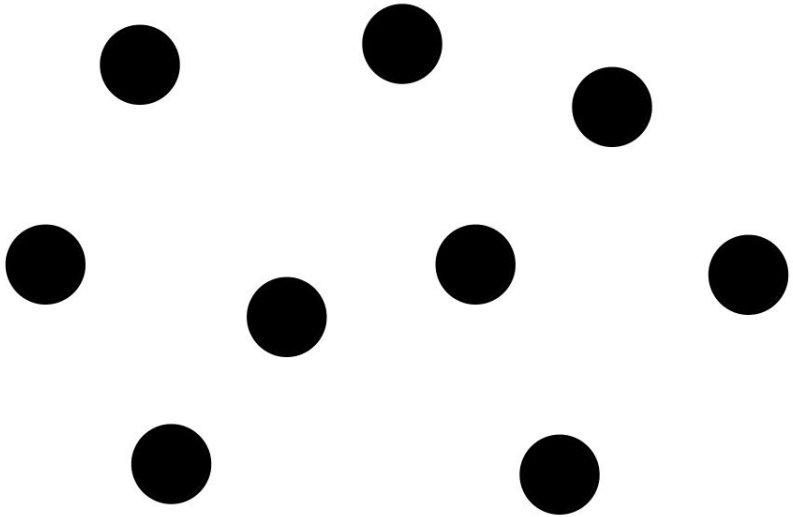
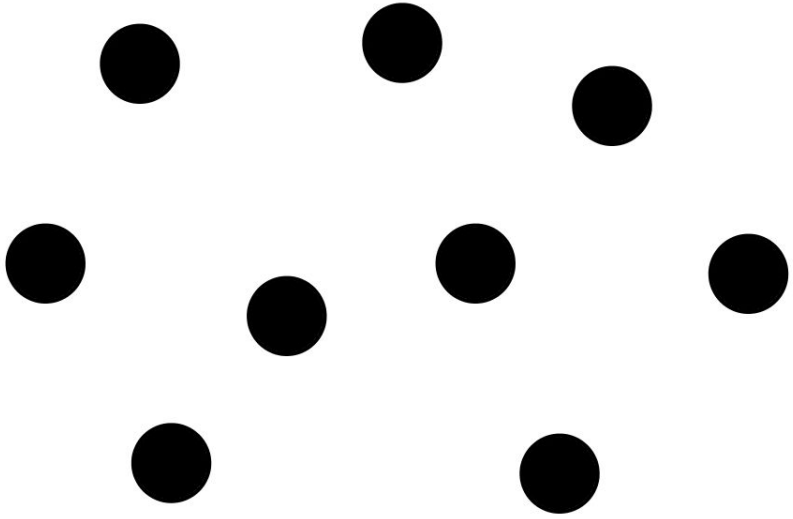
Neural network methods for working with graphs

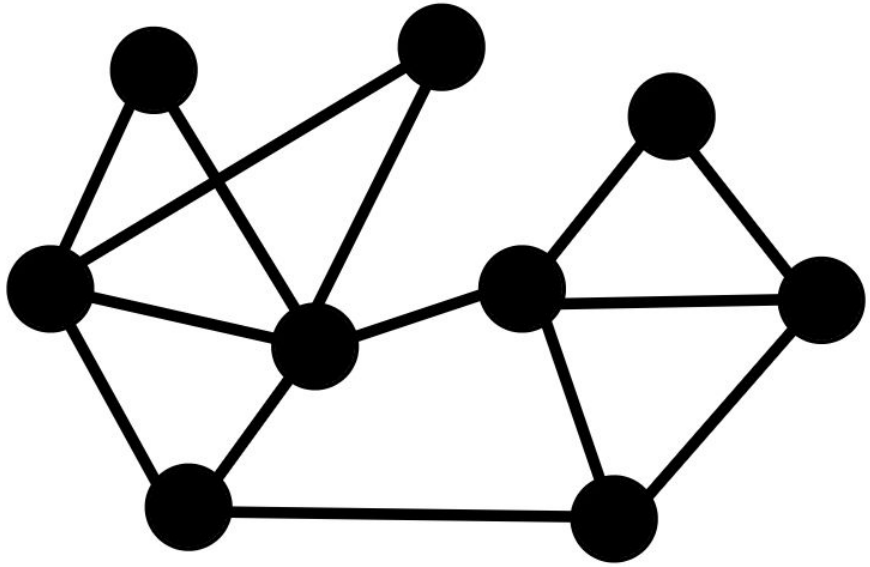
Vasileva Anna

Motivation



Motivation





Graph

Many Types of Data are Graphs

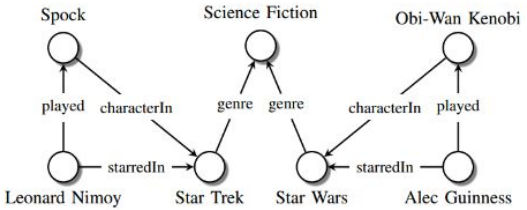


Image credit: Maximilian Nickel et al

Knowledge Graphs

Image credit: Medium

Social Networks

Citation Networks

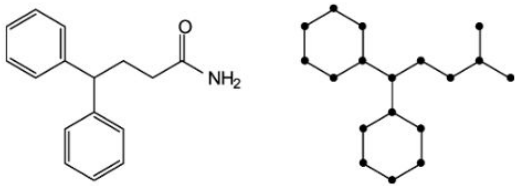
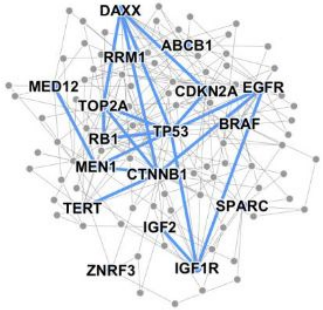


