




# Chair for Computer Aided Medical Procedures (CAMP) Master Seminar on Graph Deep Learning for Medical Applications

**IN2107 : Graph Deep Learning in Medical Imaging**

 Seminar Course

SoSe 2021  
Chair for Computer Aided Medical Procedures (CAMP)  
Preliminary meeting: Feb 8, 16:00  
Zoom link: Available on TUMonline  
<http://campar.in.tum.de/Chair/TeachingSs21GDLMA>

Image from NIH ChestX-ray8 dataset - [arxiv.org/abs/1705.02315](https://arxiv.org/abs/1705.02315)

Anees Kazi, Mahsa Ghorbani, Roger Mukul  
Prof. Dr. Nassir Navab

# Chair for Computer Aided Medical Procedures & Augmented Reality

Computer Vision

Robotics

Augmented Reality

AI for medicine



About tutors you will interact:



About me:



About me:



the  
tutors



# Team



**Anees Kazi : Senior Research Scientist,**  
[anees.kazi@tum.de](mailto:anees.kazi@tum.de)  
Research area: multi-modal data analysis, graph deep learning for healthcare



**Roger David Soberanis Mukul, Senior Research Scientist,** [roger.soberanis@tum.de](mailto:roger.soberanis@tum.de)  
Research area: CNN, GCN for medical applications, segmentation and localization



**Mahsa Ghorbani, PhD Candidate:**  
[mahsa.ghorbani@tum.de](mailto:mahsa.ghorbani@tum.de)  
Research area: Disease prediction, representation learning, multi-layered graphs



**Azade Farshad, PhD Candidate:**  
[azade.farshad@tum.de](mailto:azade.farshad@tum.de)  
Research area: Meta-learning, Scene graphs, Generative Models



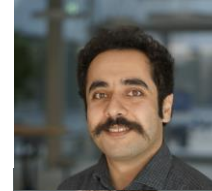
**Ashkan Khakzar, PhD Candidate:**  
[ashkan.khakzar@tum.de](mailto:ashkan.khakzar@tum.de)  
Research area: Explainable Machine Learning, Representation Learning



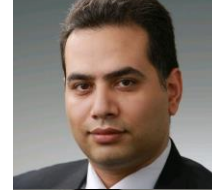
**Shahrooz Faghihroohi:**  
[shahrooz.faghihroohi@tum.de](mailto:shahrooz.faghihroohi@tum.de)  
Research area: Medical Image Reconstruction Spatio-Temporal Analysis, Generative Models



**Shun-Cheng Wu: PhD Candidate,**  
[shuncheng.wu@tum.de](mailto:shuncheng.wu@tum.de)  
Research area: semantic scene reconstruction and understanding.



**Ario Sadafi, PhD Candidate::**  
[ario.sadafi@helmholtz-muenchen.de](mailto:ario.sadafi@helmholtz-muenchen.de)  
Research area: Multiple instance learning, Active Learning, Single cell analysis



**Yousef Yeganeh, PhD Candidate:**  
[y.yeganeh@tum.de](mailto:y.yeganeh@tum.de)  
Research area: Transformers, Federated learning, OCT



**Timo Löhr, PhD Candidate:**  
[timo.loehr@tum.de](mailto:timo.loehr@tum.de)  
Research area: longitudinal disease prediction, graph learning, spatio-temporal analysis



**Mehrdad Salehi, PhD Candidate:**  
[mehrdad.salehi@tum.de](mailto:mehrdad.salehi@tum.de)  
Research areas: Image-Guided Interventions, DL in medical applications, ultrasound imaging



# Basic Info about the course

- **Type:** Master Seminar (IN2107)
- **Language:** English
- **SWS:** 2
- **ECTS:** 5 Credits
- **Webpage:**  
<http://campar.in.tum.de/Chair/TeachingSs21GDLM>  
A
- **Time:**
  - <https://wiki.tum.de/display/gdlma/GDLMA+SoSe21>
  - Every Tuesday 12:00 pm to 2:00 pm starting from 18.05.2021
- **Location:**
  - Virtual Meeting Room (Zoom)



# Evaluation:

## Presentation 45%

- 20 minutes + 10 minutes Q&A
- Slides (Powerpoint, Latex, see website for templates)
- They should cover all relevant aspects of the paper
  - Motivation
  - Methodology
  - Experimental results
  - Take Home Message
  - Discussion
- Self-contained (review of state of the art is necessary!)
- Presentation guidelines will be released later.
- **All students are expected to attend all presentations and interact during Q&A**

## Blog Post (45%)

- Blog post explaining the main ideas of the paper.
  - Motivation + Contributions
  - Methodology
  - Results & Discussion
- 1000-1200 words paper summary + 200-300 words your own review
- Students will be requested to comment on each other's blog posts.
- The website where the posts will be uploaded is [1].
- You can later privately share your blog posts in other websites as well (eg Medium).
- Upload the blog post two weeks before presentation. There will be discussion until presentation

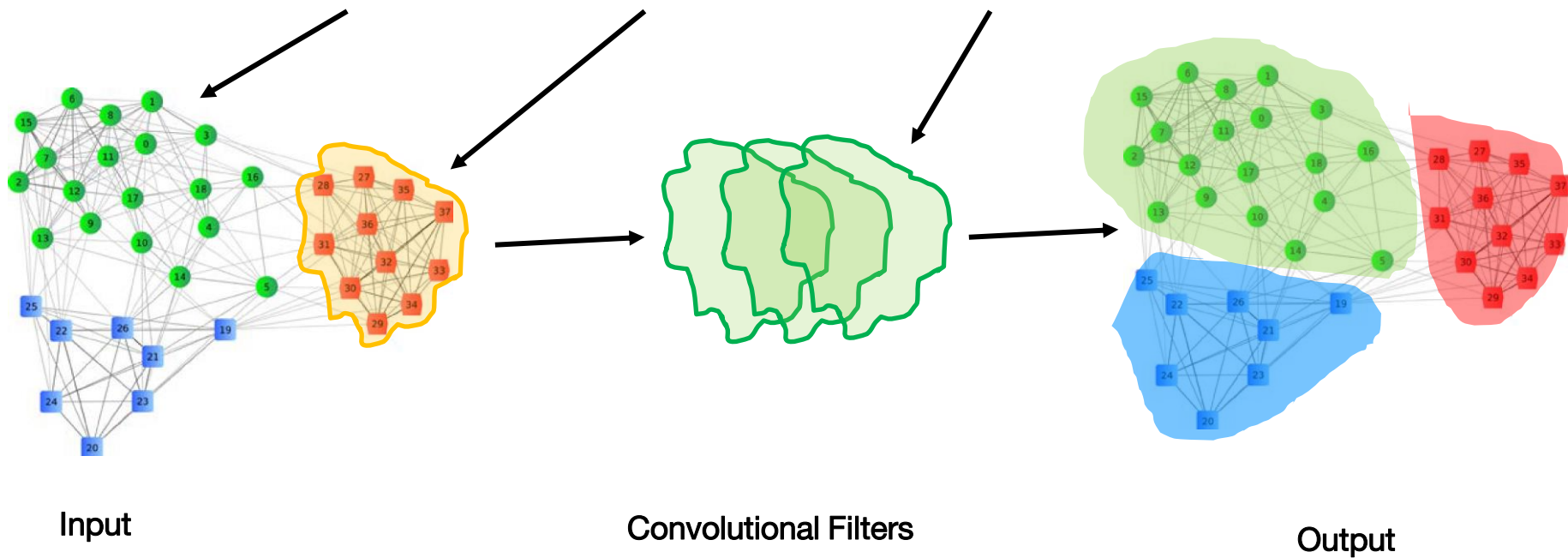
Attendance (10%)

Attendance (10%)

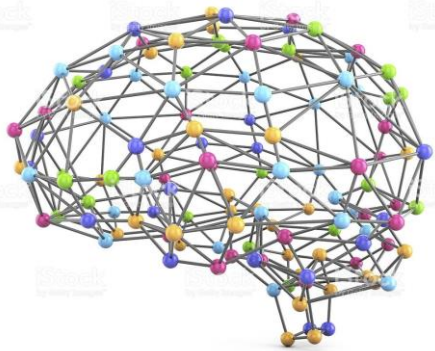




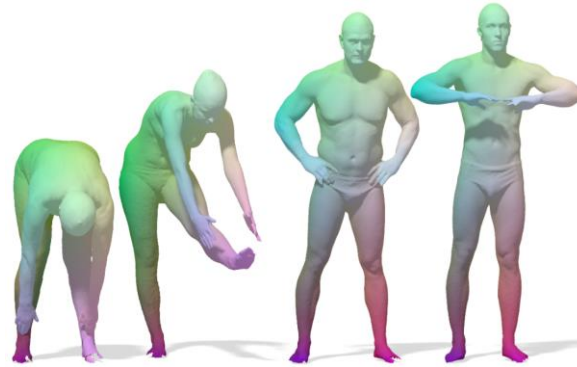
# Graph Convolutional Networks



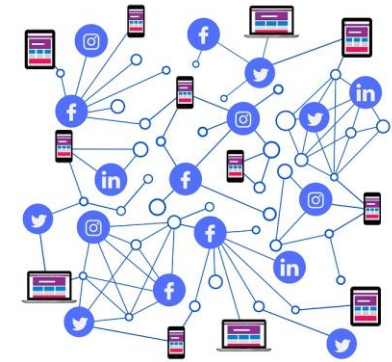
# Why Graph Structured Data?



