCN (max)	76.1 ± 5.5	73.0 ± 2.0	72.3 ± 2.7	90.3 ± 1.3	76.3 ± 2.0
TAGCN (random)	75.5 ± 1.1	67.1 ± 2.6	73.1 ± 2.8	85.8 ± 1.4	76.0 ± 1.7
TAGCN (mean+var)	76.1 ± 6.2	72.9 ± 2.4	74.0 ± 4.3	91.5 ± 1.7	79.5 ± 1.3
TAGCN (mean+max)	74.5 ± 8.3	74.6 ± 2.9	71.6 ± 2.7	90.6 ± 1.7	78.7 ± 2.6

Graph Pooling

Traditional CNN Pooling



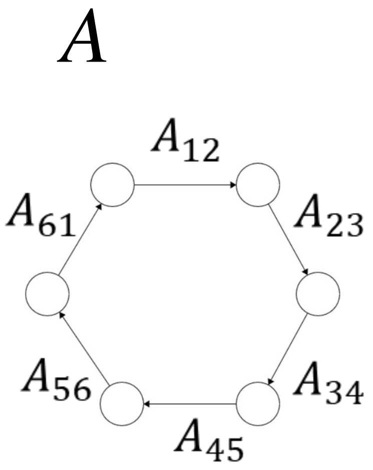
0	4	0	2
2	0	1	8
1	2	3	4
5	6	7	8

4	4	8
2	3	8
6	7	8

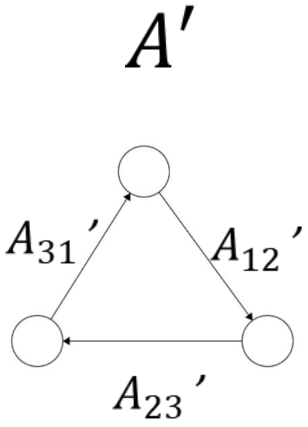
Max pool with
2x2 filters
and stride of 1

Graph Pooling

Graph Structure:



Graph pooling



Data: X $\begin{matrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{matrix}$

Challenge:
Need to know both the new data and the new graph

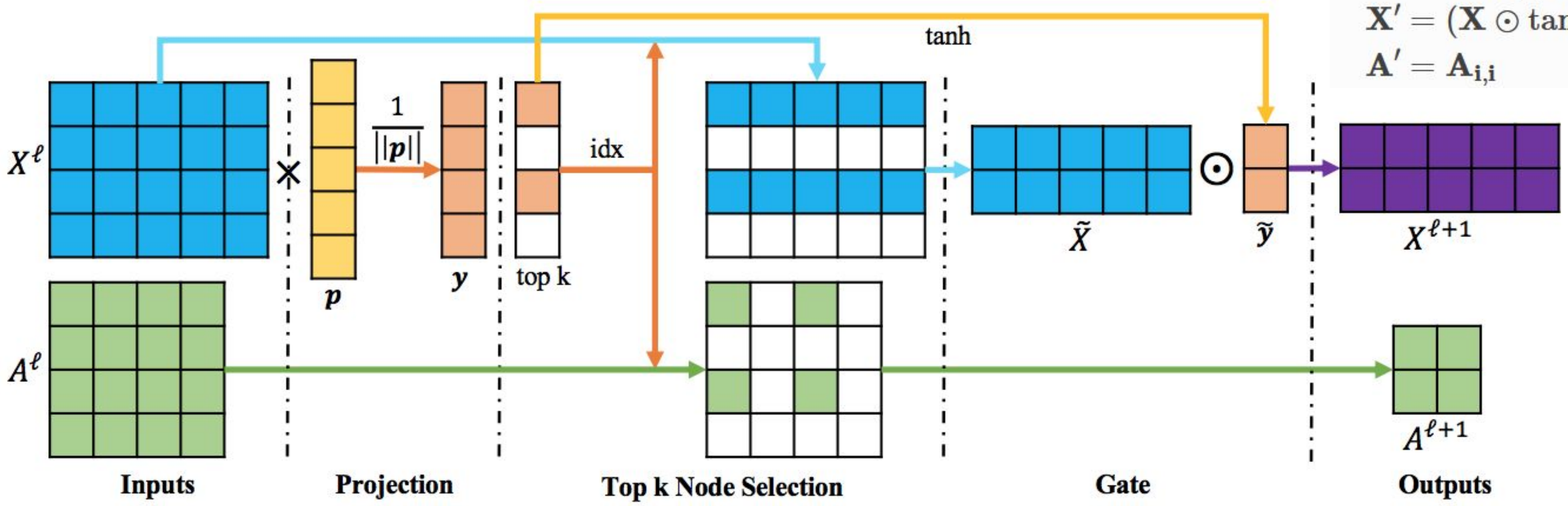
Must be permutation invariant

X' $\begin{matrix} x_1' \\ x_2' \\ x_3' \end{matrix}$

Top-k Pool (originally gpool)

- Autoencoder with a trainable projection vector

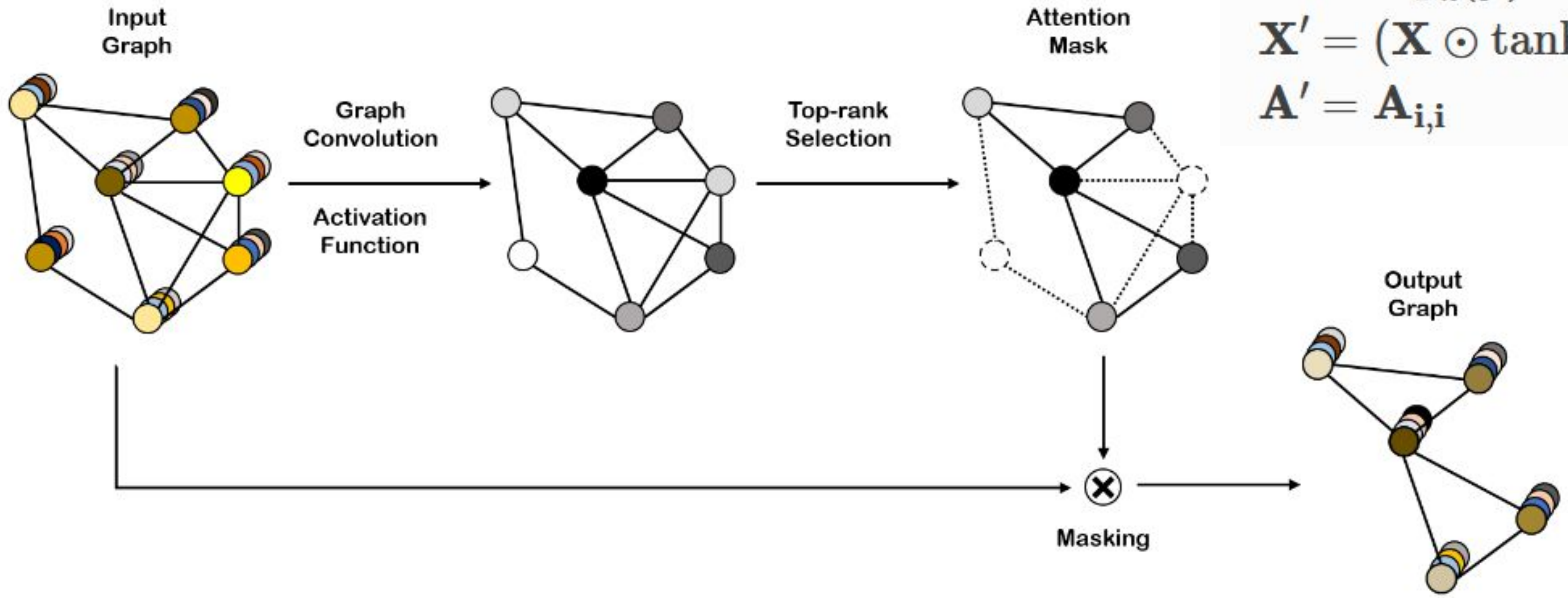
$$\mathbf{y} = \sigma\left(\frac{\mathbf{Xp}}{\|\mathbf{p}\|}\right)$$
$$\mathbf{i} = \mathrm{top}_k(\mathbf{y})$$
$$\mathbf{X}' = (\mathbf{X} \odot \tanh(\mathbf{y}))_{\mathbf{i}}$$
$$\mathbf{A}' = \mathbf{A}_{\mathbf{i},\mathbf{i}}$$