Image credit: MDPI

Molecules

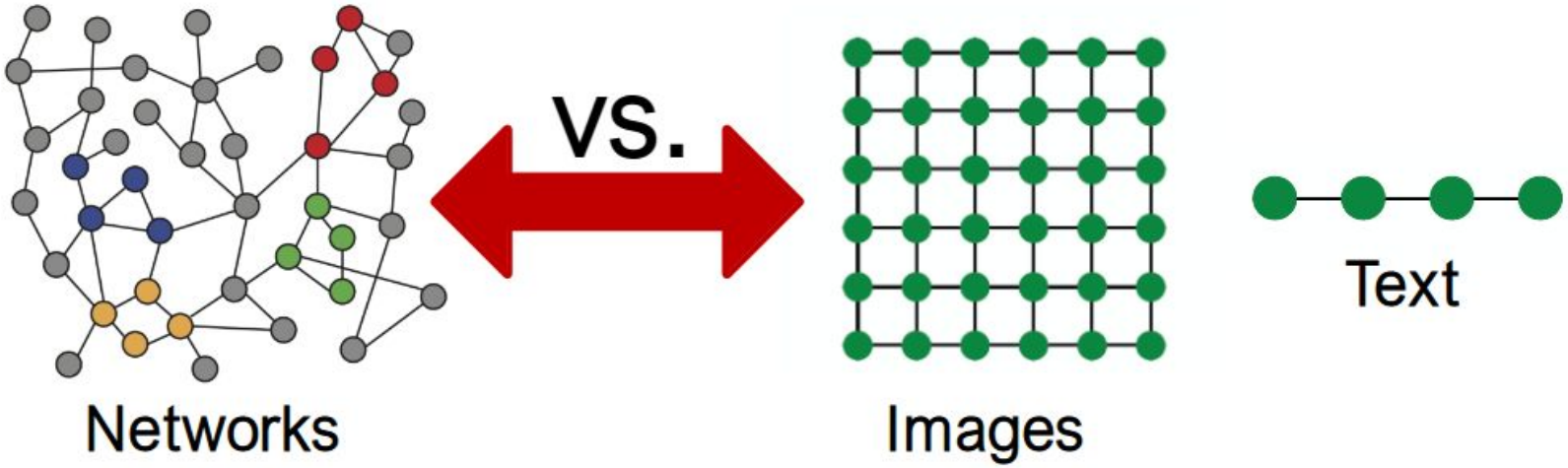


Disease Pathways

Image credit: visitlondon.com

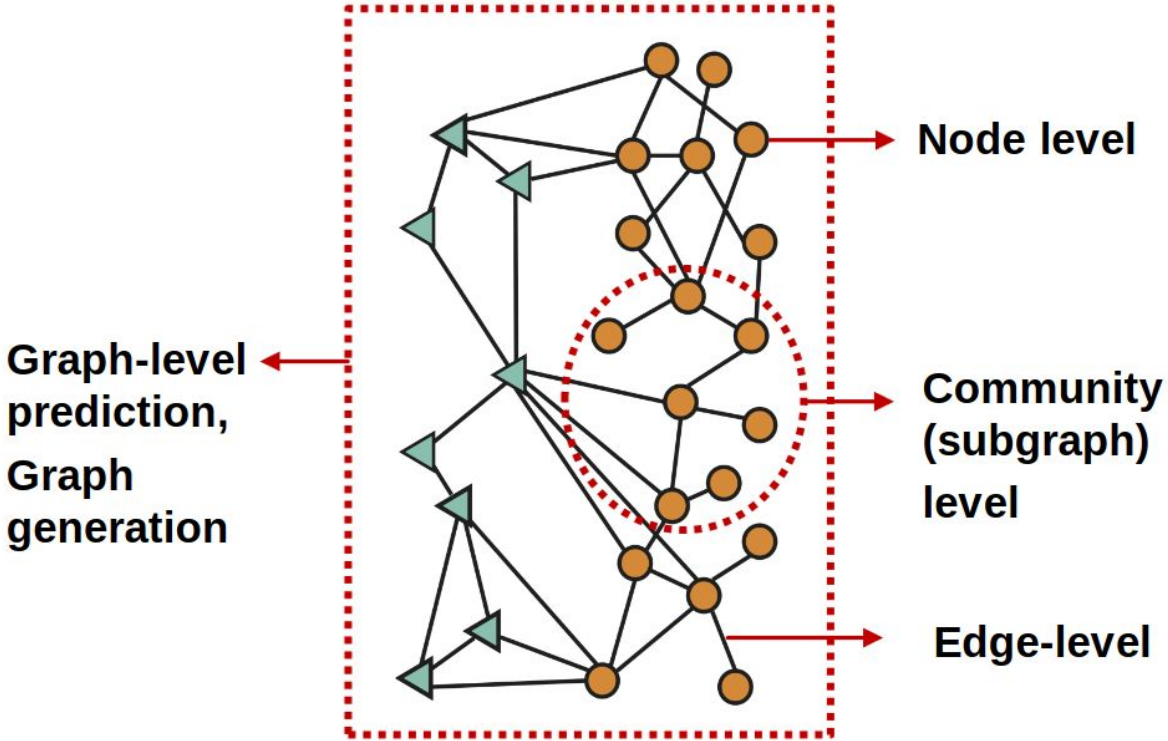
Underground Networks

Graphs vs Images & Texts



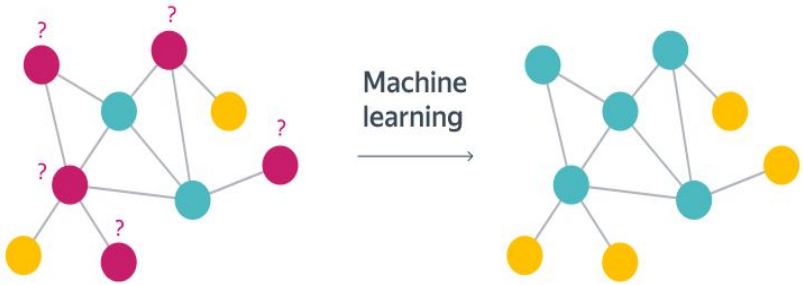
- Arbitrary size and complex topological structure
- No fixed node ordering or reference point

