Introduction to Geometric Deep Learning with Implementation

Yueh-Hua Tu 杜岳華

2020.01.10

Outline

- Brief history and nowadays AI
- Introduction to graph
- Spectral graph theory
- Graph signal processing
- Taxonomy of GNN
- Tasks
- Models
- Applications
- Connection to other fields

Brief history

1998 LeNet - LeCun

2003 Language model - Bengio

2005 GNN - Franco Scarselli, Marco Gori, Gabriele Monfardini

2006 RBM - Hinton

2009 GNN - Marco Gori, Gabriele Monfardini, Franco Scarselli

2012 AlexNet - Hinton

2013 Word2Vec - Google

2015 Deep learning - Hinton

Why graph neural network?

Nowadays AI

- Computer vision
- Natural language processing
- Speech
- Game, including cheese game, real-time strategy (RTS)

All of them lie in Euclidean domain.

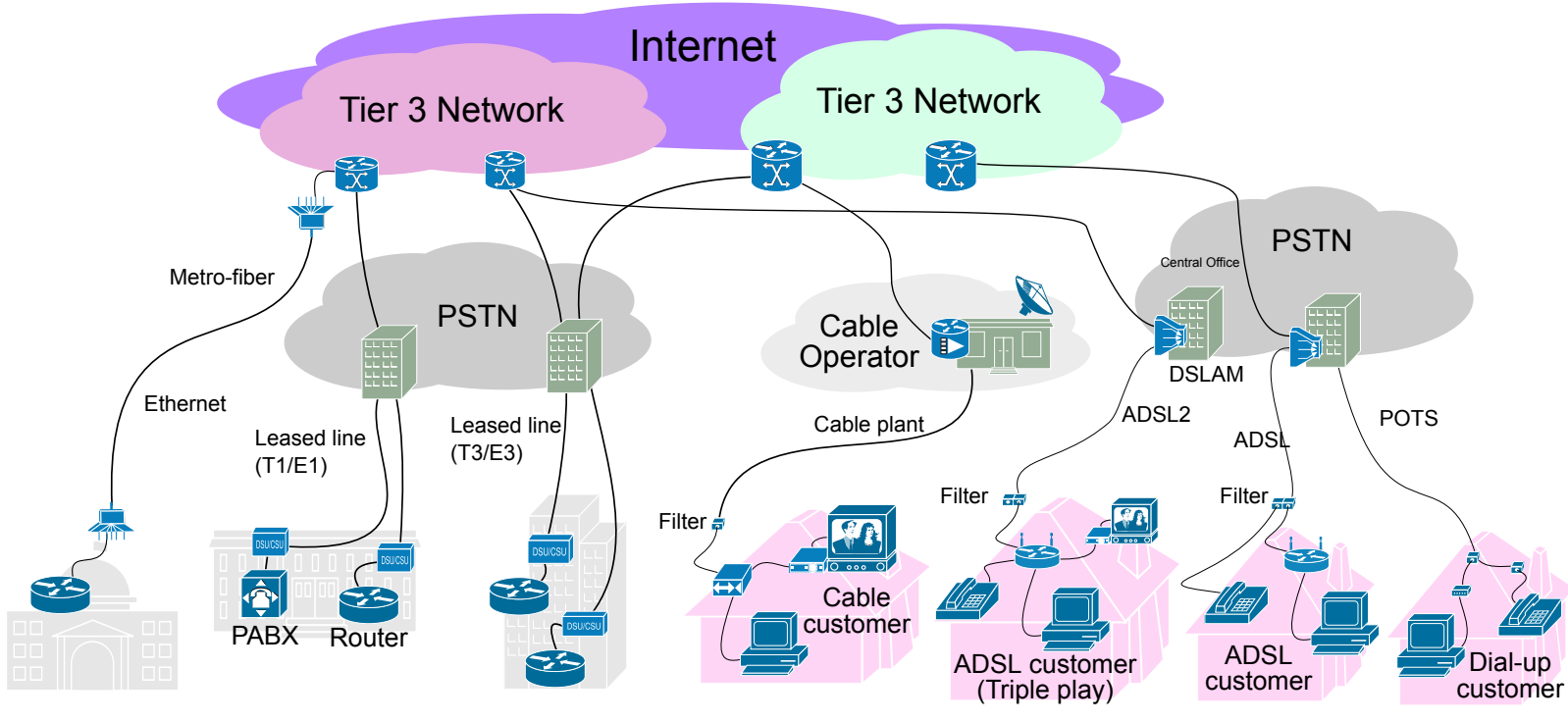
Many scientific data lie in non-Euclidean space

- Social networks in socail science
- Traffic networks
- Gene regulatory networks in biology
- Molecular structure in chemistry
- Knowledge graph
- 3D object surfaces in computer graphics

Graphs in our life

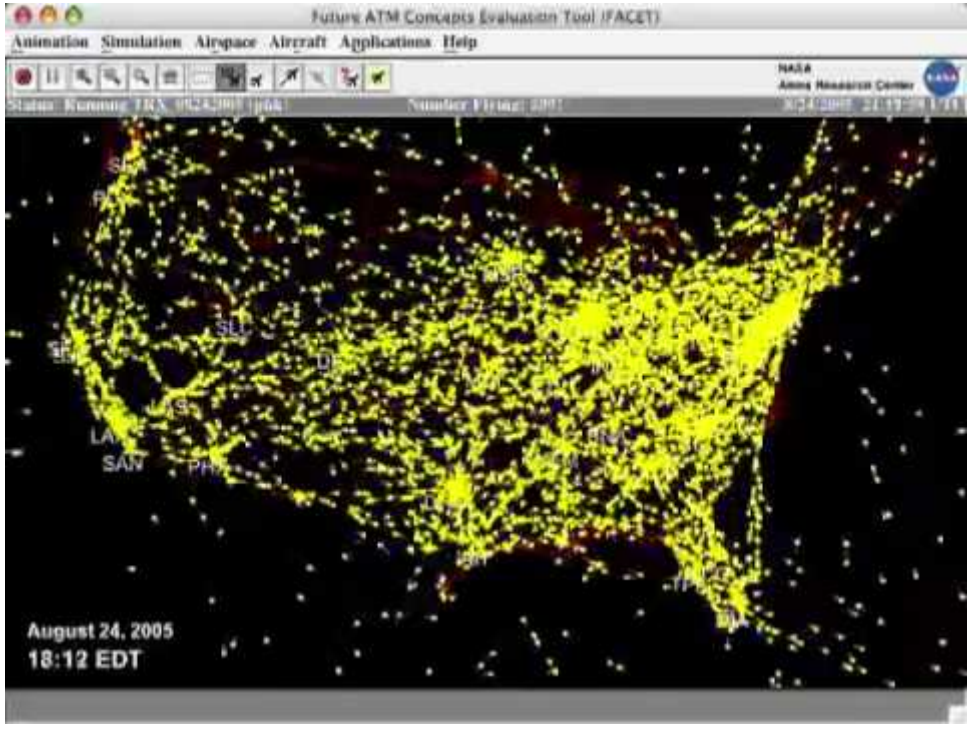
- Social network
- Biological network (gene regulatory network, protein-protein interaction network)
- Computer graphics
- Traffic network
- Knowledge graph

Internet



[picture source (https://en.wikipedia.org/wiki/Internet_service_provider)](https://en.wikipedia.org/wiki/Internet_service_provider)

Traffic network

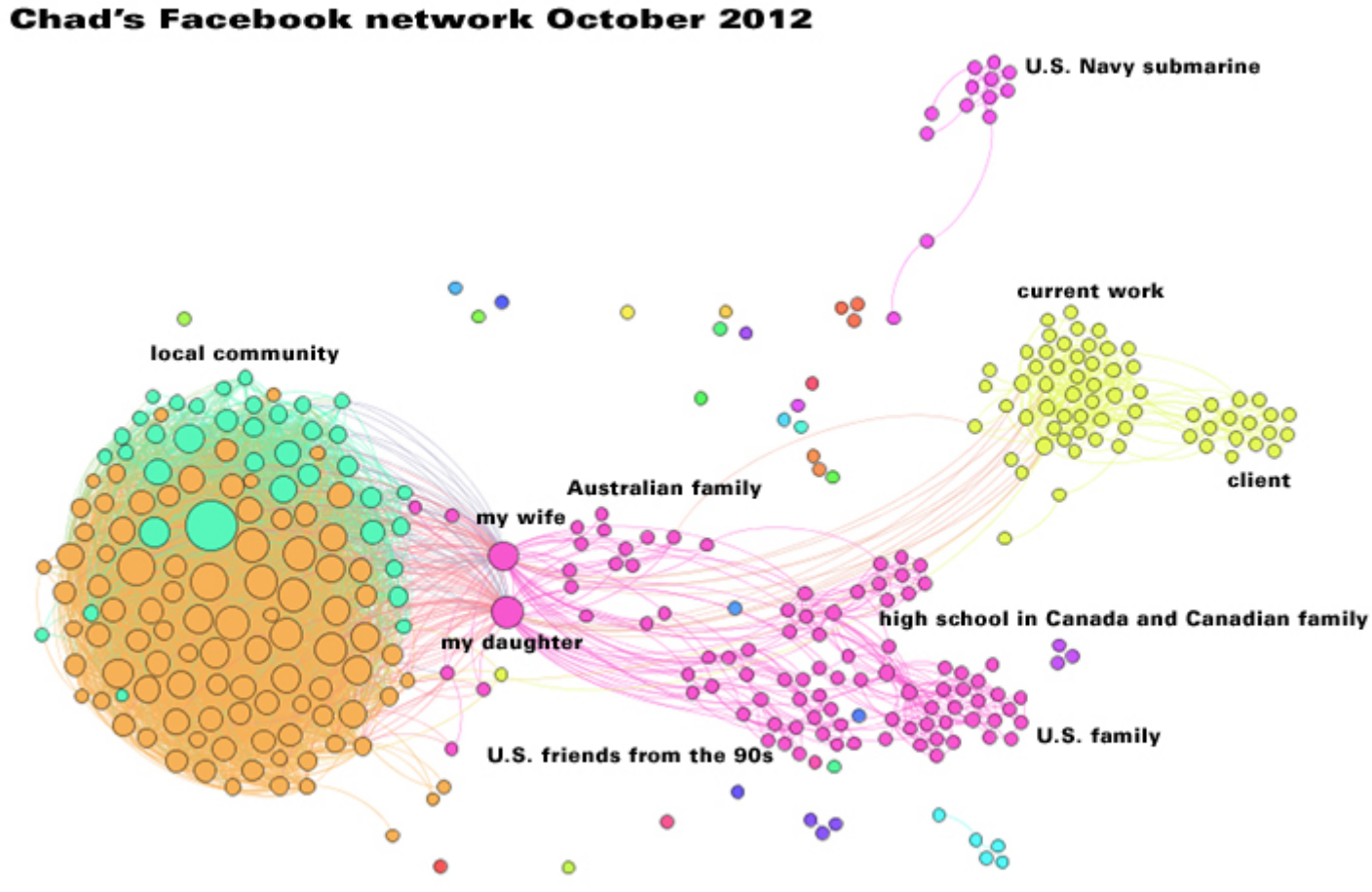


(https://www.youtube.com/watch?v=d9r3H4iHFZk)

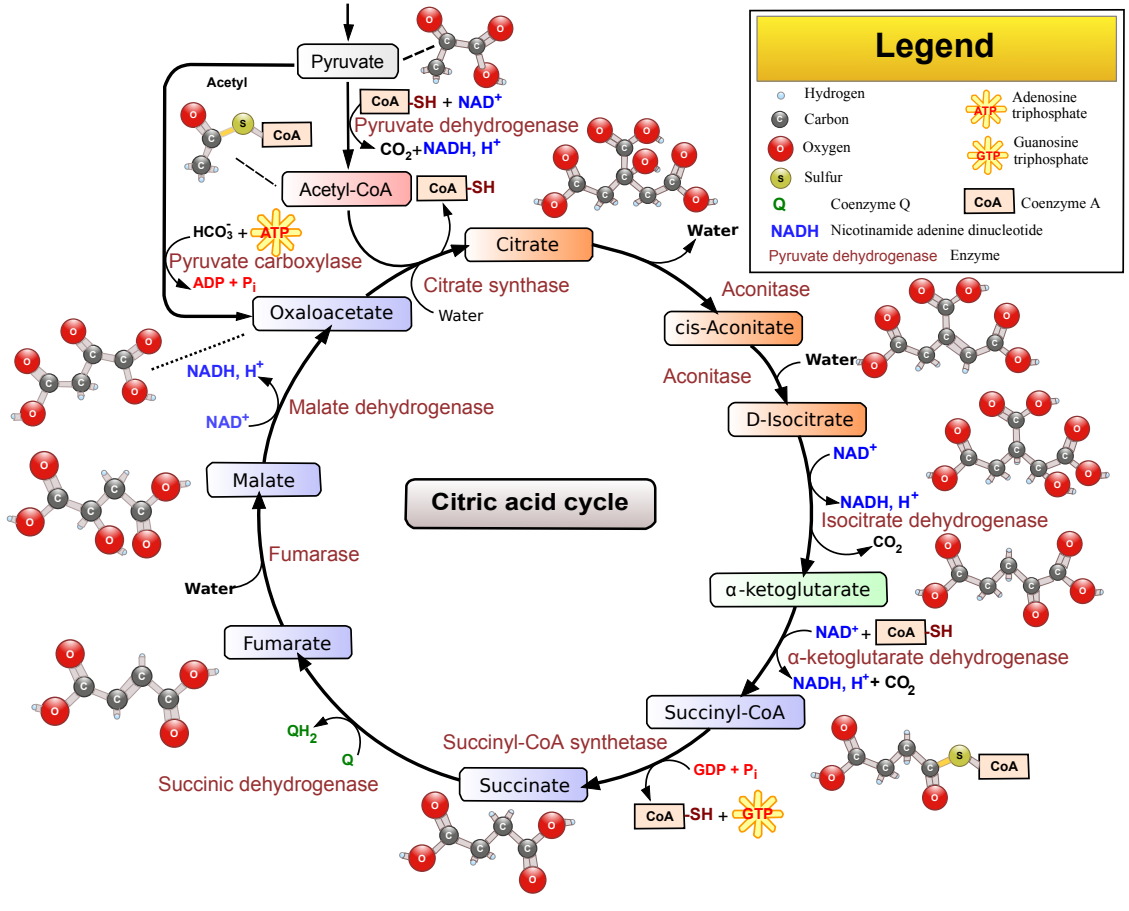
Global flight network

picture source (http://studentwork.prattsi.org/infovis/labs/visualizations-of-the-global-flights-network/)

Social network

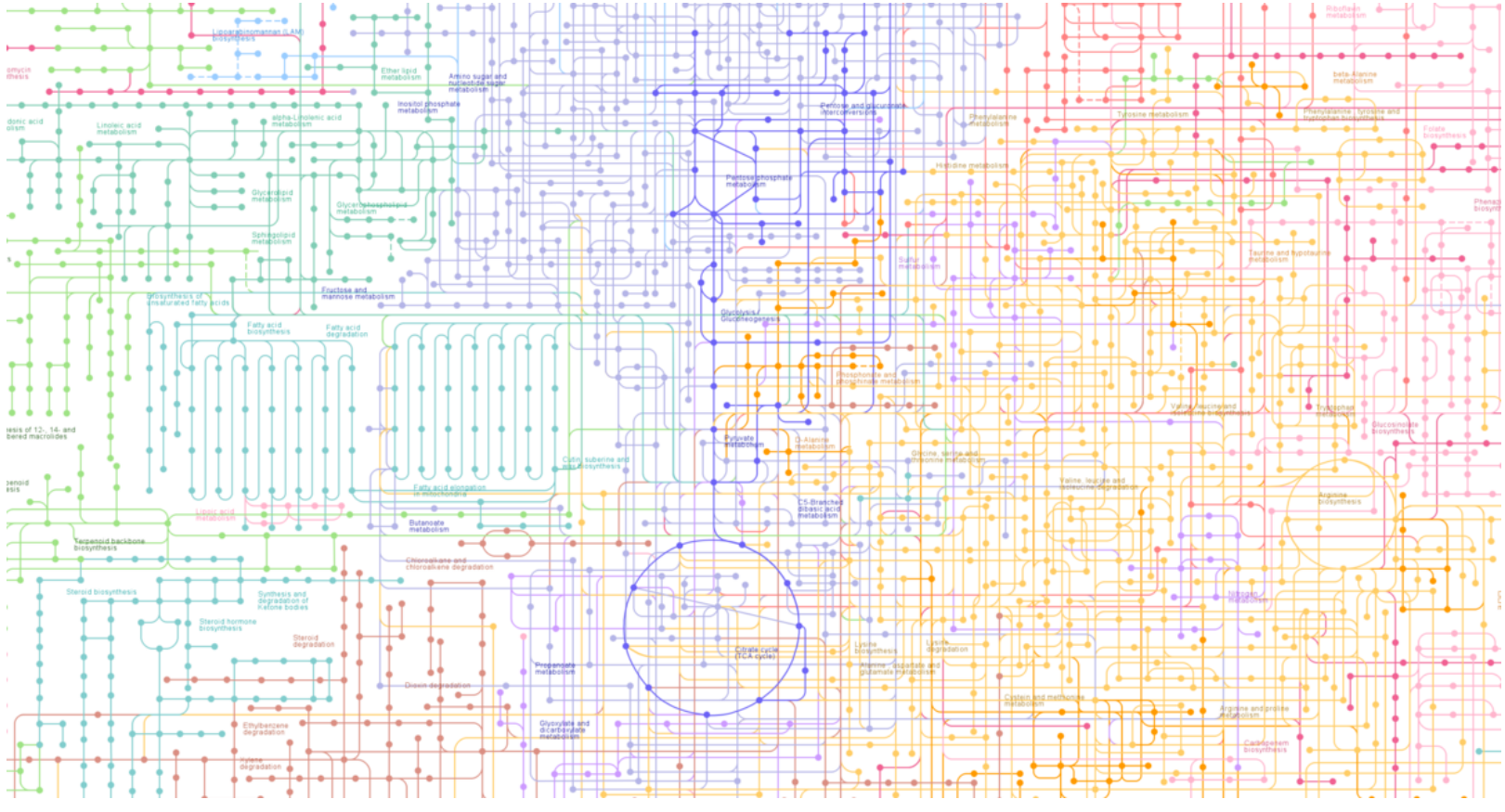


Biological pathway



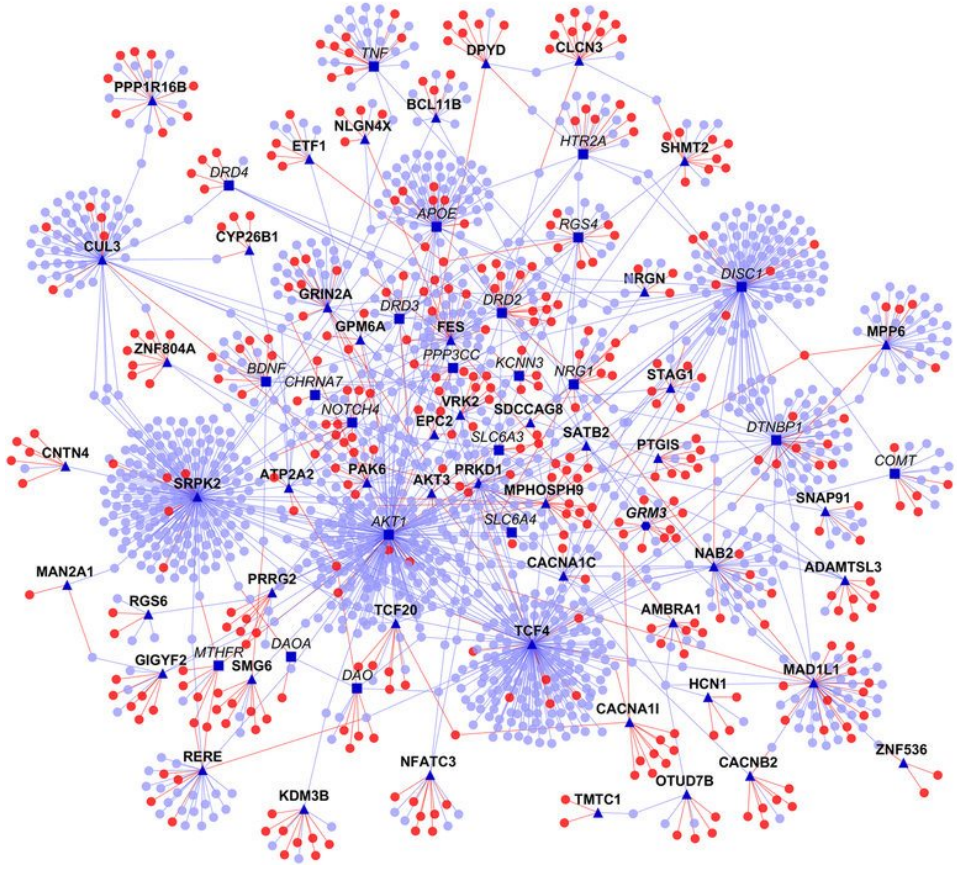
[picture source (https://en.wikipedia.org/wiki/Citric_acid_cycle)](https://en.wikipedia.org/wiki/Citric_acid_cycle)