Gao and Ji: Graph U-Net (ICLR 2019 submission) and Cangea et al.: Towards Sparse Hierarchical Graph Classifiers (NeurIPS-W 2018)

Self-attention Graph Pooling (sagpool)

$$\mathbf{y} = \mathrm{GNN}(\mathbf{X}, \mathbf{A})$$
$$\mathbf{i} = \mathrm{top}_k(\mathbf{y})$$
$$\mathbf{X}' = (\mathbf{X} \odot \tanh(\mathbf{y}))_{\mathbf{i}}$$
$$\mathbf{A}' = \mathbf{A}_{\mathbf{i},\mathbf{i}}$$



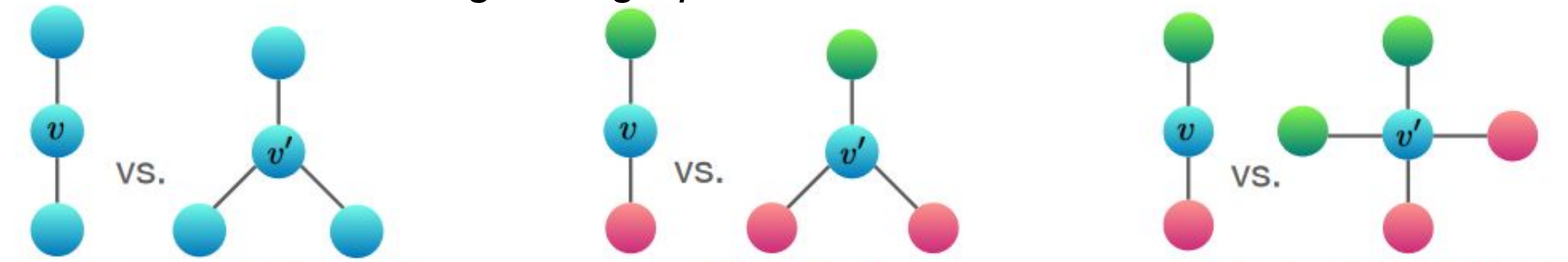
Lee, Lee, Kang: Self-Attention Graph Pooling (ICML 2019)

Graph CNN Implementation

- https://pytorch-geometric.readthedocs.io/en/latest/
- Pytorch geometric – based on standard pytorch, compatible with pytorch lightning
- Provides:
  - Efficient graph data handling
  - Provides implementations of several standard graph convolution and pooling layers
  - Provides base classes to define your own graph convolution and pooling layer
  - Access to a wide variety of graph datasets, e.g., for graph and node classification

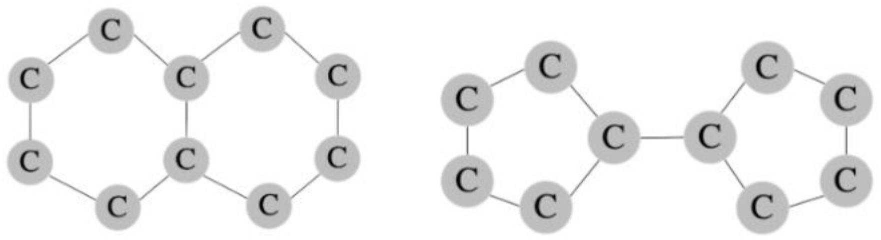
How expressive are Graph NNs

Mean/max aggregation operators assign v, v' same embedding even though the graph structures are different



(a) Mean and Max both fail (b) Max fails (c) Mean and Max both fail

Xu et. Al., "How Powerful Are GNNs?", 2019.



A_1 (Decalin) A_2 (Bicyclopentyl)

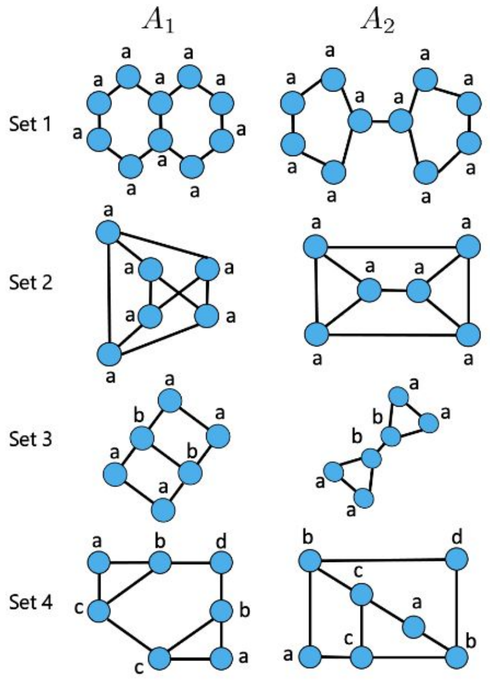
Shi, et. Al. "A dual approach to graph CNNs", 2020.

How expressive are Graph NNs

Possible Solution: Use both the nodal shift (A) and spectral shift (M)

$$\left.\begin{aligned} \mathbf{A} &= \mathrm{GFT}^{-1}\boldsymbol{\Lambda}\mathrm{GFT} \\ \mathbf{M} &= \mathrm{GFT}\boldsymbol{\Lambda}^{*}\mathrm{GFT}^{-1} \end{aligned}\right\}$$

Perform graph convolutions using both shifts



Summary of Results in Terms of Classification Accuracy for Synthetic Datasets

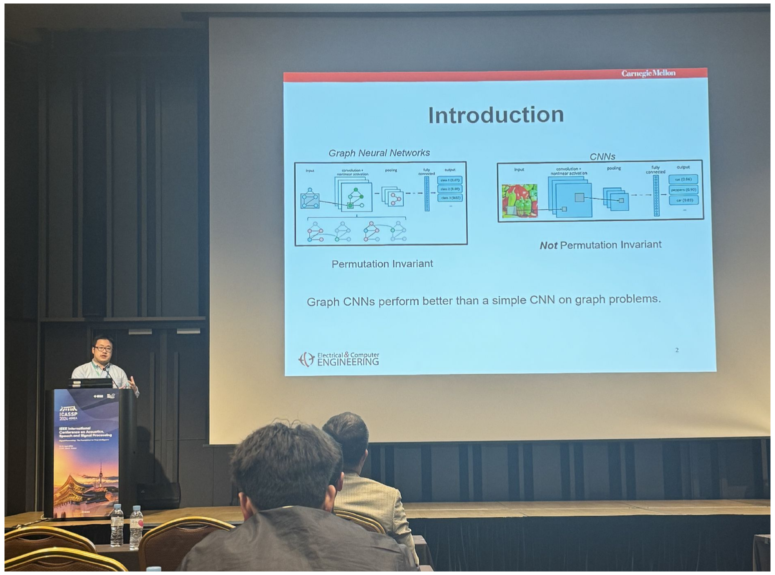
		Set 1	Set 2	Set 3	Set 4
GCN	A	0.495 ± 0.039	0.495 ± 0.014	0.491 ± 0.009	0.493 ± 0.011
	M	0.980 ± 0.008	0.580 ± 0.017	0.997 ± 0.003	0.819 ± 0.027
	$A + M$	$\mathbf{0.983 \pm 0.005}$	0.602 ± 0.036	$\mathbf{0.998 \pm 0.002}$	0.860 ± 0.031
TAGCN	$P(A)$	0.488 ± 0.028	0.484 ± 0.022	0.493 ± 0.007	0.490 ± 0.019
	$P(M)$	0.981 ± 0.007	$\mathbf{0.712 \pm 0.062}$	$\mathbf{0.998 \pm 0.002}$	$\mathbf{0.993 \pm 0.006}$
	$P(A + M)$	$\mathbf{0.983 \pm 0.005}$	0.698 ± 0.023	0.998 ± 0.003	$\mathbf{0.993 \pm 0.006}$

Shi, et. Al. “A dual approach to graph CNNs”, 2020.

