



Boulder

Graph Neural Networks



[YouTube Playlist](#)

Maziar Raissi

Assistant Professor

Department of Applied Mathematics

University of Colorado Boulder

maziar.raissi@colorado.edu



Boulder

DeepWalk: Online Learning of Social Representations



[YouTube Video](#)

Learning latent representations of vertices in a network

- BlogCatalog – Flickr – YouTube
- Network Classification – Anomaly Detection
- Content Recommendation – Missing Link Prediction

$G = (V, E)$ → a social network

V → members of the network

$E \subset V \times V$ → connections

$G_L = (V, E, X, Y)$ → partially labeled social network

$X \in \mathbb{R}^{|V| \times S}$ → attributes

S → size of the feature space

$Y \in \mathbb{R}^{|V| \times |\mathcal{Y}|}$

\mathcal{Y} → set of labels

Goal: Learn $X_E \in \mathbb{R}^{|V| \times d}$ with d small

Random Walks

\mathcal{W}_{v_i} → random walk rooted at vertex v_i

\mathcal{W}_{v_i} → stochastic process with random variables $\mathcal{W}_{v_i}^1, \mathcal{W}_{v_i}^2, \dots$

$\mathcal{W}_{v_i}^{k+1}$ → vertex chosen at random from neighbors of vertex $\mathcal{W}_{v_i}^k$

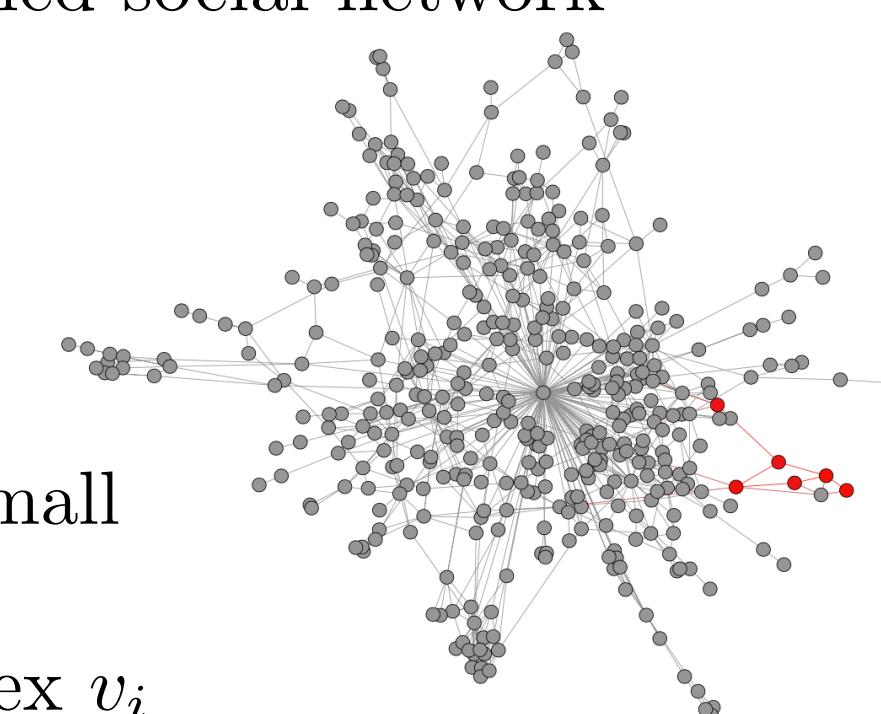
Language Modeling

$\Pr(v_i | v_1, \dots, v_{i-1})$ → probability of observing vertex v_i given all the previous vertices visited so far in the random walk

$\Phi : v \in V \mapsto \mathbb{R}^d$

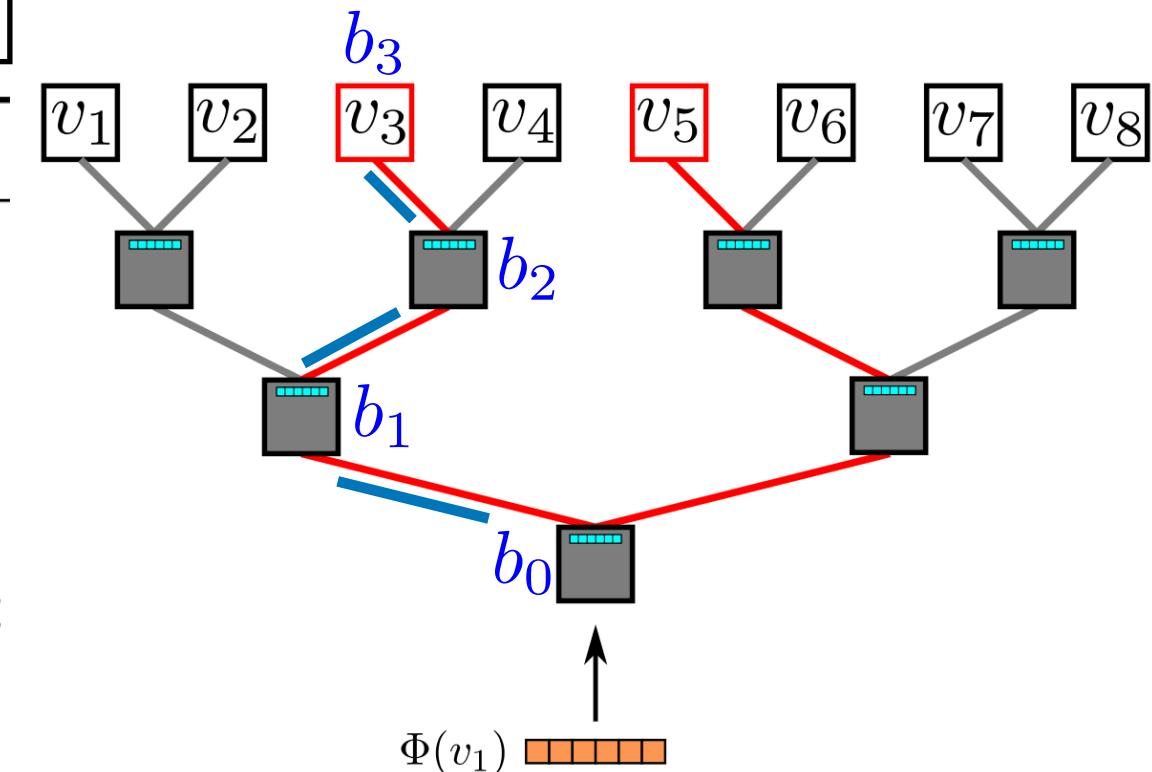
Represent Φ by a matrix of free parameters $X_E \in \mathbb{R}^{|V| \times d}$

$\Pr(v_i | \Phi(v_1), \dots, \Phi(v_{i-1}))$ → unfeasible as the walk length grows



$$\underset{\Phi}{\text{minimize}} \quad -\log \Pr(\{v_{i-w}, \dots, v_{i+w}\} \setminus v_i | \Phi(v_i))$$

Hierarchical Softmax



Algorithm 1 DEEPWALK(G, w, d, γ, t)

Input: graph $G(V, E)$

window size w

embedding size d

walks per vertex γ → “epoch”

walk length t

Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$

1: Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$

2: Build a binary Tree T from V

3: **for** $i = 0$ to γ **do**

4: $\mathcal{O} = \text{Shuffle}(V)$

5: **for each** $v_i \in \mathcal{O}$ **do**

6: $\mathcal{W}_{v_i} = \text{RandomWalk}(G, v_i, t)$ path to $u_k \leftarrow (b_0 = \text{root}, b_1, \dots, b_{\lceil \log |V| \rceil} = u_k)$

7: SkipGram($\Phi, \mathcal{W}_{v_i}, w$)

8: **end for**

9: **end for**

Algorithm 2 SkipGram($\Phi, \mathcal{W}_{v_i}, w$)

1: **for each** $v_j \in \mathcal{W}_{v_i}$ **do**

2: **for each** $u_k \in \mathcal{W}_{v_i}[j-w : j+w]$ **do**

3: $J(\Phi) = -\log \Pr(u_k | \Phi(v_j))$

4: $\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$

5: **end for**

6: **end for**

$$\Pr(\{v_{i-w}, \dots, v_{i+w}\} \setminus v_i | \Phi(v_i)) = \prod_{\substack{j=i-w \\ j \neq i}}^{i+w} \Pr(v_j | \Phi(v_i))$$

independence assumption

$\theta = \{\Phi, \Psi\}$

$$\Pr(u_k | \Phi(v_j)) = \prod_{l=1}^{\lceil \log |V| \rceil} \Pr(b_l | \Phi(v_j))$$

$$\Pr(b_l | \Phi(v_j)) = 1/(1 + e^{-\Phi(v_j) \cdot \Psi(b_l)})$$

sigmoid (binary classifier)