KEGG



[picture source (https://www.genome.jp/kegg/pathway.html)](https://www.genome.jp/kegg/pathway.html)

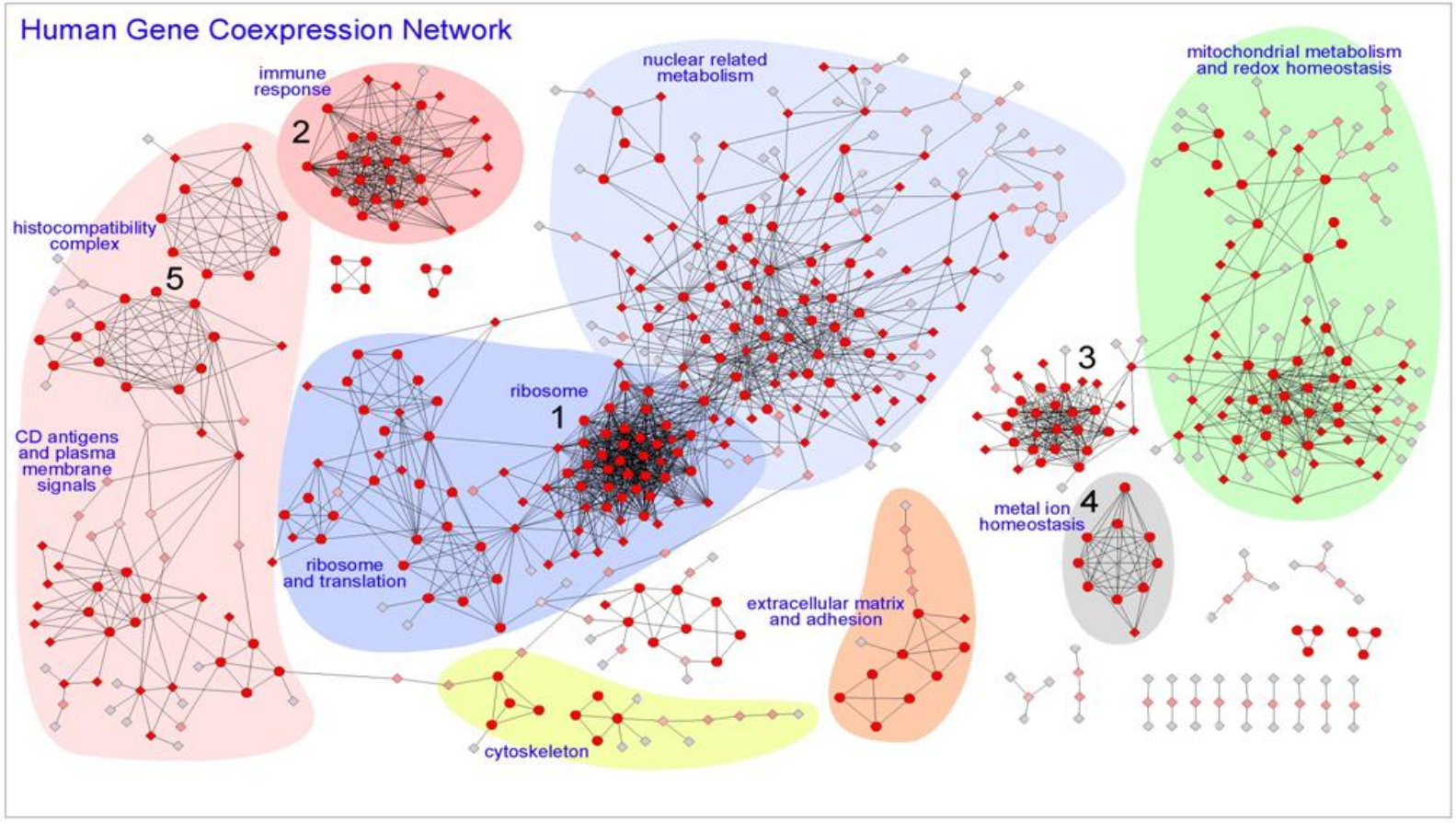
Protein-protein interaction network



Schziophrenia PPI network

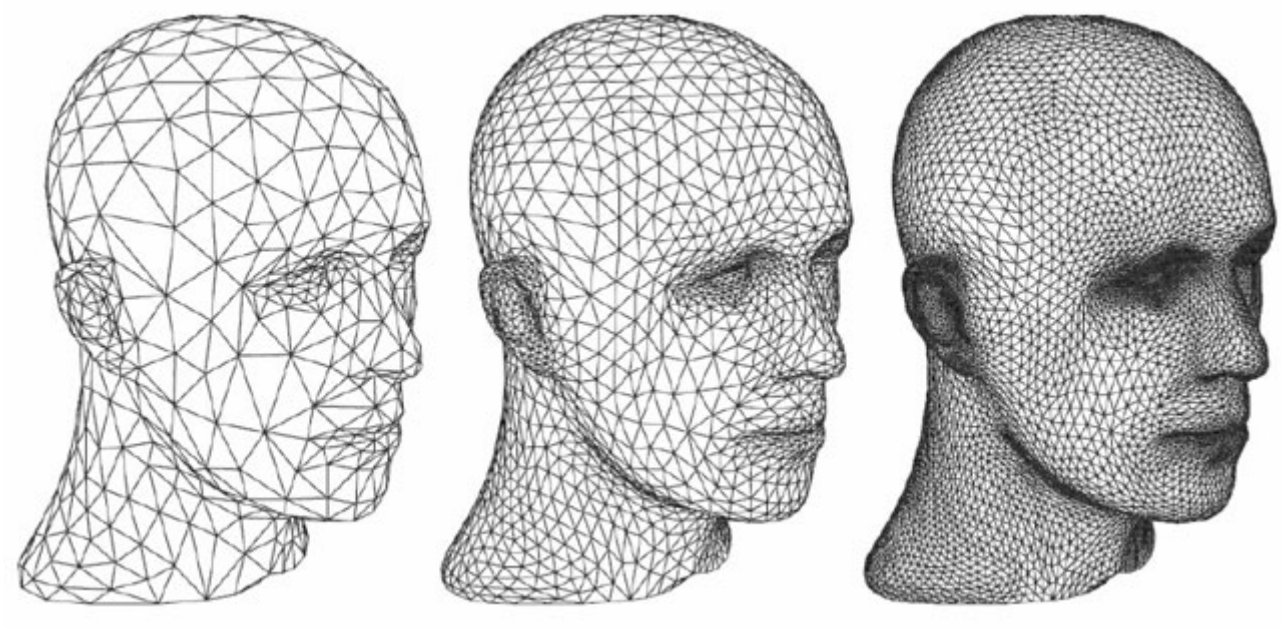
[picture source (https://en.wikipedia.org/wiki/Interactome)](https://en.wikipedia.org/wiki/Interactome)

Human gene coexpression network



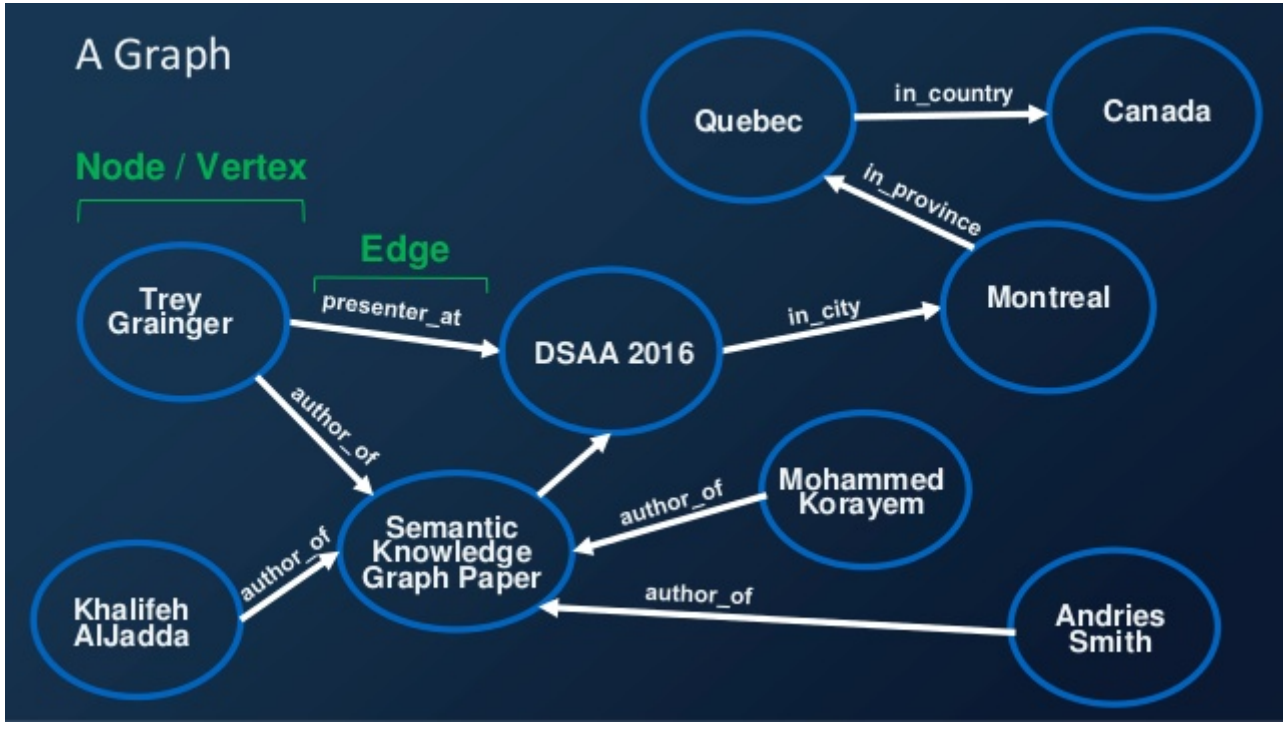
picture source (http://bioinfow.dep.usal.es/coexpression/)

Computer graphics



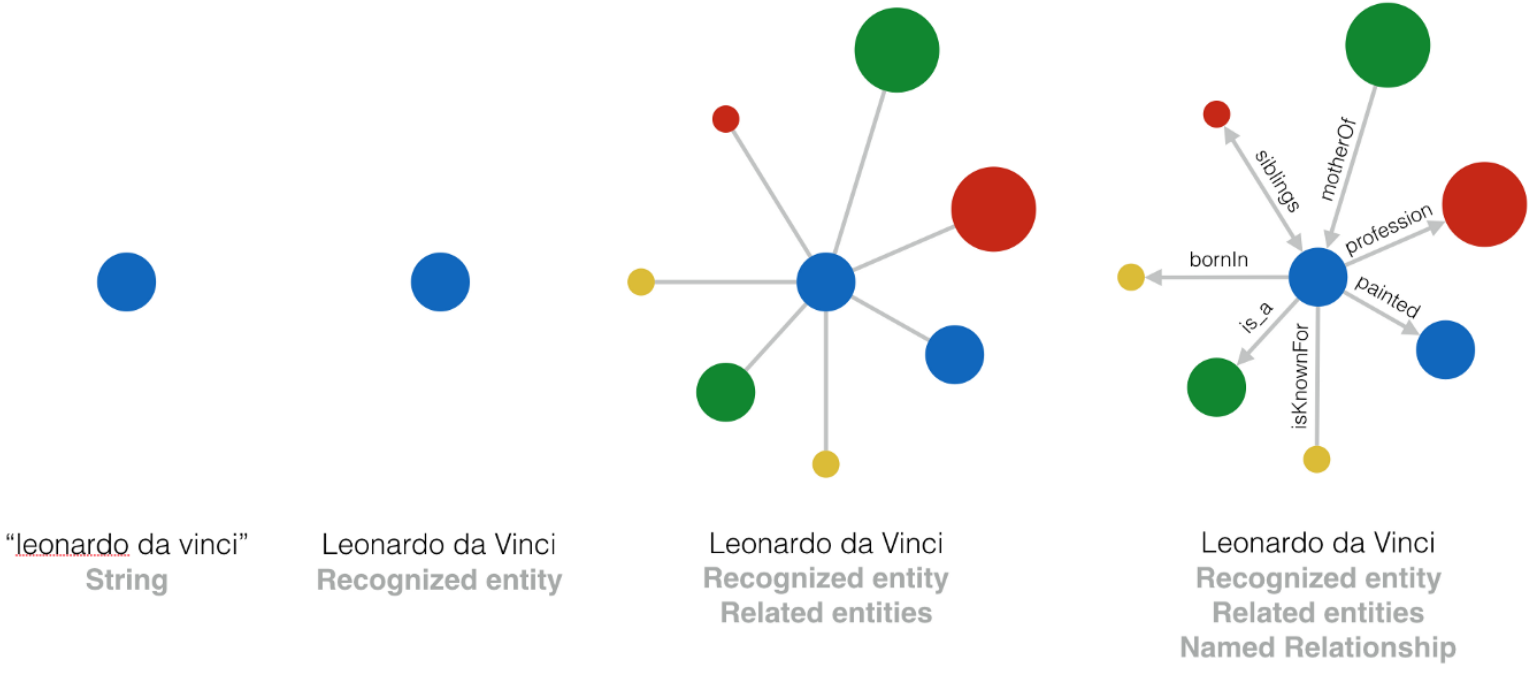
picture source (https://computergraphics.stackexchange.com/questions/2018/what-is-tessellation-in-computer-graphics)

Knowledge graph



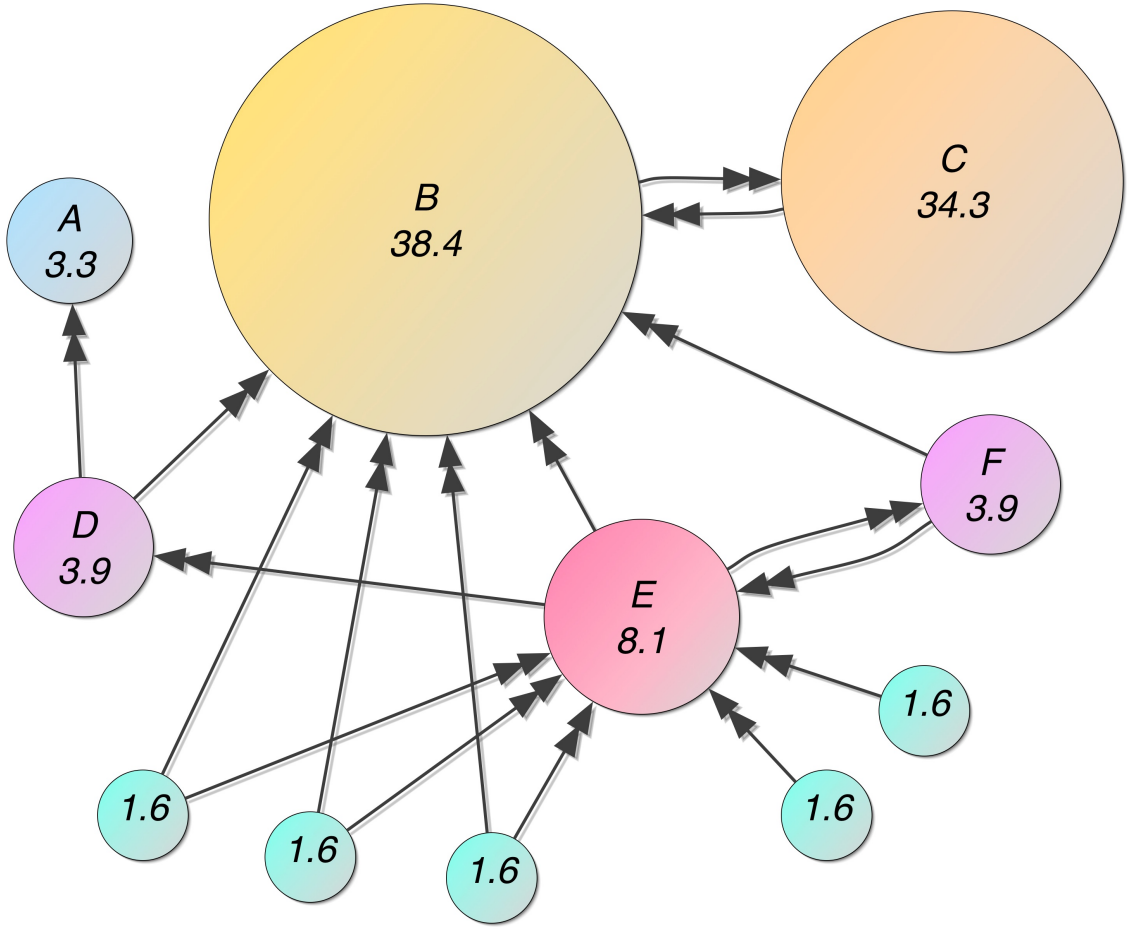
picture source (https://www.slideshare.net/treygrainger/the-semantic-knowledge-graph)

NLP and knowledge graph



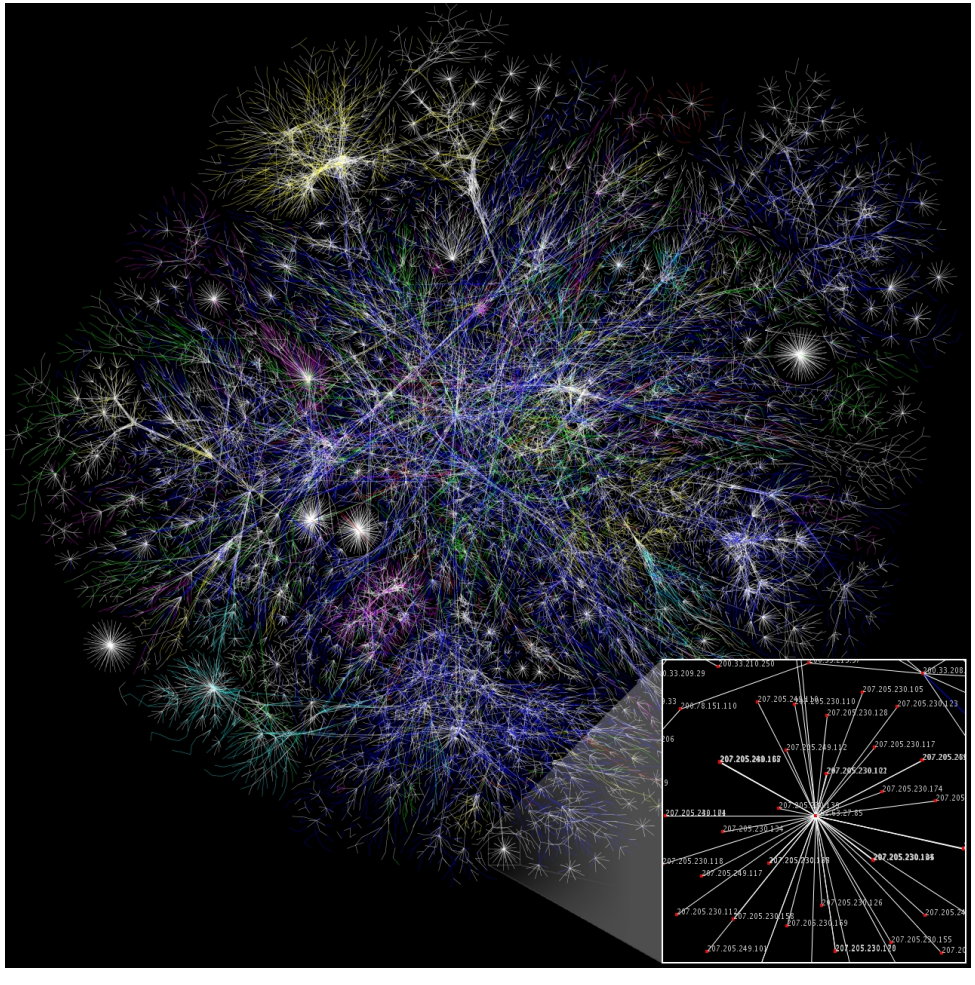
picture source (https://medium.com/@sderymail/challenges-of-knowledge-graph-part-1-d9ffe9e35214)