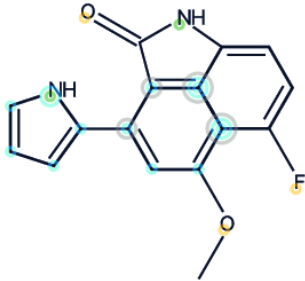
Brain connection



Shape analysis



Social network



Chemistry



Point cloud segmentation



Telecommunication

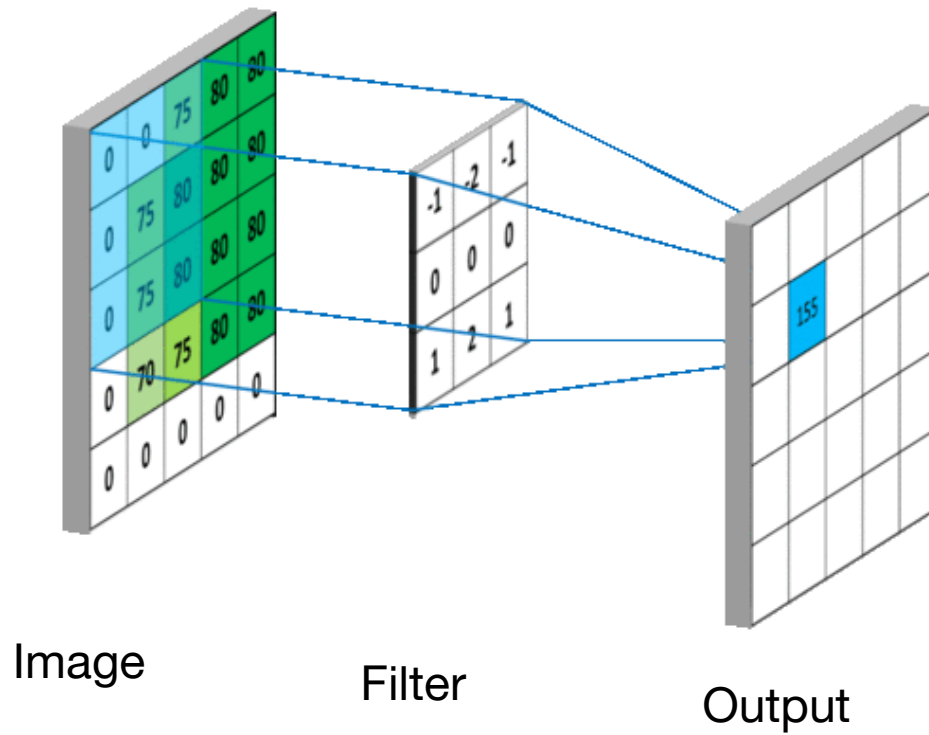


# Content

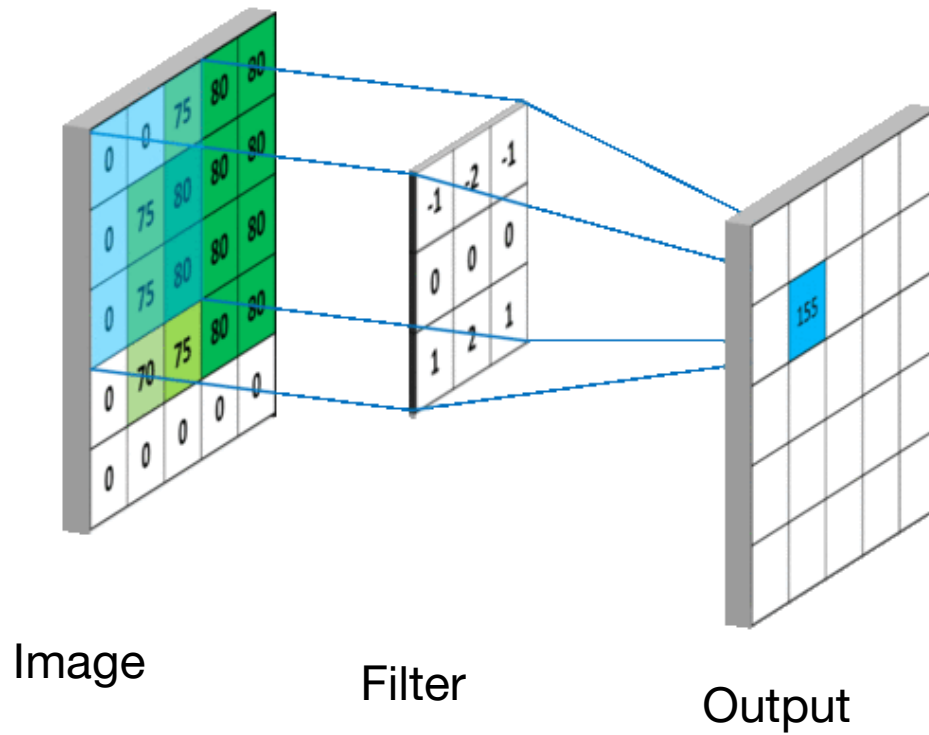
1. CNNs in a nutshell:
  - Working of CNN focusing on the input (grid structure), filters
2. Graphs
  - Graph structured data
  - Example of graph structured data
  - Problem in applying CNN to Graphs
  - Problems to be solved with GCNs
3. GCN in a nutshell
  - Block diagram of GCN
  - Type of GCNs
  - Explaining each block of GCN with the math
4. Drawbacks of GCNs
5. Applications of GCN
  - Computer vision
  - Medical
  - Brain
  - Chemistry
6. Challenges and future of GCNs



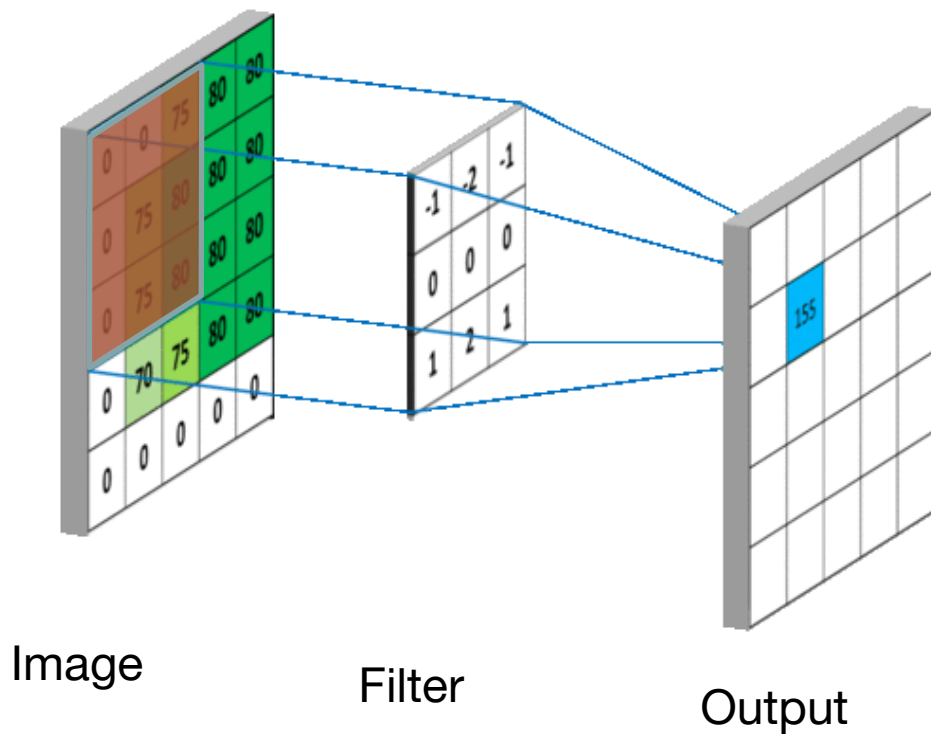
# Convolution in a nutshell



# Convolution in a nutshell



# Convolution in a nutshell



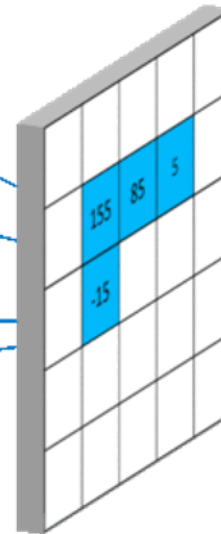
- Regular grid structure
  - Each pixel has exactly 8 neighbors
  - Distance of each pixel from the central pixel is constant
  - Output is also a grid
- Filters:
  - Regular grid structure
- Output
  - Regular grid



# Unstructured Input?



Will the grid shaped filter work?

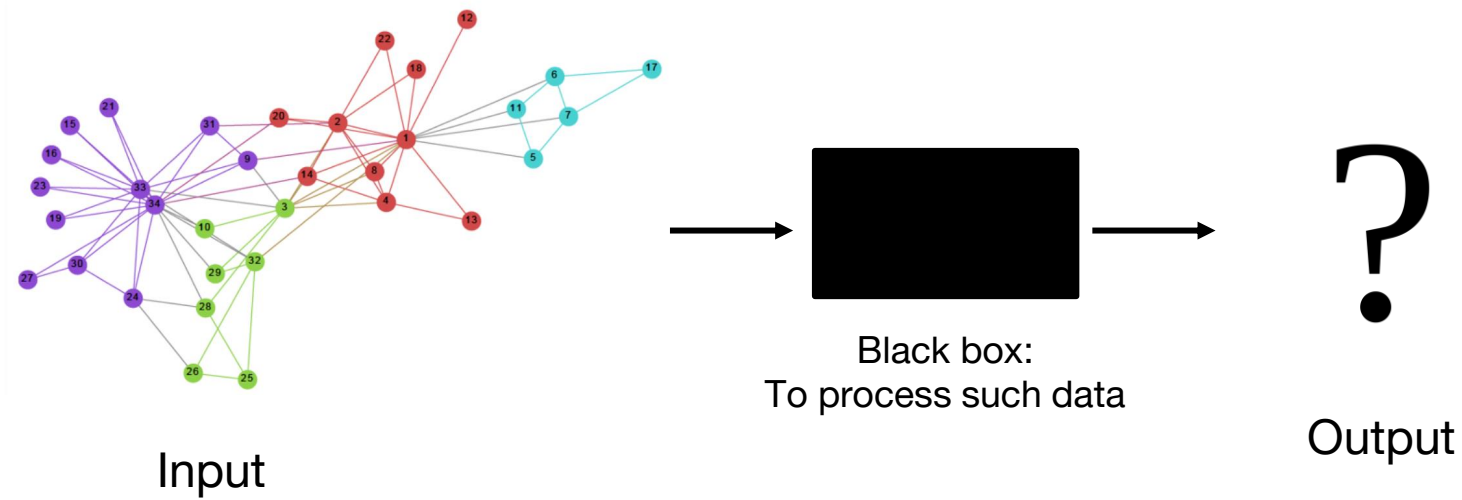


Will the output be a grid?

- No grid
- Number of neighbors are not fixed
- Distance between the nodes is not fixed

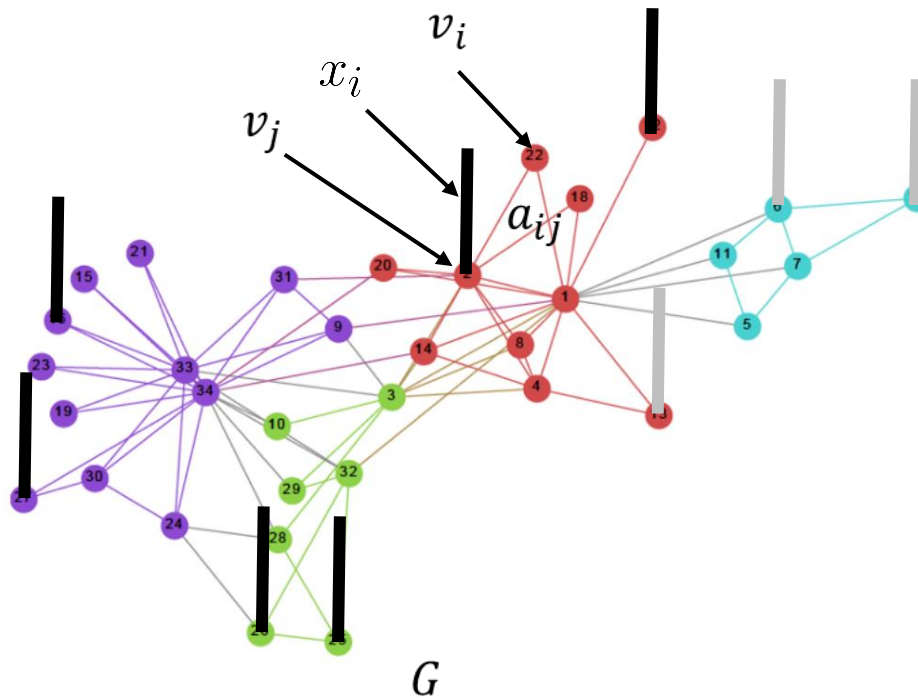


# How to Deal with Graph Structured Data?





# Lets find out the input setting



Notations:

Graph:  $G: \{V, A, W\}$

Node:  $v_i, v_j \in V$

Edge:  $a_{ij} \in A$

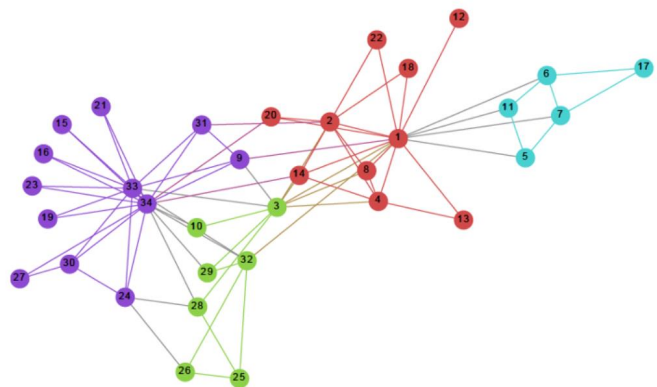
$A$  : Affinity matrix

$V$  : Set of nodes

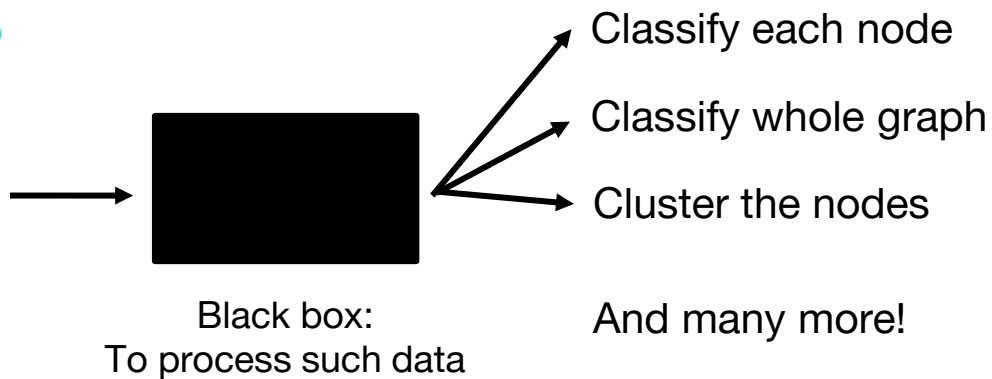
$W$ : Weight matrix

$x_i$  : Node feature vector

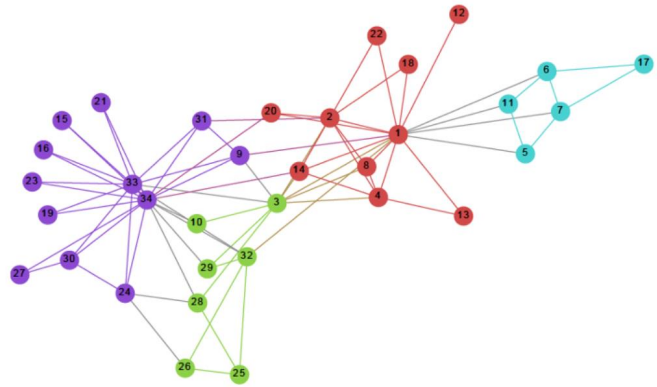
# What could be the possible outputs?



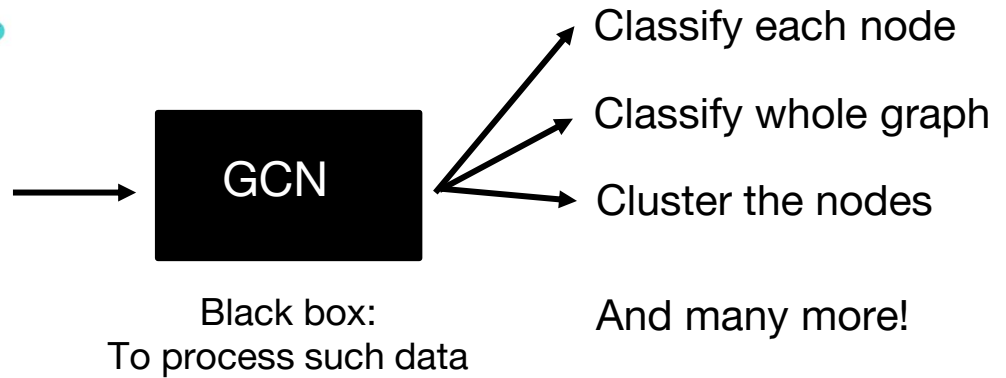
Input



# What could be the possible outputs?



Input



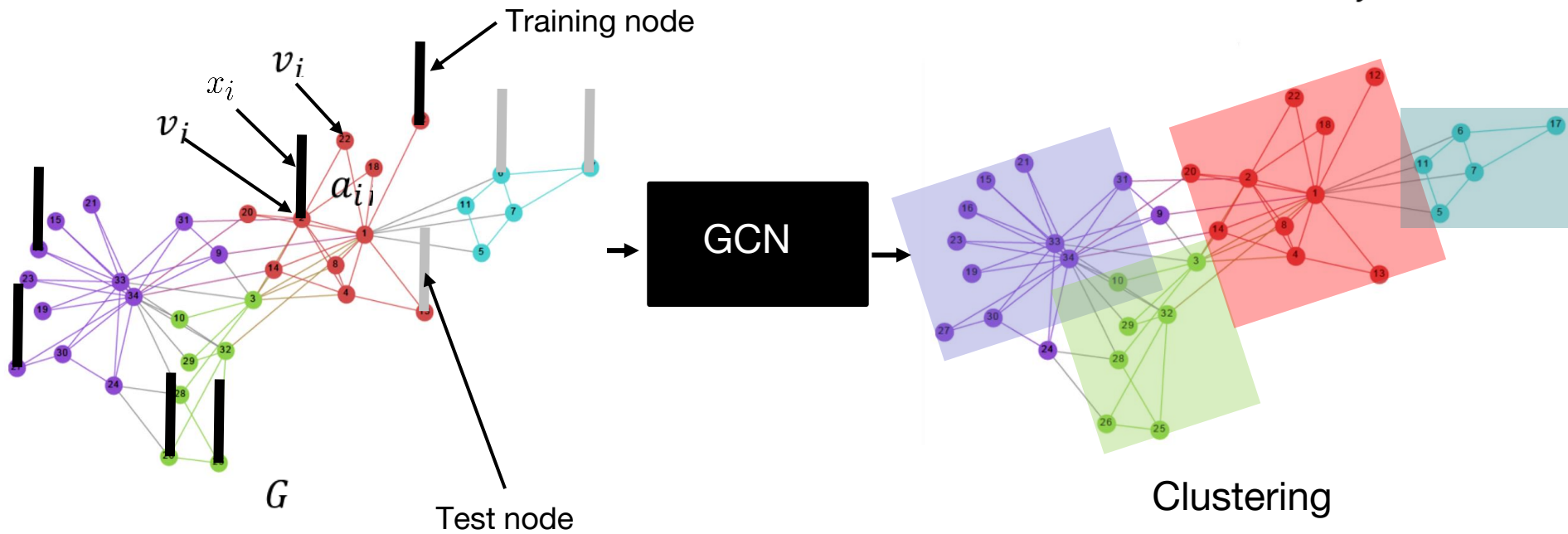
# Full pipeline

Notations:

Graph:  $G: \{V, E, W\}$

Node:  $v_i, v_j \in V$

Edge:  $e_{ij} \in E$

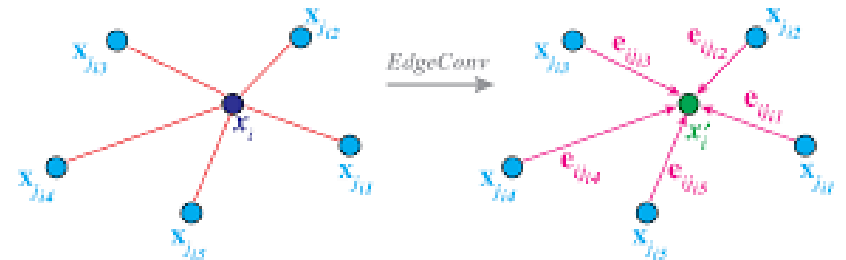
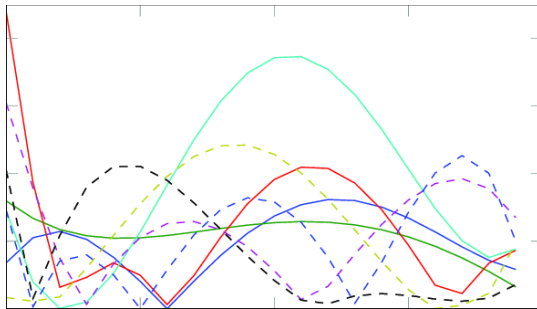