$\Psi(b_\ell) \rightarrow$ representation assigned to b_ℓ

Huffman Coding

assign shorter paths to the frequent vertices in the random walk



Boulder

LINE: Large-scale Information Network Embedding



[YouTube Video](#)

– visualization – node classification – link prediction – recommendation

$G = (V, E)$ → information network (e.g., language, social and citation networks)

V → set of vertices (data objects)

E → set of edges

$e = (u, v)$ → an edge

w_{uv} → weight

$(u, v) = (v, u) \& w_{uv} = w_{vu} \implies$ undirected

– directed (e.g., citation networks)

– undirected (e.g., social networks of users in Facebook)

– weighted (e.g., word co-occurrence networks)

Represent each vertex $v \in V$ by a low-dimensional vector in \mathbb{R}^d

$$f_G : V \rightarrow \mathbb{R}^d, d \ll |V|$$

First-order proximity

$(v_i, v_j) \rightarrow$ undirected edge

$p_1(v_i, v_j) \rightarrow$ joint probability btw vertices v_i & v_j

$$p_1(v_i, v_j) = \sigma(u_i^T u_j)$$

$u_i \in \mathbb{R}^d \rightarrow$ vector representation of vertex v_i

$p_1 \rightarrow$ distribution over $V \times V$

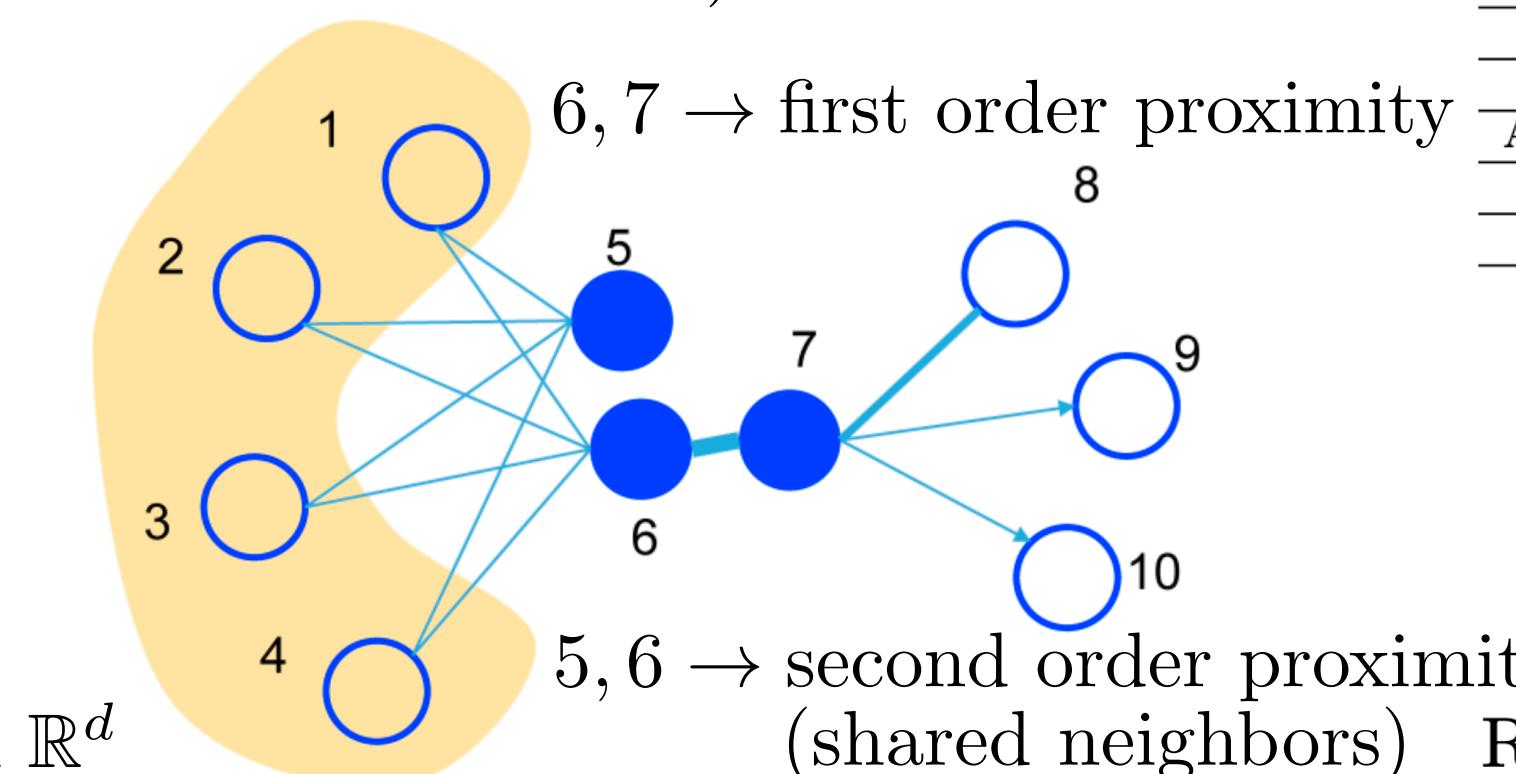
$\hat{p}_1 \rightarrow$ empirical counterpart of p_1

$$\hat{p}_1(v_i, v_j) = \frac{w_{ij}}{W}, \quad W = \sum_{(v_i, v_j) \in E} w_{v_i v_j}$$

$O_1 = d(\hat{p}_1, p_1) \rightarrow$ e.g., KL-divergence

$$O_1 = - \sum_{(v_i, v_j) \in E} w_{v_i v_j} \log p_1(v_i, v_j) \quad \text{ignoring the constants}$$

Tang, Jian, et al. "Line: Large-scale information network embedding." *Proceedings of the 24th international conference on world wide web*. 2015.



	Language Network	Social Network	
Name	WIKIPEDIA	Flickr	YOUTUBE
Type	undirected,weighted	undirected,binary	undirected,binary
V	1,985,098	1,715,256	1,138,499
E	1,000,924,086	22,613,981	2,990,443
Avg. degree	504.22	26.37	5.25
#Labels	7	5	47
#train	70,000	75,958	31,703

Citation Network	
DBLP(AUTHORCITATION)	DBLP(PAPERCITATION)
directed,weighted	directed,binary
524,061	781,109
20,580,238	4,191,677
78.54	10.73
7	7
20,684	10,398

Results of word analogy on WIKIPEDIA data.

Algorithm	Semantic (%)	Syntactic (%)	Overall (%)	Running time
GF	61.38	44.08	51.93	2.96h
DeepWalk	50.79	37.70	43.65	16.64h
SkipGram	69.14	57.94	63.02	2.82h
LINE-SGD(1st)	9.72	7.48	8.50	3.83h
LINE-SGD(2nd)	20.42	9.56	14.49	3.94h
LINE(1st)	58.08	49.42	53.35	2.44h
LINE(2nd)	73.79	59.72	66.10	2.55h

$$O_2 = - \sum_{(v_i, v_j) \in E} w_{v_i v_j} \log p_2(v_j | v_i) \rightarrow \text{setting } \lambda_i = d_i \text{ ignoring the constants}$$

$$\log p_2(v_j | v_i) \approx \log \sigma(\bar{u}_j^T u_i) + \sum_{k=1}^K \mathbb{E}_{v_k \sim P(v_k)} \log \sigma(-\bar{u}_k^T u_i)$$

$K \rightarrow$ number of negative samples (i.e., edges)

$$P(v) \propto d_v^{3/4}$$

$d_v \rightarrow$ out-degree of vertex v

Sample edges according to their weights $(w_1, \dots, w_{|E|})$



Boulder

node2vec: Scalable Feature Learning for Networks



[YouTube Video](#)