PageRank



picture source (https://en.wikipedia.org/wiki/PageRank)

Network science

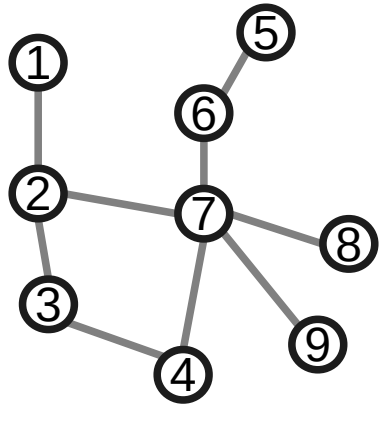


picture source (https://en.wikipedia.org/wiki/Network_science)

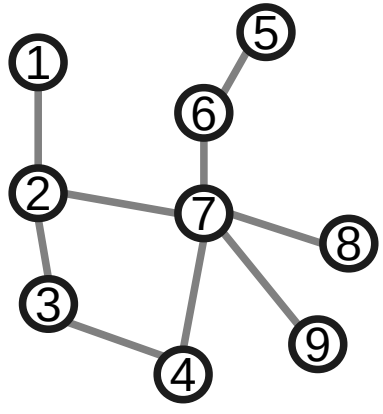
Introduction to Graph

Graph

$$G = (\mathcal{V}, \mathcal{E})$$
$$\mathcal{V} = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$$
$$\mathcal{E} = \{(1, 2), (2, 3), (2, 7), (3, 4), (4, 7), (5, 6), (6, 7), (7, 8), (7, 9)\}$$



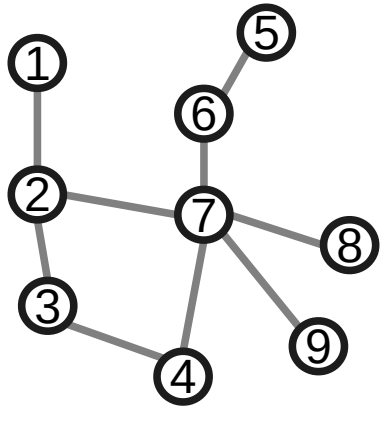
Adjacency matrix



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

Degree

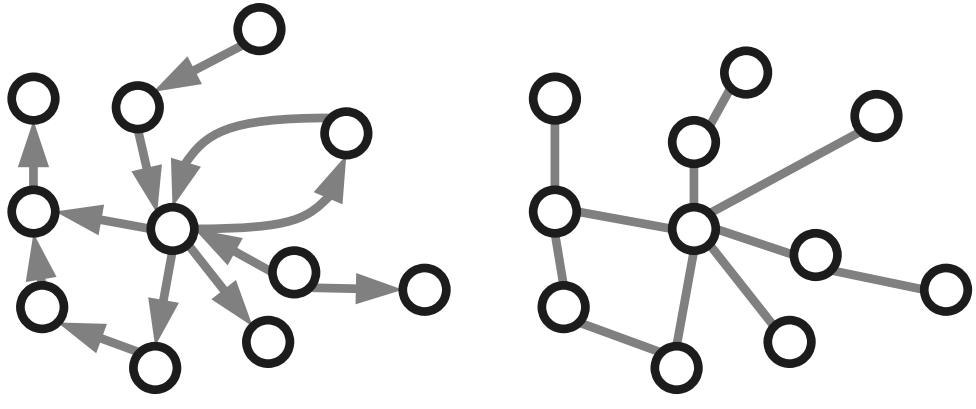
$$deg(v_i) = \text{the number of connections with } v_i$$
$$\text{ex. } deg(7) = 5$$



Degree matrix

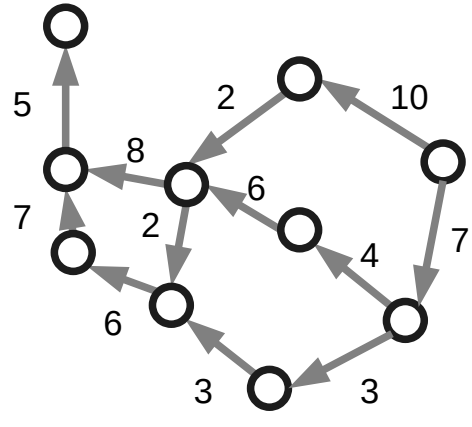
$$D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Directed graph (Digraph)



Weighted graph (network)

$$G = (\mathcal{V}, \mathcal{E}, W)$$
$$W : E \to \mathbb{R}$$



Generalized degree

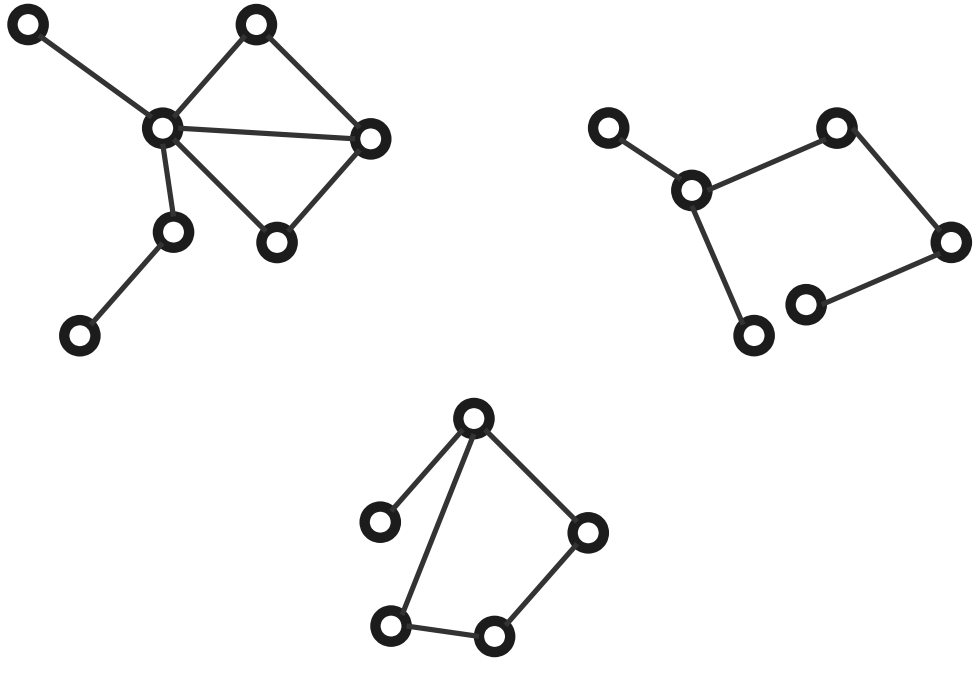
Directed graph

- Outdegree:
  $deg(i) = \sum_j w_{ij}$
- Indegree:
  $deg(j) = \sum_i w_{ij}$

Undirected graph

- Degree:
  $deg(i) = \sum_j w_{ij}$

Connected component



Random walk on graph

Random walk transition matrix

$$P = D^{-1}A$$

One-step transition

$$x_t = Px_{t-1}$$

N-step transition

$$x_t = P^n x_{t-1}$$

Reachability

∞-step transition

$$x = Px$$

Eigenvalue problem

$$Px = 1x$$

Thm

Eigenvalues of a Laplacian $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and graph has k connected components, then

$$0 = \lambda_1 = \lambda_2 = \cdots = \lambda_k < \lambda_{k+1} \leq \cdots \leq \lambda_n$$

Spectral Graph Theory

Laplacian Matrix

(Unnormalized) Laplacian matrix

$$L = D - A$$

$$L = \begin{bmatrix} 2 & -1 & 0 & 0 & -1 \\ -1 & 3 & -1 & 0 & -1 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ -1 & -1 & 0 & -1 & 3 \end{bmatrix}$$

Random walk normalized Laplacian

$$L^{rw} = D^{-1}L = I - D^{-1}A$$

Symmetric normalized Laplacian matrix

$$L = D^{-\frac{1}{2}}LD^{-\frac{1}{2}} = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$$

Graph as Operator

Graph A as an operator

$$y = Ax \Leftrightarrow y(i) = \sum_{j \in \mathcal{N}(i)} x(j)$$

Quadratic form of Graph A

$$x^T Ax = \sum_{j \in \mathcal{N}(i)} x(i)x(j)$$

Laplacian Matrix as Difference Operator

Difference operator

$$Lx = \sum_{j \in \mathcal{N}(i)} x(i) - x(j)$$

$$L = D - A = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

Quadratic Form of Laplacian Matrix

Measure of "smoothness"

$$x^T L x = \sum_{j \in \mathcal{N}(i)} (x(i) - x(j))^2$$

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

Compare with differential operator

$$f''(x) = \lim_{h \to \infty} (\frac{f(x+h) - f(x)}{h} - \frac{f(x) - f(x-h)}{h})/h$$

$$= \lim_{h \to \infty} \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

$$f''(N) = f(N+1) - 2f(N) + f(N-1)$$

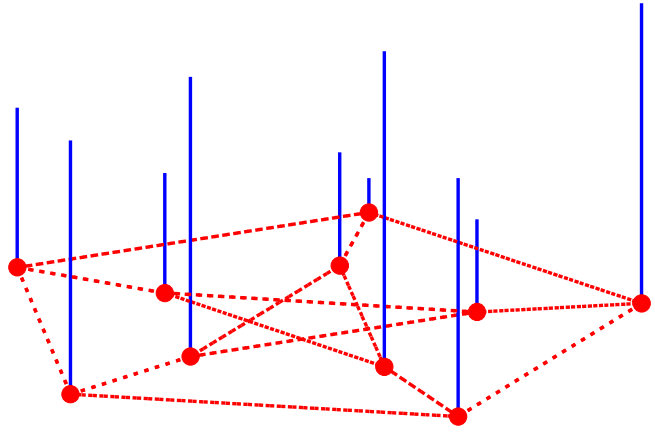
Graph Signal Processing

Graph Signals

Graph signals provide a suitable form to encode structure information within signal processing

Signals are formulated as functions

$$f : \mathcal{V} \rightarrow \mathbb{R}^n$$



Two paradigms in classical signal processing

Time domain

- Vertex (spatial) domain

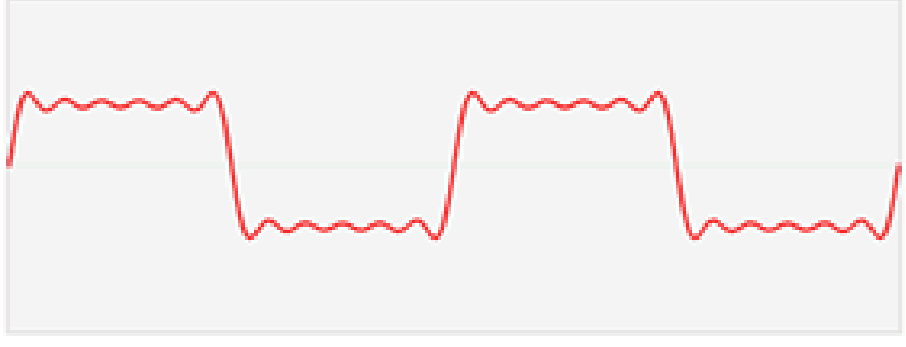
Frequency domain

- Frequency (graph spectral) domain

Fourior Transformation

$$\mathcal{F}\{f\}(w) = \int_{-\infty}^{\infty} f(x) e^{-iwx} \, dx$$

How to generalize classical signal processing to irregular domain?



Wikipedia - Fourier transform

Graph Laplacian

Laplacian

$$Lf = \sum_{j \in \mathcal{N}(i)} A_{ij}(f(i) - f(j))$$

Analog to Laplace operator in vector calculus

$$\Delta f = \nabla^2 f = \sum_i \frac{\partial^2}{\partial x_i^2} f$$

Classical Fourier Transformation

Fourier Transformation

$$F(w) = \mathcal{F}\{f\}(w) = \langle e^{iwt}, f \rangle = \int_{-\infty}^{\infty} e^{-iwt} f(t) dt$$

Inverse Fourier Transformation

$$\mathcal{F}^{-1}\{F\}(t) = \int_{-\infty}^{\infty} e^{iwt} F(w) dw$$

Eigenfunction

$$\Delta \phi_i = \lambda_i \phi_i$$

$$L x_i = \lambda_i x_i$$

Graph Fourier Transformation

Eigenvector

$$L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U \Lambda U^T$$

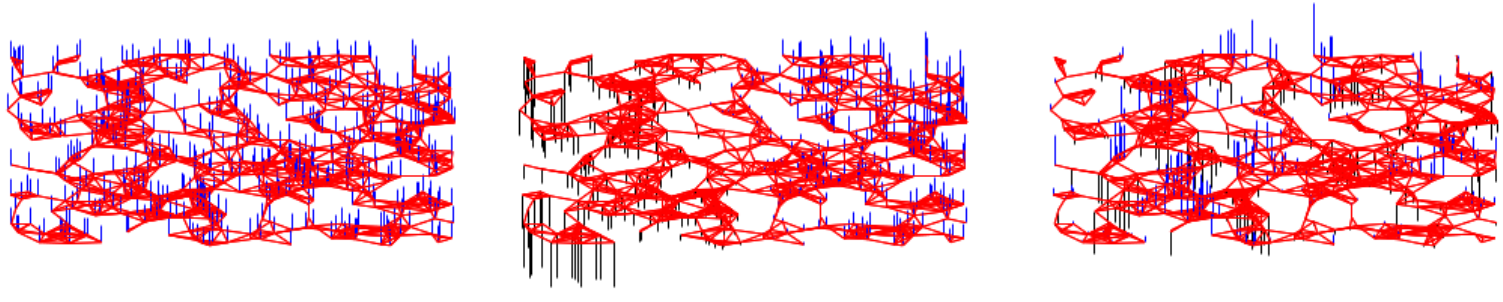
Graph Fourier Transformation

$$F(w) = \mathcal{F}\{f\}(w) = \langle U, f \rangle = \sum_{i=1}^{\infty} U^T(i) f(i) = U^T f$$

Inverse Graph Fourier Transformation

$$\mathcal{F}^{-1}\{F\}(x) = \sum_{j=1}^{\infty} U(j) F(j) = UF$$

Graph eigenvalues represent how ”smooth” it is



$\mathbf{u}_0$ $\mathbf{u}_1$ $\mathbf{u}_{50}$

Graph Convolution

Classical convolution

$$(f * g)(x) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau$$

$$= \int_{-\infty}^{\infty} f(w) g(w) e^{iwt} dw$$

Graph convolution

$$(f * g)(i) = \sum_{l=0}^{N-1} f(\lambda_l) g(\lambda_l) u_l(i)$$

Graph Spectral Filtering

Classical spectral filtering

$$f(t) \xrightarrow[\text{Transform}]{\text{Fourier}} F(w) \xrightarrow{\text{Filtering}} G(w)F(w) \xrightarrow[\text{Transform}]{\text{Inverse}} (f*g)(t)$$

Graph spectral filtering

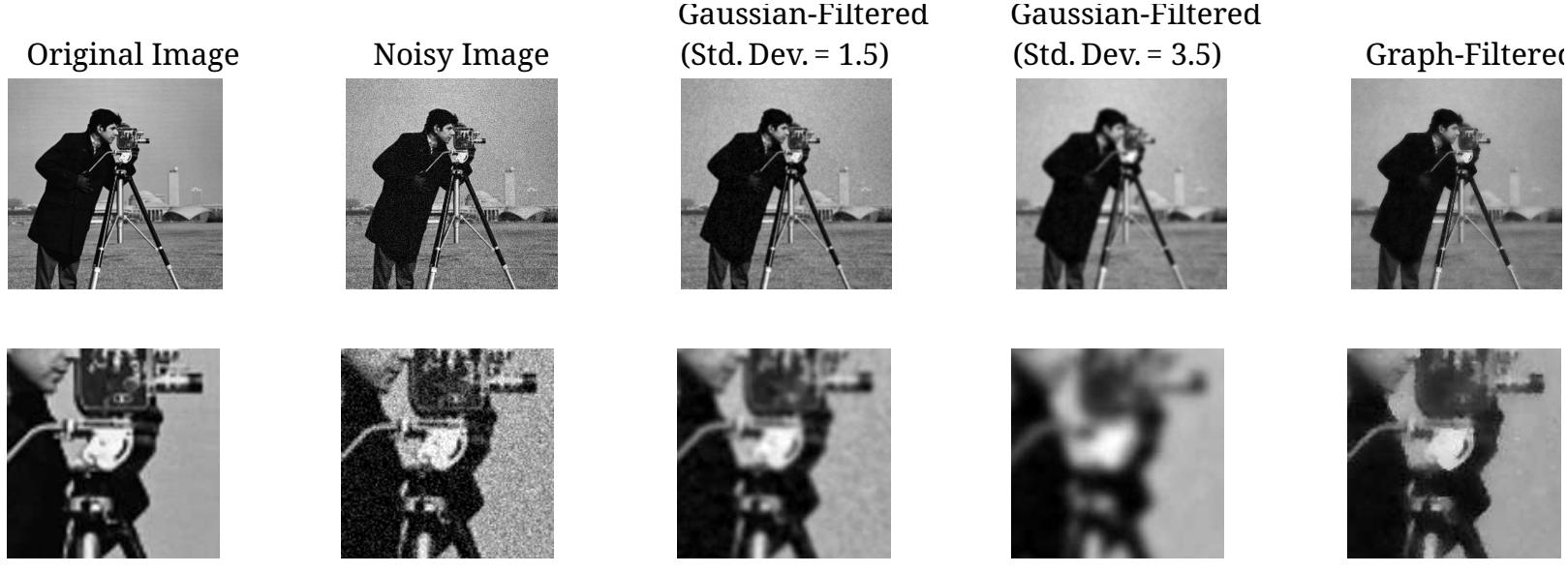
$$f(i) \xrightarrow[\text{Transform}]{\text{Fourier}} F(\lambda_l) \xrightarrow{\text{Filtering}} G(\lambda_l)F(\lambda_l) \xrightarrow[\text{Transform}]{\text{Inverse}} (f*g)(i)$$

Graph Fourier Transform: $F(\lambda_l) = U^T f$

Graph Spectral Filtering: $G(\lambda_l)F(\lambda_l) = G(\Lambda)U^T f$

Inverse Graph Fourier Transform: $(f*g)(i) = UG(\Lambda)U^T f$

Effect of Graph Spectral Filtering



Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., & Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. IEEE Signal Processing Magazine, 30(3), 83–98. doi: 10.1109/msp.2012.2235192

Taxonomy of GNN

(Semi-)supervised model

- Graph convolution networks
- Graph attention networks
- Graph spatial-temporal networks

Unsupervised model

- Graph autoencoders
- Graph generative networks

Tasks

Node-level tasks

- node regression, node classification
- graph convolution layer gives node's latent representations

Edge-level tasks

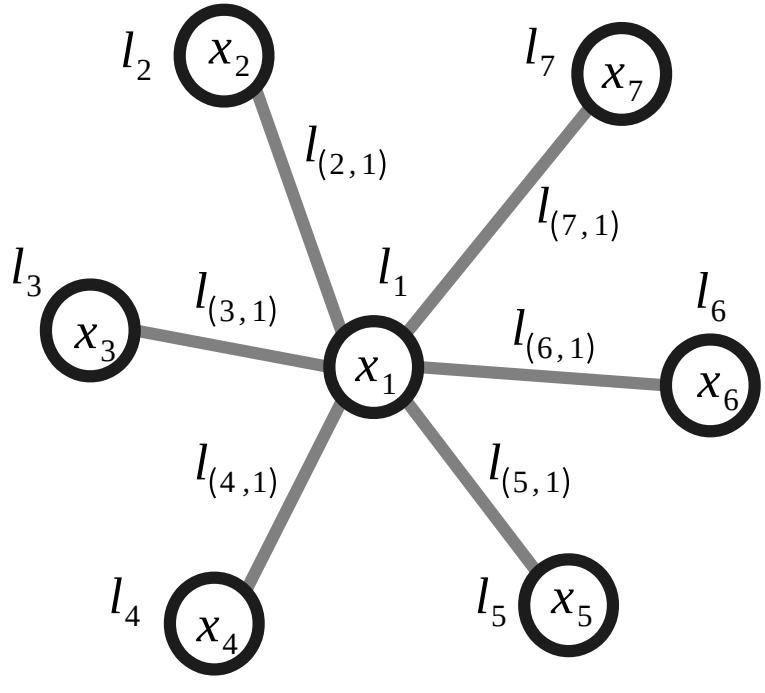
- link prediction, edge classification
- additional function would take two nodes' latent representations as input of graph convolution layer