Different Types of Tasks

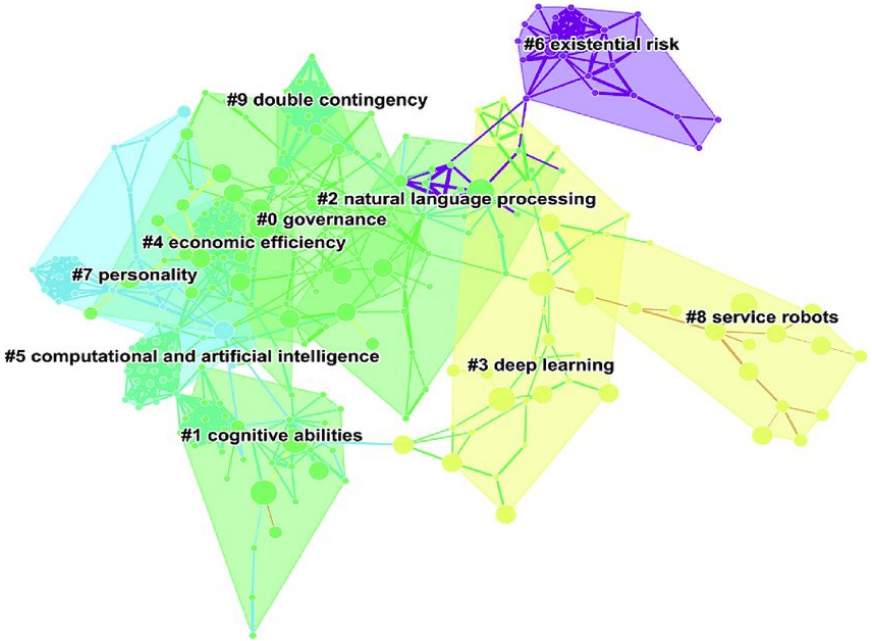


Node-level Tasks

- Node classification

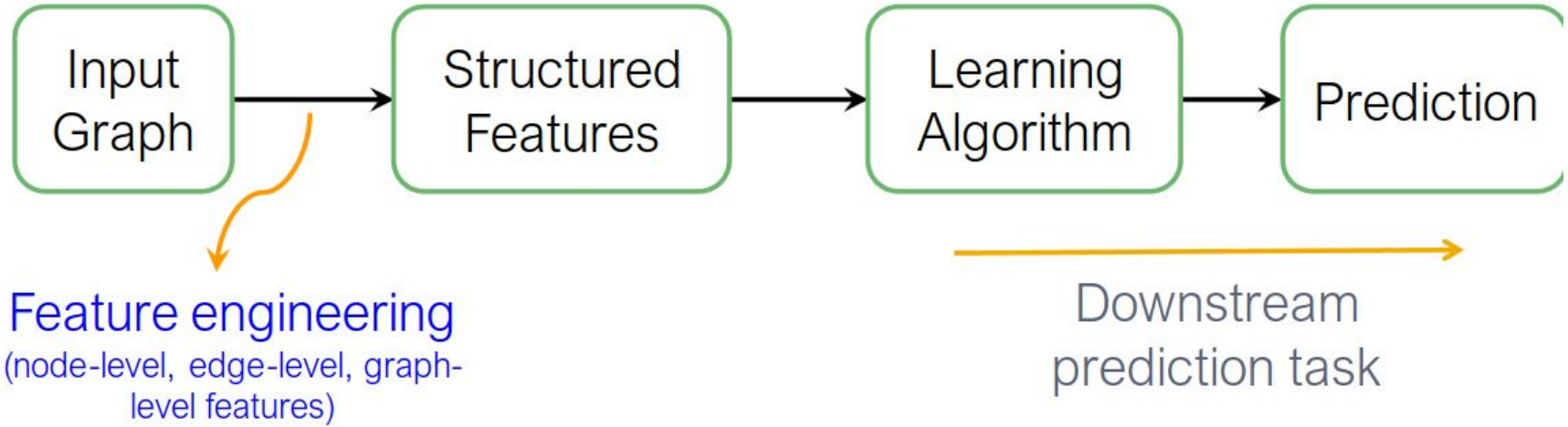


- Node-level graph clasterisation

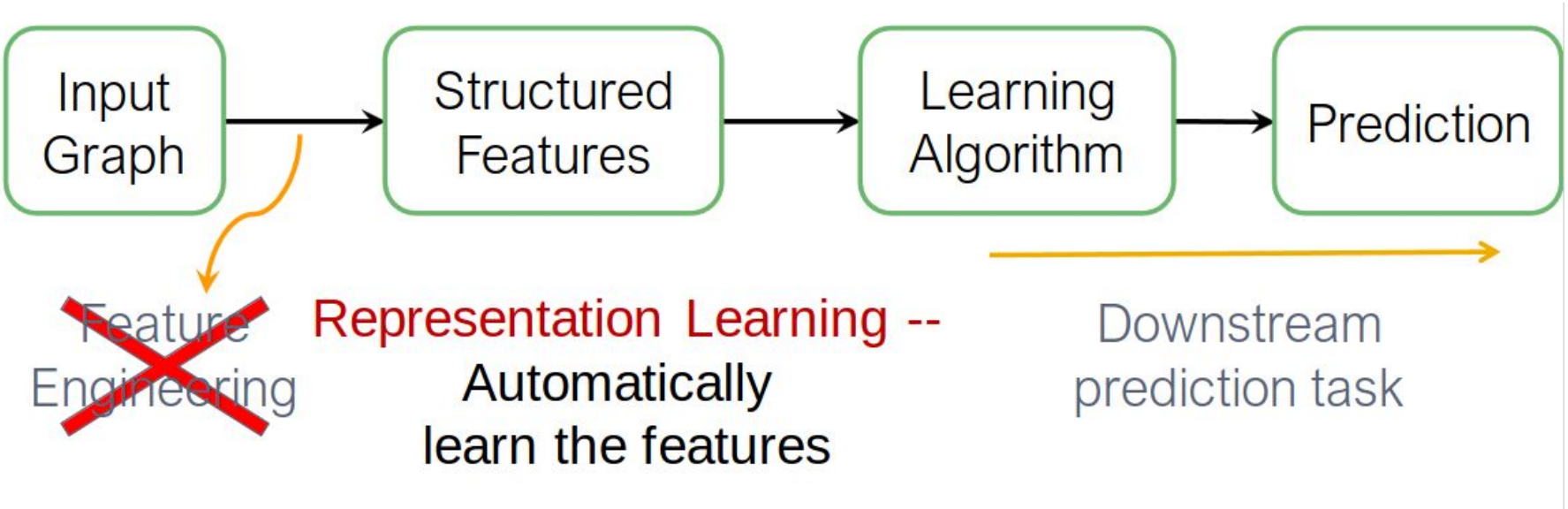


Traditional ML Approach for Node-level Tasks

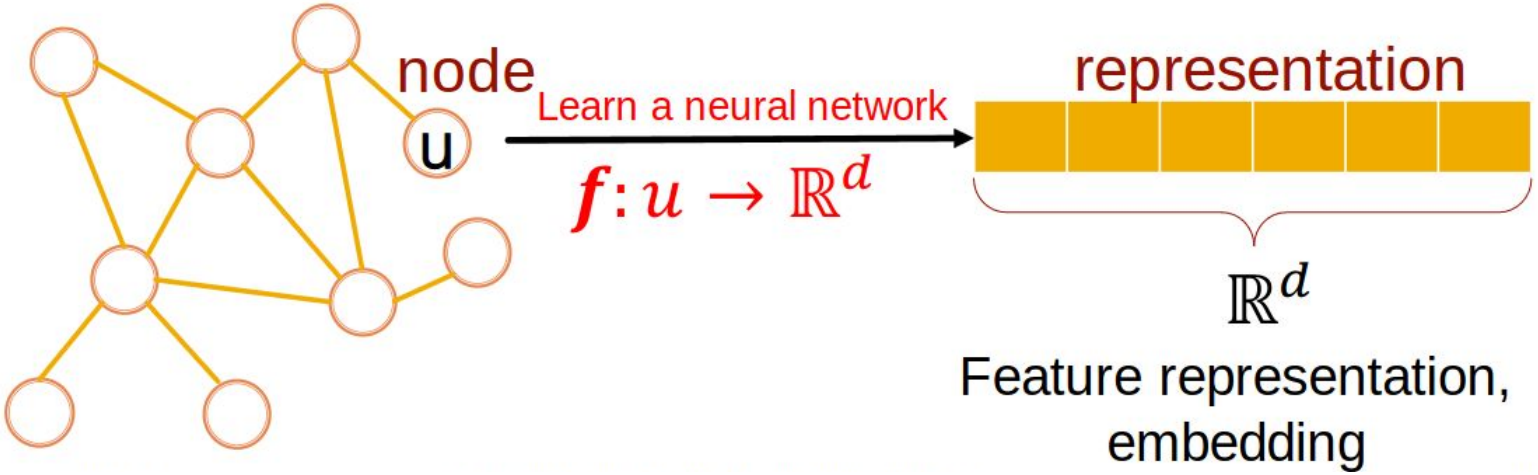
- Extract various features that would describe graph topological structure