$G = (V, E) \rightarrow$ a given network

$f : V \rightarrow \mathbb{R}^d$

a matrix of size $|V| \times d$

$u \in V \rightarrow$ source node

$N_S(u) \subset V \rightarrow$ network neighborhood of node u

$S \rightarrow$ neighborhood sampling strategy

Skip-gram

$$\max_f \sum_{u \in V} \log \Pr(N_S(u) | f(u))$$

Conditional Independence

$$\Pr(N_S(u) | f(u)) = \prod_{n_i \in N_S(u)} \Pr(n_i | f(u))$$

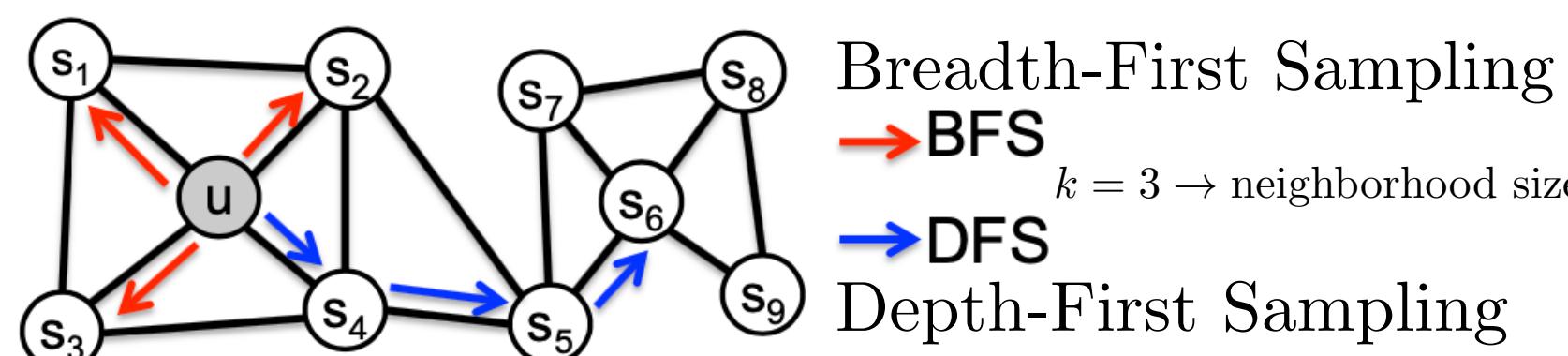
Symmetry in feature space

$$\Pr(n_i | f(u)) = \frac{\exp(f(n_i) \cdot f(u))}{\sum_{v \in V} \exp(f(v) \cdot f(u))}$$

$$\max_f \sum_{u \in V} [-\log Z_u + \sum_{n_i \in N_S(u)} f(n_i) \cdot f(u)]$$

$$Z_u = \sum_{v \in V} \exp(f(u) \cdot f(v)) \rightarrow \text{expensive to compute}$$

approximate using negative sampling



Random Walk

$c_i \rightarrow i\text{-th node in the walk}$

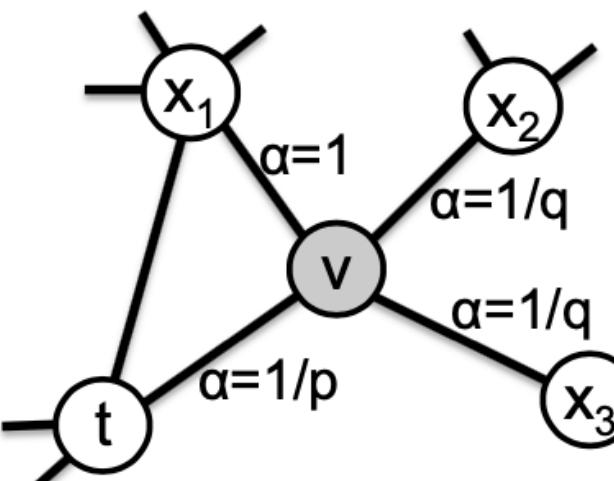
$c_0 = u$

$$p(c_i = x | c_{i-1} = v) = \begin{cases} \frac{1}{z} \pi_{vx} & \text{if } (v, x) \in E \\ 0 & \text{otherwise} \end{cases}$$

$z \rightarrow$ normalization constant

$\pi_{vx} \rightarrow$ unnormalized transition probability
btw nodes v & x

$w_{vx} (= 1) \rightarrow$ edge weights



$$\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}$$

$$\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

A walk that just transitioned from t to v and is now evaluating its next step out of node v .
 $d_{tx} \rightarrow$ shortest path distance between nodes t and x

$$p > q \& p > 1 \implies \frac{1}{p} < 1 \& \frac{1}{p} < \frac{1}{q} \implies$$

lower weight on the prob. of returning to t

$$p < q \& p < 1 \implies \frac{1}{p} > 1 \& \frac{1}{p} > \frac{1}{q} \implies$$

higher weight on the prob. of returning to t

$p \rightarrow$ return parameter

$q > 1 \implies$ local

$q < 1 \implies$ global

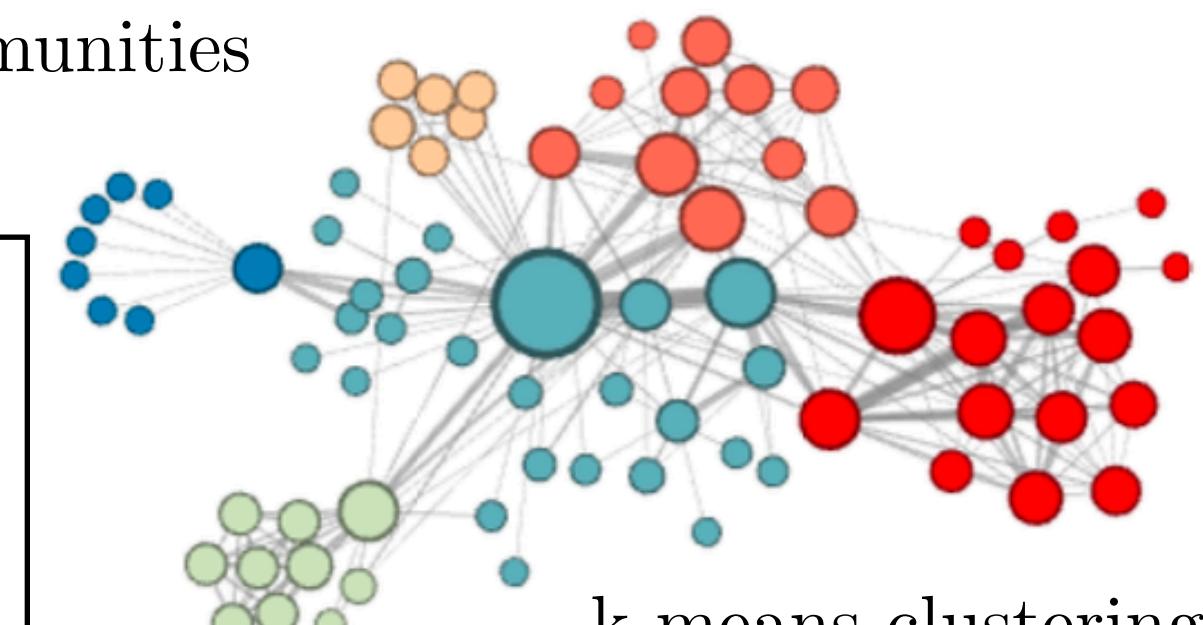
(approximately) interpolate between BFS & DFS

Edge Features

Operator	Symbol	Definition
Average	\boxplus	$[f(u) \boxplus f(v)]_i = \frac{f_i(u) + f_i(v)}{2}$
Hadamard	\boxdot	$[f(u) \boxdot f(v)]_i = f_i(u) * f_i(v)$
Weighted-L1	$\ \cdot\ _1$	$\ f(u) \cdot f(v)\ _{1i} = f_i(u) - f_i(v) $
Weighted-L2	$\ \cdot\ _2$	$\ f(u) \cdot f(v)\ _{2i} = f_i(u) - f_i(v) ^2$

network communities

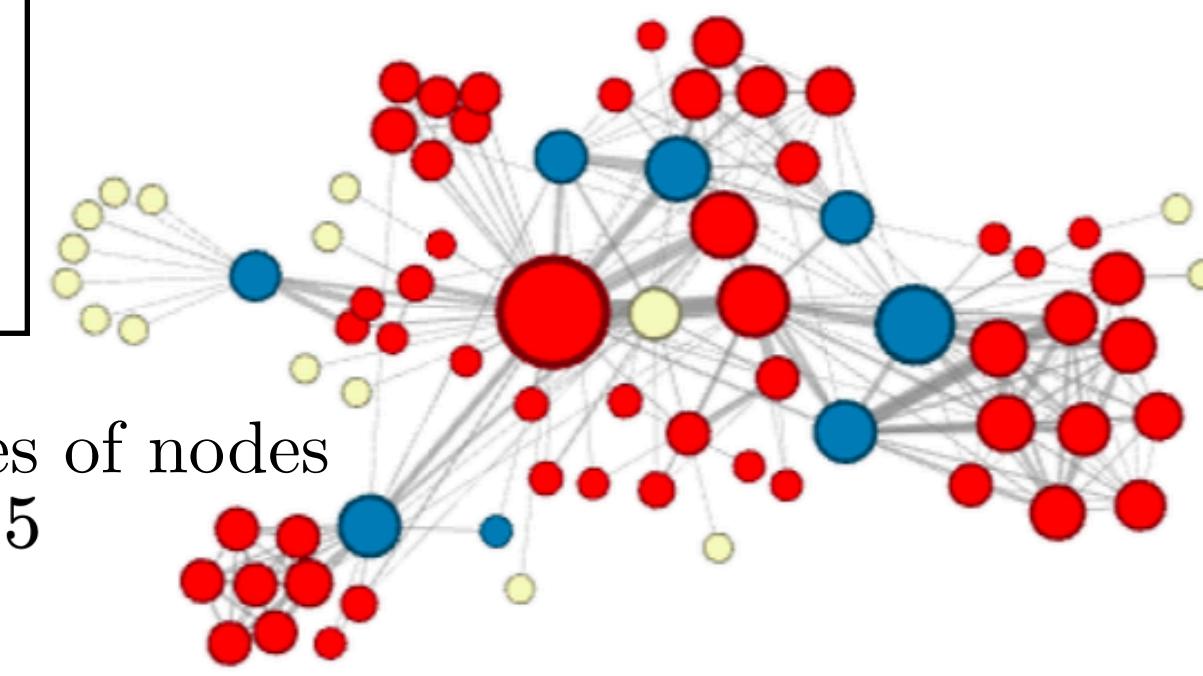
$$p = 1, q = 2$$



Nodes correspond to characters in the novel Les Misérables and edges connect coappearing characters. The network has 77 nodes and 254 edges.

structural roles of nodes

$$p = 1, q = 0.5$$





Boulder

Semi-Supervised Classification with Graph Convolutional Networks



[YouTube Playlist](#)

Consider the problem of classifying nodes (such as documents) in a graph (such as a citation network), where labels are only available for a small subset of nodes.