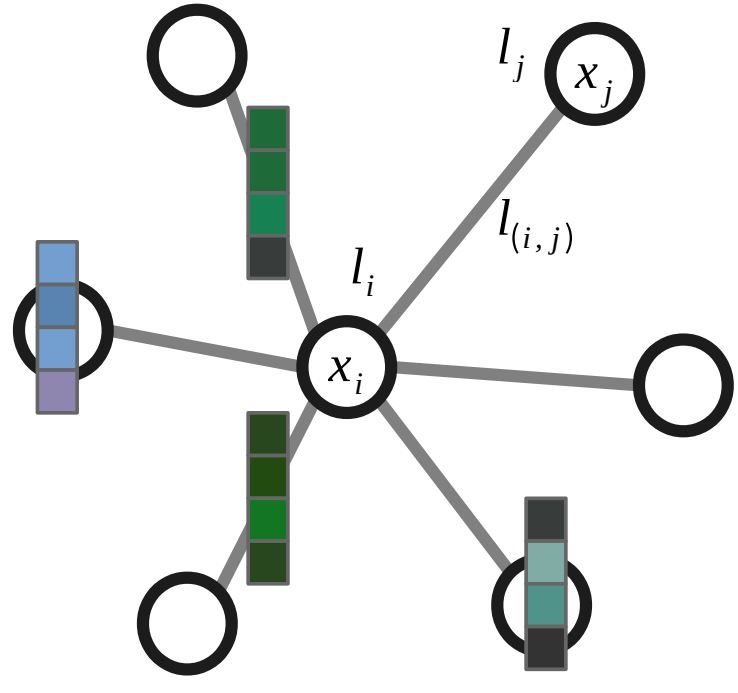
Graph-level (global-level) tasks

- graph classification
- graph pooling gives the reduced graph information

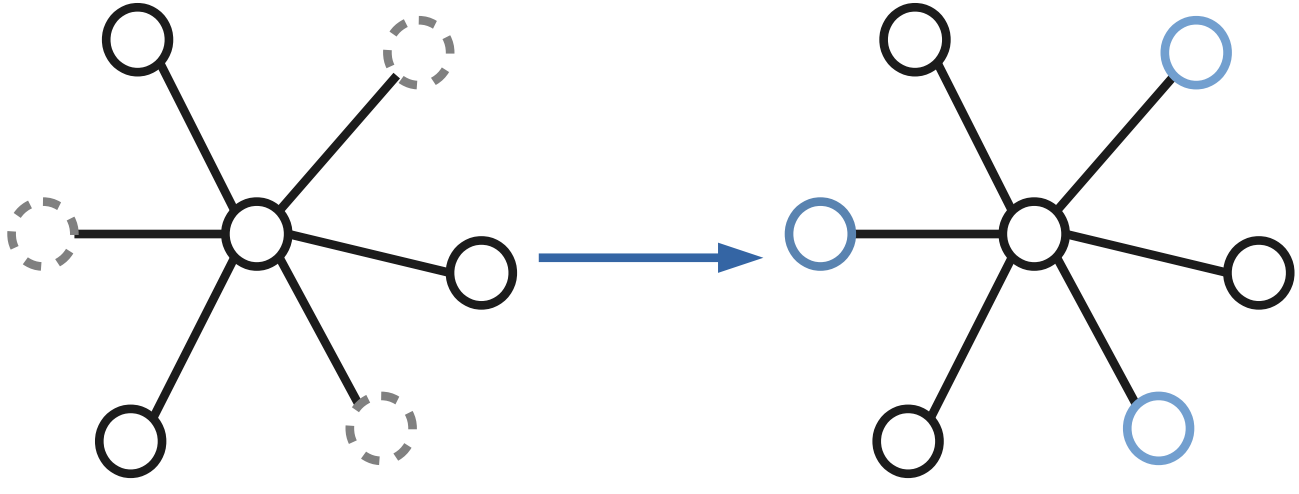
Graph signals and labels

W: adjacency (or weighted) matrix, X: input node features, l: node/edge labels

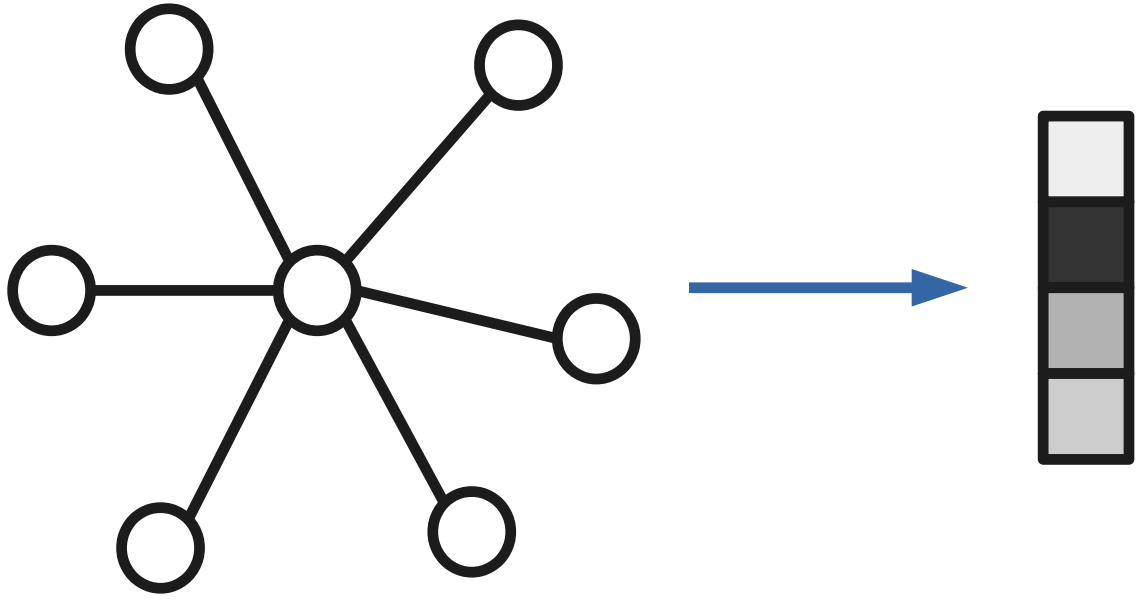




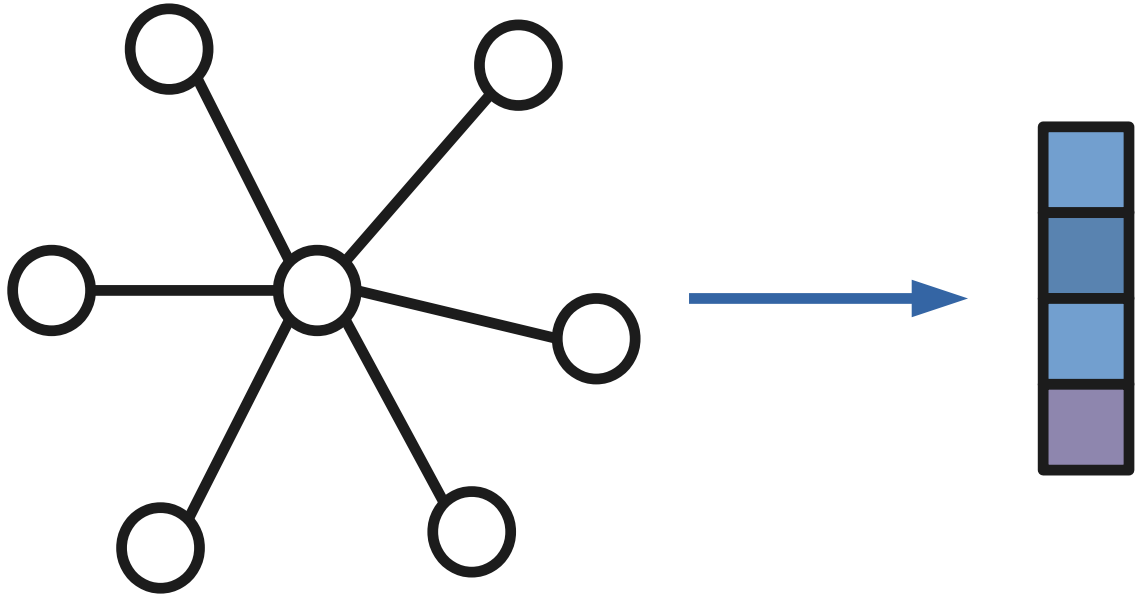
Semi-supervised learning for node classification



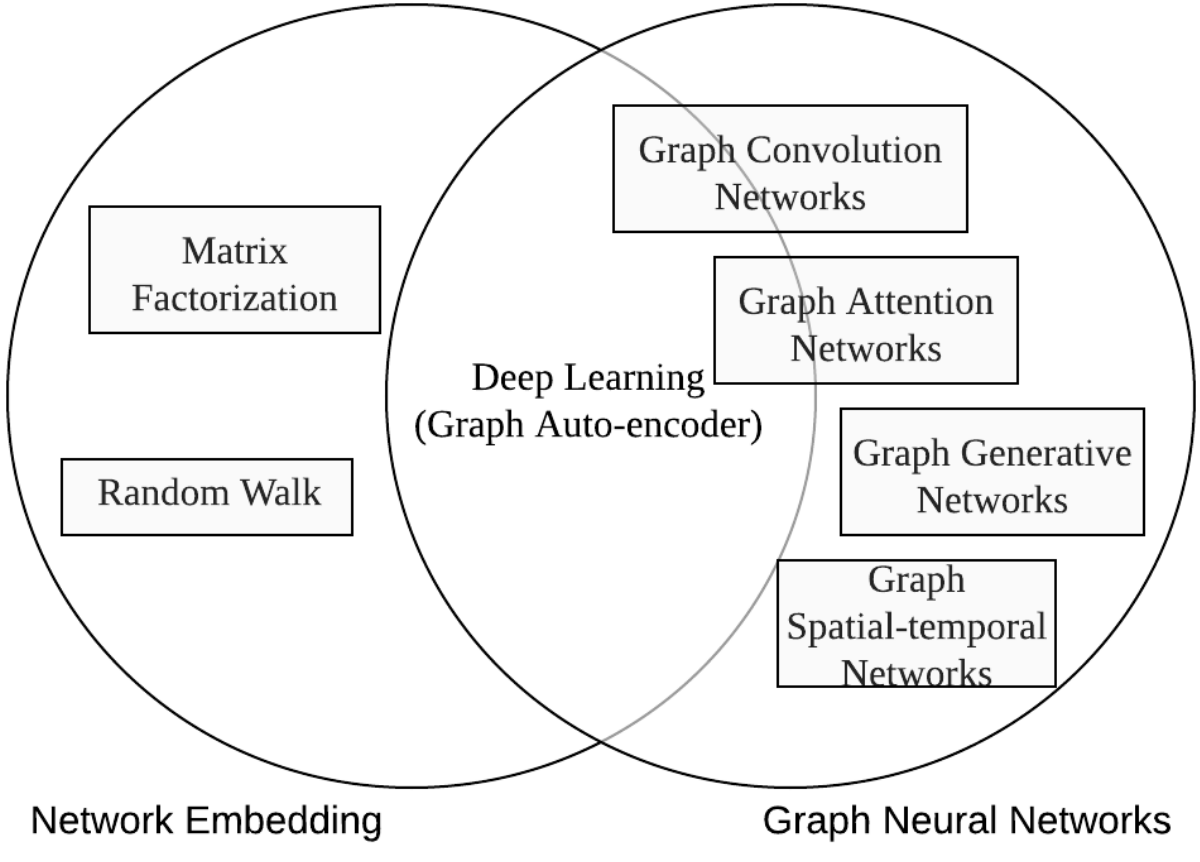
Supervised learning for graph classification



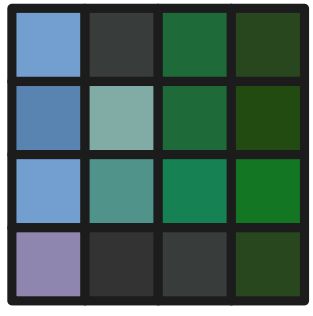
Unsupervised learning for graph embedding

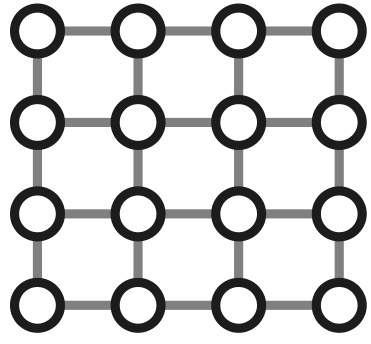


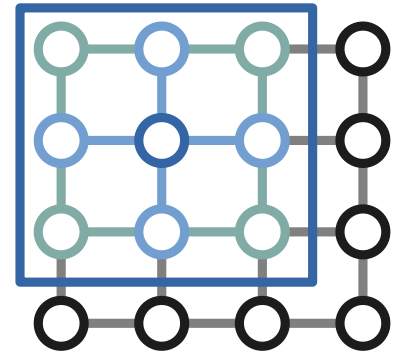
GNN and network embedding

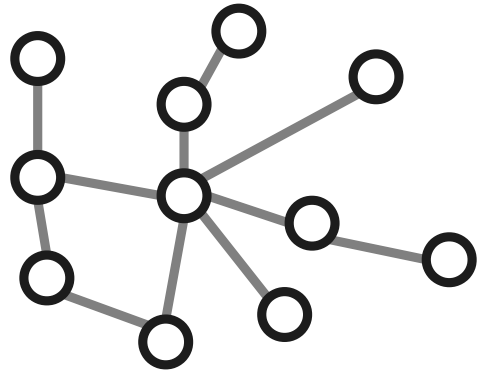


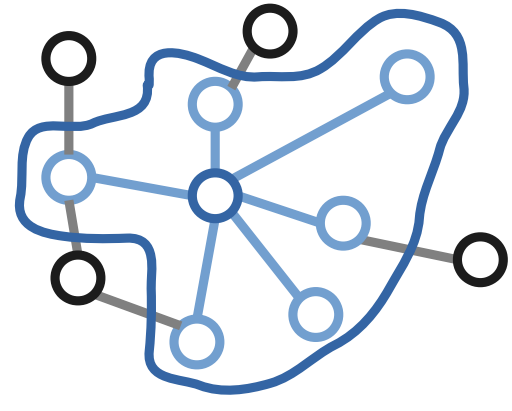
Convolution on graph





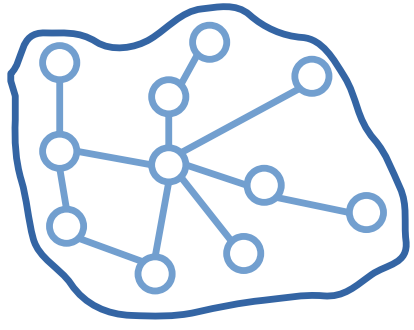




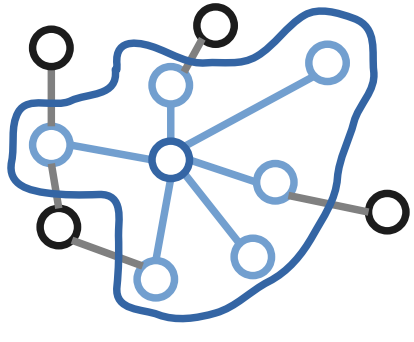


Graph convolution networks

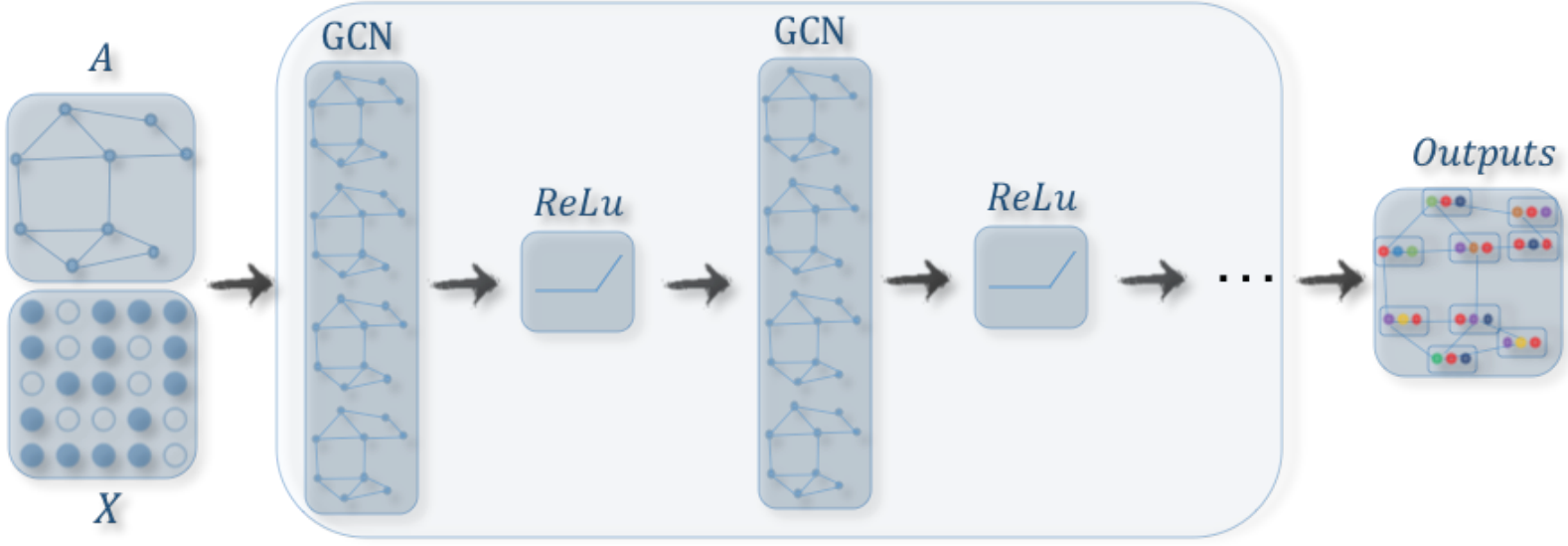
Spectral-based GCN



Spatial-based GCN



Spectral-based GCN



Single-node layer

$$x_j^{t+1} = \sigma(\tilde{A}x_i^t) = \sigma(U\Theta_{i,j}^k U^T x_i^t)$$

Spectral CNN

Graph convolution

A input x and a filter $g \in \mathbb{R}^N$ is defined as

$$x *_G g = \mathcal{F}^{-1}\{\mathcal{F}\{x\} \odot \mathcal{F}\{g\}\} = U(U^T x \odot U^T g)$$

$$= U g_\theta U^T x$$

$$g_\theta = diag(U^T g)$$

This consumes large computation and memeory!

Chebyshev spectral CNN (ChebNet)

For efficiency, define filters as **Chebyshev polynomials**.

$$g_\theta = \sum \theta_i T_k(\tilde{\Lambda})$$

First-order ChebNet

First-order approximation of ChebNet

$$\begin{aligned} x *_G g_\theta &= \theta(I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}})x \\ &= \theta \tilde{A} x \end{aligned}$$

$$X^{t+1} = \tilde{A} X^t \Theta$$

Spatial-based GCN

Recurrent-based GCN (spatial)

Apply **the same** graph convolution layer to update hidden representations.