Last Year: Case Study: GNNs = CNNs + B.C.

- Presented Thursday (4/18/24) at ICASSP 2024 in South Korea

- Combines GSP and GCNNs


Carnegie Mellon
Introduction
Graph Neural Networks
CNNs
Permutation Invariant
Not Permutation Invariant
Graph CNNs perform better than a simple CNN on graph problems.
Electrical & Computer ENGINEERING

J. Shi, Shreyas Chaudhari, and J. M. F. Moura, “Graph Convolutional Neural Networks in the Companion Model,” International Conference on Acoustics, Speech, and Signal Processing (ICASSP) (2024).

This Year: Case Study: Inferring the Graph Structure of Images for Graph Neural Networks

- Presenting at GSP Workshop 2025 in Montreal, Canada

M. Gowda, J. Shi, A. Santos, J. M. F. Moura, “Inferring the Graph Structure of Images for Graph Neural Networks,” GSP Workshop 2025.

Mayur Gowda is currently an IDL student!!

Main Idea: How do we represent images using graphs?

MNIST

28 x 28 pixel images

→

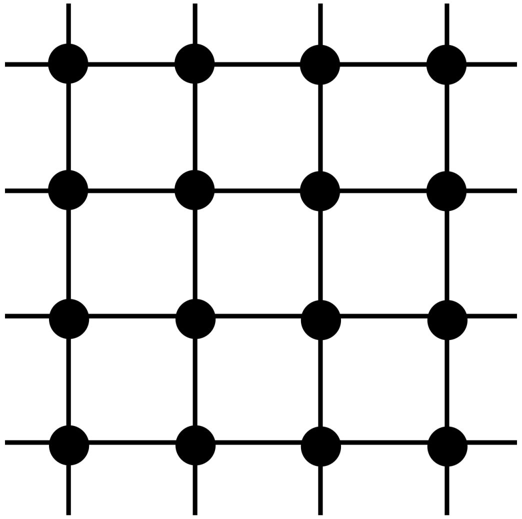
Grid Graph

28 * 28 = 764 nodes

764 x 764 adj

→

Graph CNNs (GCN, GAT, ...)



Most intuitive:

Nodes = pixels, Edges connect Adjacent Pixels,

Features = pixel intensities

Main Idea: How do we represent images using graphs?

Can we produce a better **graph** representation of images to increase the accuracy for downstream geometric deep learning tasks?

Capturing pixel-wise similarities. Engineering features.

Many methods in the literature: Superpixels, Color Similarity

Main Idea: How do we represent images using graphs?

MNIST

28 x 28 pixel images

→

Inferring the Graph of Networked Dynamical Systems

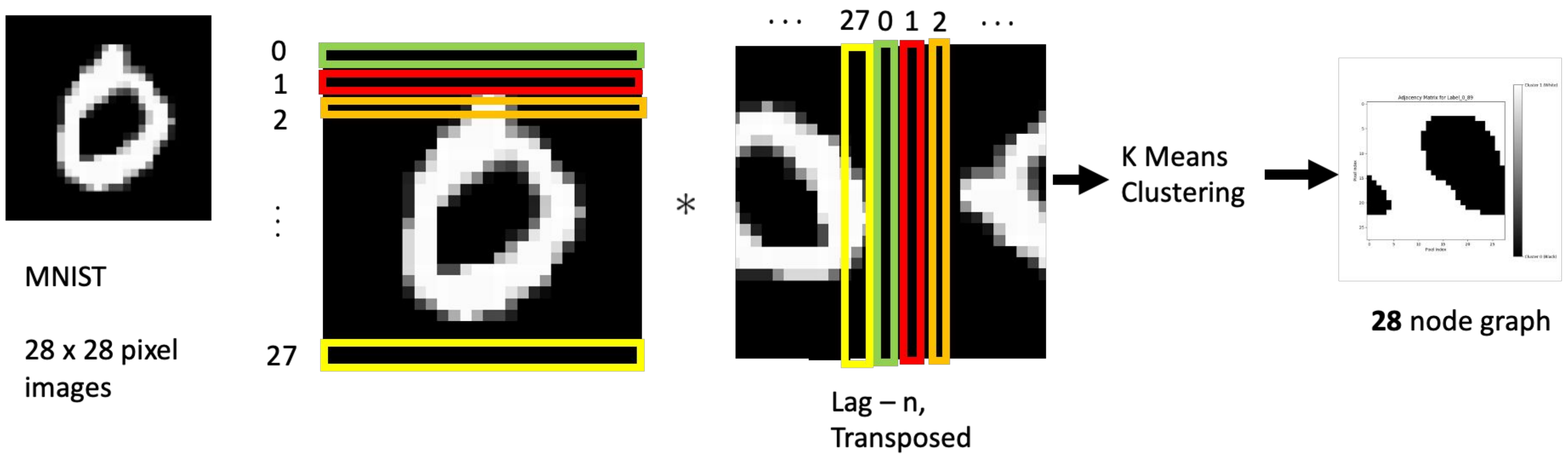
Santos, Rente, Seabra, Moura

→

Graph CNNs (GCN, GAT, ...)

Our method uses the method developed by our collaborators for time series correlation.

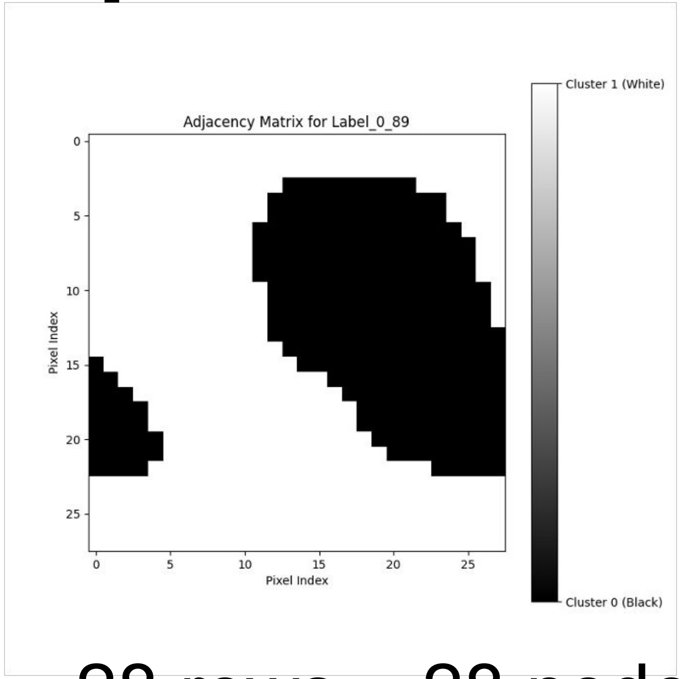
Clustering MNIST Using Rows

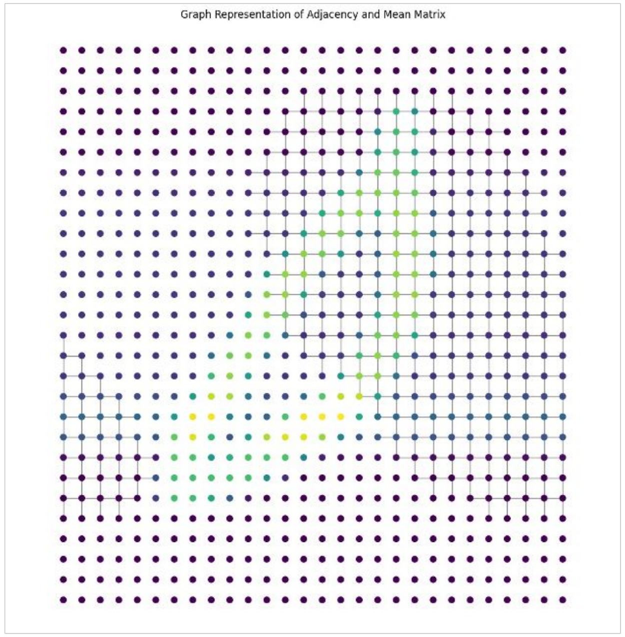


Compare each row of image with every other row

Redundant, non-local for n between 0 to 27

Representation after Clustering





28 rows = 28 nodes

28 x 28 adjacency matrix, which we superimpose the image on

Is this a good idea?