# Types of GCNs

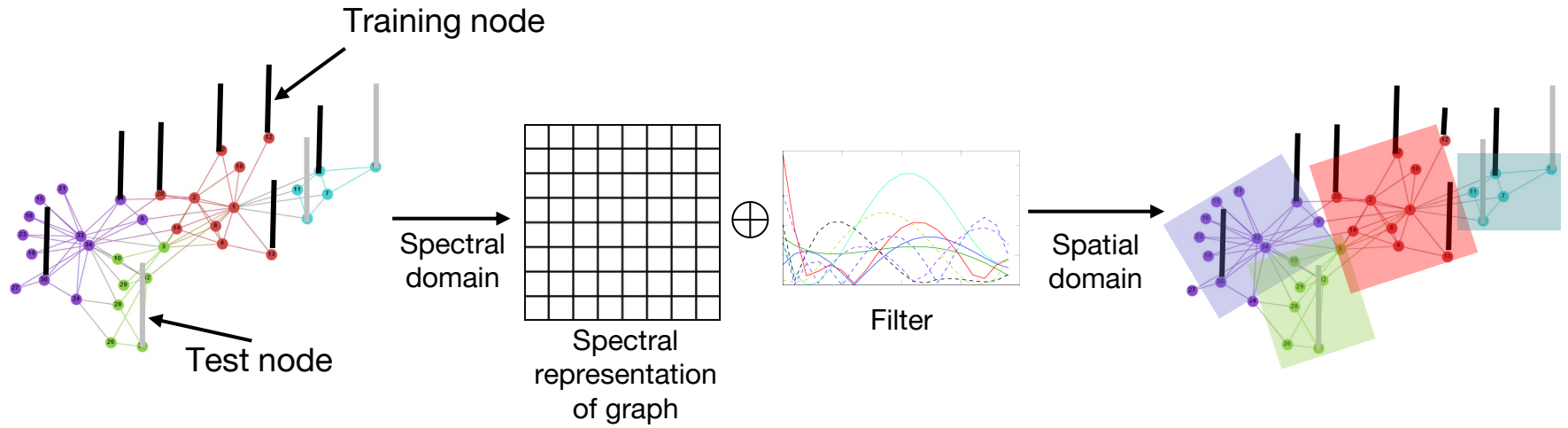
GCN

Spectral

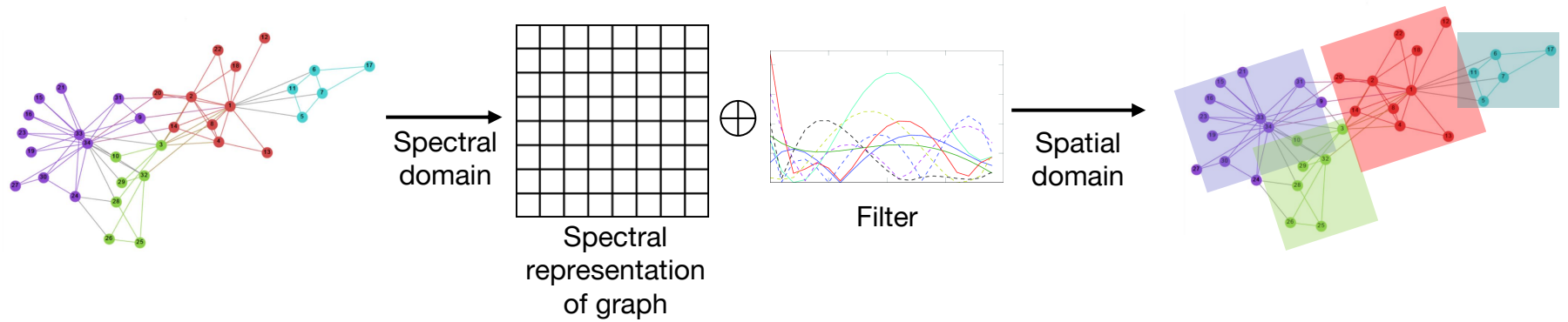
Spatial



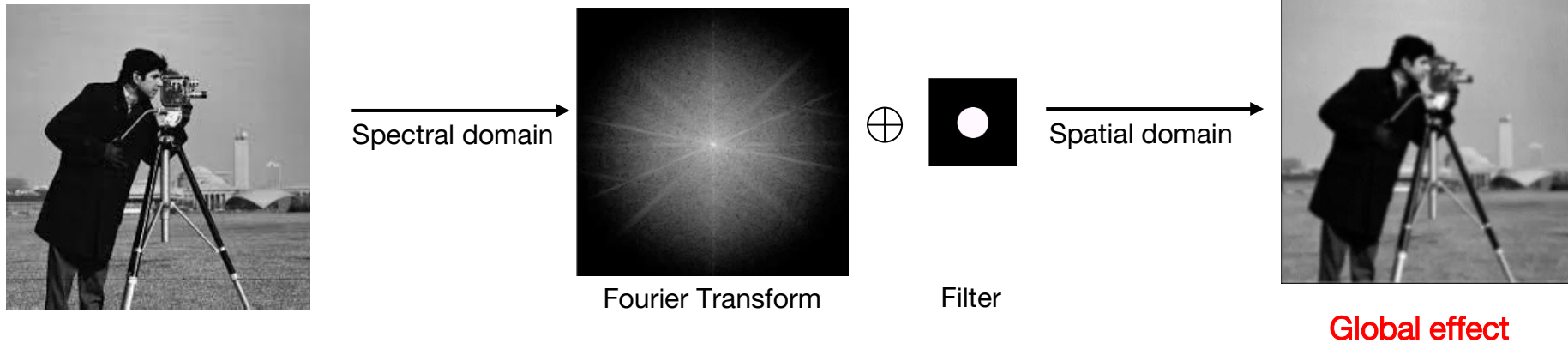
# GCN- spectral approach



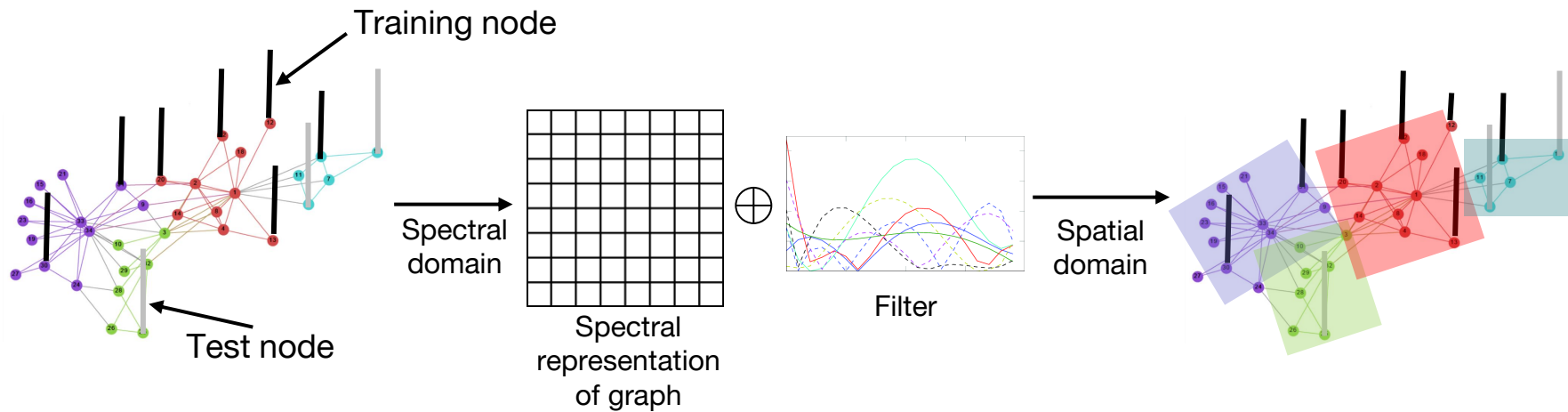
# GCN- spectral approach



# Regular image



# GCN- spectral approach



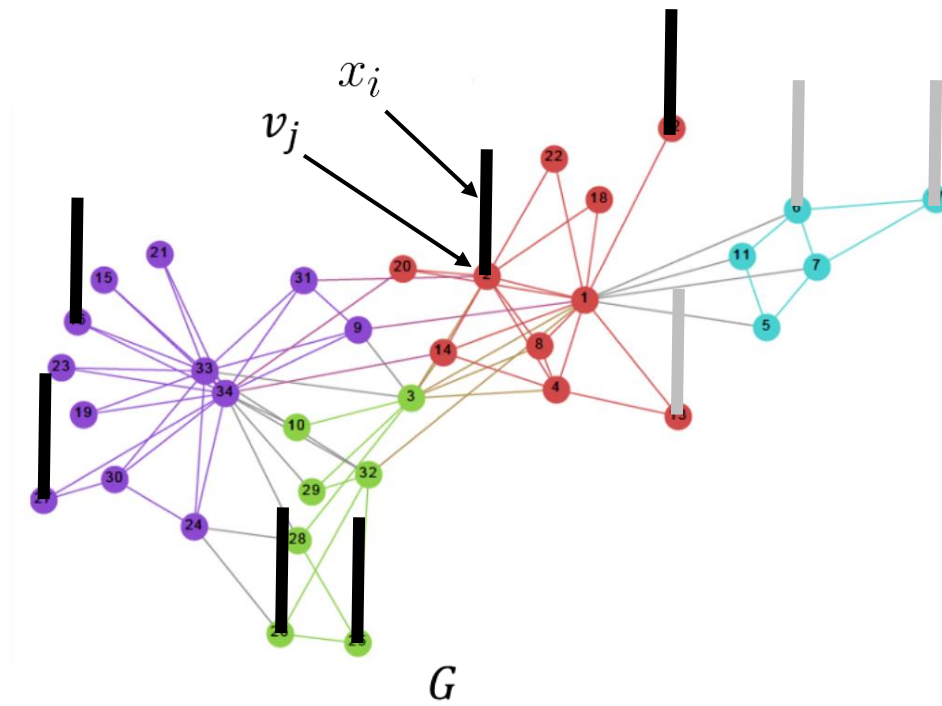
- Step 1: Compute normalized graph Laplacian
- Step 2: Eigen value decomposition
- Step 3: Convert node signals to spectral domain
- Step 4: Filtering
- Step 5: Inverse Fourier transform





# Two main components in the input setting

1. Graphs  $G \rightarrow A$
2. Node features  $x_i$



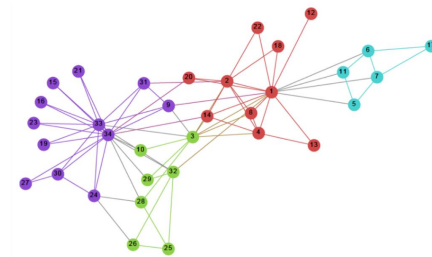
## Mathematical background

$$\text{Step 1: } L = I_N - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$$

Where,

$L$  is the normalized graph Laplacian.  $I_N$  is the identity showing the self loop and  $L \in \mathbb{R}^{N \times N}$

$D$  is the diagonal degree matrix,  $D_{ii} = \sum_j A_{ij}$  and  $D \in \mathbb{R}^{N \times N}$



$$\text{Step 2: Eigen value decomposition: } L = U\Lambda U^T$$

where,  $\Lambda = \text{diag}(\lambda_0, \lambda_1, \dots, \lambda_{N-1}) \in \mathbb{R}^{N \times N}$

---

Fourier transform of signal  $x$

$$\hat{x} = U^T x \in \mathbb{R}^N$$

Inverse Fourier transform of signal

$$x = U\hat{x} \in \mathbb{R}^N$$

Let us define a function  $g_\theta$  which brings learnable filters.

$$y = g_\Theta(L)x$$



# Mathematical background

$$\begin{aligned} y &= g_{\Theta}(L)x \\ y &= g_{\Theta}(U \Lambda U^T)x \\ &= U g_{\Theta}(\Lambda) U^T x \end{aligned}$$

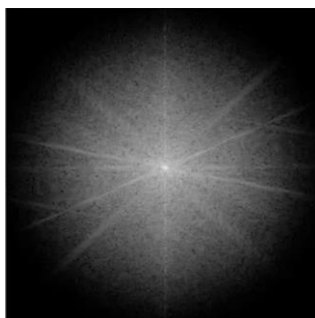
Step 3: Fourier Transform of  $x$

Analogy

Regular image



→  
Spectral domain



Fourier Transform



# Mathematical background

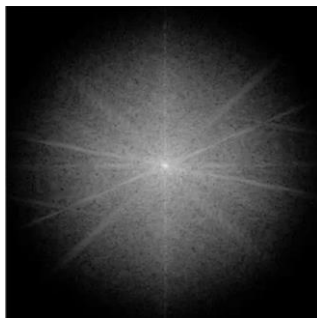
$$\begin{aligned} y &= g_{\Theta}(L)x \\ y &= g_{\Theta}(U\Lambda U^T)x \\ &= U g_{\Theta}(\Lambda) U^T x \end{aligned}$$

Step 4: Filtering of  $x$  in Fourier domain

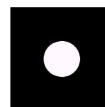
## Analogy Regular image



Spectral domain



Fourier Transform



Filter



# Mathematical background

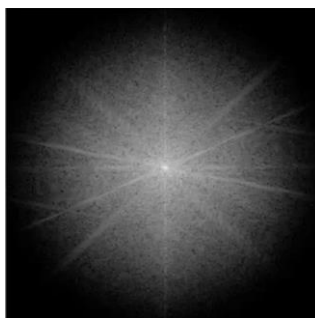
$$\begin{aligned} y &= g_{\Theta}(L)x \\ y &= g_{\Theta}(U \Lambda U^T)x \\ &= U g_{\Theta}(\Lambda) U^T x \end{aligned}$$

Step 5: Inverse Fourier Transform of  $x$

## Analogy Regular image



Spectral domain



Fourier Transform



Filter

Spatial domain



Global effect



## Mathematical background

$$\begin{aligned}y &= g_{\Theta}(L)x \\y &= g_{\Theta}(U \Lambda U^T)x \\&= \underbrace{U g_{\Theta}(\Lambda) U^T}_{\text{Computationally expensive}} x\end{aligned}$$

- Computationally expensive
- Not localized



# Mathematical background

$$\begin{aligned}y &= g_{\Theta}(L)x \\y &= g_{\Theta}(U \Lambda U^T)x \\&= \underbrace{U g_{\Theta}(\Lambda) U^T}_{\text{Computationally expensive}} x\end{aligned}$$

- Computationally expensive
- Not localized

So, polynomial parameterization over Lambda.