Composition-based GCN (spatial)

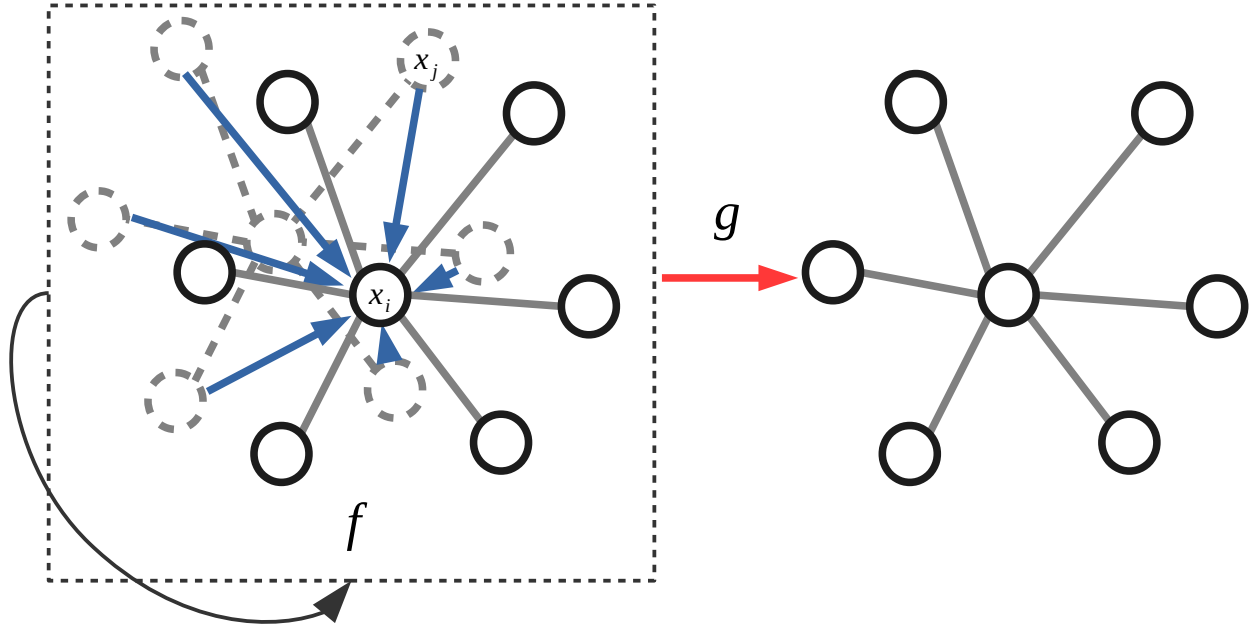
Apply **different** graph convolution layers to update hidden representations.

Recurrent-based GCN

Graph neural network (GNN)

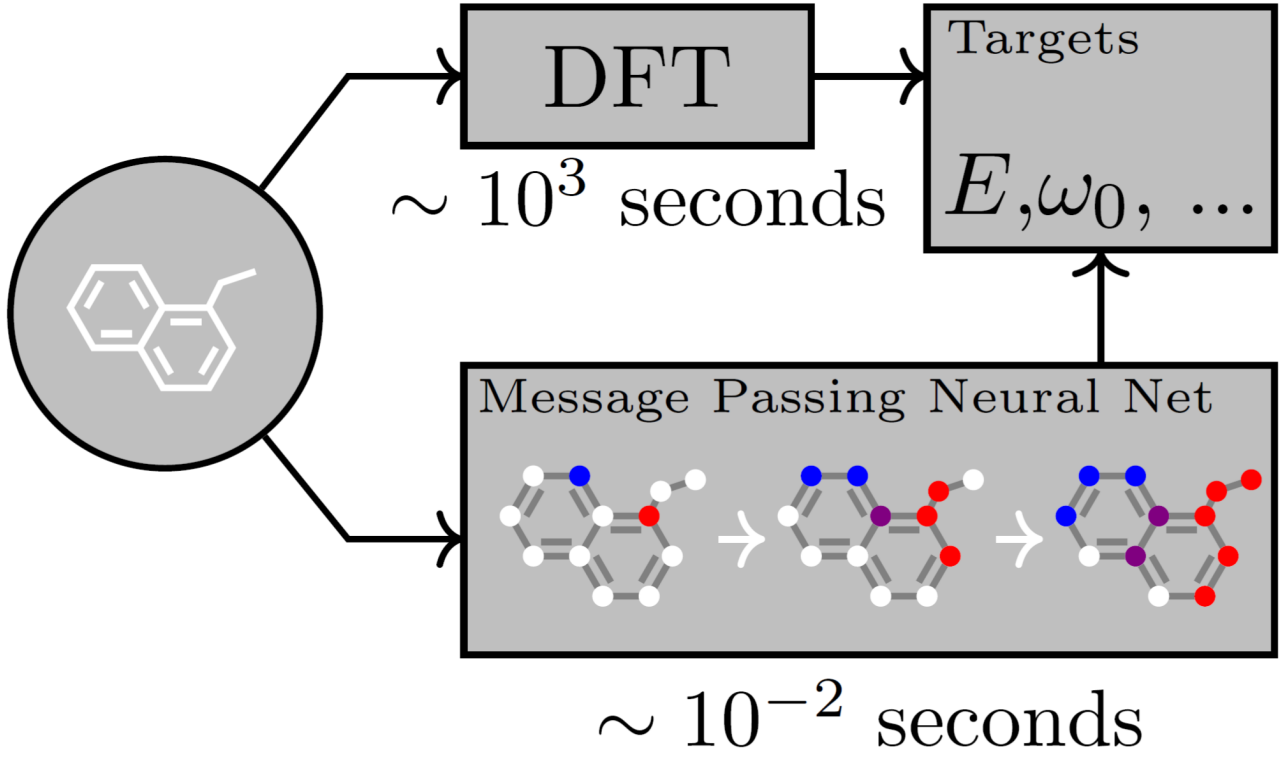
$$x_i(t+1) = f(l_i, l_e, x_{ne[i]}(t), l_{ne[i]})$$

$$o_i(t) = g(x_i(t), l_i)$$

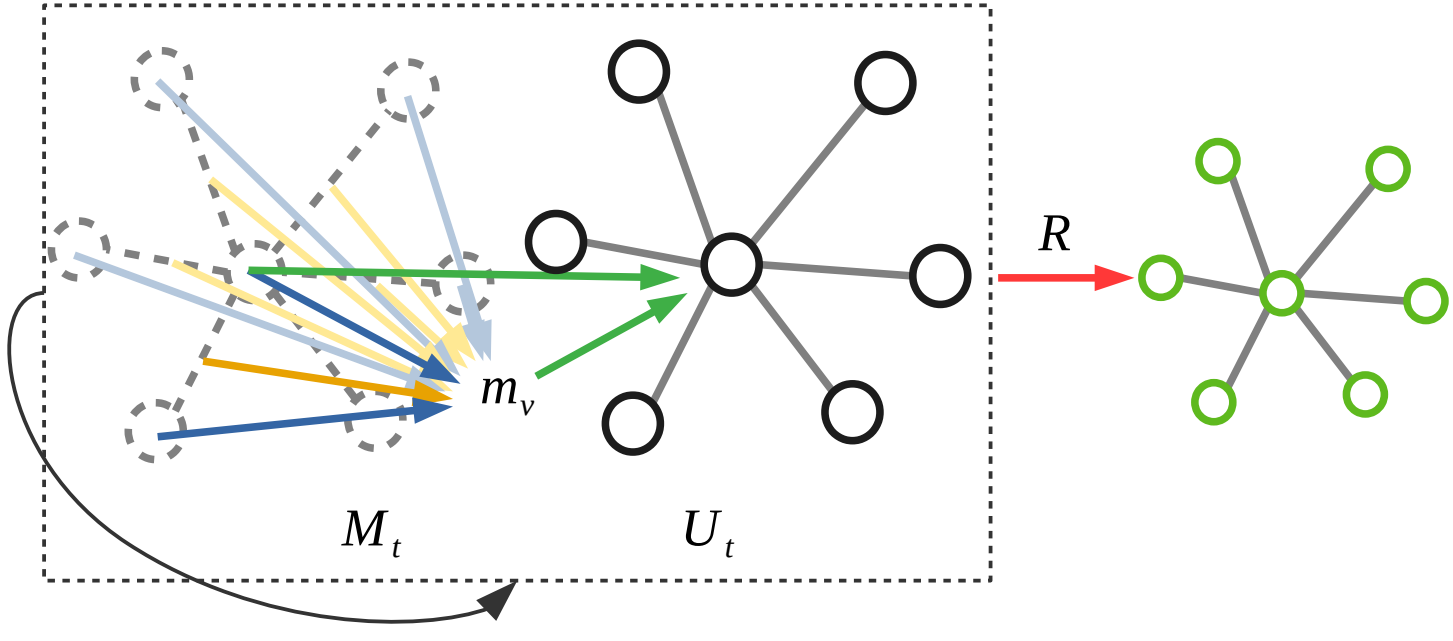


Composition-based GCN

Message passing neural networks (MPNN)



MPNN



MPNN

Message passing phase

$$m_v^{t+1} = \sum_{u \in N(v)} M_t(h_v^t, h_u^t, e_{uv})$$

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$$

h_v^t: vertex features, e_{uv}: edge features, $N(v)$: neighbors of vertex v

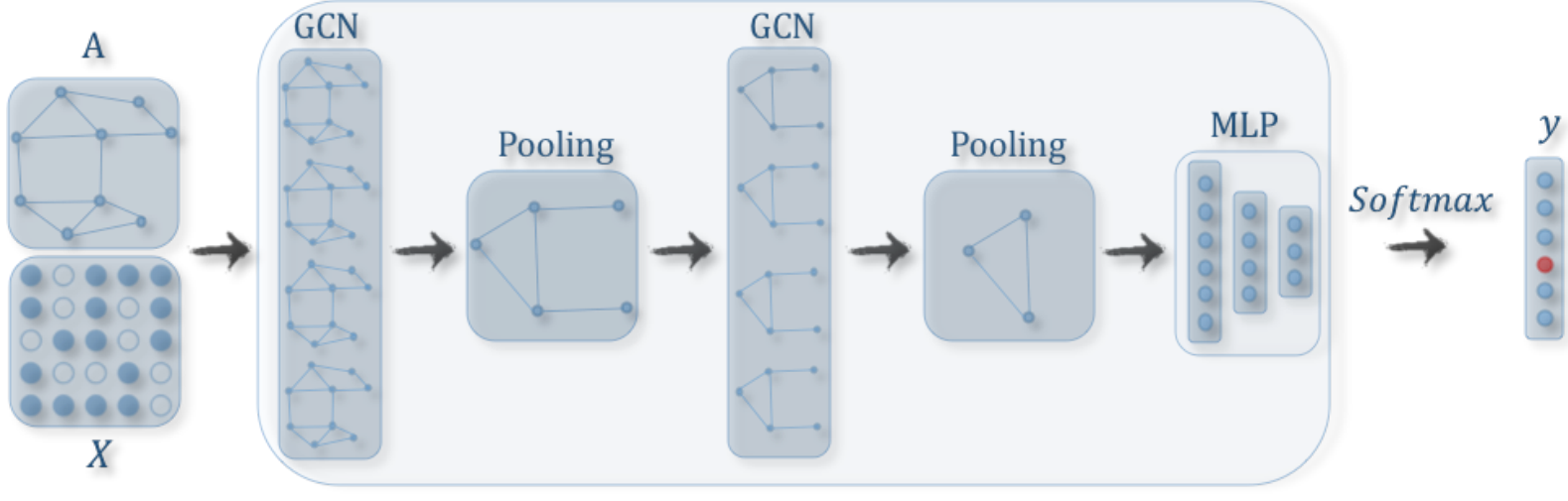
Readout phase

$$\hat{y} = R(h_v)$$

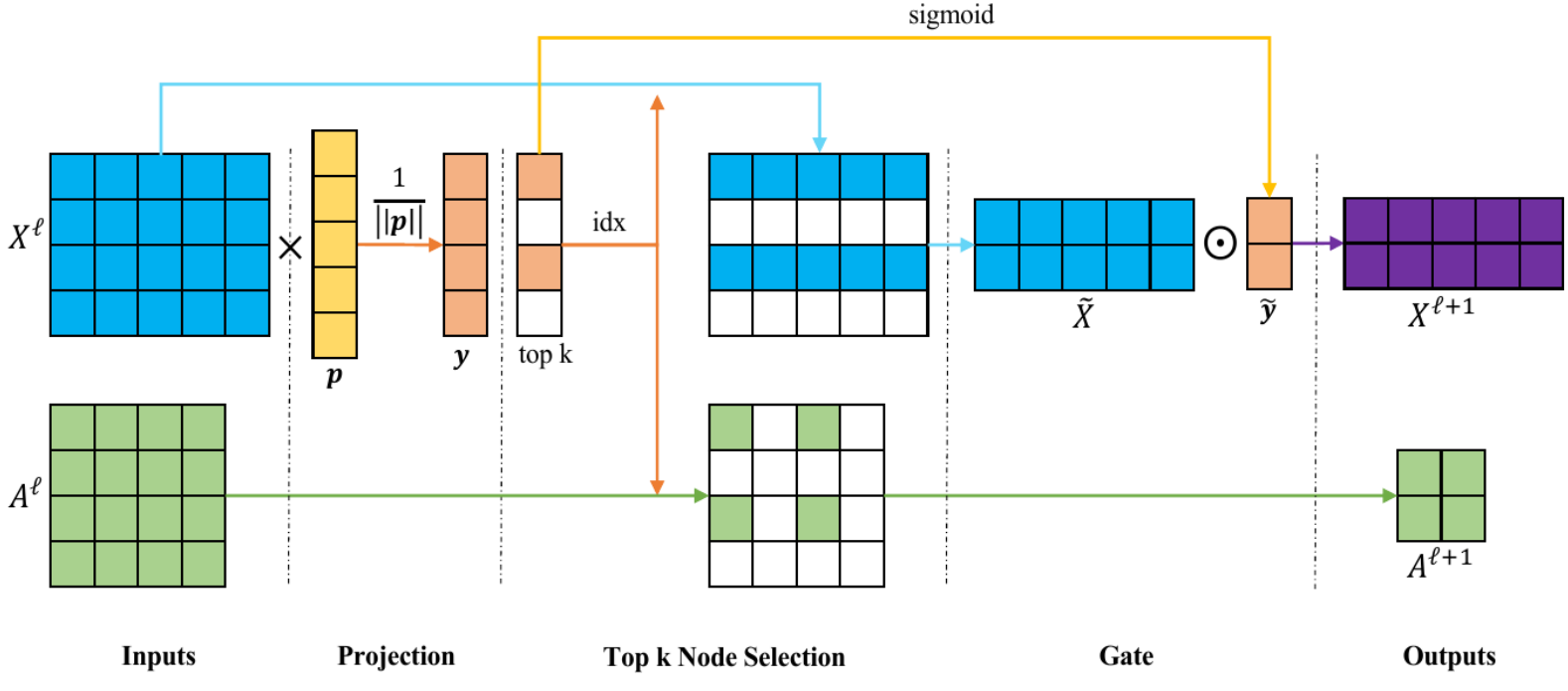
Graph pooling module

mean/max/sum pooling

Reducing number of vertecies of a graph.



Top-K pooling



[Gao, Hongyang, Ji, & Shuiwang. (2019, May 11). Graph U-Nets. (https://arxiv.org/abs/1905.05178)](https://arxiv.org/abs/1905.05178)

Graph attention networks

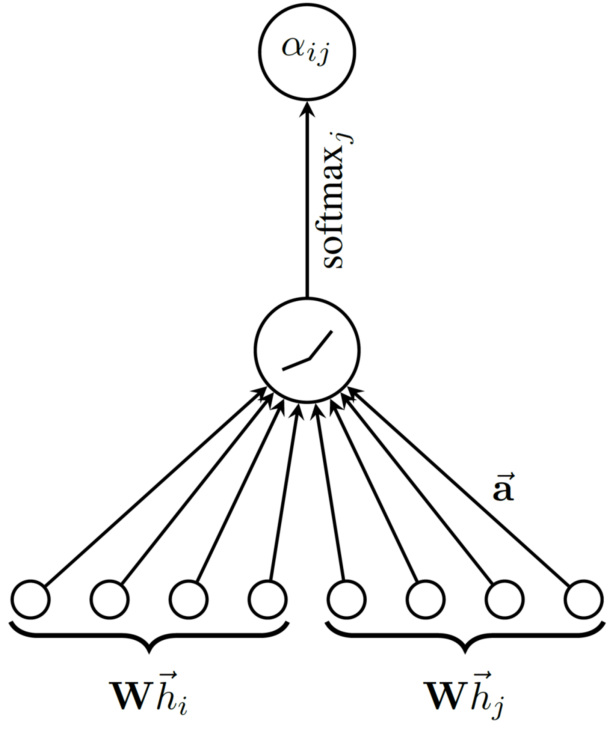
Graph attention network (GAT)

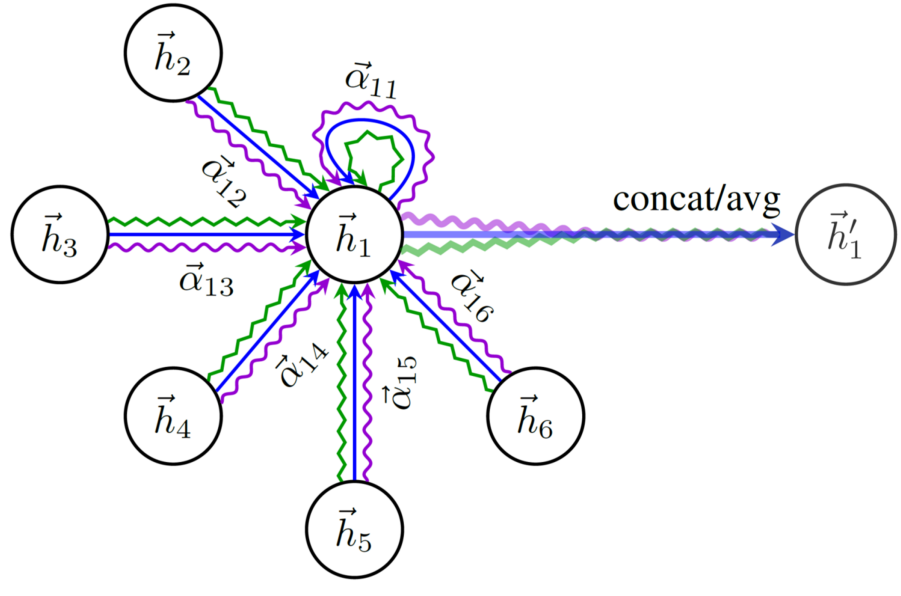
$$h_i^t = \sigma(\sum_j \alpha(h_i^{t-1}, h_j^{t-1}) W^{t-1} h_j^{t-1})$$

Multi-head attention

$$h_i^t = ||_{k=1}^K \sigma(\sum_j \alpha_k(h_i^{t-1}, h_j^{t-1}) W_k^{t-1} h_j^{t-1})$$

GAT


α_{ij}
softmax$_j$
$\vec{\mathbf{a}}$
$\mathbf{W}\vec{h}_i$
$\mathbf{W}\vec{h}_j$


$\vec{h}_2$
$\vec{\alpha}_{11}$
$\vec{\alpha}_{12}$
$\vec{h}_3$
$\vec{h}_1$
concat/avg
$\vec{h}'_1$
$\vec{\alpha}_{13}$
$\vec{\alpha}_{16}$
$\vec{\alpha}_{14}$
$\vec{\alpha}_{15}$
$\vec{h}_4$
$\vec{h}_6$
$\vec{h}_5$