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}) \rightarrow \text{undirected graph}$$

$$\mathcal{V} = \{v_i : i = 1, \dots, N\} \rightarrow \text{nodes}$$

$$(v_i, v_j) \in \mathcal{E} \rightarrow \text{edges}$$

$$A \in \mathbb{R}^{N \times N} \rightarrow \text{adjacency matrix (binary or weighted)}$$

$$D_{ii} = \sum_j A_{ij} \rightarrow \text{degree matrix}$$

$$\Delta = D - A \rightarrow \text{unnormalized graph Laplacian}$$

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$$

https://en.wikipedia.org/wiki/Laplacian_matrix#Interpretation_as_the_discrete_Laplace_operator

$$\mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_{\text{reg}}$$

neural network

supervised loss

$$\mathcal{L}_{\text{reg}} = \sum_{i,j} A_{ij} \|f(X_i) - f(X_j)\|^2 = f(X)^T \Delta f(X)$$

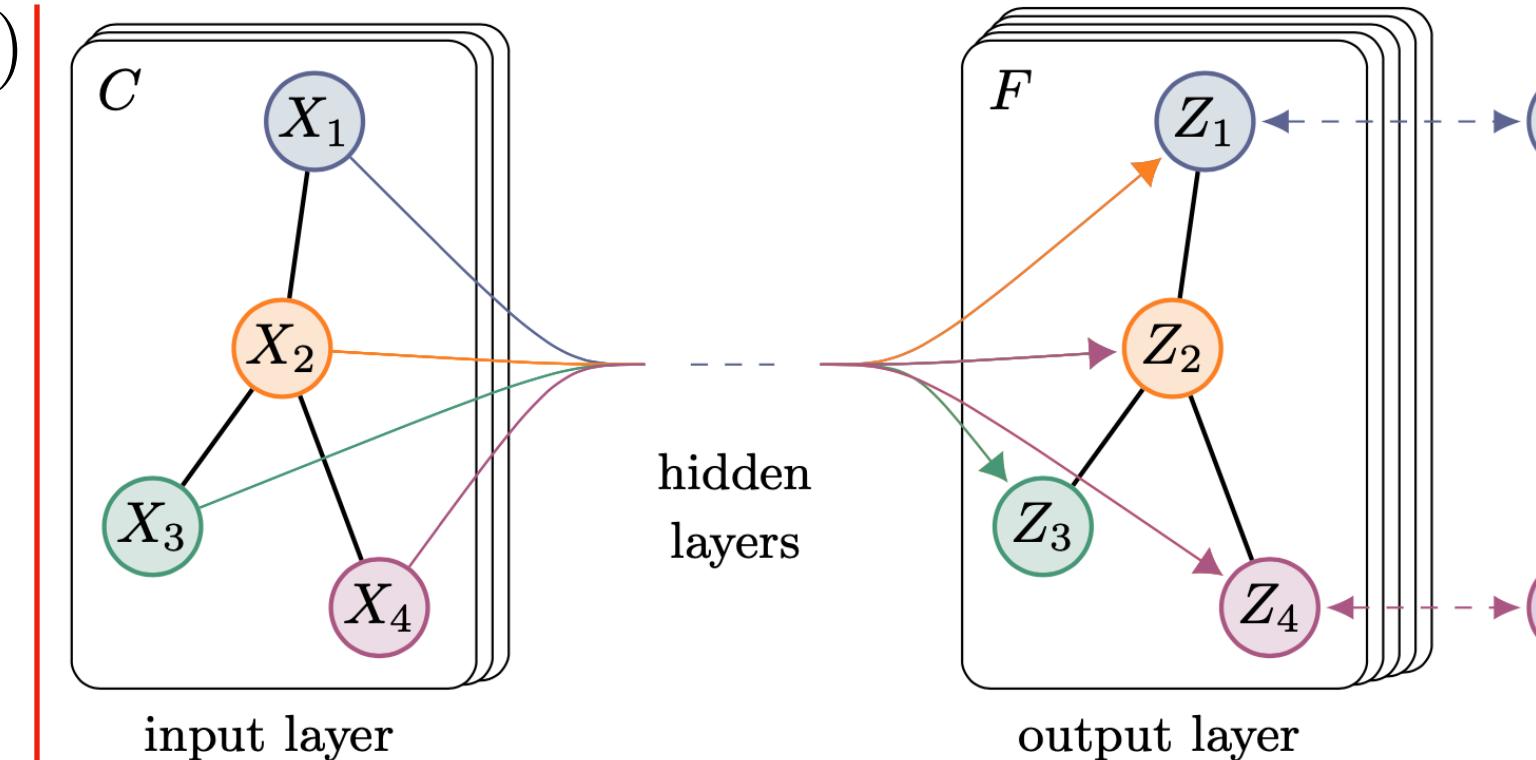
$X \rightarrow \text{matrix of node feature vectors } X_i$

Convolutions on Graphs

$$\tilde{A} = A + I \quad \tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$$

Motivated by a first-order approximation of localized spectral filters on graphs (see the paper)

$$H^{\ell+1} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^\ell W^\ell) \quad H^0 = X \quad H^\ell \in \mathbb{R}^{N \times D}$$



Graph Convolutional Network

$$\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \rightarrow \text{pre-processing step}$$

$$Z = f(X, A) = \text{softmax}\left(\hat{A} \text{ReLU}\left(\hat{A} X W^{(0)}\right) W^{(1)}\right)$$