$$g_{\theta}(\Lambda) = \sum_{r=0}^k \theta_r T_r(\Lambda)$$

$$g_{\theta} * x = \sum_{r=0}^k \theta_r T_r(L) * x$$

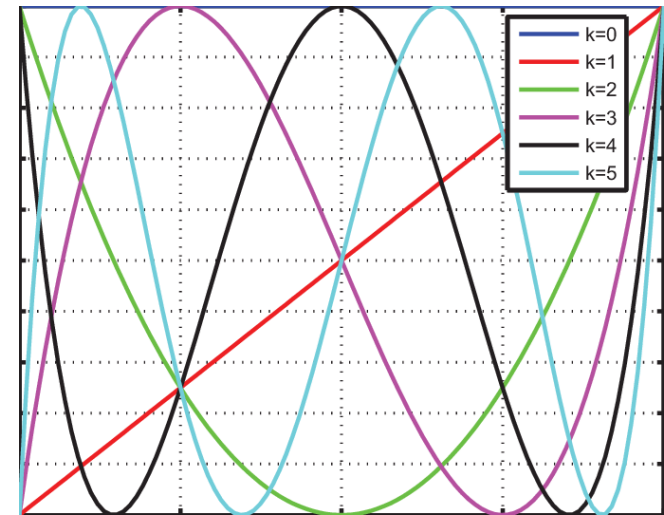
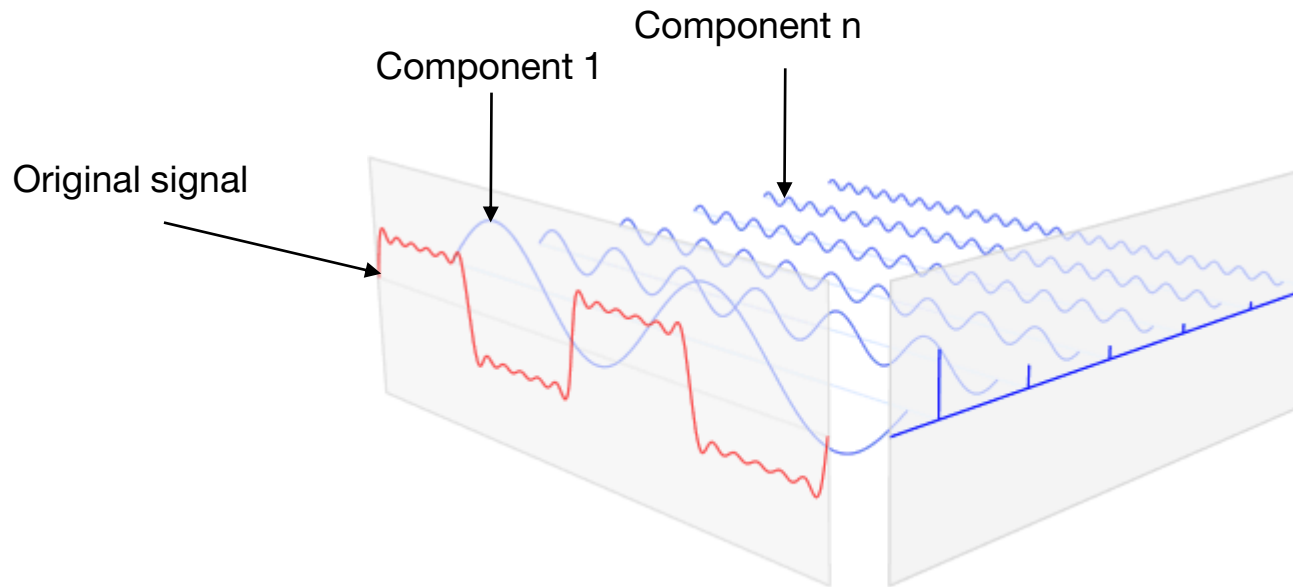


Fig. 1 Chebyshev Polynomials of the first kind [1]

[1] [https://www.researchgate.net/figure/Chebyshev-Polynomials-of-the-first-kind\\_fig1\\_287367586](https://www.researchgate.net/figure/Chebyshev-Polynomials-of-the-first-kind_fig1_287367586)

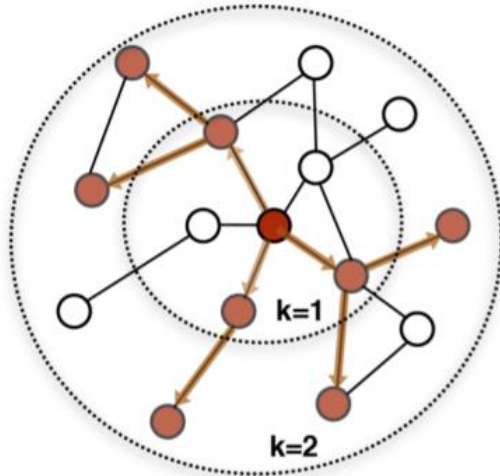
# Example with Fourier series



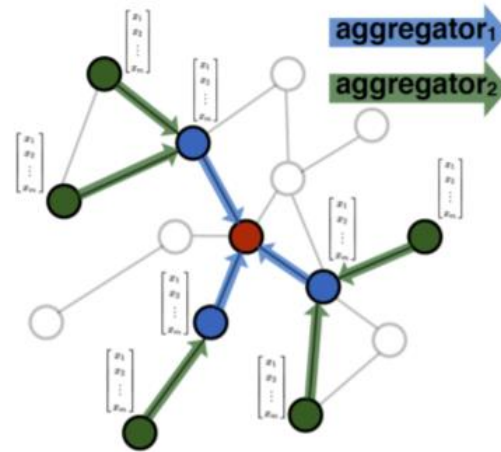
[1] Replicate the Fourier transform time-frequency domains correspondence illustration using TikZ



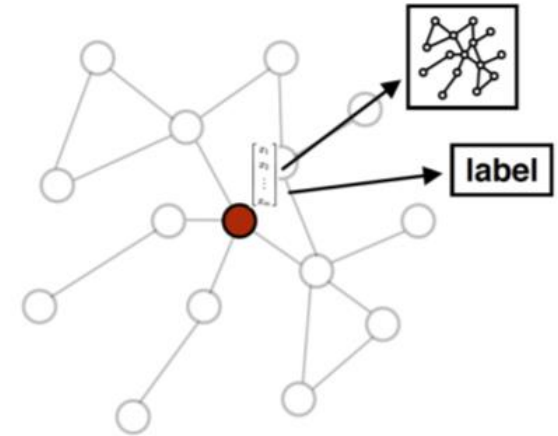
# Spatial GCN



1. Sample neighborhood



2. Aggregate feature information from neighbors



3. Predict graph context and label using aggregated information

sampling the number, with  $k$  being the hops

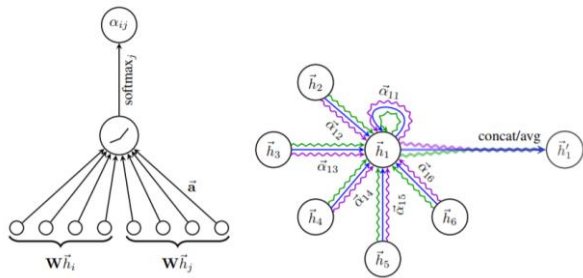
$$h_i^{(l+1)} = \sigma \left( h_i^{(l)} w_0^{(l)} + \sum_{j \in \eta} \frac{1}{c_{ij}} w_{(l)}^1 h_j^{(l)} \right)$$

## Drawbacks of GCNs

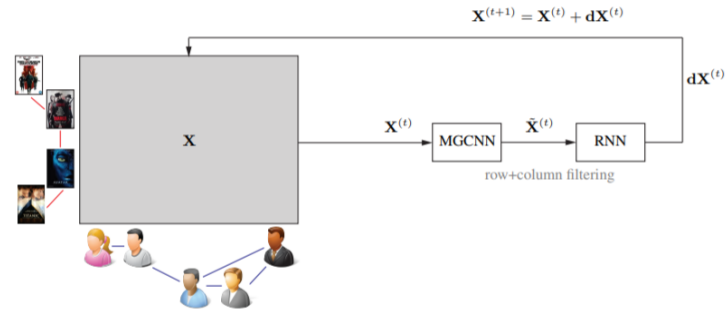
1. A GCN trained on one graph will not work on another graph structured data.
2. Spectral versions require predefined graph.
3. At all levels in this network, the filters are limited to 3x3 in size (which in it self is still OK) and are also essentially **fixed to be the same kernel across all layers and all units in entire network**, up to a constant multiplier.
4. Graph construction is critical.
5. Graph learning is still an open challenge.



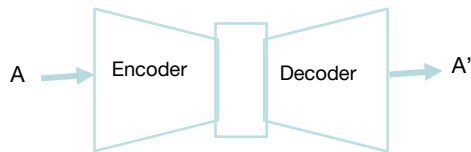
# Methodological advancements:



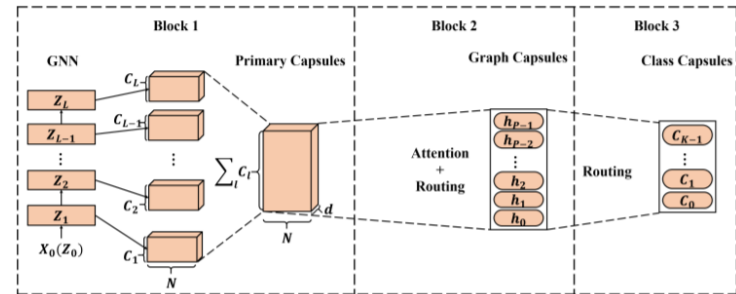
Graph attention networks [1]



Matrix completion [2]



Graph auto encoders [4]

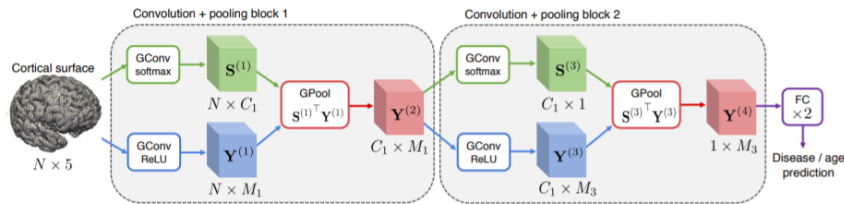


Capsule Graph Neural Networks [5]

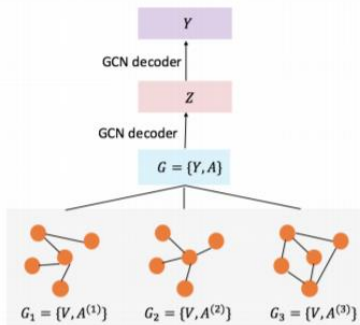
[1] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P. and Bengio, Y., 2017. Graph attention networks ICLR 2018.  
 [2] Monti, F., Bronstein, M. and Bresson, X., 2017. Geometric matrix completion with recurrent multi-graph neural networks. In *Advances in Neural Information Processing Systems* (pp. 3697-3707).  
 [3] Klicpera, J., Bojchevski, A. and Günnemann, S., 2018. Predict then Propagate: Graph Neural Networks meet Personalized PageRank, ICLR 2019.  
 [4] Kipf, T.N. and Welling, M., 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.  
 [5] Verma, S. and Zhang, Z.L., 2018. Graph capsule convolutional neural networks. ICLR 2019]

# Application based advancements

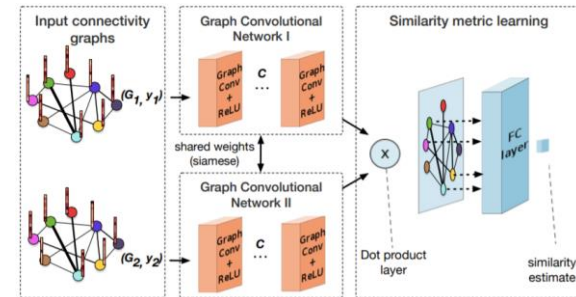
- Medical



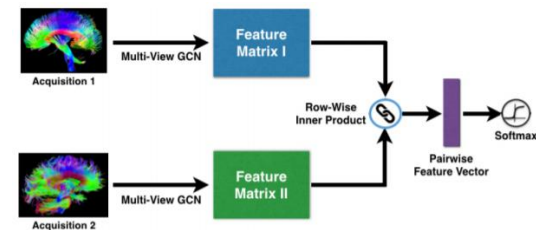
Brain surface analysis [1]



Drug similarity integration [4]



Metric Learning for brain connectivity analysis [2]



Multi-view GCN for [5]

[1] Gopinath, K., Desrosiers, C. and Lombaert, H., Adaptive Graph Convolution Pooling for Brain Surface Analysis. Accepted in IPMI 2019.