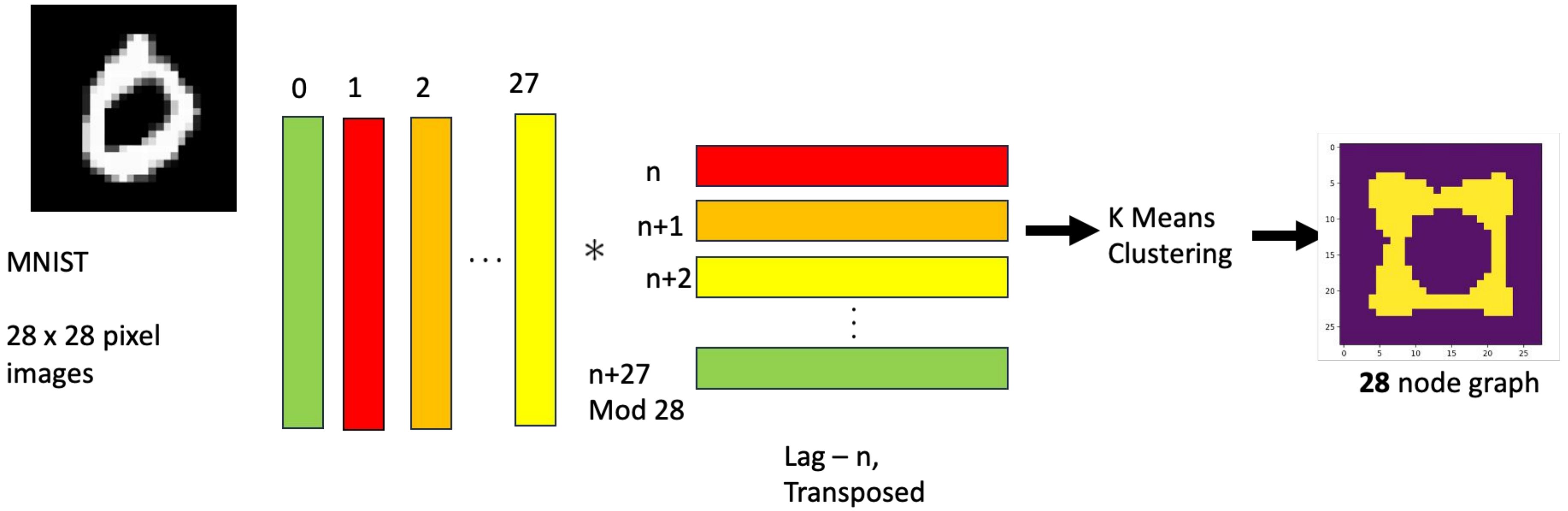
Problems

Directly using the 28 x 28 adjacency matrix for the row graph as the image?

The graph only relates the rows not the relationships between the 784 pixels!

Should have 784 x 784 adjacency matrix like the grid graph!

Column Graph

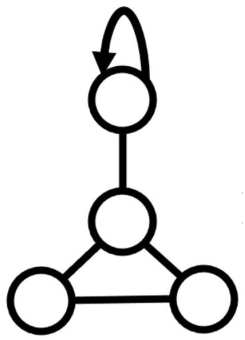


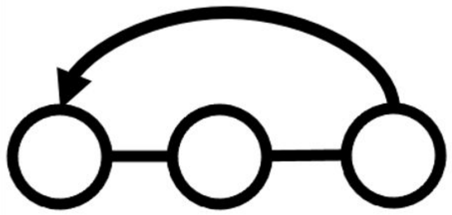
Compare each column of image with every other column

Same issues as row. Only relates the 28 columns!

Kronecker Products in GSP

Suppose we have data defined on a graph (like an electrical grid), but it also is a time series…

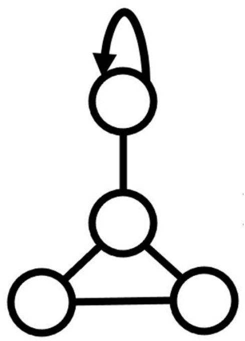


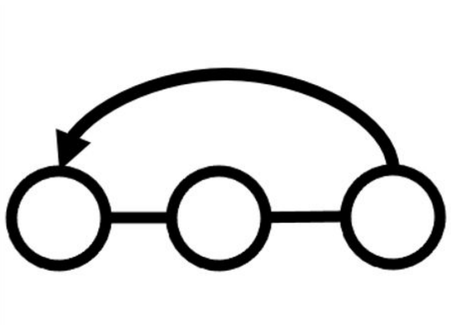


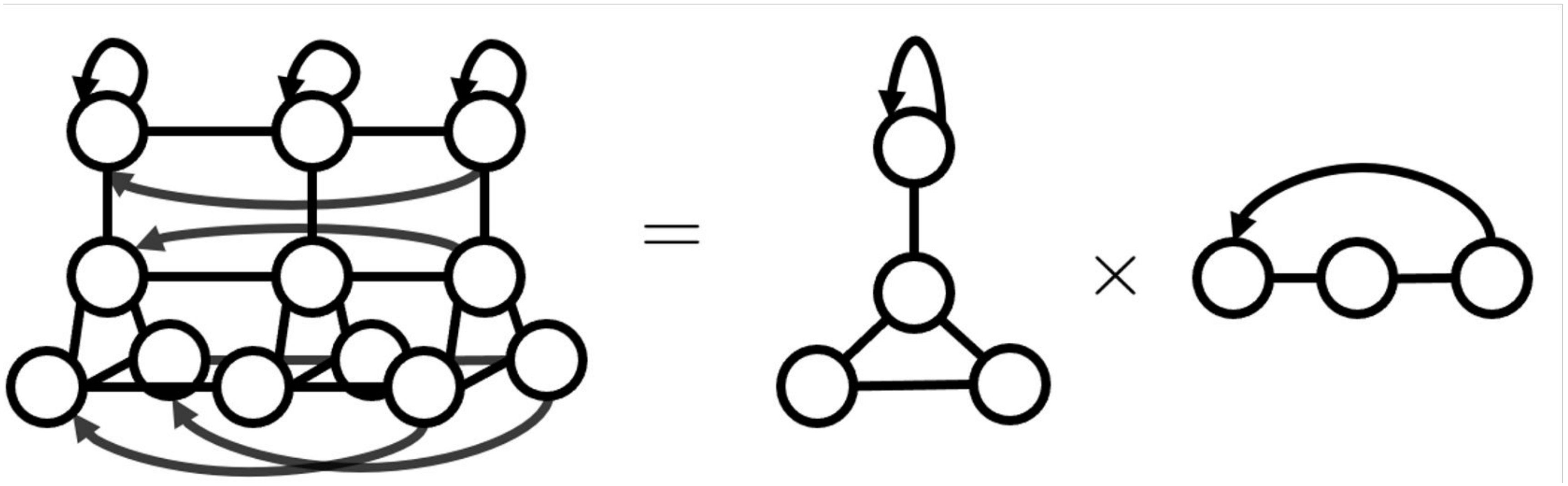
Two Methods: Represent it on the space graph (left) where each node is a vector of time values. Or Represent it on a time graph (right) where each node is a vector of space values (rarely done).

Is there a better way?