$$W^{(0)} \in \mathbb{R}^{C \times H}$$

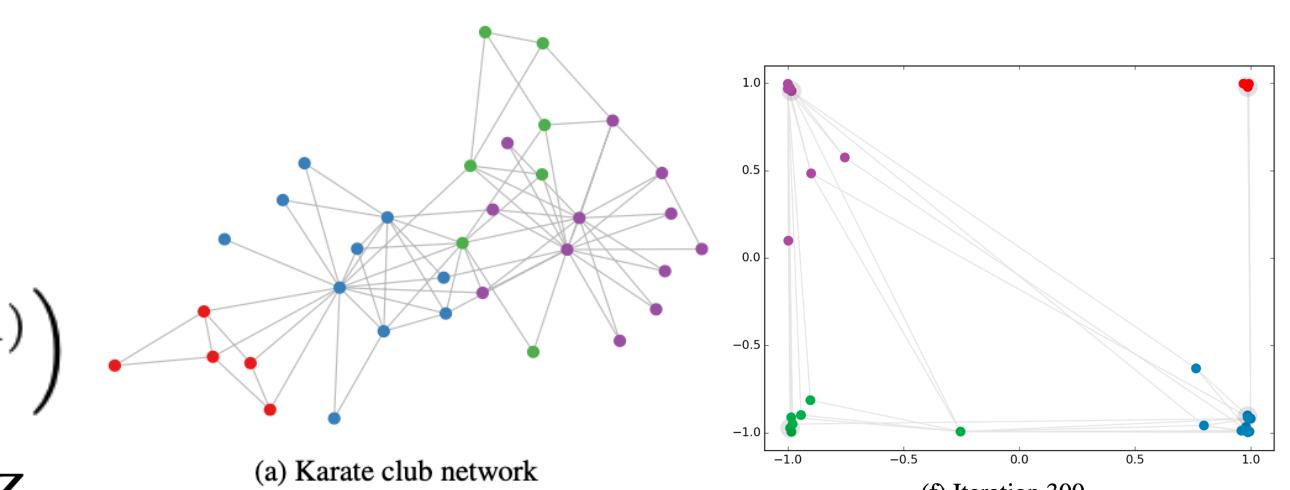
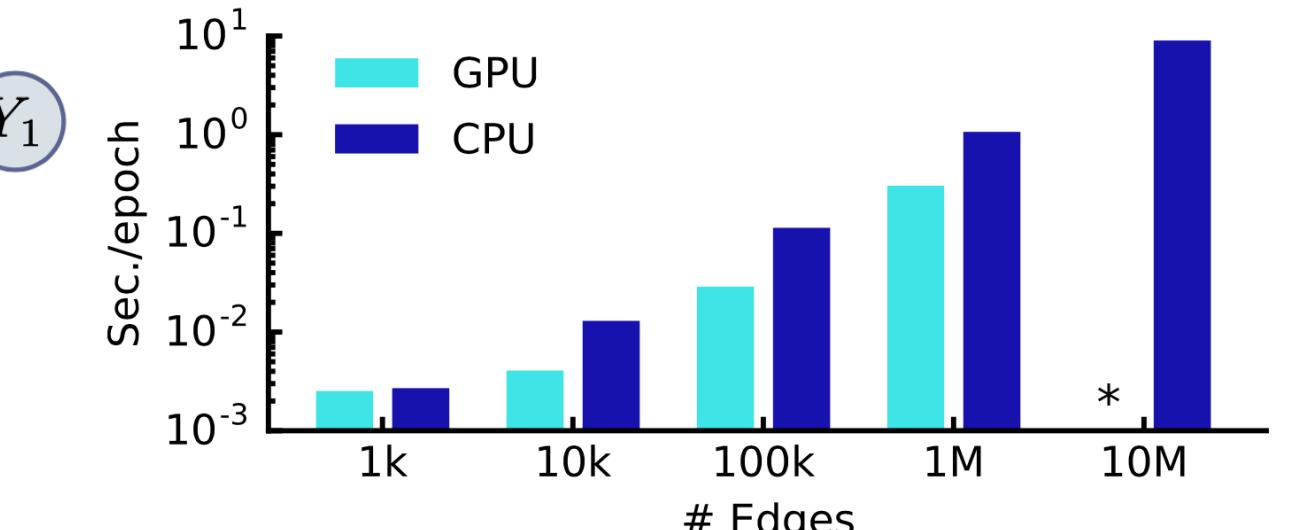
$$W^{(1)} \in \mathbb{R}^{H \times F}$$

$$\mathcal{L} = - \sum_{l \in \mathcal{Y}_L} \sum_{f=1} Y_{lf} \ln Z_{lf}$$

cross-entropy error

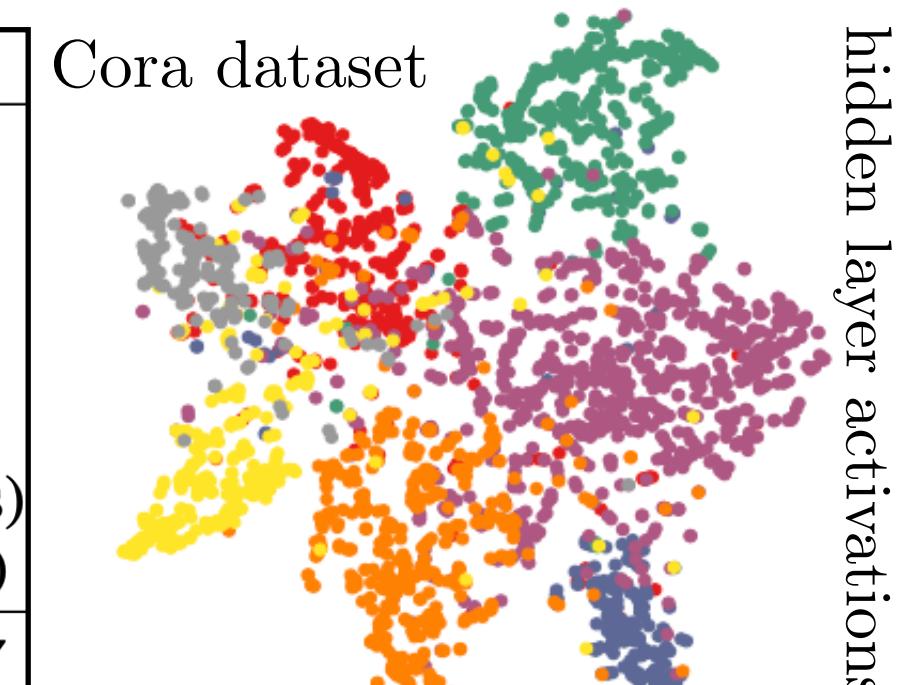
set of node indices that have labels

Drawback: batch gradient descent



Dataset	Type	Nodes	Edges	Classes	Features	Label rate
Citeseer	Citation network	3,327	4,732	6	3,703	0.036
Cora	Citation network	2,708	5,429	7	1,433	0.052
Pubmed	Citation network	19,717	44,338	3	500	0.003
NELL	Knowledge graph	65,755	266,144	210	5,414	0.001

Method	Citeseer	Cora	Pubmed	NELL
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [28]	59.6	59.0	71.1	26.7
LP [32]	45.3	68.0	63.0	26.5
DeepWalk [22]	43.2	67.2	65.3	58.1
ICA [18]	69.1	75.1	73.9	23.1
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	70.3 (7s)	81.5 (4s)	79.0 (38s)	66.0 (48s)
GCN (rand. splits)	67.9 ± 0.5	80.1 ± 0.5	78.9 ± 0.7	58.4 ± 1.7



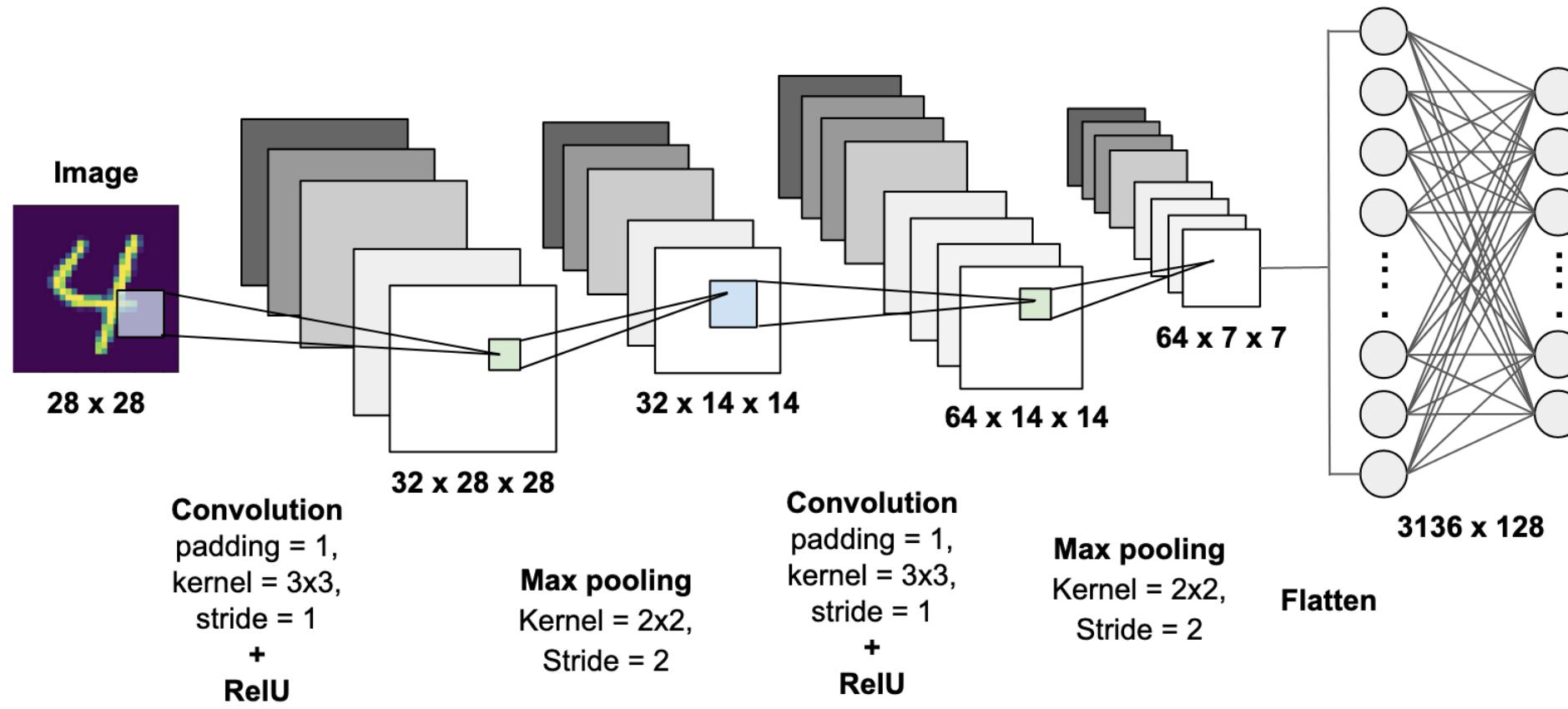


Boulder

Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering



[YouTube Video](#)