[2] Ktena, S.I., Parisot, S., Ferrante, E., Rajchl, M., Lee, M., Glocker, B. and Rueckert, D., 2018. Metric learning with spectral graph convolutions on brain connectivity networks. *NeuroImage*, 169, pp.431-442.

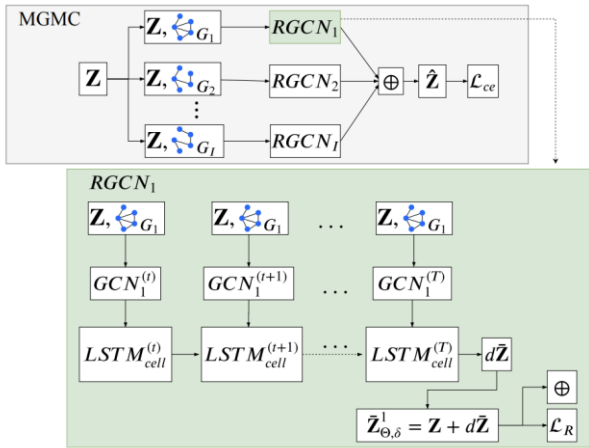
[3] Anirudh, R. and Thiagarajan, J.J., 2019, April. Bootstrapping graph convolutional neural networks for autism spectrum disorder classification. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 3197-3201). IEEE.

[4] Ma, T., Xiao, C., Zhou, J. and Wang, F., 2018. Drug similarity integration through attentive multi-view graph auto-encoders. *arXiv preprint arXiv:1804.10850*.

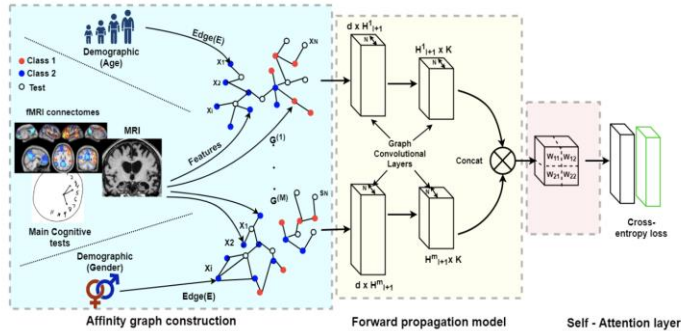
[5] Zhang, X., He, L., Chen, K., Luo, Y., Zhou, J. and Wang, F., 2018. Multi-view graph convolutional network and its applications on neuroimage analysis for parkinson's disease. In *AMIA Annual Symposium Proceedings* (Vol. 2018, p. 1147). American Medical Informatics Association.



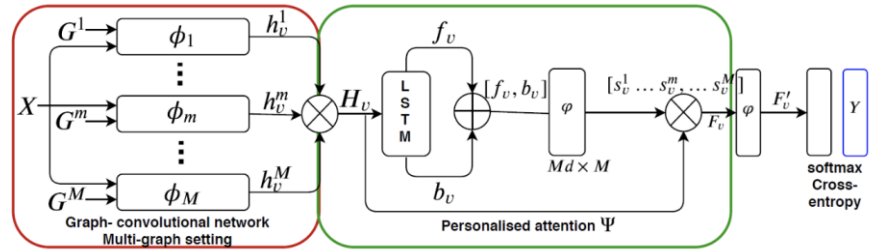
# Works from CAMP!



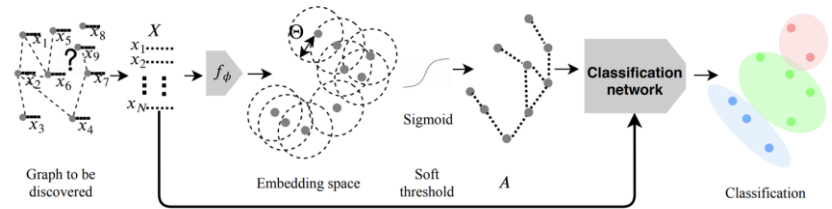
1) Matrix completion



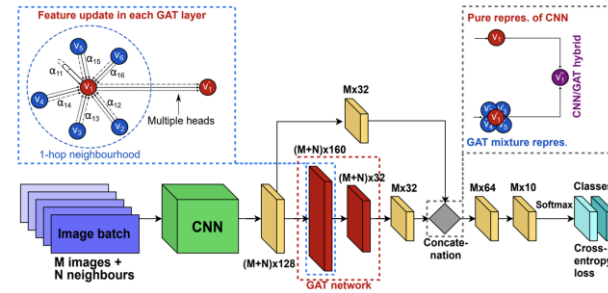
2) Multiple graph scenario



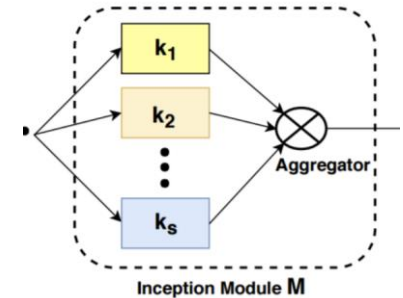
3) Personalised medicine



4) Graph learning



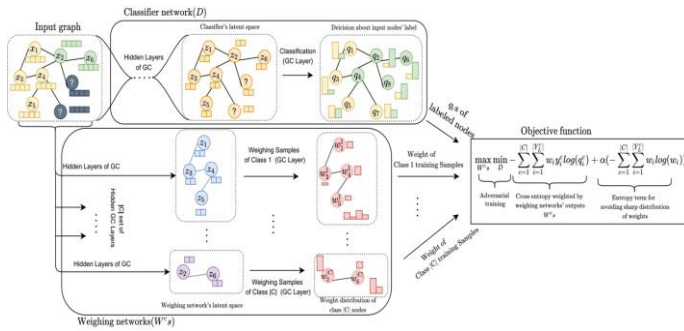
5) CNN GAT



6) inceptionGCN

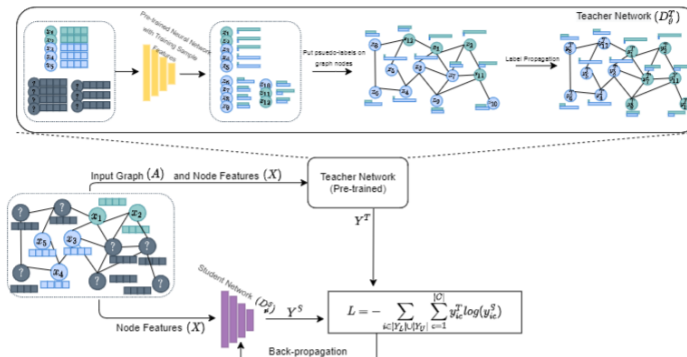
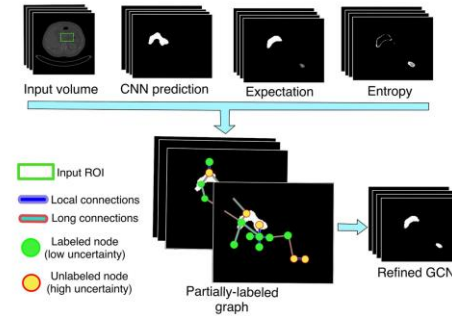


# State of the art of CAMP



## An Uncertainty-Driven GCN Refinement Strategy for Organ Segmentation

The work employs the volume's intensities, and CNN's prediction, expectation, and entropy to formulate the refinement problem as a semi-supervised graph learning problem



[1] Roger D. Soberanis-Mukul, Nassir Navab, and Shadi Albarqouni. An Uncertainty-Driven GCN Refinement Strategy for Organ Segmentation. MELBA ,2020

[2]Ghorbani, Mahsa, et al. "RA-GCN: Graph Convolutional Network for Disease Prediction Problems with Imbalanced Data." *arXiv preprint arXiv:2103.00221* (2021).

[3] Ghorbani, Mahsa, et al. "GKD: Semi-supervised Graph Knowledge Distillation for Graph-Independent Inference." *arXiv preprint arXiv:2104.03597* (2021).



# Future challenges

1. Scalability in spectral.
2. Modeling dynamic, temporal graphs
3. Robustness
4. Interpretability



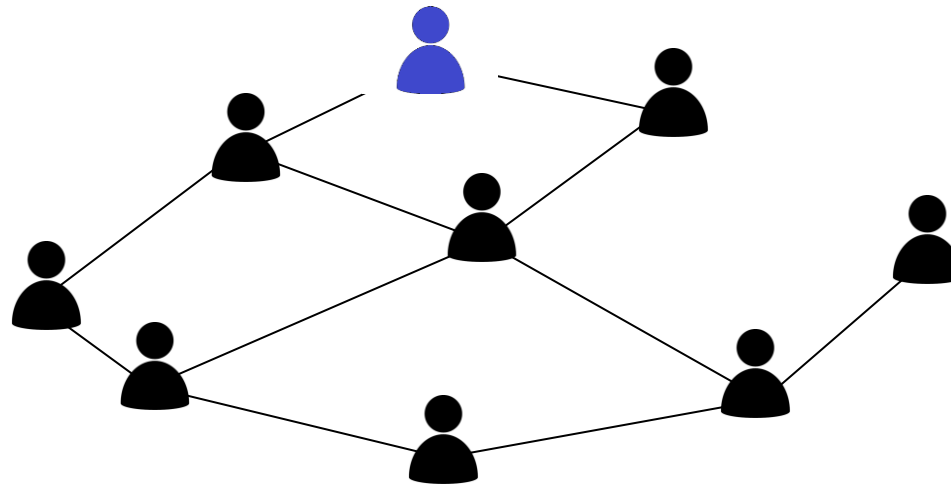
# Interesting tutorials to get some hand on!