Kronecker Products in GSP



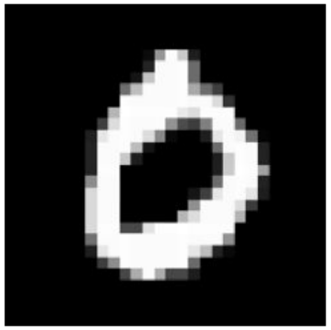




Taking the product graph gives one value per node -> very natural and intuitive

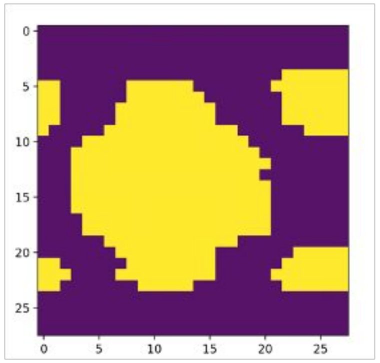
Kronecker Products in GSP

The (Cartesian) product of a line graph (row) and a line graph (column) is a grid graph

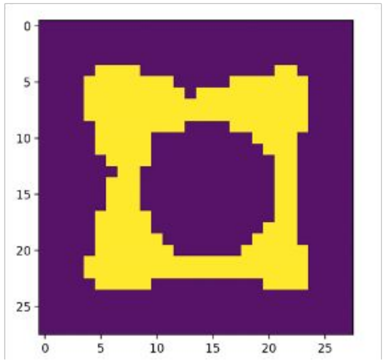


MNIST

28 x 28 pixel images



Row Graph
28 x 28

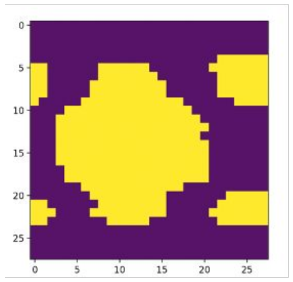


Column Graph
28 x 28

Use Kronecker Product

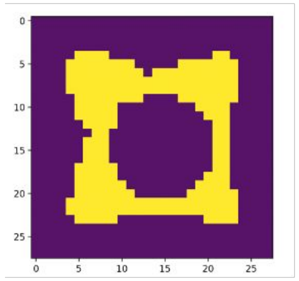
Produces 784 x 784

Clustering MNIST



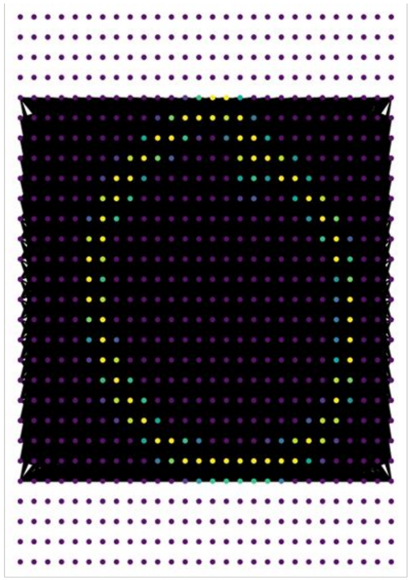
Row Graph
28 x 28

A_r



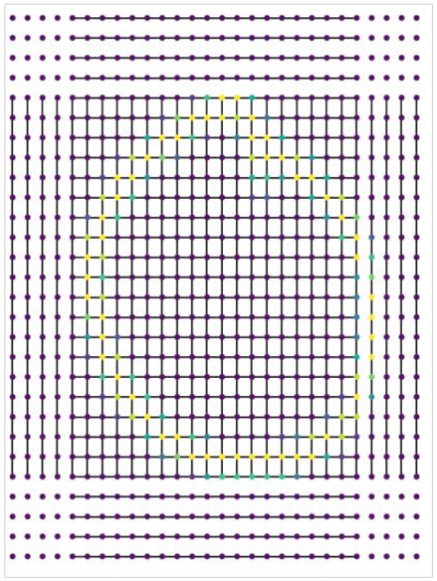
Column Graph
28 x 28

A_c



$$A_1 = A_r \otimes A_c$$

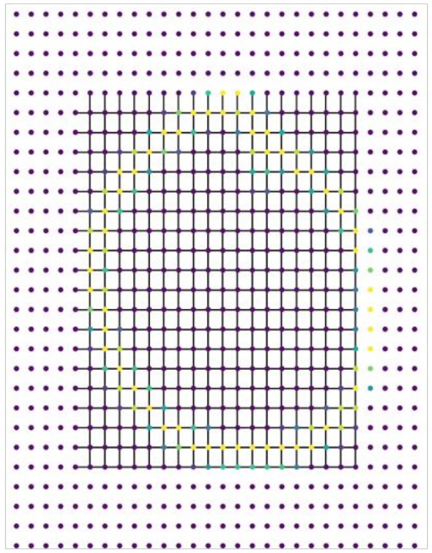
Kronecker Product
Very Dense



$$A_2 = A_r \otimes I + I \otimes A_c$$

Cartesian Product

Grid Graph?



$$A_2 \odot (A_r \otimes A_c + A_c \otimes A_r)$$

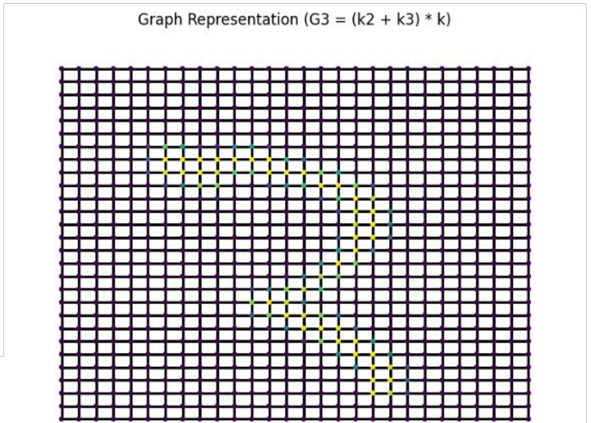
We use this one!

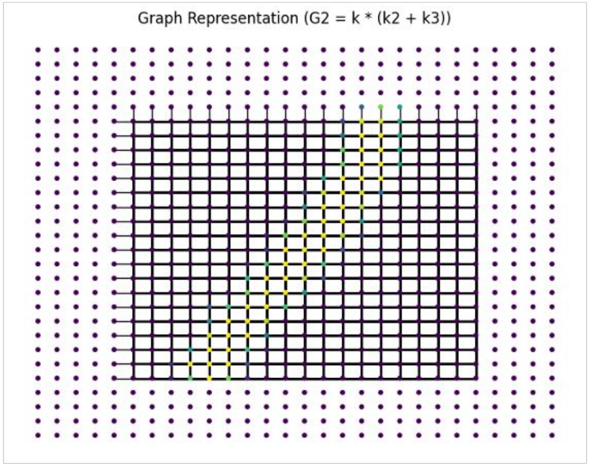
Examples:

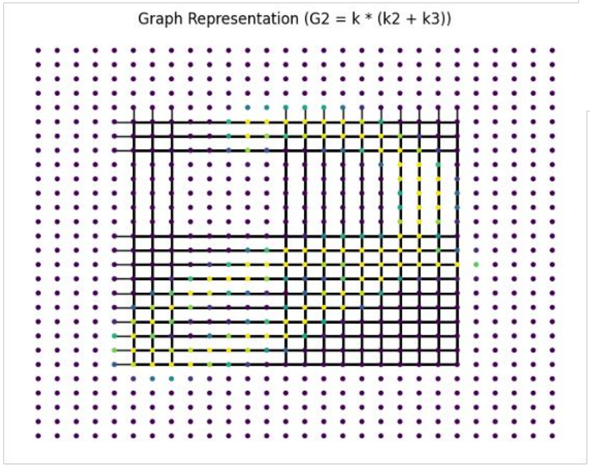
$$A_2 = A_r \otimes I + I \otimes A_c$$

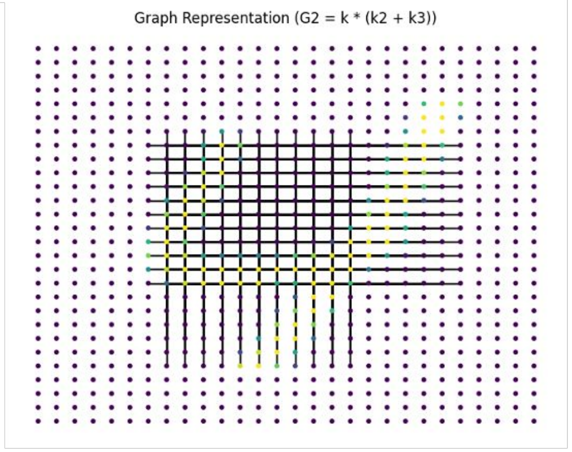
G = $A_2 \odot (A_r \otimes A_c + A_c \otimes A_r)$

Few Visual Examples of Graphs for MNIST:









What about the features?