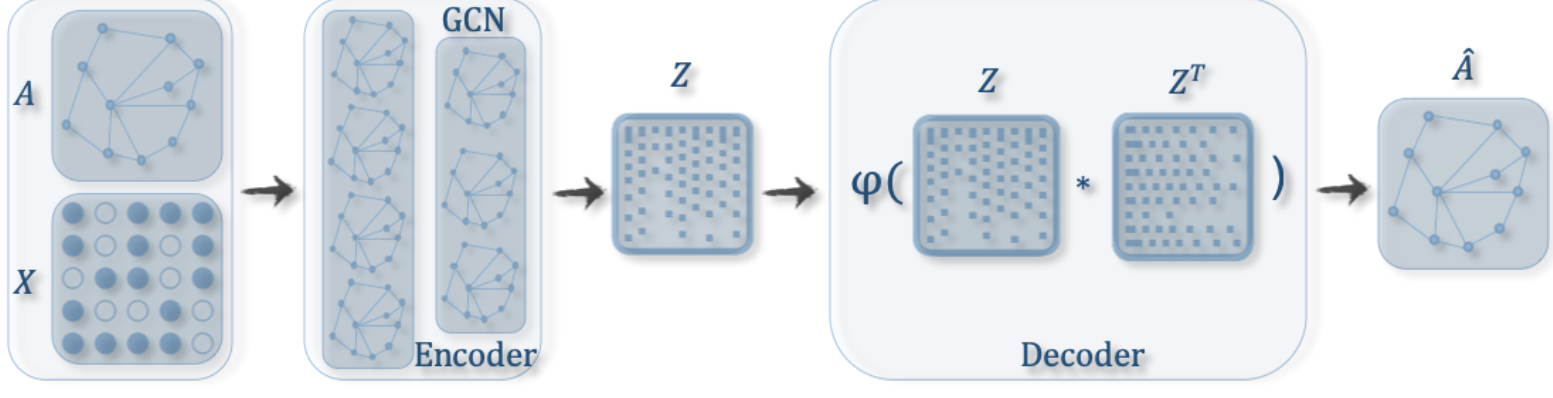
Graph autoencoder

Graph autoencoder (GAE)

$$Z = GCN(X, A) = \mathrm{ReLU}(\tilde{A}XW_0)$$
$$\hat{A} = \sigma(ZZ^T)$$

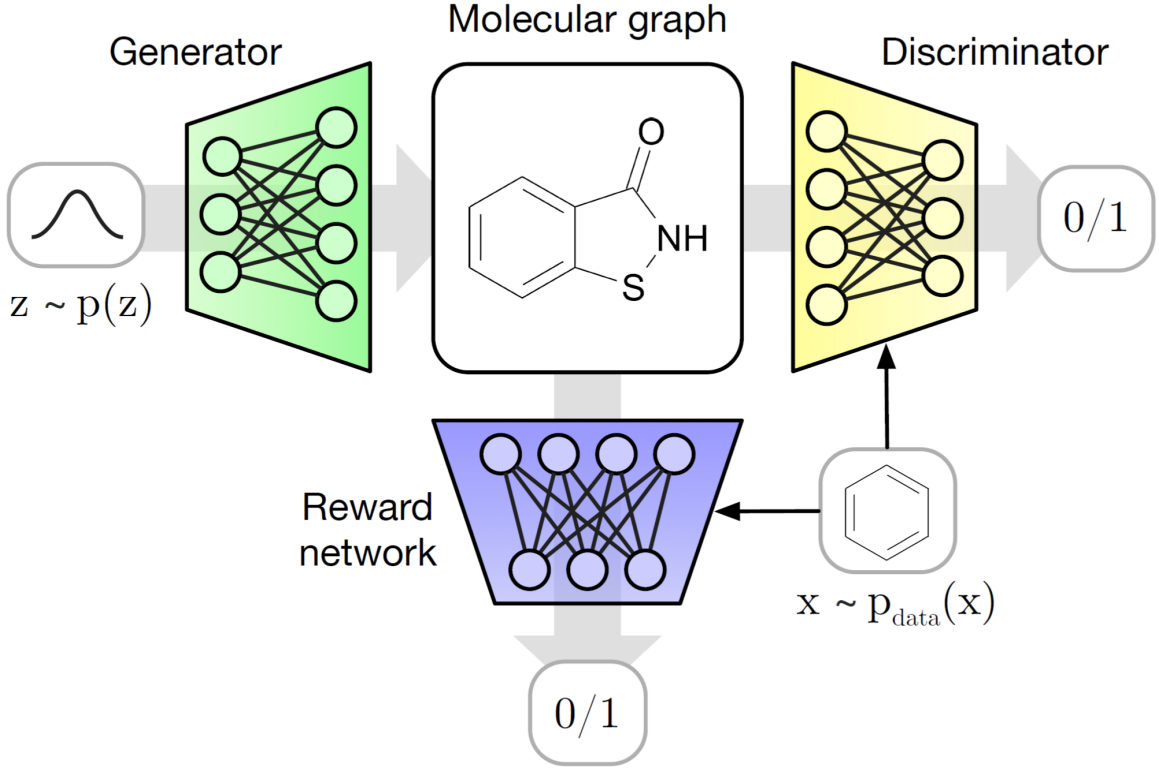
Variational graph autoencoder (VGAE)

$$\mu = GCN_{\mu}(X, A) = \tilde{A}\,\mathrm{ReLU}(\tilde{A}XW_0)\,W_{\mu}$$
$$\log\sigma = GCN_{\sigma}(X, A) = \tilde{A}\,\mathrm{ReLU}(\tilde{A}XW_0)\,W_{\sigma}$$
$$q(Z \mid X, A) = \prod_{i=1}^{N} \mathcal{N}(z_i \mid \mu_i, diag(\sigma_i^2))$$
$$p(A \mid Z) = \sigma(ZZ^T)$$

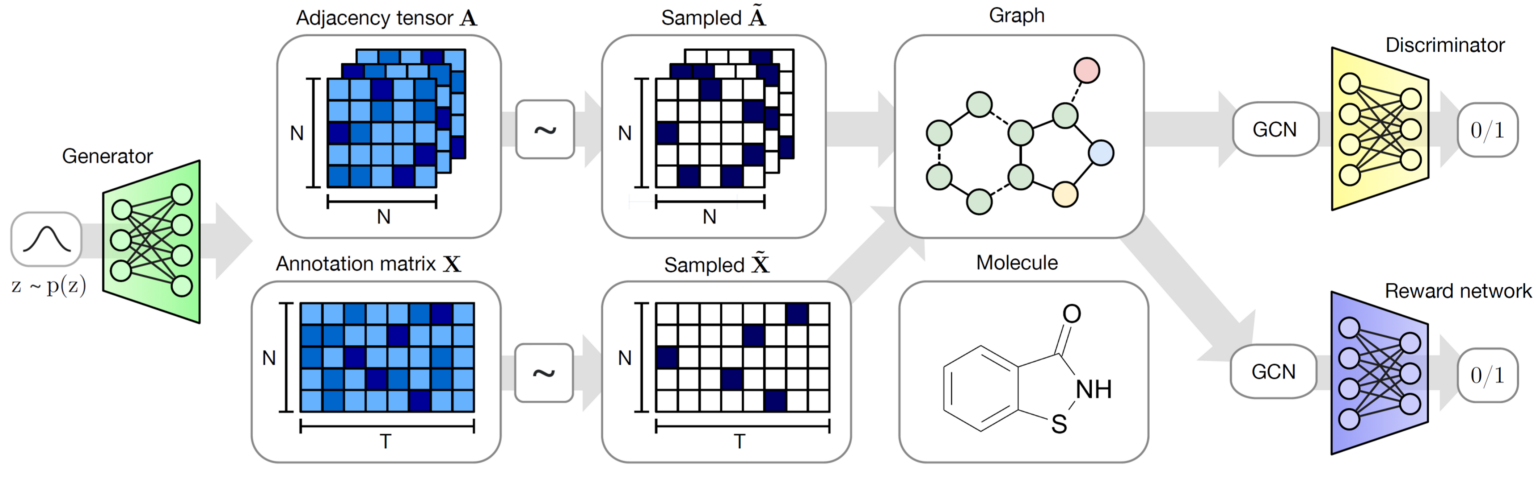


Graph generative networks

MolGAN

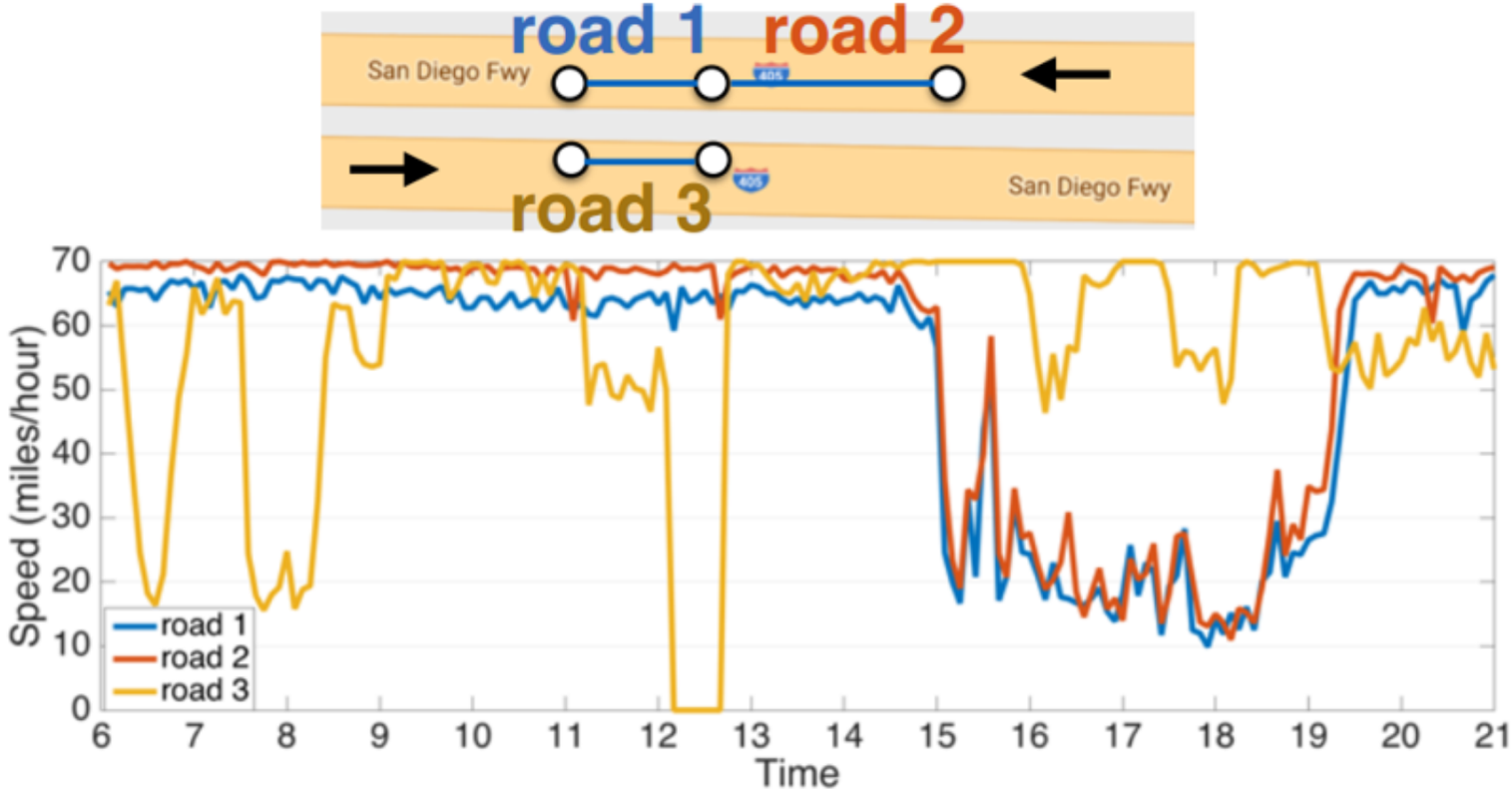


MolGAN


Generator
z ~ p(z)
Adjacency tensor A
N
N
Annotation matrix X
N
T
Sampled Ã
N
N
Sampled X̃
N
T
Graph
Molecule
O
NH
S
GCN
GCN
Discriminator
0/1
Reward network
0/1

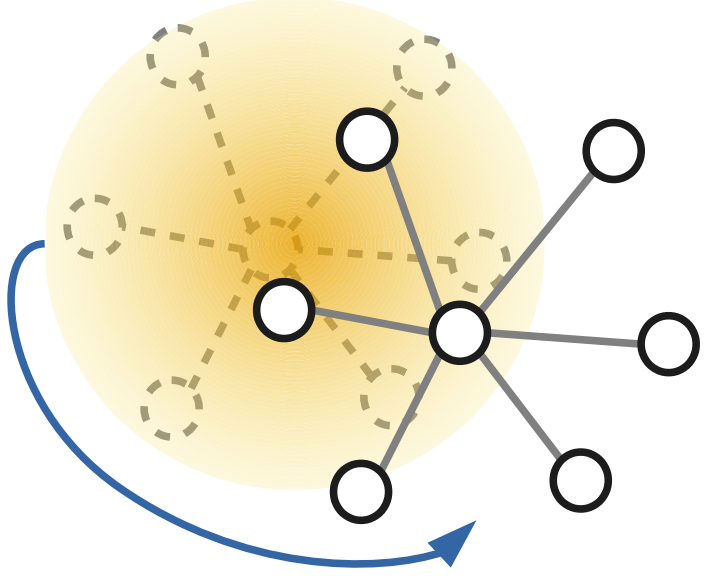
Graph spatial-temporal networks

Diffusion convolutional recurrent neural network (DCRNN)



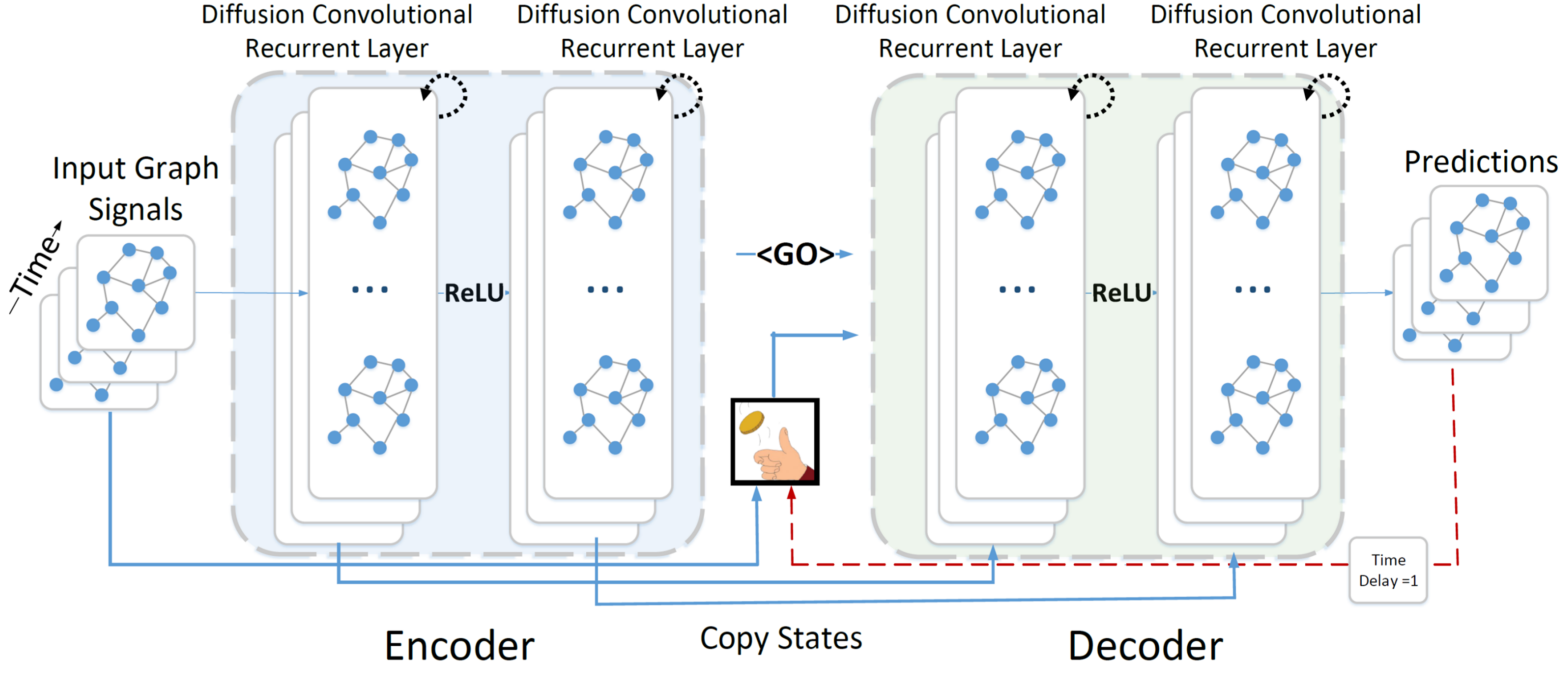
DCRNN

diffusion convolution + sequence to sequence architecture + scheduled sampling technique



DCRNN

diffusion convolution + sequence to sequence architecture + scheduled sampling technique



Diffusion Convolution

Diffusion convolution

$$X *_G f_\theta = \sum_{k=0}^{K-1} (\theta_{k1} (D_O^{-1} A)^k + \theta_{k2} (D_I^{-1} A^T)^k) X$$

$\theta \in \mathbb{R}^{K \times 2}$: (training) parameter for filter

$D_O^{-1} A, D_I^{-1} A^T$: transition matrix for diffusion process and reverse one

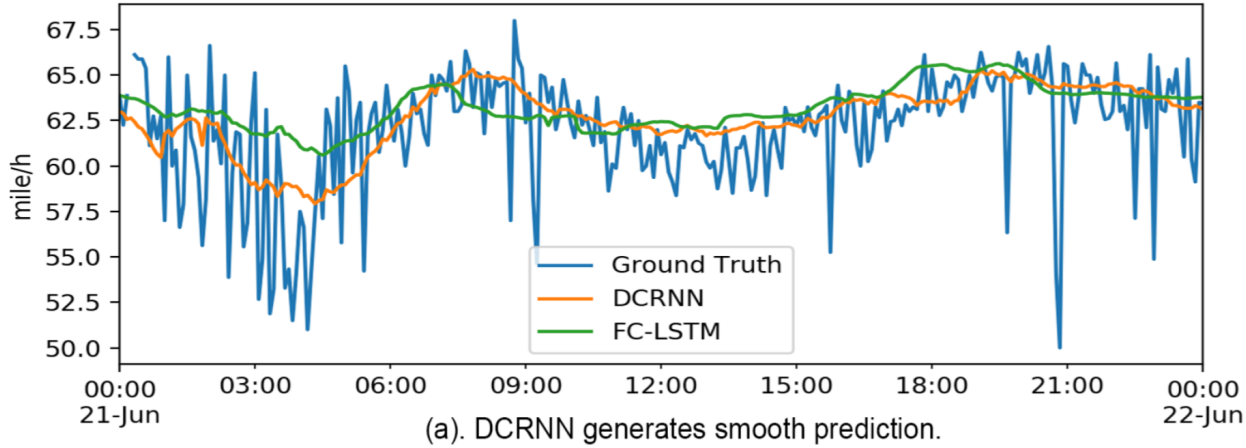
Diffusion convolution layer

$$H = \sigma(\sum_{p=1}^{P} X_p *_G f_{\theta_p})$$

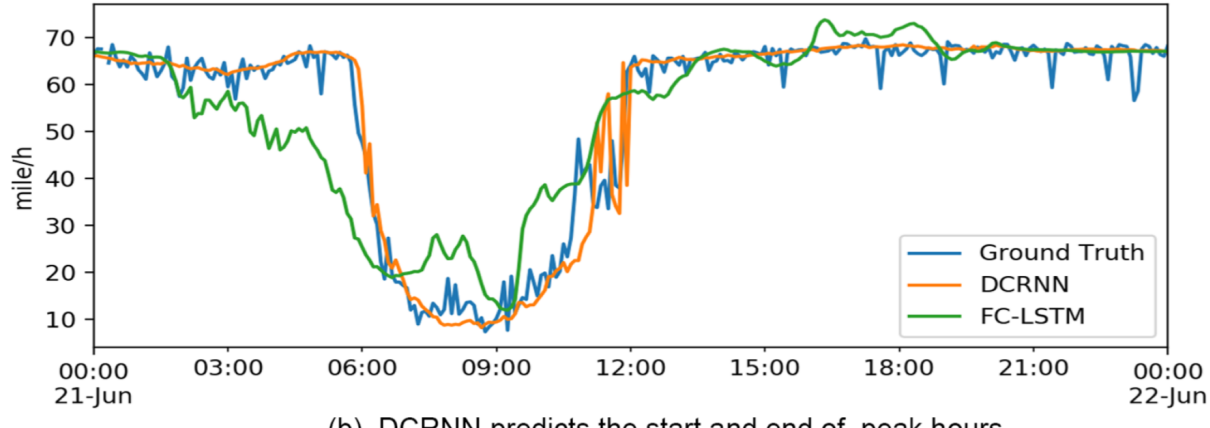
$X \in \mathbb{R}^{N \times P}$: input features

$H \in \mathbb{R}^{N \times Q}$: output

DCRNN

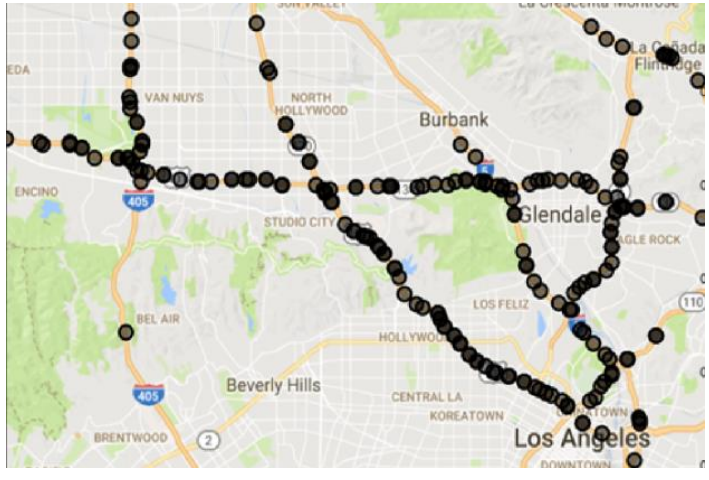


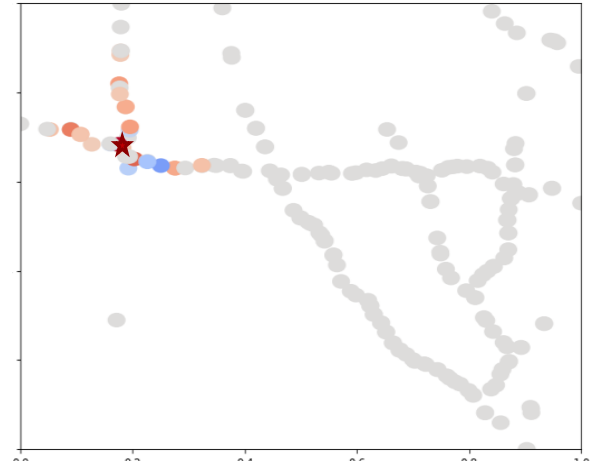
(a). DCRNN generates smooth prediction.