  - Node degree, node centrality, ..



Our Approach for Node-level Tasks

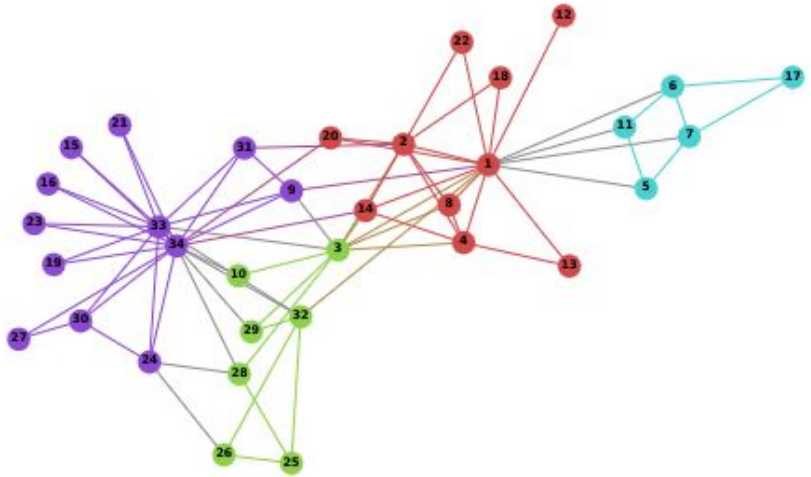


Representation Learning

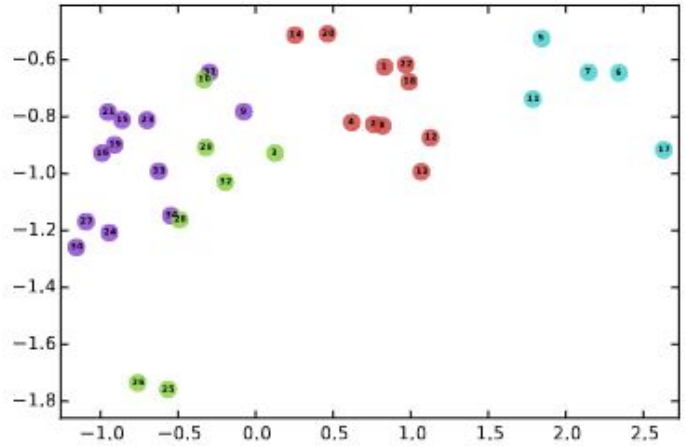


- Similarity of nodes -> similarity of embeddings
- Encodes network information
- Low dimensional, continuous, adaptable
- Can be used in many tasks