1 Feature: Pixel values.

Can we use more descriptive features/graph signals besides pixel values?

What about the features?

- Mean at each pixel position:

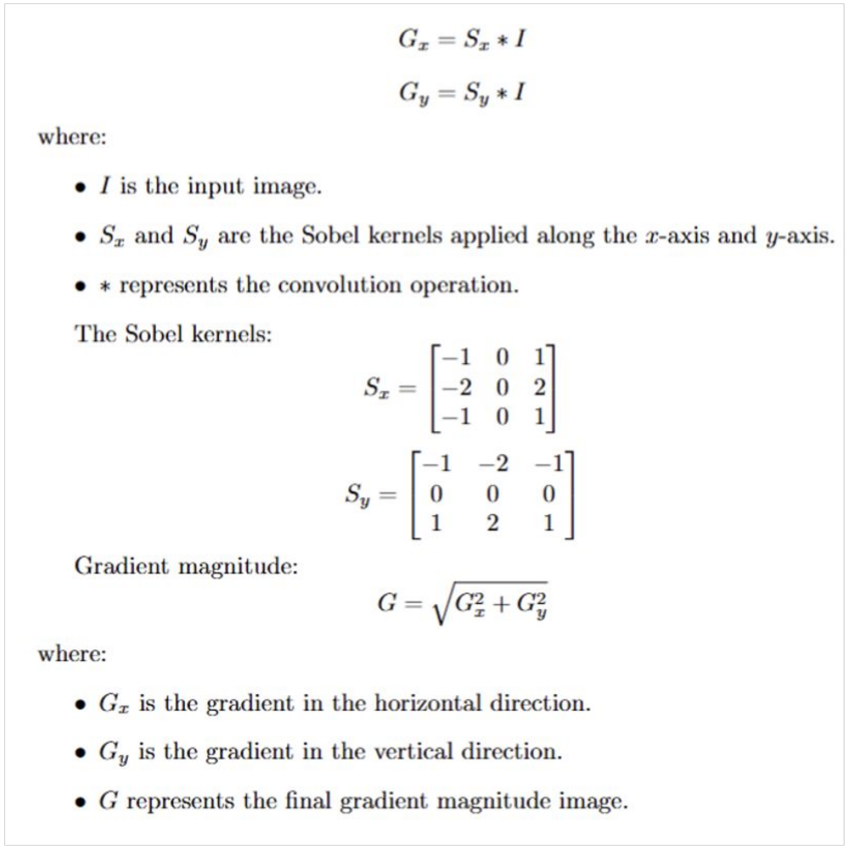
$$M(i,j) = \frac{1}{9}\sum_{m=-1}^{1}\sum_{n=-1}^{1} I(i+m, j+n)$$

- Variance at each pixel position:

$$\operatorname{Var}[X] = \operatorname{E}[X^2] - (\operatorname{E}[X])^2.$$

What about the features?

- Gradient Magnitude:



$$G_x = S_x * I$$

$$G_y = S_y * I$$

where:

- I is the input image.
- S_x and S_y are the Sobel kernels applied along the x-axis and y-axis.
- $*$ represents the convolution operation.

The Sobel kernels:

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

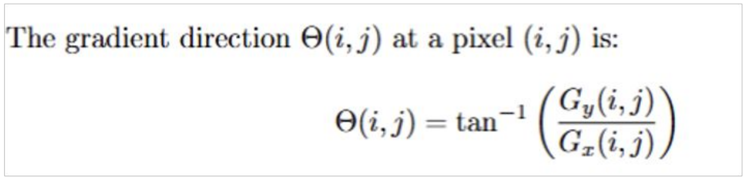
Gradient magnitude:

$$G = \sqrt{G_x^2 + G_y^2}$$

where:

- G_x is the gradient in the horizontal direction.
- G_y is the gradient in the vertical direction.
- G represents the final gradient magnitude image.

- Gradient Direction:

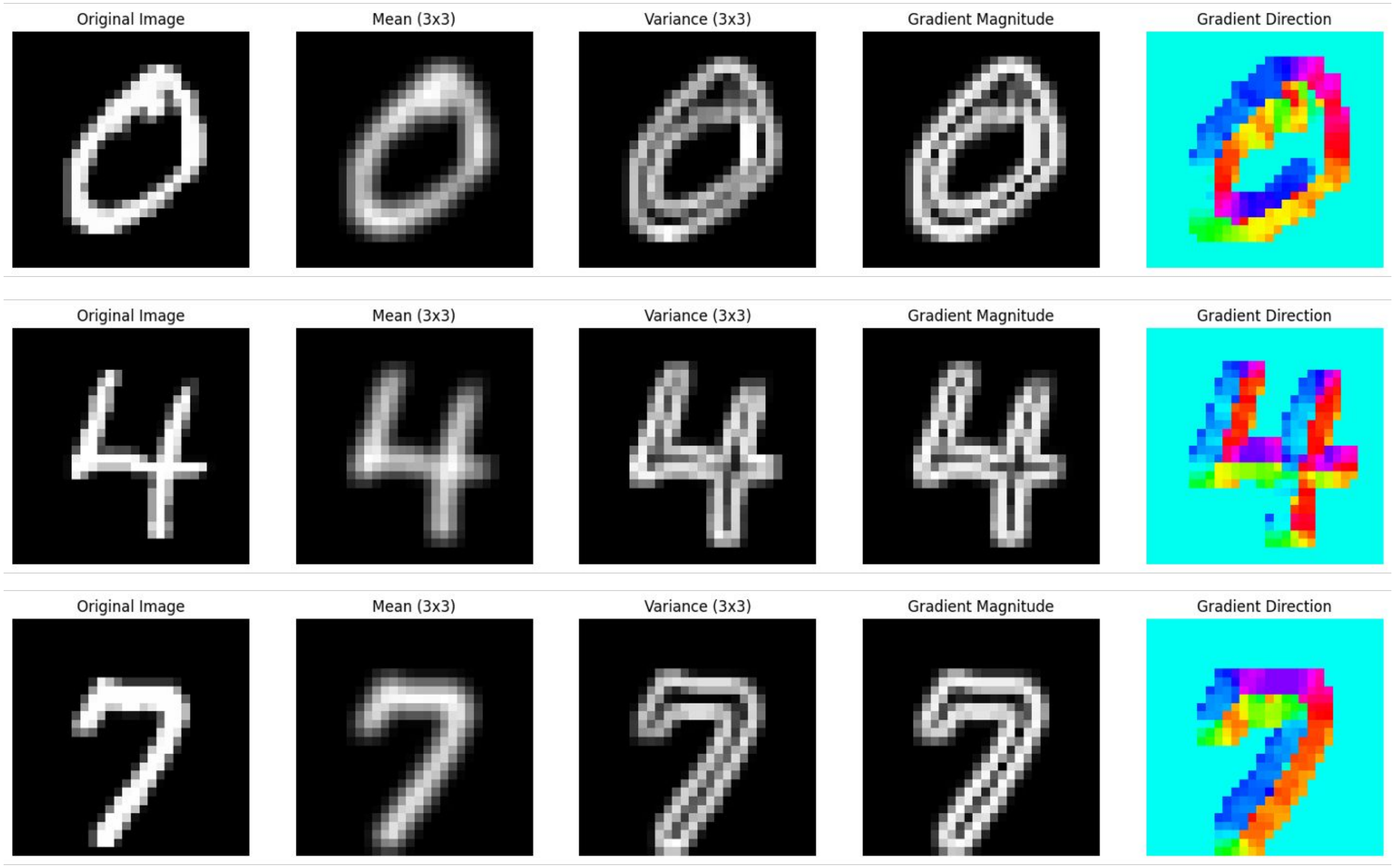


The gradient direction $\Theta(i,j)$ at a pixel (i,j) is:

$$\Theta(i,j) = \tan^{-1}\left(\frac{G_y(i,j)}{G_x(i,j)}\right)$$

Electrical & Computer ENGINEERING

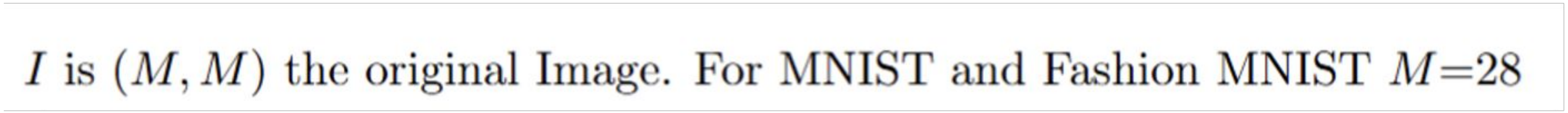
Example Features



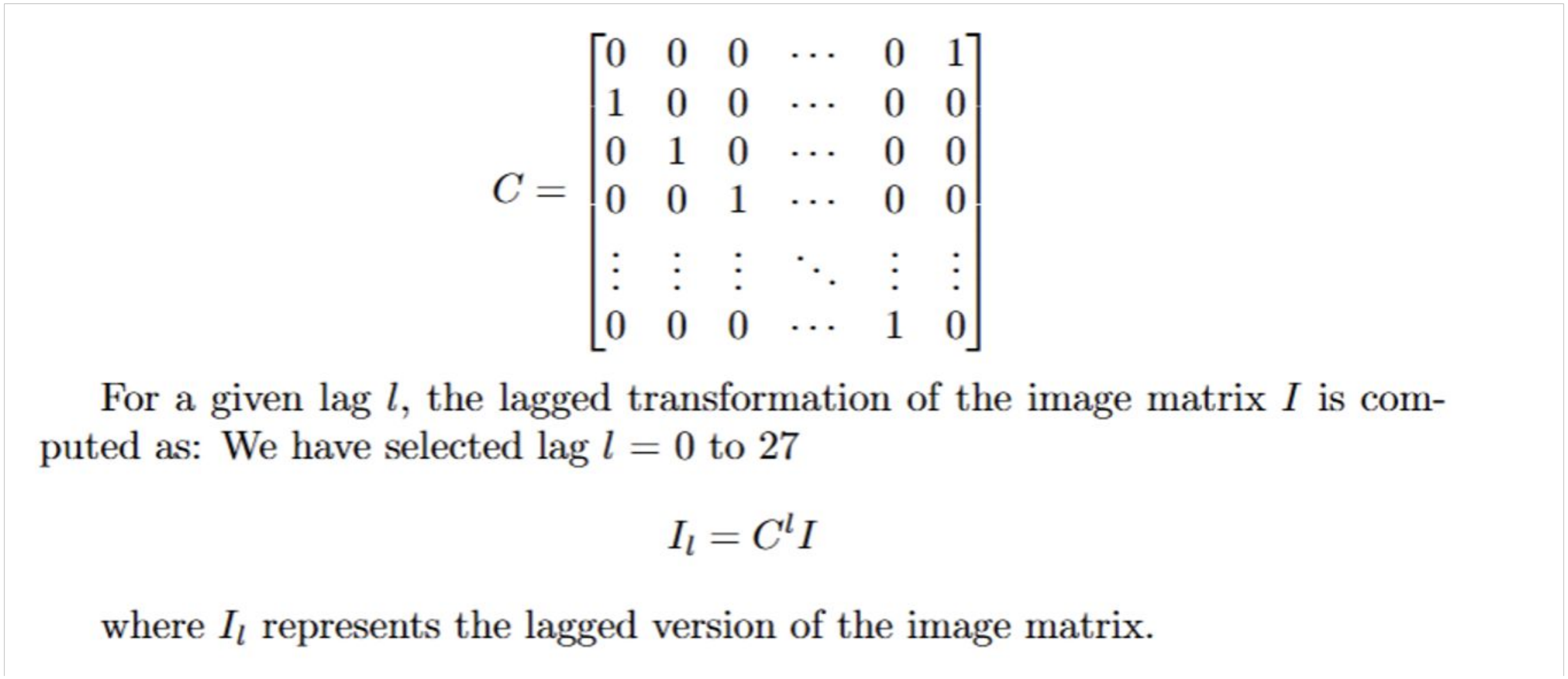
Features also affect accuracy!

GNN Model	Number of Node Features	Test Accuracy (mean_acc ± std_deviation)
GCN	1 (Pixel Intensity)	0.5505 ± 0.0289
GCN	4 (Mean, Variance, Gradient Magnitude, Gradient Direction)	0.7206 ± 0.0401
GAT	4 (Mean, Variance, Gradient Magnitude, Gradient Direction)	0.7818 ± 0.0006

Node Features for Row and Column Graphs



I is (M, M) the original Image. For MNIST and Fashion MNIST M=28



$$C = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix}$$

For a given lag l, the lagged transformation of the image matrix I is computed as: We have selected lag $l = 0$ to 27

$$I_l = C^l I$$

where I_l represents the lagged version of the image matrix.

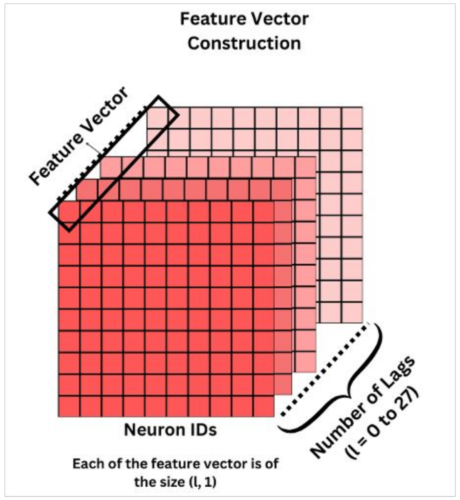
Node Features for Row and Column Graphs

The node feature matrix for lag l is given by:

$$F_l = \frac{I + I_l^T}{2}$$

where: I is the original image matrix; I_l^T is the transposed lagged version; The result is an element-wise average.