Regular grids (e.g., image, video and speech)

Irregular domains (e.g., social networks, brain connectomes or words' embedding) represented by graphs

$\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ → undirected and connected graph

\mathcal{V} → finite set of $|\mathcal{V}| = n$ vertices

\mathcal{E} → set of edges

$W \in \mathbb{R}^{n \times n}$ → weighted adjacency matrix

(encoding the connection weight btw two vertices)

Example (image): 8-NN graph of the 2D grid

$n = |\mathcal{V}| = 976$ ($28^2 = 784$ pixels plus 192 fake nodes)

$|\mathcal{E}| = 3198$

$W_{ij} = \exp\left(-\frac{\|z_i - z_j\|_2^2}{\sigma^2}\right)$ z_i is the 2D coordinate of pixel i

\curvearrowleft a signal on the nodes of the graph (e.g., an image)

$x \in \mathbb{R}^n, x_i \rightarrow$ value of x at node i

- user data on social networks
- gene data on biological regulatory networks
- log data on telecommunication networks
- text documents on word embeddings

Graph Laplacian

$D \in \mathbb{R}^{n \times n} \rightarrow$ diagonal degree matrix

$$D_{ii} = \sum_j W_{ij}$$

$L = D - W \rightarrow$ combinatorial definition

$L = I_n - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} \rightarrow$ normalized definition

$L \rightarrow$ real symmetric positive semidefinite matrix

$\{u_l \in \mathbb{R}^n\}_{l=1}^{n-1} \rightarrow$ complete set of orthonormal eigenvectors

Graph Fourier Modes

$\{\lambda_l\}_{l=1}^{n-1} \rightarrow$ corresponding ordered real nonnegative eigenvalues

Frequencies of the Graph

$U = [u_0, \dots, u_{n-1}] \in \mathbb{R}^{n \times n} \rightarrow$ Fourier Basis

$\Lambda = \text{diag}([\lambda_0, \dots, \lambda_{n-1}]) \in \mathbb{R}^{n \times n}$

$L = U \Lambda U^T \rightarrow$ diagonalized

$\hat{x} = U^T x \rightarrow$ Fourier Transform

$x = U \hat{x} \rightarrow$ Inverse Fourier Transform

$*_{\mathcal{G}} \rightarrow$ graph convolution

$x *_{\mathcal{G}} y := U((U^T x) \odot (U^T y))$

Spectral Filtering

$g_\theta(L)x = g_\theta(U \Lambda U^T)x = U g_\theta(\Lambda) U^T x$

signal x filtered by g_θ

$$f_\theta(\Lambda) = \text{diag}(\theta), \theta \in \mathbb{R}^n$$

– non-localized in space

– learning complexity $\mathcal{O}(n)$

Polynomial Parametrization

Chebyshev Polynomials

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$$

$$T_0(x) = 1, T_1(x) = x$$

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda}), \tilde{\Lambda} = \frac{2\Lambda}{\lambda_{\max}} - I_n$$

Model	Accuracy
Classical CNN	99.33
Proposed graph CNN	99.14

The proposed spectral filters are provable to be strictly localized in a ball of radius K , i.e. K hops from the central vertex.

$$\theta \in \mathbb{R}^K$$

$$g_\theta(L)x = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})x$$

$$\tilde{L} = 2L/\lambda_{\max} - I_n$$

$$\bar{x}_k = T_k(\tilde{L})x$$

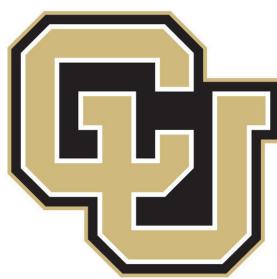
$$\bar{x}_k = 2\tilde{L}\bar{x}_{k-1} - \bar{x}_{k-2}$$

$$\bar{x}_0 = x$$

$$\bar{x}_1 = \tilde{L}x$$

Graph Coarsening and Pooling (see the paper)

$\mathcal{O}(K|\mathcal{E}|)$ operations



Boulder

Inductive Representation Learning on Large Graphs



[YouTube Video](#)

Large Graphs: Social or Biological Networks

- content recommendation
- identifying protein functions

Transductive: Single Fixed Graph

Inductive: Dynamic Graph

- Citation & Reddit post data
- Protein-protein interaction

Low-dimensional vector embeddings
of nodes in large graphs

- node classification
- clustering
- link prediction

Nodes: posts on Reddit or
users and videos on YouTube

GraphSAGE: Sample & aggreGatE

Node features:

- text attributes
- node profile information
- node degrees

- citation data with text attributes
- biological data with functional/molecular markers

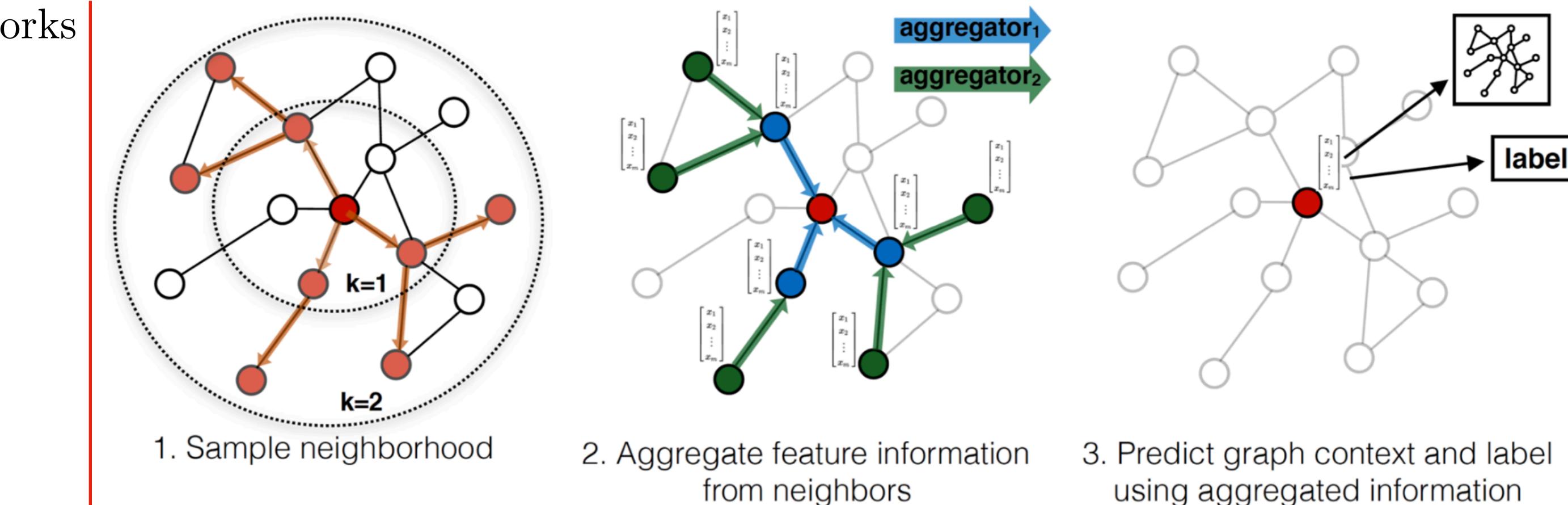
Mean aggregator → Similar to GCN

$$\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W} \cdot \text{MEAN}(\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}))$$

Pooling aggregator

$$\text{AGGREGATE}_k^{\text{pool}} =$$

$$\max(\{\sigma(\mathbf{W}_{\text{pool}} \mathbf{h}_{u_i}^k + \mathbf{b}), \forall u_i \in \mathcal{N}(v)\})$$



Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K ; weight matrices $\mathbf{W}^k, \forall k \in \{1, \dots, K\}$; non-linearity σ ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, \dots, K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output: Vector representations \mathbf{z}_v for all $v \in \mathcal{V}$

```

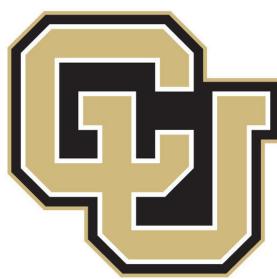
1  $\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$  ;
2 for  $k = 1 \dots K$  do
3   for  $v \in \mathcal{V}$  do
4      $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\});$ 
5      $\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W}^k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

```

$$J_{\mathcal{G}}(\mathbf{z}_u) = -\log(\sigma(\mathbf{z}_u^\top \mathbf{z}_v)) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log(\sigma(-\mathbf{z}_u^\top \mathbf{z}_{v_n}))$$

Encourages nearby nodes to have similar representations, while enforcing that the representations of disparate nodes are highly distinct