- <https://github.com/tkipf/gcn>





# Difference between transductive and inductive methods



## Inductive:

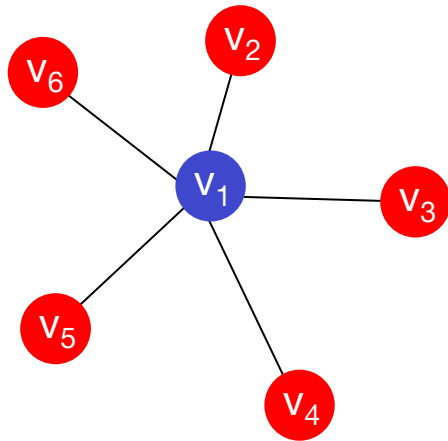
- The method works **directly on the graph** and learns the important relations based on the feature representations
- An extension to new samples is therefore possible **without retraining the network**
- The method is scalable because it can be trained **successively on different parts** of the graph structure (batchwise approach)



# General idea of inductive graph approach

## General idea:

Feature representation of one node gets updated by surrounding nodes



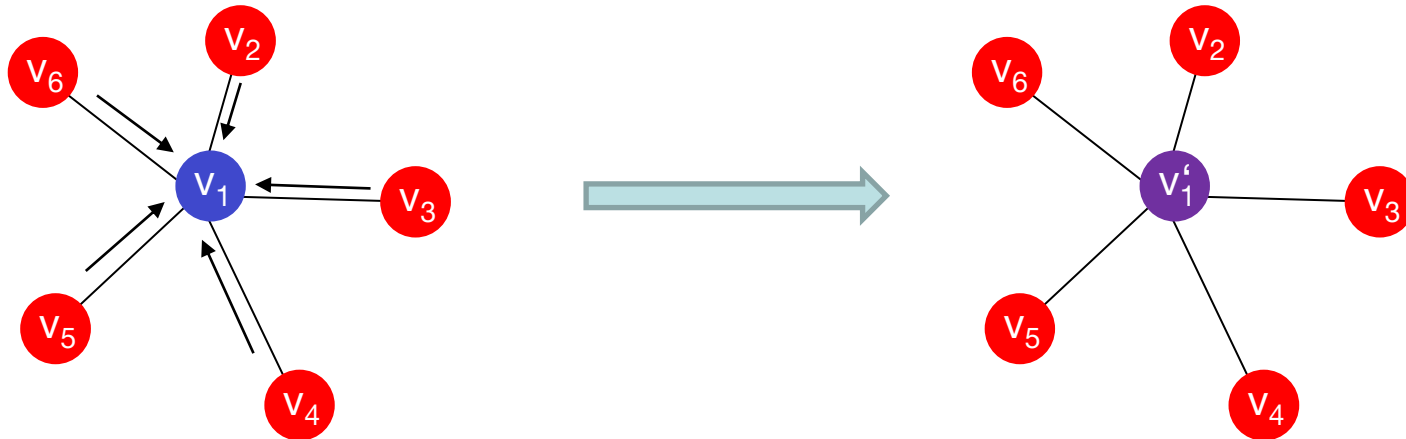
Node 1 with feature representation  $v_1$  has connections to five corresponding neighbors



# General idea of inductive graph approach

## General idea:

Feature representation of one node gets updated by surrounding nodes



Node 1 with feature representation  $v_1$  has connections to five corresponding neighbors

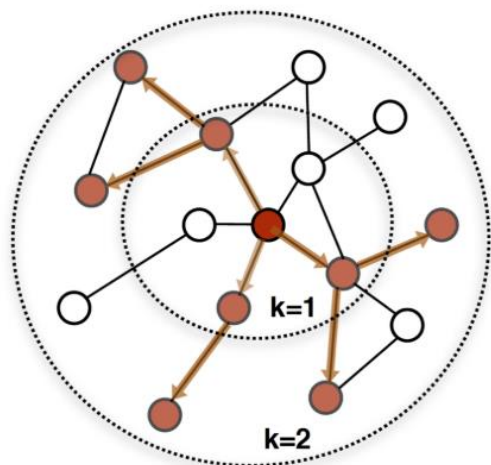
Surrounding nodes features are aggregated and result in new representation  $v'_1$



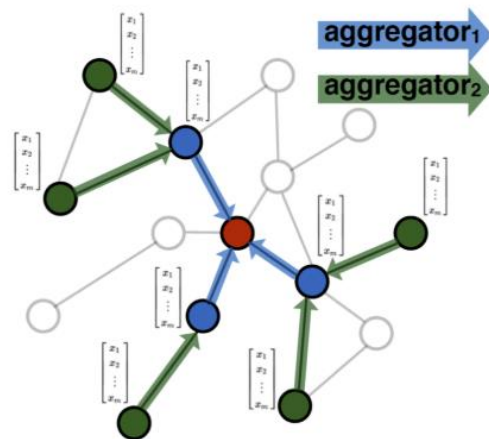
# Example 1: GraphSage

General idea:

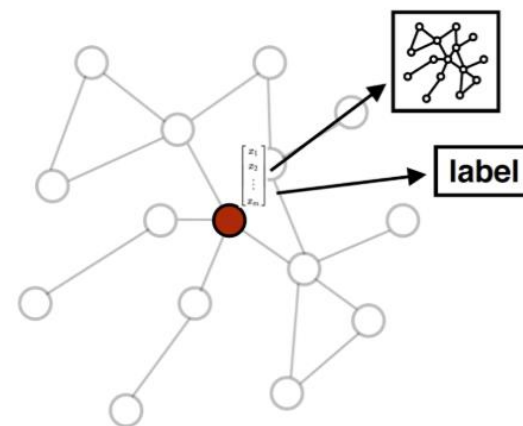
Feature representation of one node gets updated by surrounding nodes



1. Sample neighborhood



2. Aggregate feature information from neighbors



3. Predict graph context and label using aggregated information



[1] W. L. Hamilton, R. Ying, J. Leskovec, Inductive Representation Learning on Large Graphs, NIPS 2017

# Example 1: GraphSage

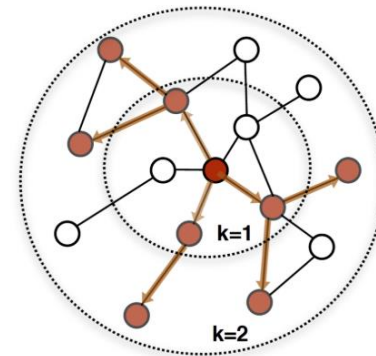
## Mathematical realization:

```
1  $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$ ;
2 for  $k = 1 \dots K$  do
3   for  $v \in \mathcal{V}$  do
4      $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$ ;
5      $\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k))$ 
6   end
7    $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$ 
8 end
9  $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$ 
```

## Concept steps for each node:

## Parameters

- Graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$
- weight matrices  $\mathbf{W}^k, \forall k \in \{1, \dots, K\}$
- input features  $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$
- depth  $K$



1. Sample neighborhood



# Example 1: GraphSage

## Mathematical realization:

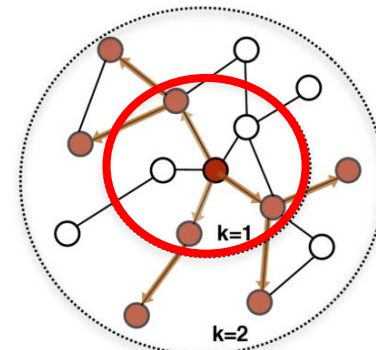
```
1  $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$  ;
2 for  $k = 1 \dots K$  do
3   for  $v \in \mathcal{V}$  do
4      $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$ ;
5      $\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k))$ 
6   end
7    $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$ 
8 end
9  $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$ 
```

## Concept steps for each node:

1. Collect all nodes which are in the **direct neighborhood** of the node of interest

## Parameters

Graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$   
weight matrices  $\mathbf{W}^k, \forall k \in \{1, \dots, K\}$   
input features  $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$   
depth  $K$



1. Sample neighborhood

[1] W. L. Hamilton, R. Ying, J. Leskovec, Inductive Representation Learning on Large Graphs, NIPS 2017

# Example 1: GraphSage

## Mathematical realization:

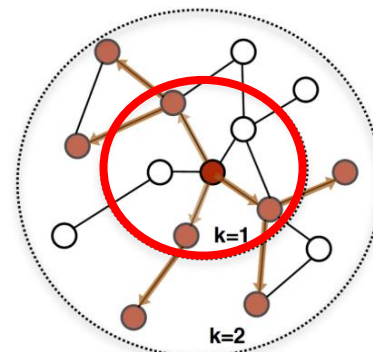
```
1  $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$ ;
2 for  $k = 1 \dots K$  do
3   for  $v \in \mathcal{V}$  do
4      $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$ ;
5      $\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k))$ 
6   end
7    $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$ 
8 end
9  $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$ 
```

## Concept steps for each node:

1. Collect all nodes which are in the **direct neighborhood** of the node of interest
2. Aggregate these nodes using an **aggregation function** (averaging, pooling, ...)

## Parameters

Graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$   
weight matrices  $\mathbf{W}^k, \forall k \in \{1, \dots, K\}$   
input features  $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$   
depth  $K$



1. Sample neighborhood



# Example 1: GraphSage

## Mathematical realization:

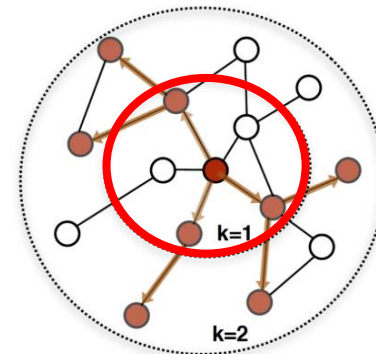
```
1  $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$ ;
2 for  $k = 1 \dots K$  do
3   for  $v \in \mathcal{V}$  do
4      $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$ ;
5      $\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k))$ 
6   end
7    $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$ 
8 end
9  $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$ 
```

## Concept steps for each node:

1. Collect all nodes which are in the **direct neighborhood** of the node of interest
2. Aggregate these nodes using an **aggregation function** (averaging, pooling, ...)
3. Concatenate the aggregated feature representation with the features of the central node

## Parameters

Graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$   
weight matrices  $\mathbf{W}^k, \forall k \in \{1, \dots, K\}$   
input features  $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$   
depth  $K$



1. Sample neighborhood





# Example 1: GraphSage

## Mathematical realization:

```
1  $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$ ;
2 for  $k = 1 \dots K$  do
3   for  $v \in \mathcal{V}$  do
4      $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$ ;
5      $\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k))$ 
6   end
7    $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$ 
8 end
9  $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$ 
```

## Concept steps for each node:

1. Collect all nodes which are in the **direct neighborhood** of the node of interest
2. Aggregate these nodes using an **aggregation function** (averaging, pooling, ...)
3. Concatenate the aggregated feature representation with the features of the central node
4. Perform a **linear transformation** of the concatenated vector to receive the new representation

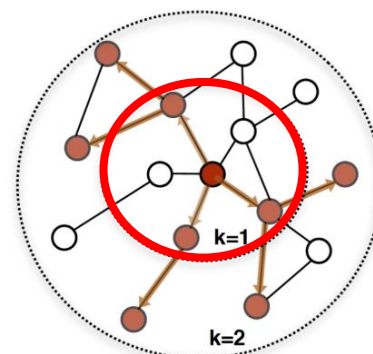
## Parameters

Graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$

weight matrices  $\mathbf{W}^k, \forall k \in \{1, \dots, K\}$

input features  $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$

depth  $K$



1. Sample neighborhood

[1] W. L. Hamilton, R. Ying, J. Leskovec, Inductive Representation Learning on Large Graphs, NIPS 2017



# Example 1: GraphSage

## Mathematical realization:

```
1  $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$ ;
2 for  $k = 1 \dots K$  do
3   for  $v \in \mathcal{V}$  do
4      $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$ ;
5      $\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k))$ 
6   end
7    $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$ 
8 end
9  $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$ 
```

## Concept steps for each node:

1. Collect all nodes which are in the **direct neighborhood** of the node of interest
2. Aggregate these nodes using an **aggregation function** (averaging, pooling, ...)
3. Concatenate the aggregated feature representation with the features of the central node
4. Perform a **linear transformation** of the concatenated vector to receive the new representation
5. **Activate and normalize** the new representation

[1] W. L. Hamilton, R. Ying, J. Leskovec, Inductive Representation Learning on Large Graphs, NIPS 2017

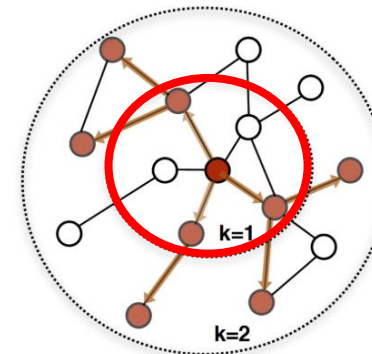
## Parameters

Graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$

weight matrices  $\mathbf{W}^k, \forall k \in \{1, \dots, K\}$

input features  $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$

depth  $K$



1. Sample neighborhood



# Example 1: GraphSage

## Mathematical realization:

```
1  $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$ ;
2 for  $k = 1 \dots K$  do
3   for  $v \in \mathcal{V}$  do
4      $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$ ;
5      $\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k))$ 
6   end
7    $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$ 
8 end
9  $\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$ 
```