Deep Walk

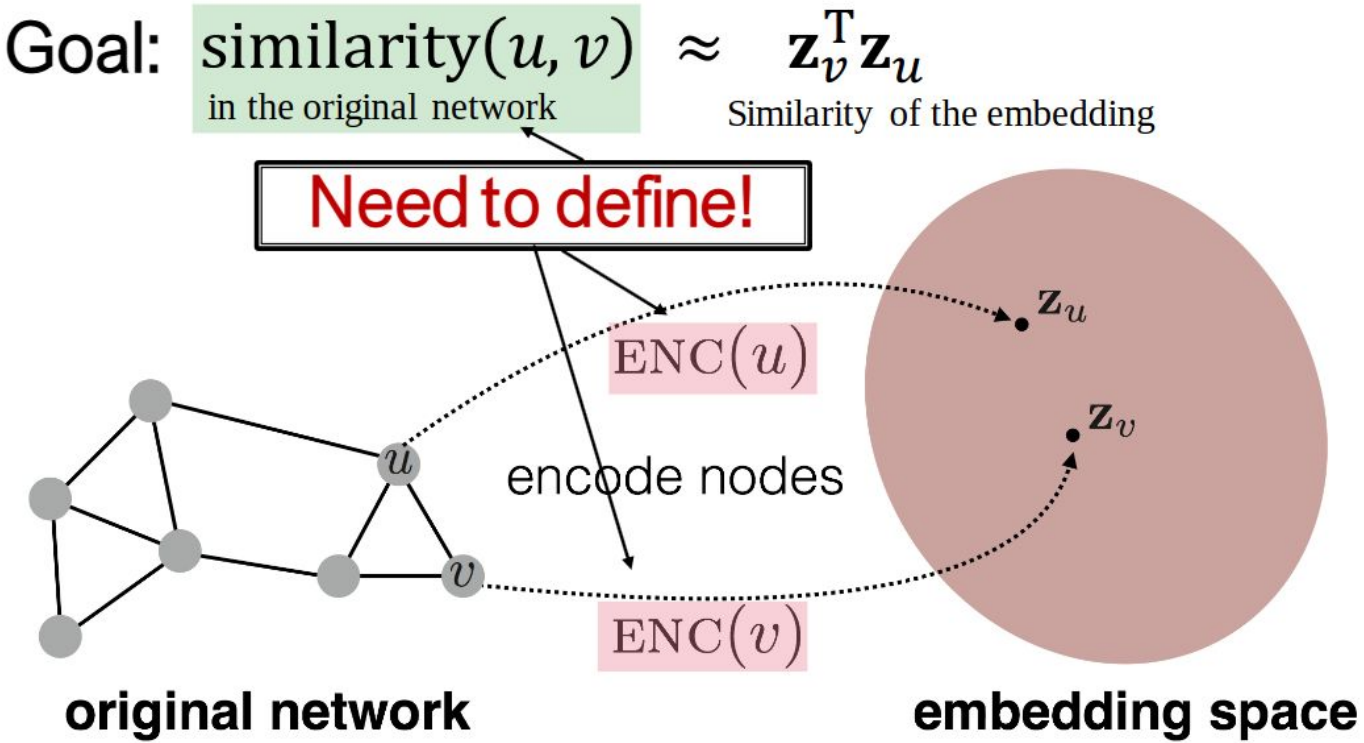


(a) Input: Karate Graph

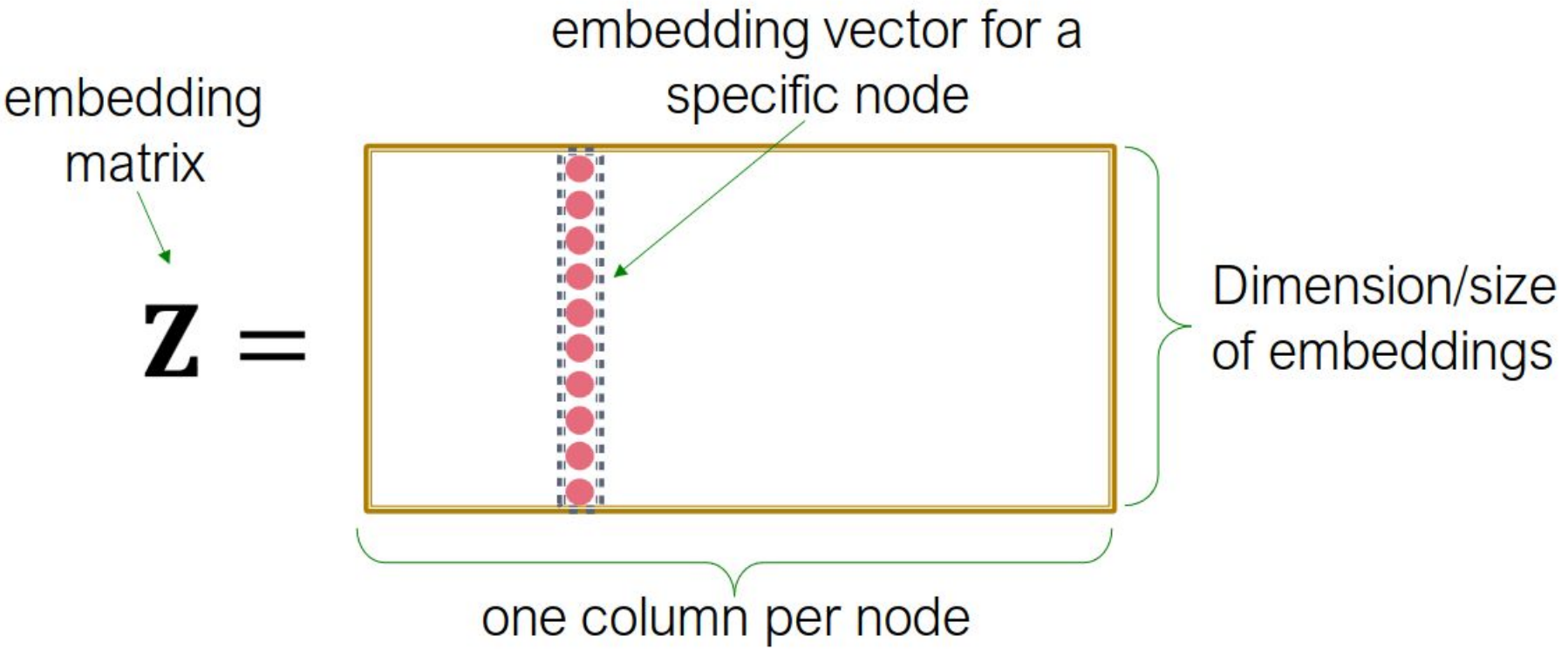


(b) Output: Representation

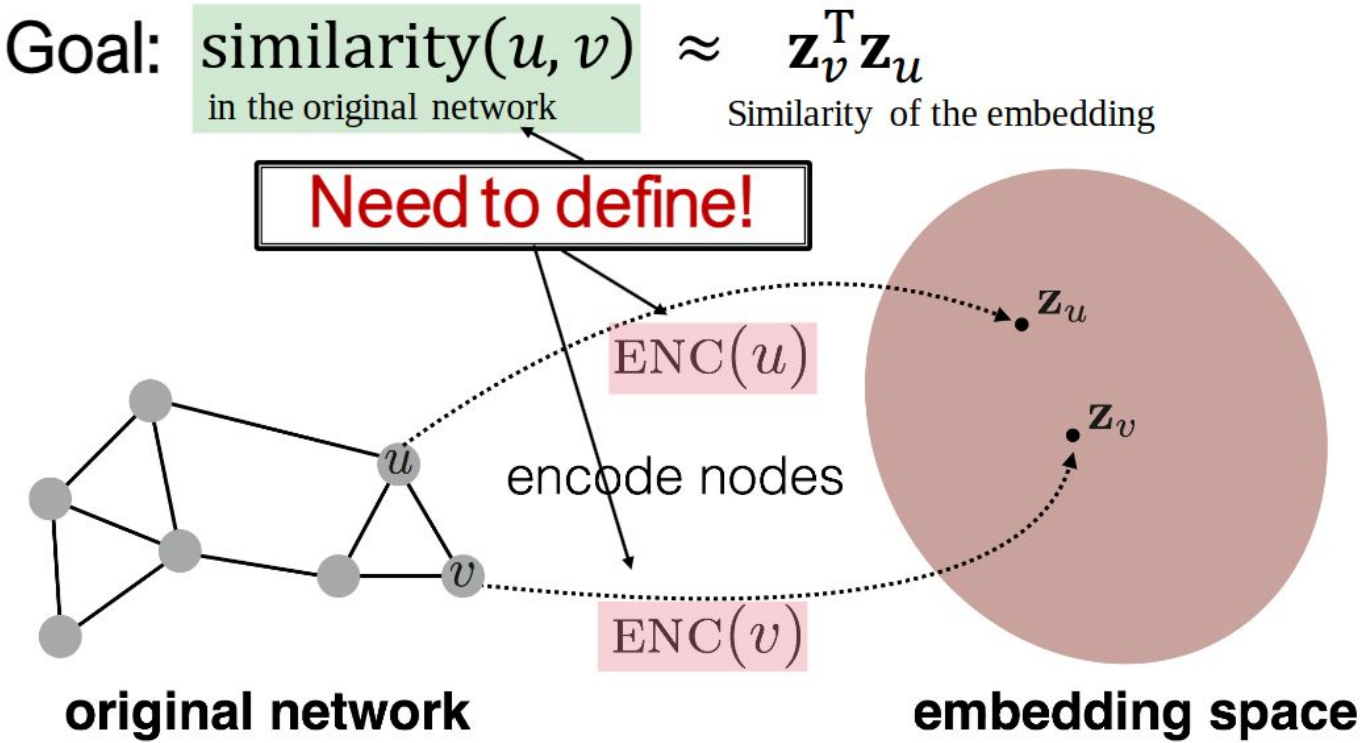
Embedding Nodes



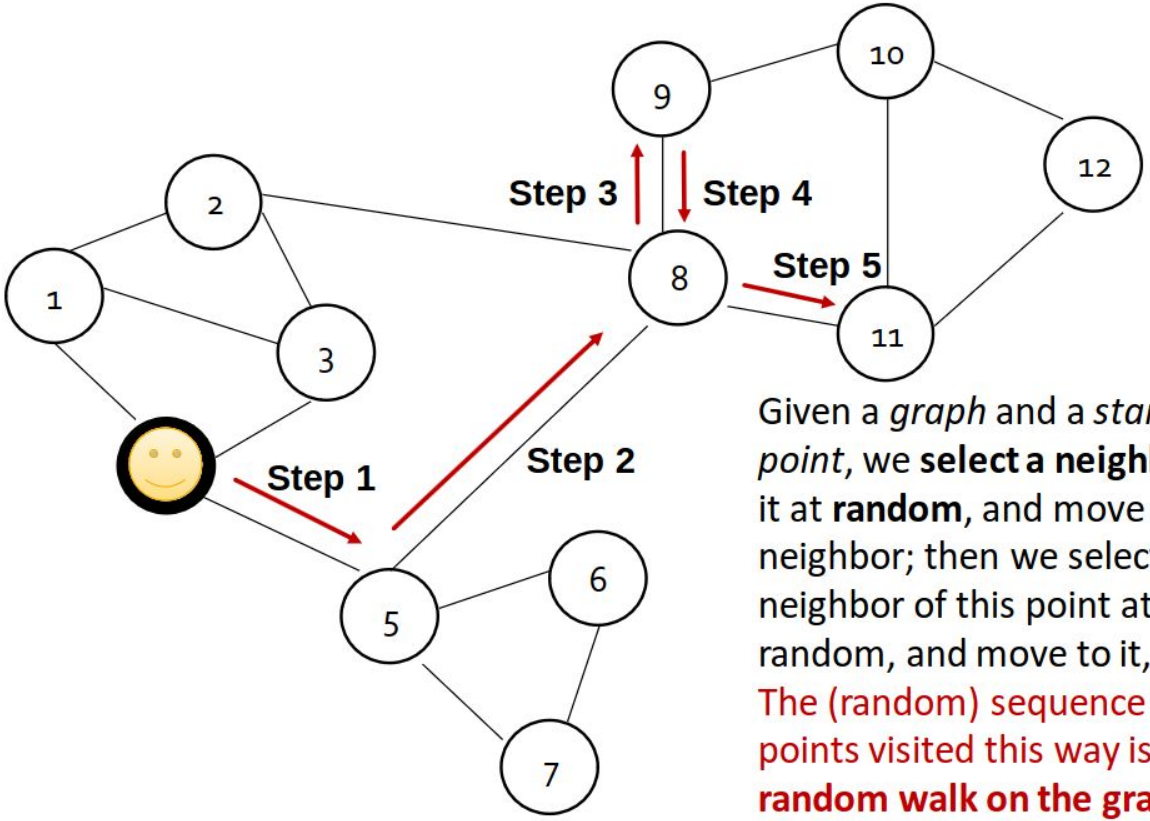
Shallow Encoding



Embedding Nodes

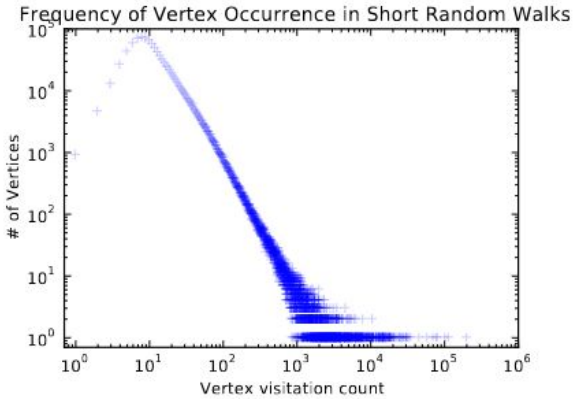


Nodes Similarity: Random Walks

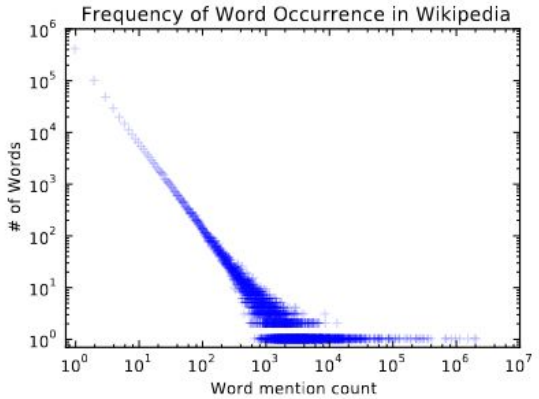


Given a *graph* and a *starting point*, we **select a neighbor** of it at **random**, and move to this neighbor; then we select a neighbor of this point at random, and move to it, etc. The (random) sequence of points visited this way is a **random walk on the graph**.

Power Law