$v \rightarrow$ a node that co-occurs near u on a fixed-length random walk
 $P_n \rightarrow$ negative sampling distribution
 $Q \rightarrow$ number of negative samples



Boulder



[YouTube Video](#)

Graph Attention Networks

GATs: Graph ATTention networks

- Cora
- Citeseer
- Pubmed

- Protein-protein interaction

Graphs: 3D meshes, social networks, telecommunication networks, biological networks or brain connectomes

Graph Attentional Layer

$$h = \{h_1, h_2, \dots, h_N\}, h_i \in \mathbb{R}^F$$

input (a set of node features)

$N \rightarrow$ number of nodes

$F \rightarrow$ number of features

$$h' = \{h'_1, h'_2, \dots, h'_N\}, h'_i \in \mathbb{R}^{F'}$$

output

$W \in \mathbb{R}^{F' \times F} \rightarrow$ weight matrix

Self-attention

$$a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$$

$e_{ij} = a(Wh_i, Wh_j) \rightarrow$ attention coefficient
(importance of node j 's features to node i)

Masked attention

$\mathcal{N}_i \rightarrow$ some neighborhood of
node i in the graph

(first-order neighbors of i , including i)

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$$

$$a(x, y) = \text{LeakyReLU}_{\alpha=2}(a^T [x \| y]), a \in \mathbb{R}^{2F'}$$

concatenation ↘

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

Multi-head Attention (K heads)

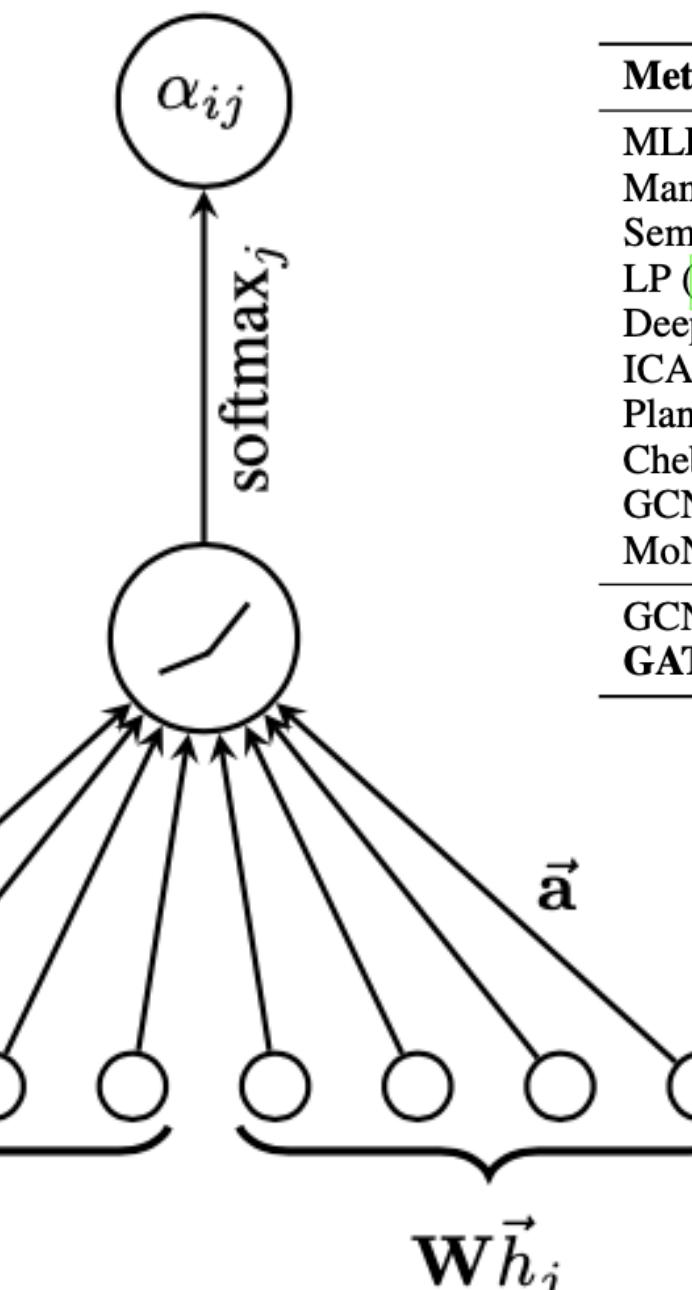
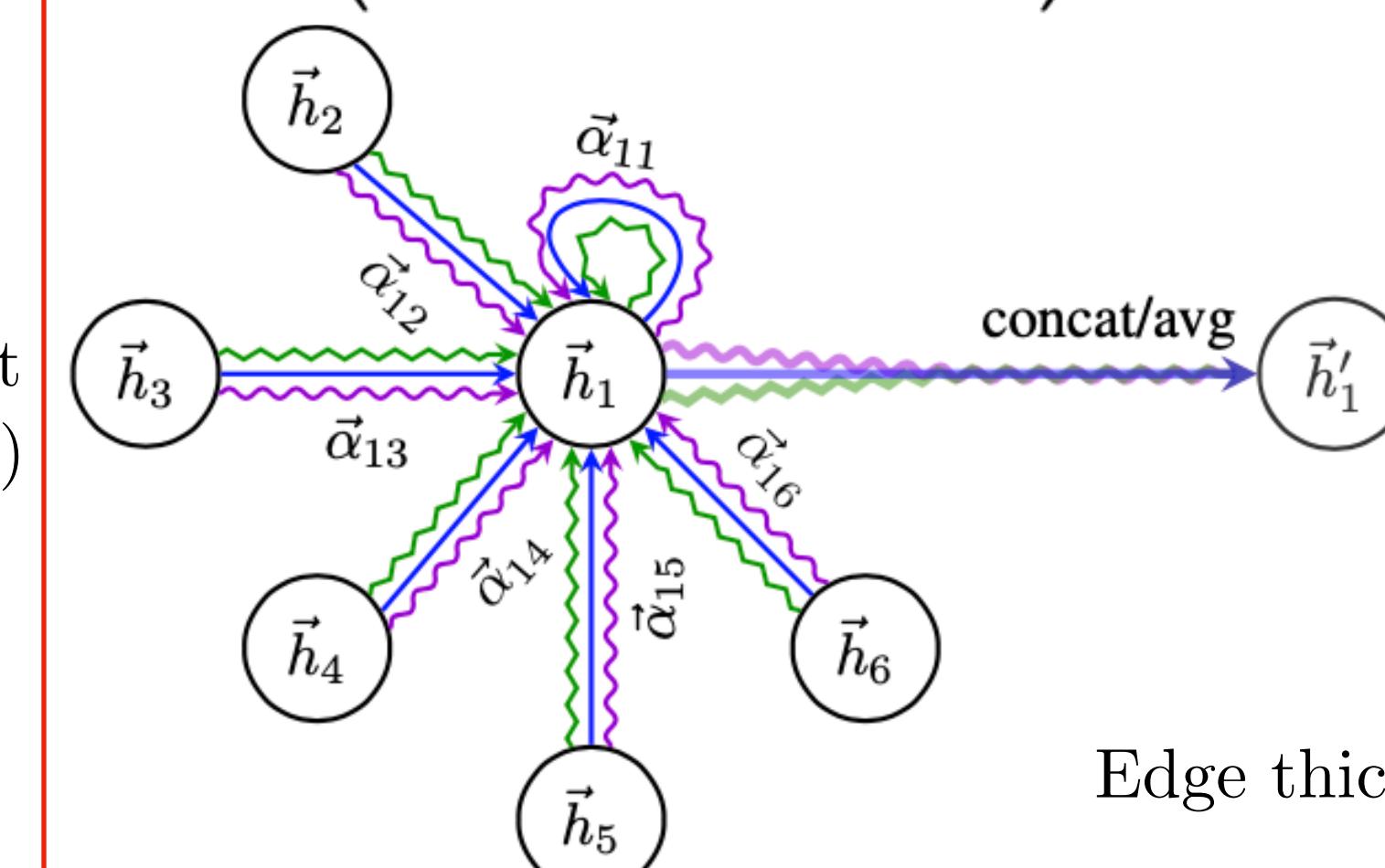
$$h'_i = \parallel_{k=1}^K \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k W^k h_j \right)$$