## Properties:

1. The amount of samples and neighborhood size can be varied, batchwise training is possible
2. The trained network is applicable also for new samples that were not part of the training
3. **But:** The network does not select which neighbors are the most important ones for an update

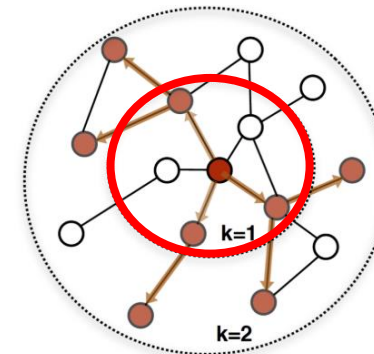
## Parameters

Graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$

weight matrices  $\mathbf{W}^k, \forall k \in \{1, \dots, K\}$

input features  $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$

depth  $K$



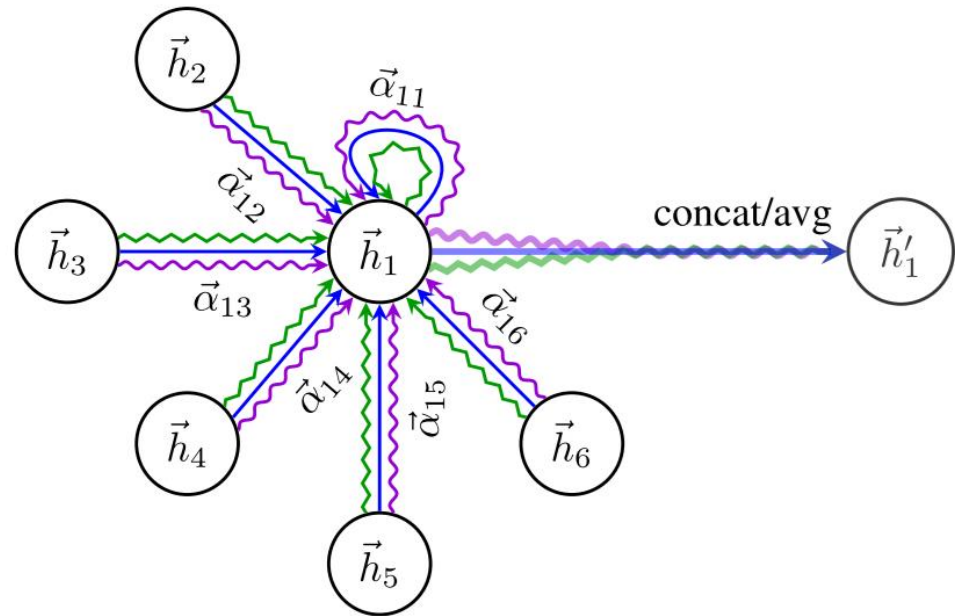
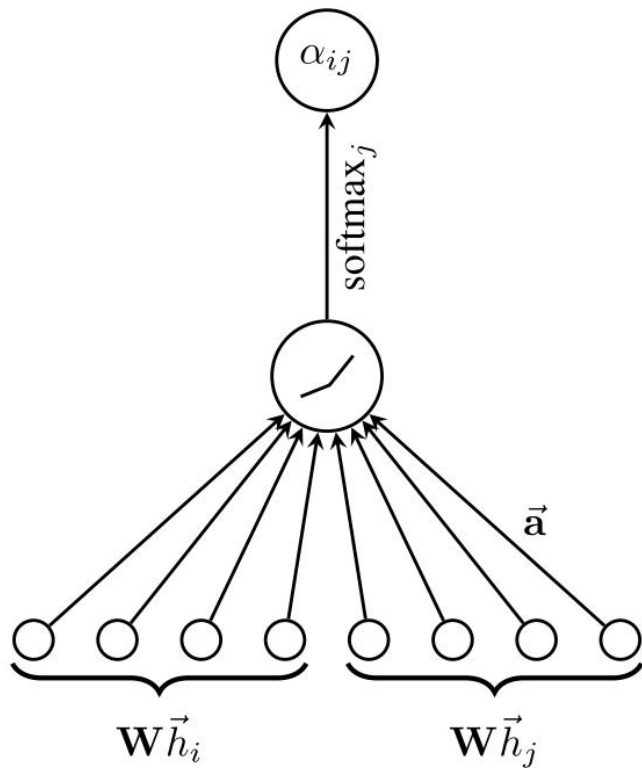
1. Sample neighborhood

[1] W. L. Hamilton, R. Ying, J. Leskovec, Inductive Representation Learning on Large Graphs, NIPS 2017

## Example 2: Graph Attention Network (GAT)

### General idea:

Network learns which neighbors are the most important for the update (attention mechanism)



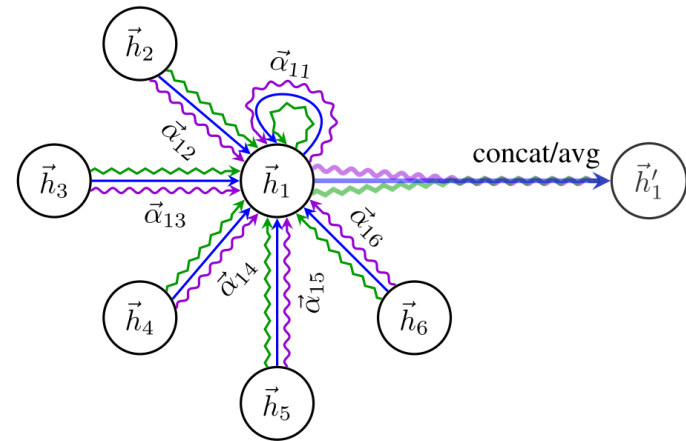
[1] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio and Y. Bengio, 2017. Graph attention networks ICLR 2018

## Example 2: Graph Attention Network (GAT)

### Mathematical realization:

An attention coefficient is calculated for each connection between a node  $j$  and the central node  $i$  using a shared linear transformation  $\mathbf{W}$  and an attention function  $a$ :

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$$



[1] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio and Y. Bengio, 2017. Graph attention networks ICLR 2018

## Example 2: Graph Attention Network (GAT)

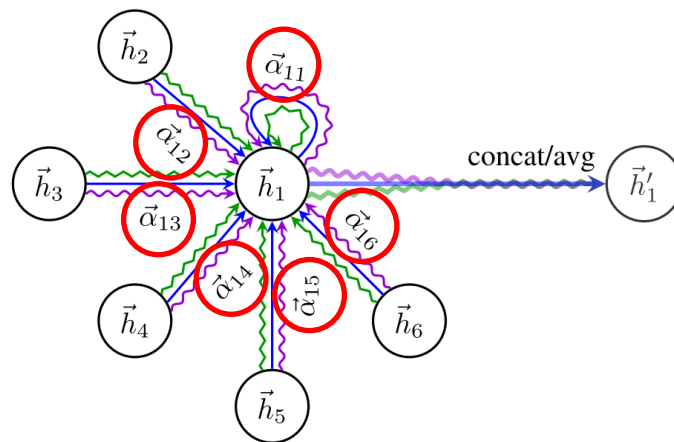
### Mathematical realization:

An attention coefficient is calculated for each connection between a node  $j$  and the central node  $i$  using a shared linear transformation  $\mathbf{W}$  and an attention function  $a$ :

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$$

For GAT, this attention function  $a$  is just another linear transformation  $\mathbf{a}$ . The resulting coefficient reflects the importance of node  $j$  for the update of node  $i$ . A softmax function is used to normalize all coefficients:

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

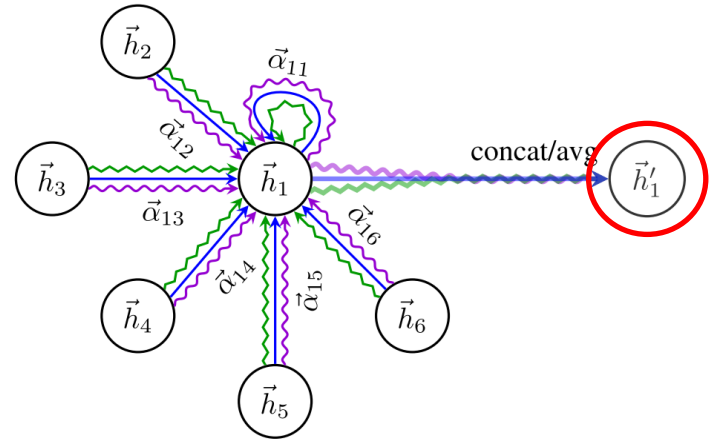


# Example 2: Graph Attention Network (GAT)

## Mathematical realization:

An attention coefficient is calculated for each connection between a node  $j$  and the central node  $i$  using a shared linear transformation  $\mathbf{W}$  and an attention function  $a$ :

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$$



For GAT, this attention function  $a$  is just another linear transformation  $\mathbf{a}$ . The resulting coefficient reflects the importance of node  $j$  for the update of node  $i$ . A softmax function is used to normalize all coefficients:

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

The calculated attention coefficients are now used to aggregate the neighborhood of node  $i$  for the update of its feature representation:

$$\vec{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}\vec{h}_j\right)$$

## Single head

[1] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio and Y. Bengio, 2017. Graph attention networks ICLR 2018



## Example 2: Graph Attention Network (GAT)

### Mathematical realization:

An attention coefficient is calculated for each connection between a node  $j$  and the central node  $i$  using a shared linear transformation  $\mathbf{W}$  and an attention function  $a$ :

$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$$

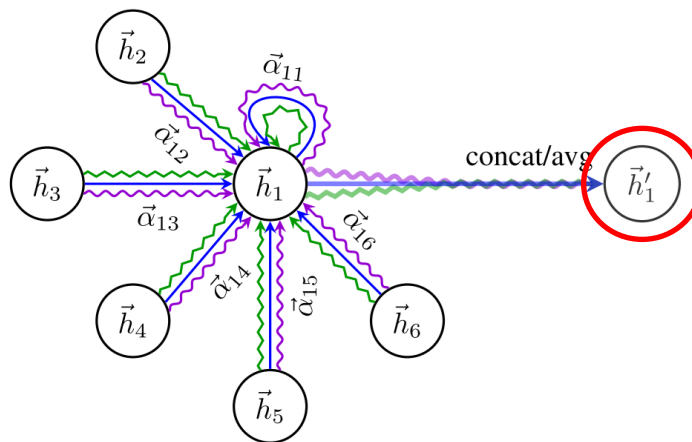
For GAT, this attention function  $a$  is just another linear transformation  $\mathbf{a}$ . The resulting coefficient reflects the importance of node  $j$  for the update of node  $i$ . A softmax function is used to normalize all coefficients:

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

The calculated attention coefficients are now used to aggregate the neighborhood of node  $i$  for the update of its feature representation:

$$\vec{h}'_i = \sigma \left( \sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}\vec{h}_j \right)$$

Single head



$$\vec{h}'_i = \left\| \sigma \left( \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j \right) \right\|_{k=1}^K$$

Multi-head

[1] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio and Y. Bengio, 2017. Graph attention networks ICLR 2018







# Thank you!

Looking forward to the state of the art in this course! All the best!

**Anees Kazi**

[anees.kazi@tum.de](mailto:anees.kazi@tum.de)

**Roger David Soberanis Mukul,**

[roger.soberanis@tum.de](mailto:roger.soberanis@tum.de)

**Mahsa Ghorbani,**

[m.ghorbani1991@gmail.com](mailto:m.ghorbani1991@gmail.com)