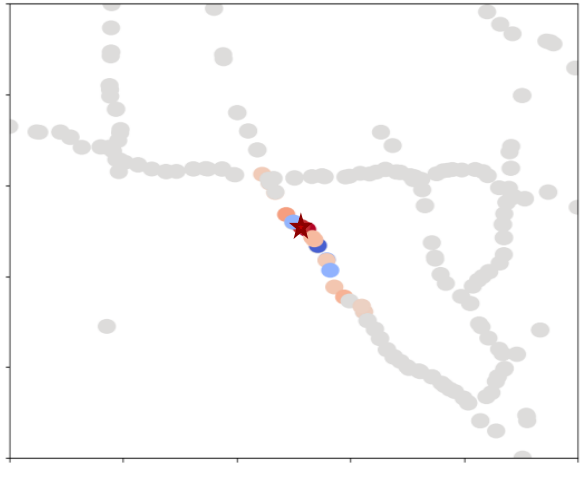


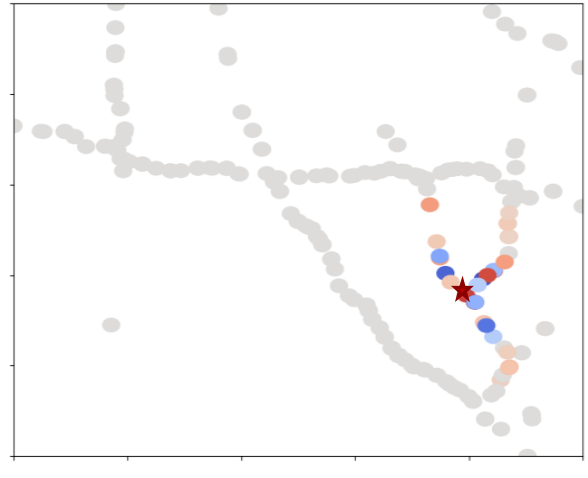
(b). DCRNN predicts the start and end of peak hours.

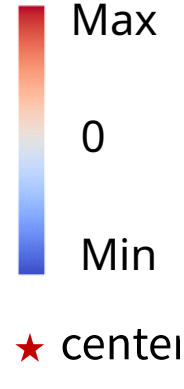
DCRNN


VAN NUYS
NORTH HOLLYWOOD
Burbank
ENCINO
405
Glendale
STUDIO CITY
LOS FELIZ
BEL AIR
HOLLYWOOD
Beverly Hills
CENTRAL LA
KOREATOWN
BRENTWOOD
Los Angeles

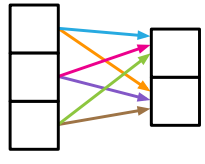




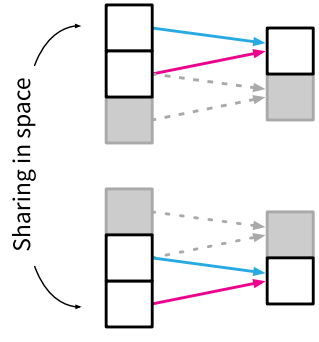



Max
0
Min
★ center

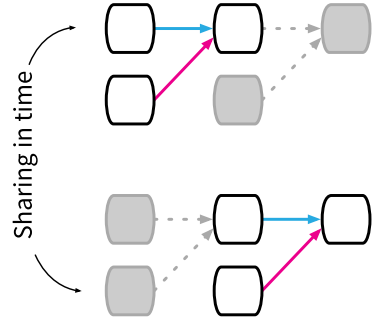
Relational inductive bias



(a) Fully connected

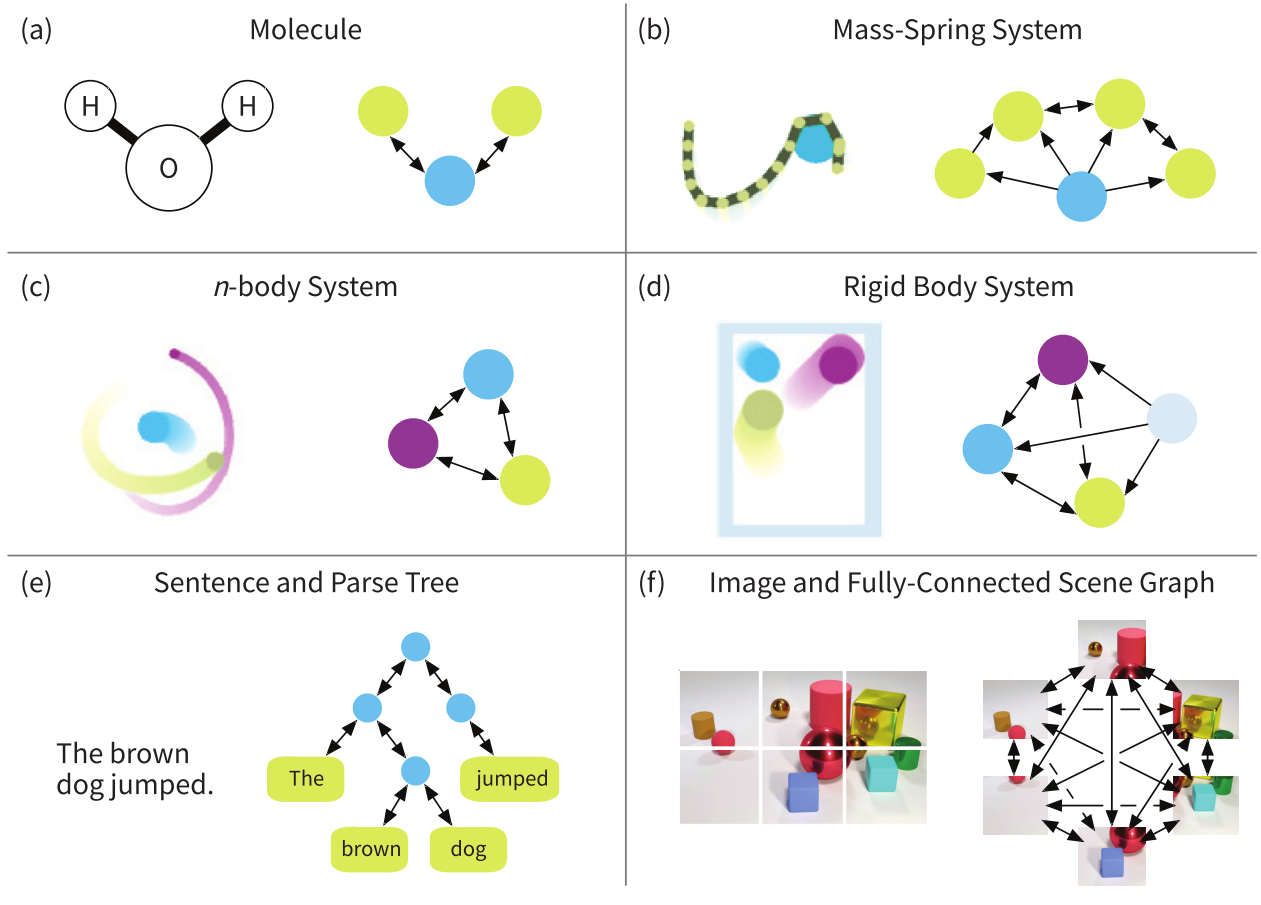


(b) Convolutional

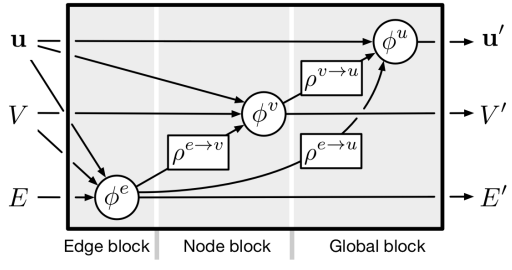


(c) Recurrent

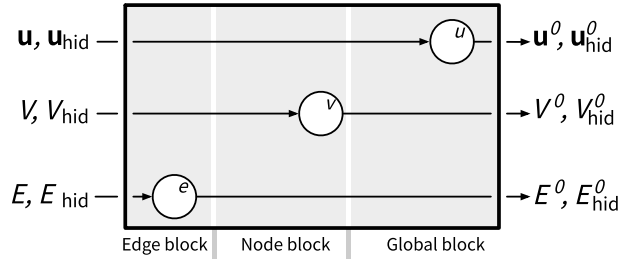
Graph as model of relations



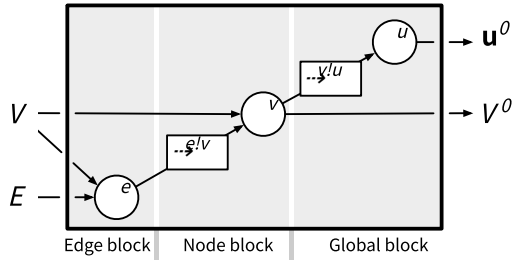
Graph network (GN) block



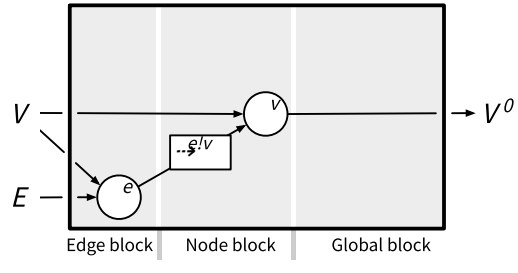
(a) Full GN block



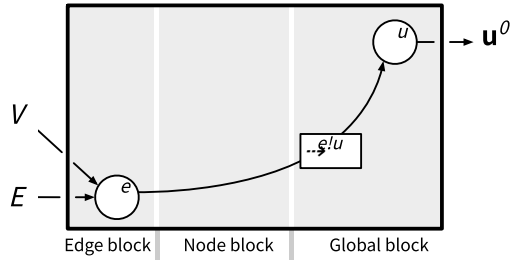
(b) Independent recurrent block



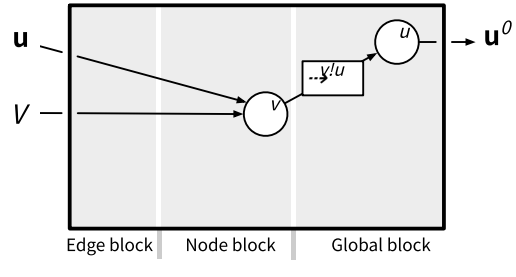
(c) Message-passing neural network



(d) Non-local neural network

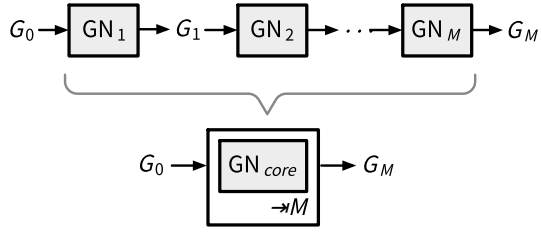


(e) Relation network

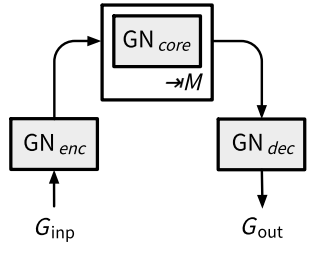


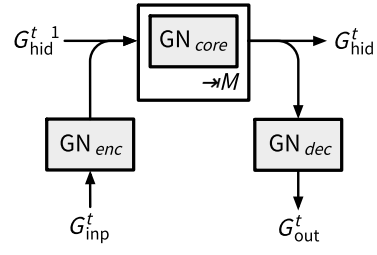
(f) Deep set

Graph network



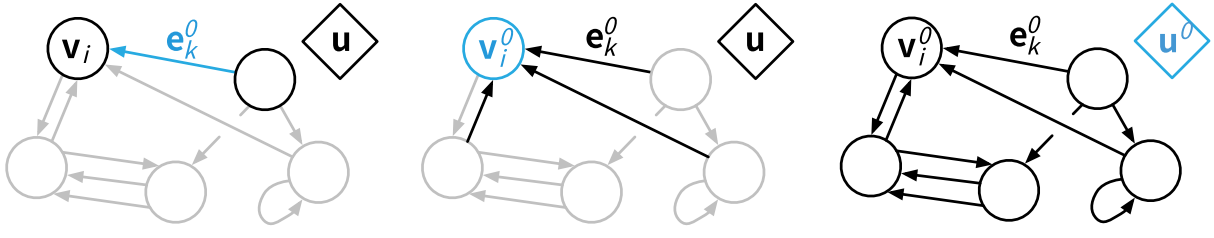
(a) Composition of GN blocks





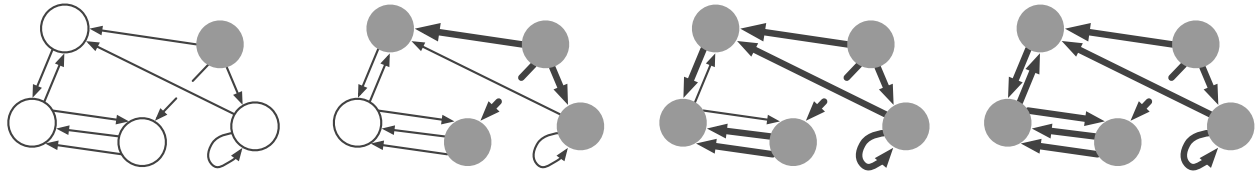
(b) Encode-process-decode(c) Recurrent GN architecture

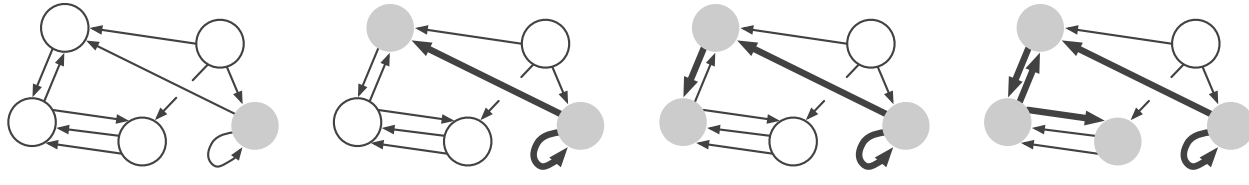
Update of graph network



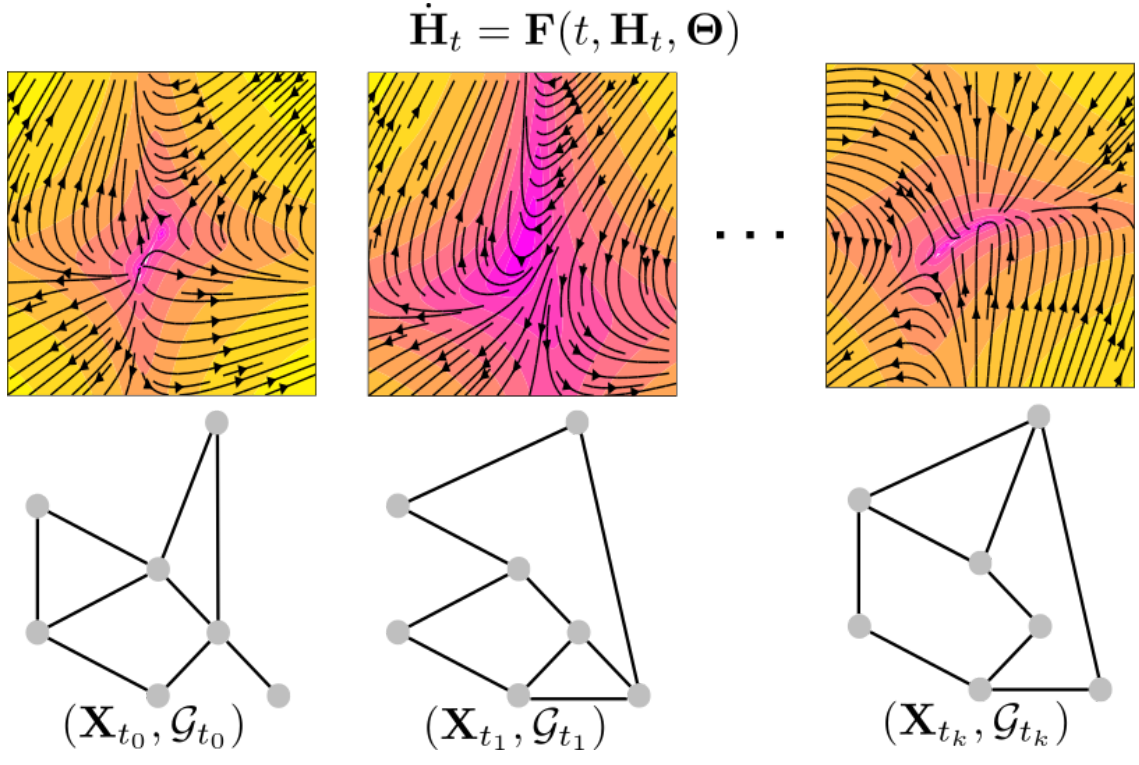
(a) Edge update (b) Node update (c) Global update

Update of graph network





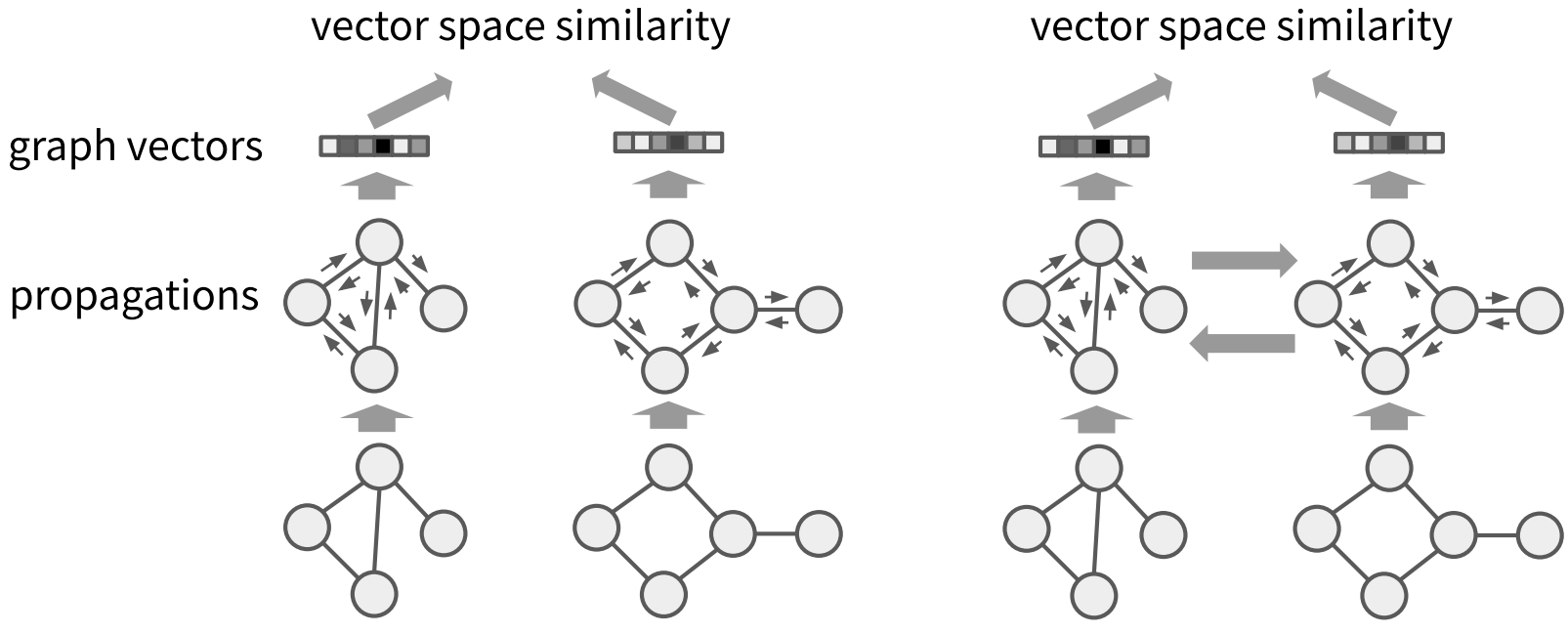
Graph Neural ODE



Michael, Stefano, Yamashita, Atsushi, Hajime, & Jinkyoo. (2019, November 18). Graph Neural Ordinary Differential Equations. (https://arxiv.org/abs/1911.07532)