concatenation

$KF' \rightarrow$ features

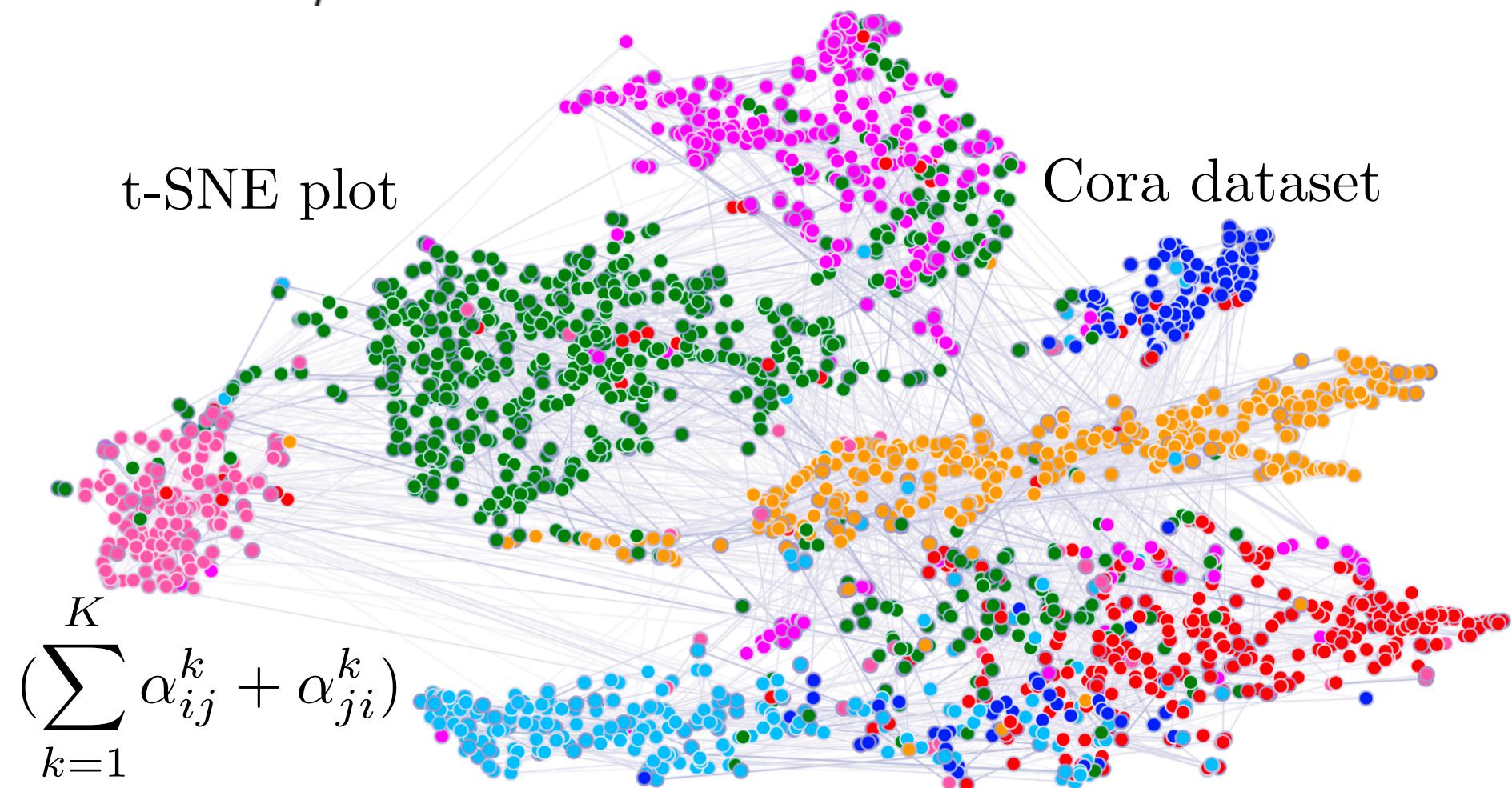
Final Prediction Layer

$$h'_i = \sigma \left(\frac{1}{K} \sum_{k=1}^K \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k W^k h_j \right)$$



$$\text{Edge thickness: } (\sum_{k=1}^K \alpha_{ij}^k + \alpha_{ji}^k)$$

t-SNE plot



Method	Transductive		
	Cora	Citeseer	Pubmed
MLP	55.1%	46.5%	71.4%
ManiReg (Belkin et al., 2006)	59.5%	60.1%	70.7%
SemiEmb (Weston et al., 2012)	59.0%	59.6%	71.7%
LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
ICA (Lu & Getoor, 2003)	75.1%	69.1%	73.9%
Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%	74.4%
GCN (Kipf & Welling, 2017)	81.5%	70.3%	79.0%
MoNet (Monti et al., 2016)	81.7 ± 0.5%	—	78.8 ± 0.3%
GCN-64*	81.4 ± 0.5%	70.9 ± 0.5%	79.0 ± 0.3%
GAT (ours)	83.0 ± 0.7%	72.5 ± 0.7%	79.0 ± 0.3%

Method	Inductive		
	PPI	F1 scores	
Random	0.396		
MLP	0.422		
GraphSAGE-GCN (Hamilton et al., 2017)	0.500		
GraphSAGE-mean (Hamilton et al., 2017)	0.598		
GraphSAGE-LSTM (Hamilton et al., 2017)	0.612		
GraphSAGE-pool (Hamilton et al., 2017)	0.600		
GraphSAGE*	0.768		
Const-GAT (ours)	0.934 ± 0.006		
GAT (ours)	0.973 ± 0.002		

2.
precision · recall
precision + recall



Boulder



[YouTube Video](#)

How Powerful Are Graph Neural Networks?

$G = (V, E) \rightarrow \text{graph}$

$X_v \rightarrow \text{node feature vector for } v \in V$

Node Classification

$y_v \rightarrow \text{associated label to node } v \in V$

$h_v \rightarrow \text{representation vector of } v \in V$

$$y_v = f(h_v)$$

Graph Classification

$\{G_1, \dots, G_N\} \subset \mathcal{G} \rightarrow \text{set of graphs}$

$\{y_1, \dots, y_N\} \subset \mathcal{Y} \rightarrow \text{set of labels}$

$h_G \rightarrow \text{representation vector}$

$y_G = g(h_G) \rightarrow \text{predicted label of an entire graph}$

Graph Neural Networks (GNNs)

$k \rightarrow k\text{-th layer of a GNN (}k\text{-th iteration)}$

$a_v^{(k)} = \text{AGGREGATE}^{(k)}(\{h_u^{(k-1)} : u \in \mathcal{N}(v)\})$
set of nodes adjacent to $v \supset$

$h_v^{(k)} = \text{COMBINE}^{(k)}(h_v^{(k-1)}, a_v^{(k)})$

$h_v^{(0)} = X_v$

Graph Classification

$h_G = \text{READOUT}(\{h_v^K : v \in G\})$

summation or group-level pooling function

GraphSage

$a_v^{(k)} = \text{MAX}(\{\text{ReLU}(W h_u^{(k-1)}), \forall u \in \mathcal{N}(v)\}) \rightarrow \text{aggregate}$

$W[h_v^{(k-1)}, a_v^{(k)}] \rightarrow \text{combine}$

Graph Convolutional Network (GCN)

$h_v^{(k)} = \text{ReLU}(W \text{ MEAN}(\{h_u^{k-1}, \forall u \in \mathcal{N}(v) \cup \{v\}\}))$

aggregate and combine

Xu, Keyulu, et al. "How powerful are graph neural networks?." arXiv preprint arXiv:1810.00826 (2018).

Graph Isomorphism Network (GIN)

$$h_v^{(k)} = \text{MLP}^{(k)} \left(\left(1 + \epsilon^{(k)} \right) \cdot h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)} \right)$$

$$h_G = \text{CONCAT} \left(\text{READOUT} \left(\{h_v^{(k)} | v \in G\} \right) \mid k = 0, 1, \dots, K \right)$$

Discriminative and representational power of GIN is equal to the power of the Weisfeiler-Lehman test.

Weisfeiler-Lehman (WL) Test

Graph isomorphism problem: Are two graphs topologically identical?

The WL test iteratively (1) aggregates the labels of nodes and their neighborhoods, and (2) hashes the aggregated labels into unique new labels. The algorithm decides that two graphs are non-isomorphic if at some iteration the labels of the nodes between the two graphs differ.

Proof (Sketch)

– GNNs are at most as powerful as the WL test in distinguishing graph structures

Let G_1 and G_2 be any two non-isomorphic graphs. If a graph neural network $\mathcal{A} : \mathcal{G} \rightarrow \mathbb{R}^d$ maps G_1 and G_2 to different embeddings, the Weisfeiler-Lehman graph isomorphism test also decides G_1 and G_2 are not isomorphic.

– Under some mild conditions on the neighbor aggregation and graph readout functions, the resulting GNN is as powerful as the WL test.

a) \mathcal{A} aggregates and updates node features iteratively with

$$h_v^{(k)} = \phi \left(h_v^{(k-1)}, f \left(\{h_u^{(k-1)} : u \in \mathcal{N}(v)\} \right) \right),$$

where the functions f , which operates on multisets, and ϕ are injective.

b) \mathcal{A} 's graph-level readout, which operates on the multiset of node features $\{h_v^{(k)}\}$, is injective.



Boulder



Questions?

[YouTube Playlist](#)