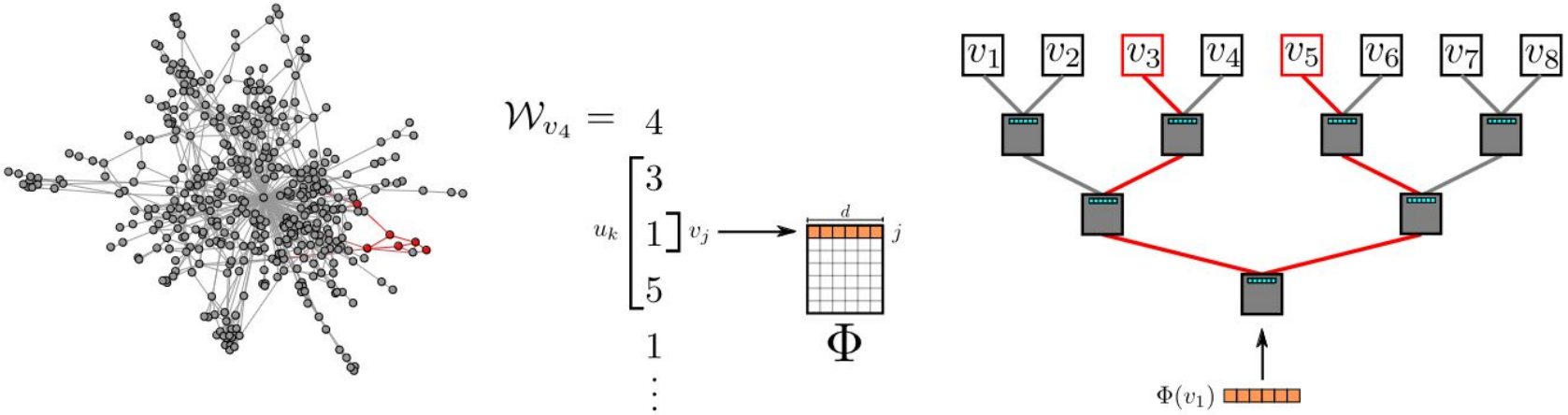
(a) YouTube Social Graph



(b) Wikipedia Article Text

Figure 2: The power-law distribution of vertices appearing in short random walks (2a) follows a power-law, much like the distribution of words in natural language (2b).

Algorithm



(a) Random walk generation. (b) Representation mapping. (c) Hierarchical Softmax.

Figure 3: Overview of DeepWalk. We slide a window of length $2w + 1$ over the random walk $\mathcal{W}_{v_4}$, mapping the central vertex v_1 to its representation $\Phi(v_1)$. Hierarchical Softmax factors out $\Pr(v_3 \mid \Phi(v_1))$ and $\Pr(v_5 \mid \Phi(v_1))$ over sequences of probability distributions