Applications

Graph matching network

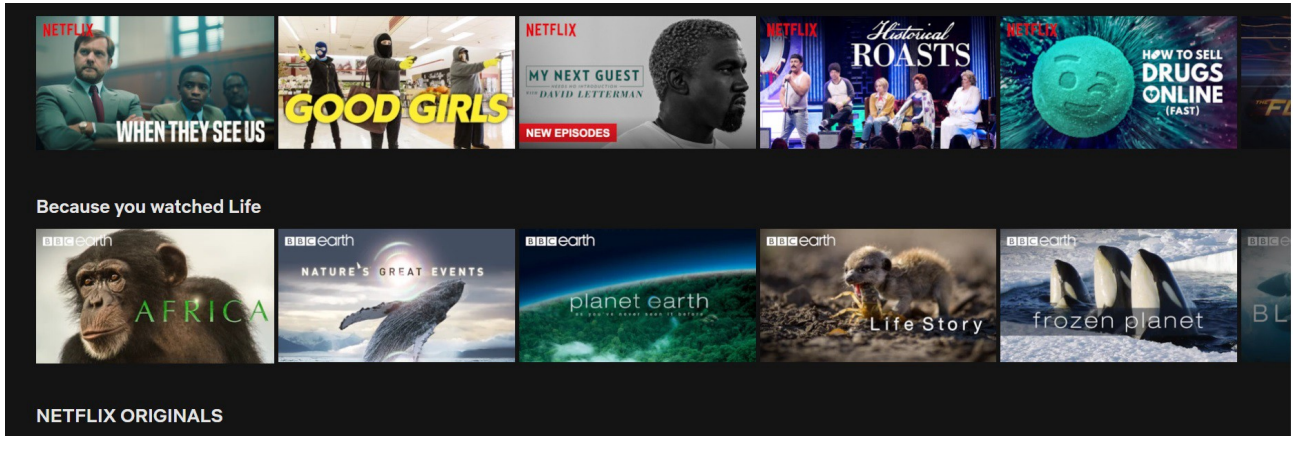


Li, Yujia, Gu, Thomas, Kohli, Pushmeet, & Oriol. (2019, May 12). Graph Matching Networks for Learning the Similarity of Graph Structured Objects. (https://arxiv.org/abs/1904.12787)

Recommender system

Matrix completion problem

Really sparse matrix (Netflix with 480k movies x 18k users with 0.011% known entries)



picture source (https://medium.com/@sajad1009/netflix-recommendation-system-with-pyspark-3c25121a9144)

Recommender system

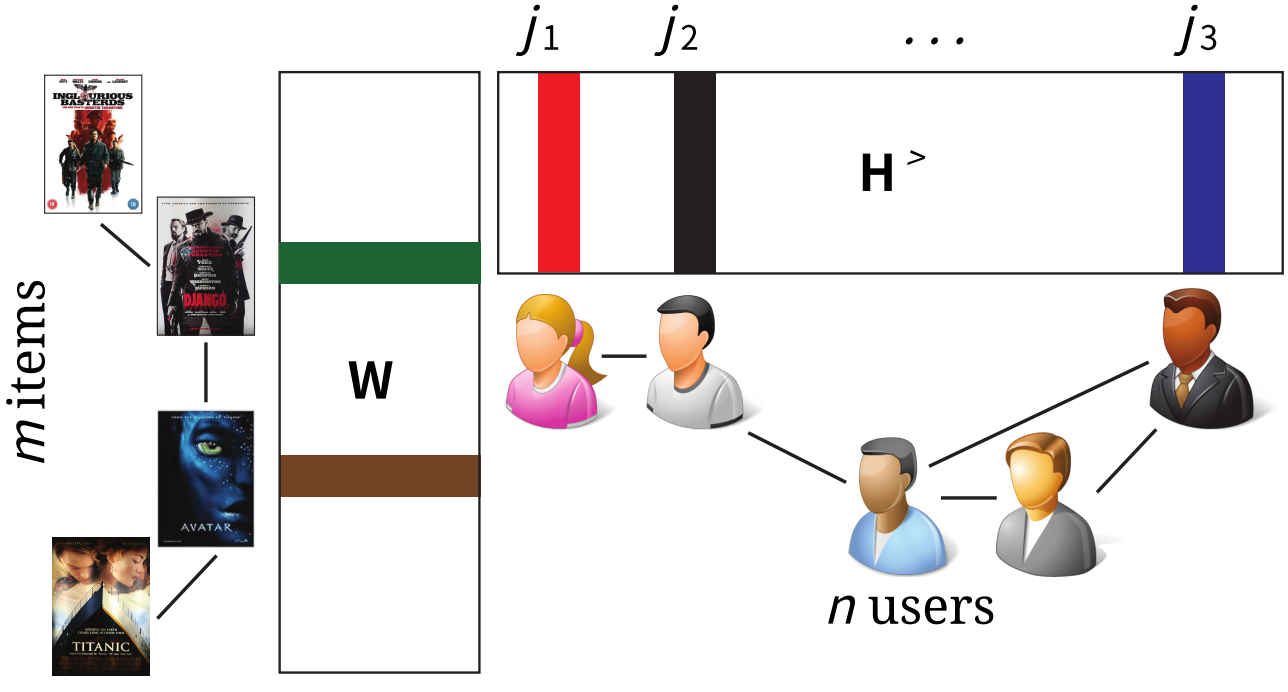
Collaborative filtering

Use colleted product ratings from users to offer recommendations.

Content filtering

Use similarity between products and users to recommend products.

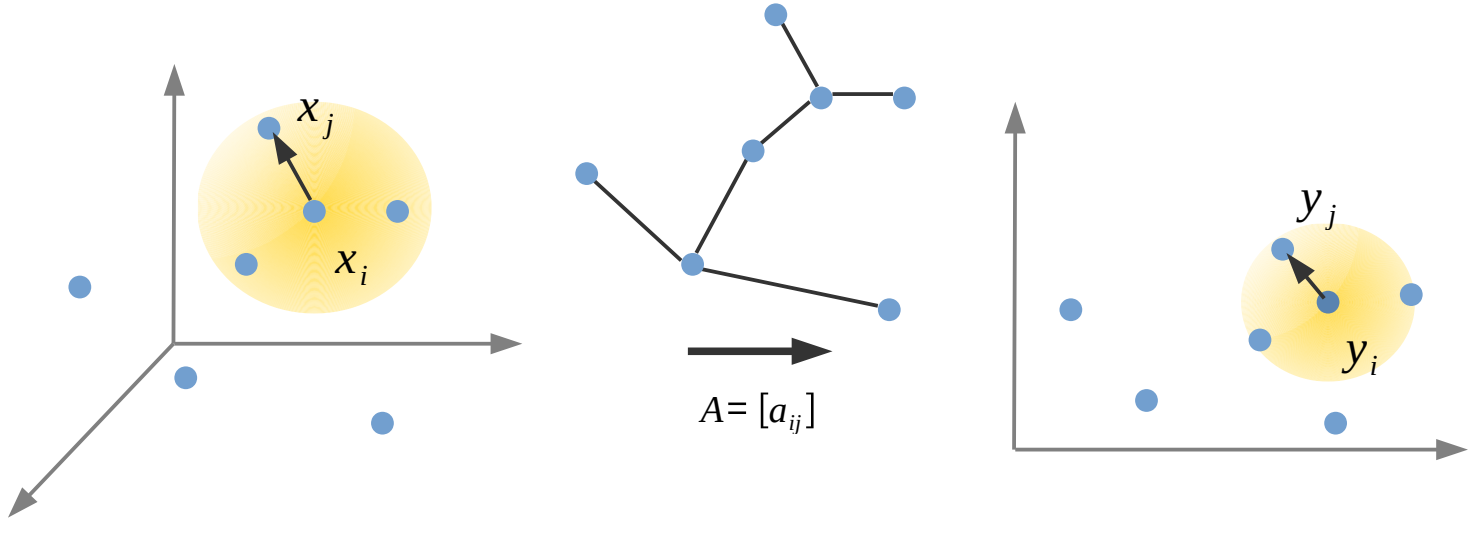
Recommender system



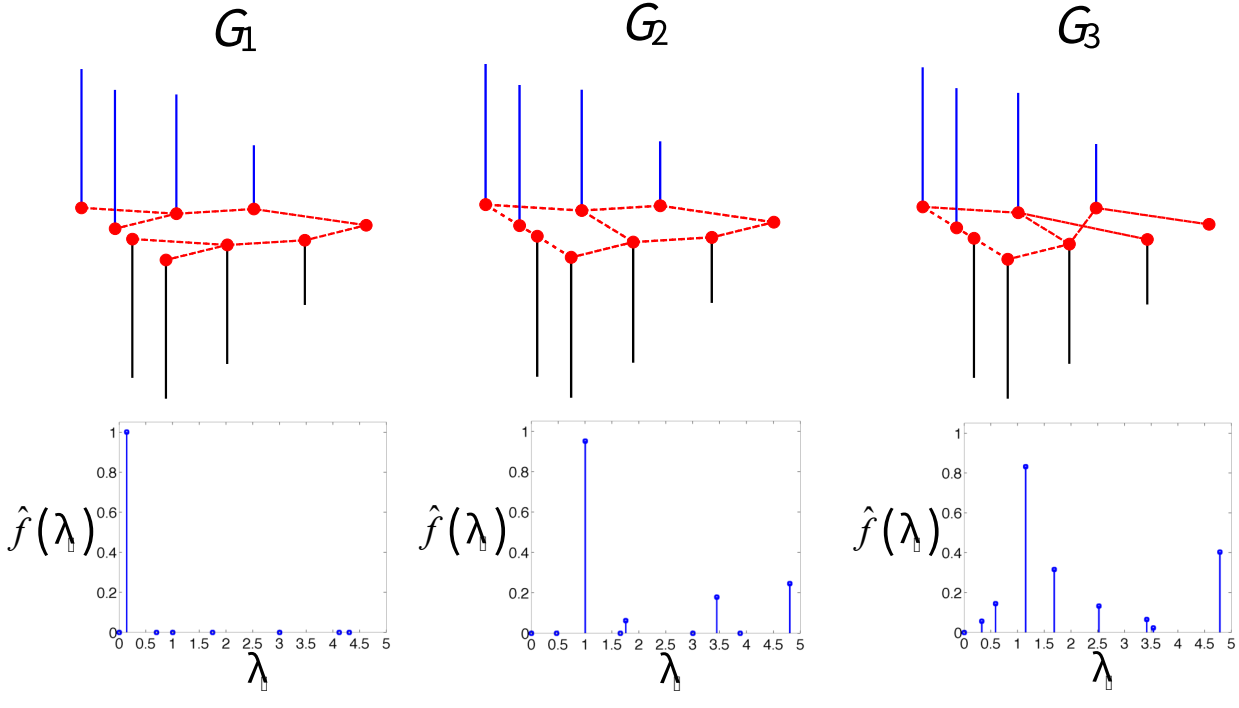
Monti, Federico, Bronstein, M., M., Bresson, & Xavier. (2017, April 22). Geometric Matrix Completion with Recurrent Multi-Graph Neural Networks. (https://arxiv.org/abs/1704.06803)

Connection to other fields

Dimensional Reduction: Geometric deep learning layer acts as dimension reduction via graph



How important the underlying graph is?



Physics: Heat diffusion process

From Newton's law of cooling...

$G = (\mathcal{V}, \mathcal{E}), \mathcal{V} \to \infty, \mathcal{E} \to \infty$ $\phi_i : \mathcal{V} \to \mathbb{R}^n$, heat at vertex i

ϕ, heat distribution across whole graph

$$\frac{d\phi_i}{dt} = -k \sum_j W_{ij}(\phi_i - \phi_j)$$

k: heat capacity

$$= -k(\phi_i \sum_j W_{ij} - \sum_j W_{ij}\phi_j)$$

$$= -k(\phi_i deg(i) - \sum_j W_{ij}\phi_j)$$

Heat equation on graph

$$= -k(\phi_i deg(i) - \sum_j W_{ij}\phi_j)$$

$$= -k\sum_j(\delta_{ij}deg(i) - W_{ij})\phi_j$$

$$\frac{d\phi}{dt} = -k(D - W)\phi$$
$$= -kL\phi$$

Laplacian matrix as discrete Laplace operator

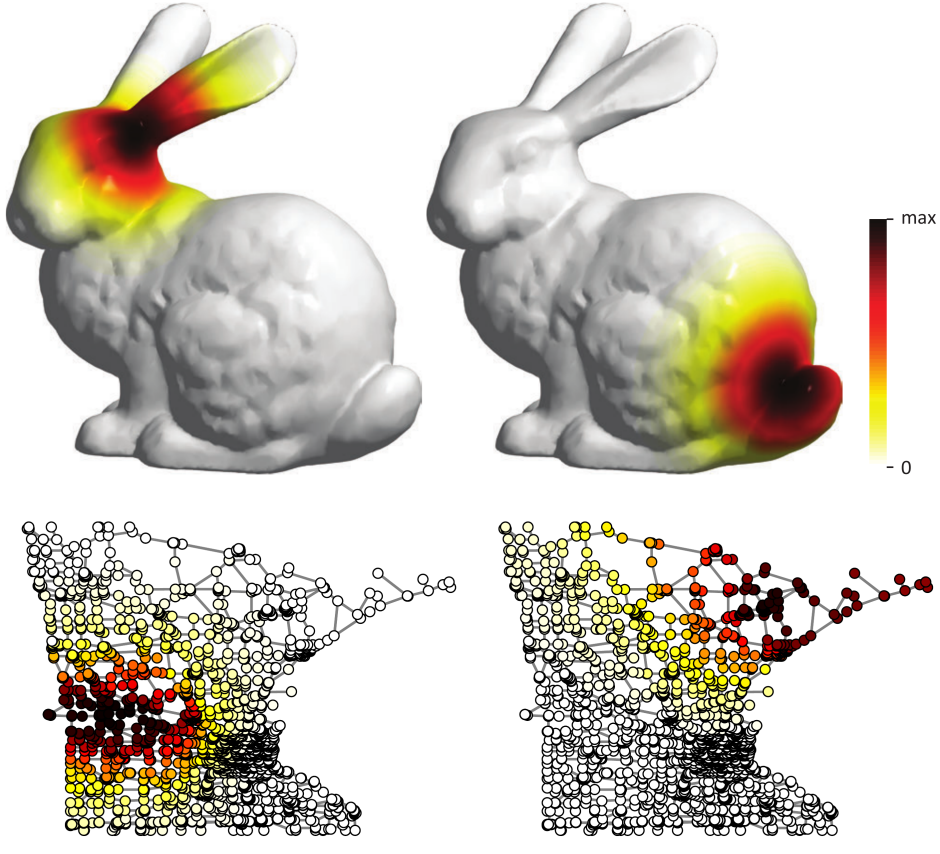
Graph Laplacian

$$\frac{d\phi}{dt} + kL\phi = 0$$

Heat equation

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u$$

Graphs are discrete approximation to manifold

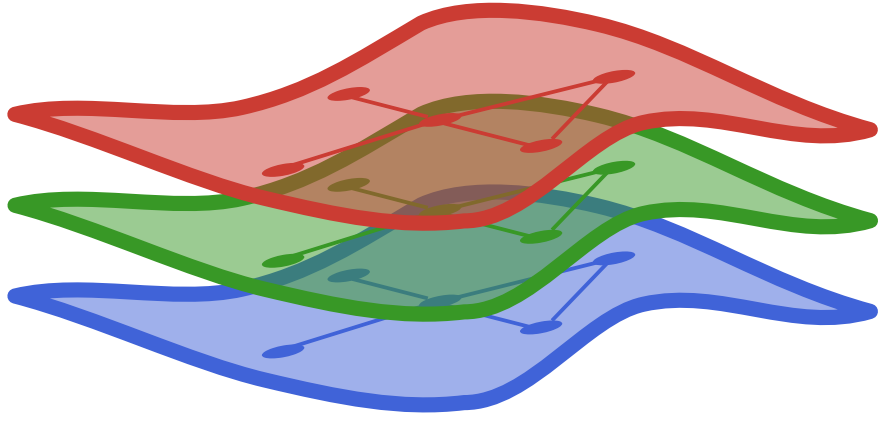


Geometric deep learning: going beyond Euclidean data (https://arxiv.org/abs/1611.08097)

GeometricFlux.jl: Geometric Deep Learning framework in Julia

GeometricFlux

https://github.com/yuehhua/GeometricFlux.jl
(https://github.com/yuehhua/GeometricFlux.jl)



Aims

- extend **Flux** deep learning framework in Julia
- support of CUDA GPU with **CuArrays** (other interfaces are welcome)
- integrate with existing **JuliaGraphs** ecosystem

Graphs are usually sparse

I worked on SparseArrays and CuArrays.CUSPARSE

Layers

- Convolution layers
    - MessagePassing
    - GCNConv
    - GraphConv
    - ChebConv
    - GatedGraphConv
    - GATConv
    - EdgeConv
- Pooling layers
    - GlobalPool
    - TopKPool (WIP)
    - MaxPool
    - MeanPool
    - sum/sub/prod/div/max/min/mean pool
- Embedding layers
    - InnerProductDecoder

Models

- VGAE
- GAE

Compatible with layers in Flux

In [ ]:
```
## Model
model = Chain(GCNConv(g, num_features=>1000, relu),
              GCNConv(g, 1000=>500, relu),
              Dense(500, 7),
              softmax)
```

Use it as you use Flux

In [ ]:

```
## Loss
loss(x, y) = crossentropy(model(x), y)
accuracy(x, y) = mean(onecold(model(x)) .== onecold(y))

## Training
ps = Flux.params(model)
train_data = [(train_X, train_y)]
opt = ADAM(0.01)
evalcb() = @show(accuracy(train_X, train_y))

Flux.train!(loss, ps, train_data, opt, cb=throttle(evalcb, 10))
```

Construct layers from SimpleGraph/SimpleWeightedGraph

In [ ]:

```
g = SimpleGraph(5)
add_edge!(1, 2); add_edge!(3, 4)
GCNConv(g, num_features=>1000, relu)
```

Use Zygote

Use message passing scheme

$$m_v^{t+1} = \sum_{u \in N(v)} M_t(h_v^t, h_u^t, e_{uv})$$

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$$

h_v^t: vertex features, e_{uv}: edge features, $N(v)$: neighbors of vertex v

Accept changing graph dynamically!

Performance

- Multithreaded scatter function
- Scatter function on GPU
- Threading on CPU (in v1.3!)

Some work to do

- Support CUDA/CUSPARSE
- More layers/models
- Datasets inetegration
- Sparse array computation optimization

Thank you for attention

References

- Graph Signal Processing
    - Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., & Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. IEEE Signal Processing Magazine, 30(3), 83–98. doi: 10.1109/msp.2012.2235192
    - Ortega, A., Frossard, P., Kovacevic, J., Moura, J. M. F., & Vandergheynst, P. (2018). Graph Signal Processing: Overview, Challenges, and Applications. Proceedings of the IEEE, 106(5), 808–828. doi: 10.1109/jproc.2018.2820126

References

- Geometric Deep Learning
    - A Comprehensive Survey on Graph Neural Networks
    - Relational inductive biases, deep learning, and graph networks
    - Geometric deep learning: going beyond Euclidean data
- Models
    - The Graph Neural Network Model
    - Variational Graph Auto-Encoders
    - Neural Message Passing for Quantum Chemistry
    - DIFFUSION CONVOLUTIONAL RECURRENT NEURAL NETWORK: DATA-DRIVEN TRAFFIC FORECASTING
    - GRAPH ATTENTION NETWORKS
    - MolGAN: An implicit generative model for small molecular graphs