Feature Vectors:

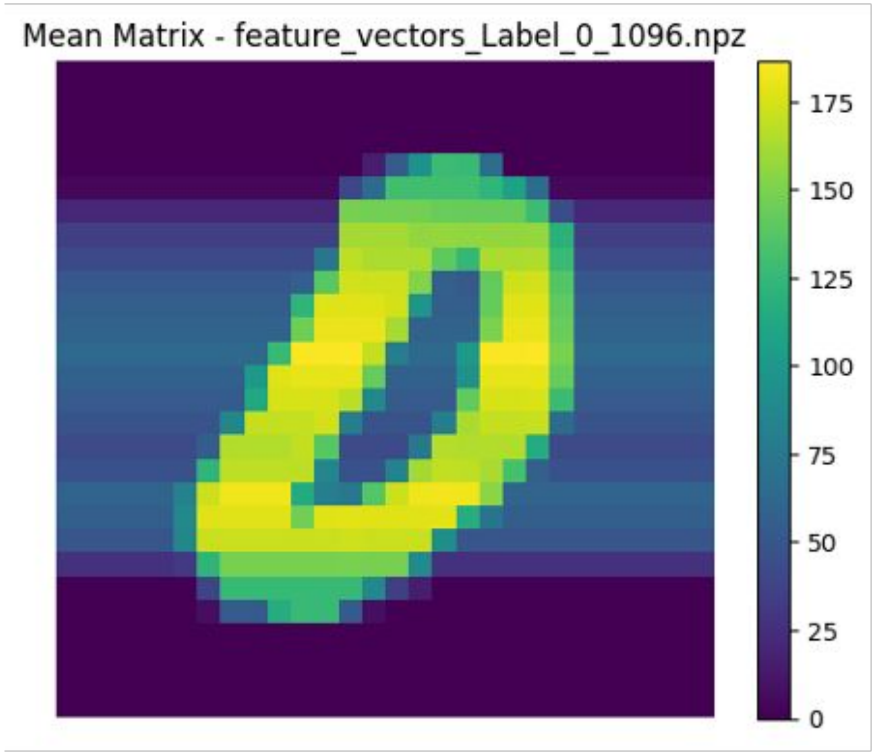

Feature Vector Construction
Feature Vector
Neuron IDs
Each of the feature vector is of the size (l, 1)
Number of Lags (l = 0 to 27)

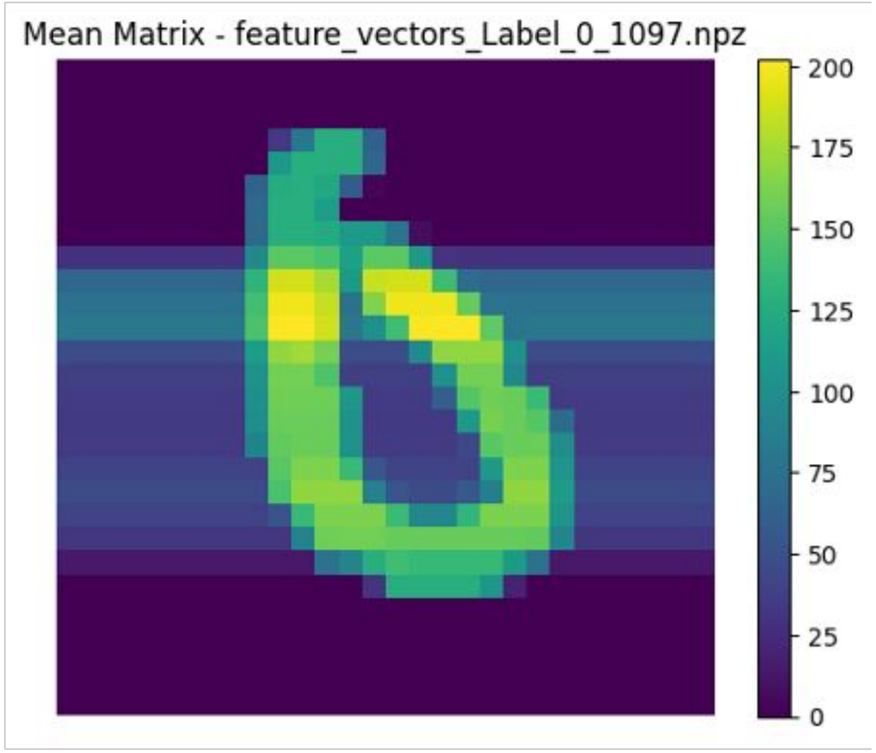
Node Feature Matrix :

$$\frac{1}{N}\sum_{l=0}^{27} F_l$$

Where N is the total number of lags we are taking $N = 28$

Node Features for Row and Column Graphs



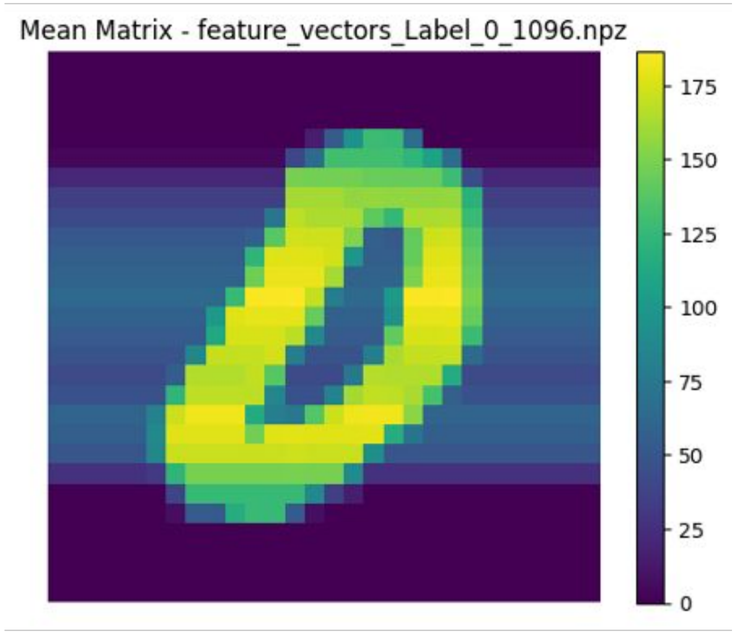


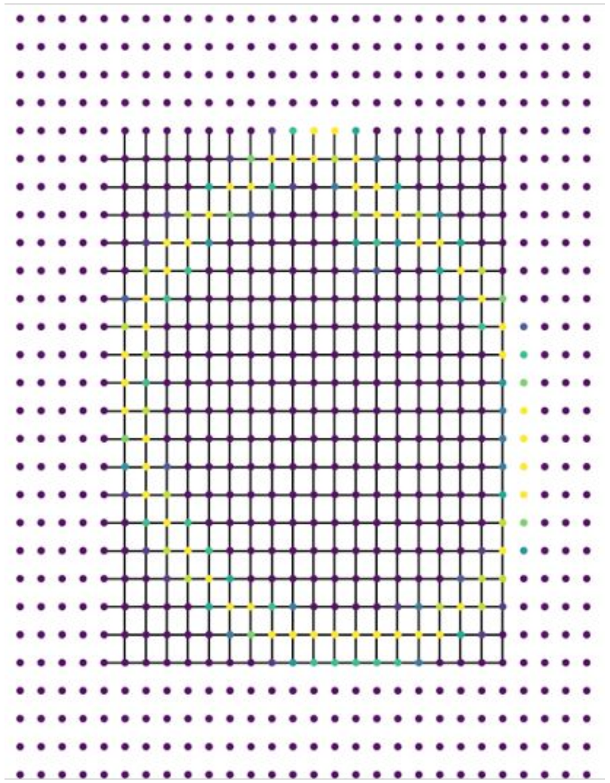
•Each row in the above image is fed as a feature vector to the nodes in the corresponding graph representation of the original image.

•i.e. each node as a node feature vector of size (28,1)

Similar process for column.

How do we map the (28,1) node feature for row and column to the 784 node Kronecker?





Why not just Kronecker the row and column feature matrices as well?

Results

Dataset Name	Grid Graph		Row Graph		Column Graph		Modified Cartesian Graph (Single Node Feature = Pixel Intensity)		Modified Cartesian Graph (4 Features/node, formed using modified cartesian method)		Modified Cartesian Graph (784 Features/node, formed using modified cartesian method)	
	GCN	GAT	GCN	GAT	GCN	GAT	GCN	GAT	GCN	GAT	GCN	GAT
MNIST	66.33	87.95	71.35	92.58	87.18	95.07	0.5505 ± 0.0289		0.7206 ± 0.0401	0.7818 ± 0.0006	0.9497 ± 0.002455	0.9691 ± 0.00016

1) GAT is better than GCN.
2) Kronecker does better than row/column, which does better than grid.
3) More useful features = better accuracy
4) More representative graph = better accuracy

Electrical & Computer ENGINEERING