corresponding to the paths starting at the root and ending at v_3 and v_5. The representation Φ is updated to maximize the probability of v_1 co-occurring with its context $\{v_3, v_5\}$.

Algorithm

Algorithm 1 DeepWalk(G, w, d, γ, t)

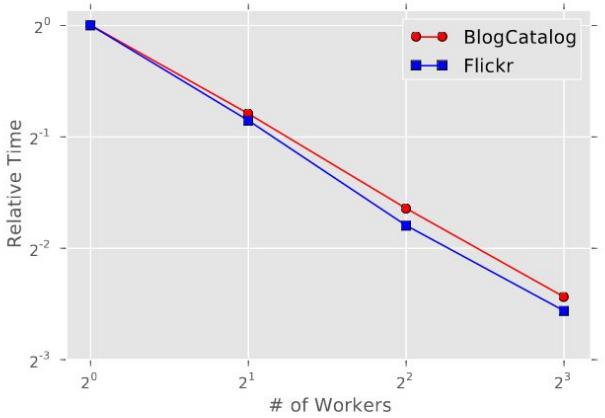
Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t
Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$

Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$
Build a binary Tree T from V
for $i = 0$ to γ **do**
  $\mathcal{O} = \text{Shuffle}(V)$
  for each $v_i \in \mathcal{O}$ **do**
    $\mathcal{W}_{v_i} = RandomWalk(G, v_i, \text{t})$
    SkipGram(Φ, $\mathcal{W}_{v_i}$, w)
  end for
end for

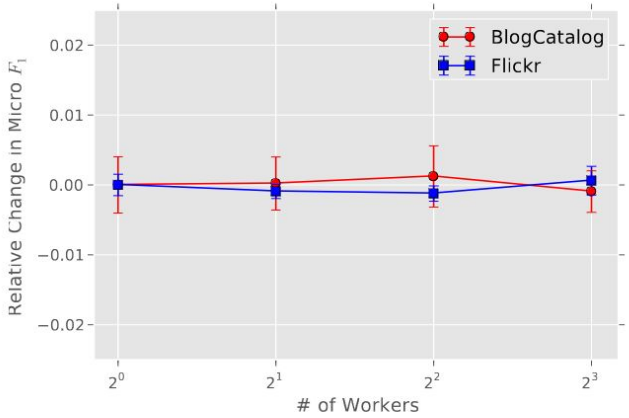
Algorithm 2 SkipGram(Φ, $\mathcal{W}_{v_i}$, w)

for each $v_j \in \mathcal{W}_{v_i}$ **do**
  for each $u_k \in \mathcal{W}_{v_i}[j - w : j + w]$ **do**
    $J(\Phi) = -\log \Pr(u_k \mid \Phi(v_j))$
    $\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$
  end for
end for

Parallelizability



(a) Running Time



(b) Performance

Updates that affect embedding matrix will be sparse in nature, so

- Easy to parallelize
- Easy to accommodate small changes

Datasets

Name	BLOGCATALOG	FLICKR	YOUTUBE
$\|V\|$	10,312	80,513	1,138,499
$\|E\|$	333,983	5,899,882	2,990,443
$\|\mathcal{Y}\|$	39	195	47
Labels	Interests	Groups	Groups

Table 1: Graphs used in our experiments.

- **BlogCatalog** is a **network of social relationships** provided by blogger authors. The labels represent the topic categories provided by the authors.
- **Flickr** is a **network of the contacts** between users of the photo sharing website. The labels represent the interest groups of the users such as ‘black and white photos’.
- **YouTube** is a **social network** between users of the popular video sharing website. The labels here represent groups of viewers that enjoy common video genres (e.g. anime and wrestling).

Deep Walk Performance

	% Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1(%)	DeepWalk	**36.00**	**38.20**	**39.60**	**40.30**	**41.00**	**41.30**	41.50	41.50	42.00
	SpectralClustering	31.06	34.95	37.27	38.93	39.97	40.99	**41.66**	**42.42**	**42.62**
	EdgeCluster	27.94	30.76	31.85	32.99	34.12	35.00	34.63	35.99	36.29
	Modularity	27.35	30.74	31.77	32.97	34.09	36.13	36.08	37.23	38.18
	wvRN	19.51	24.34	25.62	28.82	30.37	31.81	32.19	33.33	34.28
	Majority	16.51	16.66	16.61	16.70	16.91	16.99	16.92	16.49	17.26
Macro-F1(%)	DeepWalk	**21.30**	**23.80**	25.30	26.30	27.30	27.60	27.90	28.20	28.90
	SpectralClustering	19.14	23.57	**25.97**	**27.46**	**28.31**	**29.46**	**30.13**	**31.38**	**31.78**
	EdgeCluster	16.16	19.16	20.48	22.00	23.00	23.64	23.82	24.61	24.92
	Modularity	17.36	20.00	20.80	21.85	22.65	23.41	23.89	24.20	24.97
	wvRN	6.25	10.13	11.64	14.24	15.86	17.18	17.98	18.86	19.57
	Majority	2.52	2.55	2.52	2.58	2.58	2.63	2.61	2.48	2.62

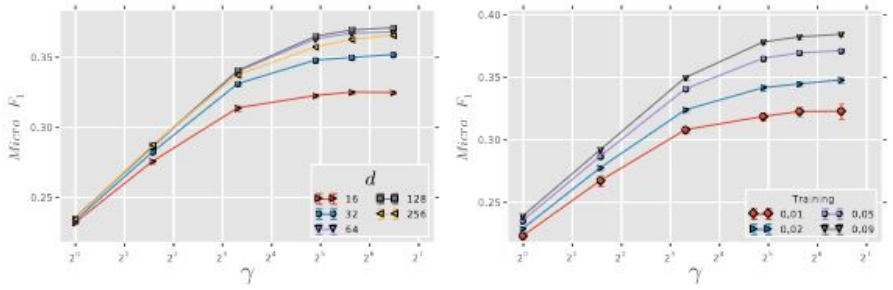
Table 2: Multi-label classification results in BlogCatalog

Deep Walk Performance

	% Labeled Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-F1(%)	DeepWalk	**37.95**	**39.28**	**40.08**	**40.78**	**41.32**	**41.72**	**42.12**	**42.48**	**42.78**	**43.05**
	SpectralClustering	—	—	—	—	—	—	—	—	—	—
	EdgeCluster	23.90	31.68	35.53	36.76	37.81	38.63	38.94	39.46	39.92	40.07
	Modularity	—	—	—	—	—	—	—	—	—	—
	wvRN	26.79	29.18	33.1	32.88	35.76	37.38	38.21	37.75	38.68	39.42
	Majority	24.90	24.84	25.25	25.23	25.22	25.33	25.31	25.34	25.38	25.38
Macro-F1(%)	DeepWalk	**29.22**	**31.83**	**33.06**	**33.90**	**34.35**	**34.66**	**34.96**	**35.22**	**35.42**	**35.67**
	SpectralClustering	—	—	—	—	—	—	—	—	—	—
	EdgeCluster	19.48	25.01	28.15	29.17	29.82	30.65	30.75	31.23	31.45	31.54
	Modularity	—	—	—	—	—	—	—	—	—	—
	wvRN	13.15	15.78	19.66	20.9	23.31	25.43	27.08	26.48	28.33	28.89
	Majority	6.12	5.86	6.21	6.1	6.07	6.19	6.17	6.16	6.18	6.19

Table 4: Multi-label classification results in YouTube

Parameter Sensitivity



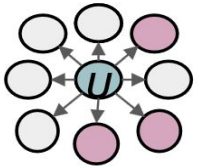
(b1) FLICKR, $T_R = 0.05$ (b2) FLICKR, $d = 128$

(b3) BLOGCATALOG, $T_R = 0.5$ (b4) BLOGCATALOG, $d = 128$

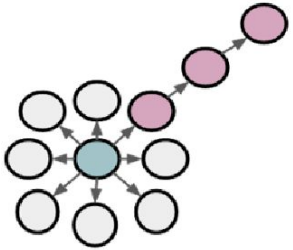
(a) Stability over number of walks, γ

node2vec

use flexible, biased random walks that can trade off between local and global views of the network



BFS:
Micro-view of neighbourhood



DFS:
Macro-view of neighbourhood

Advantages & Limitations of Random Walks Approach

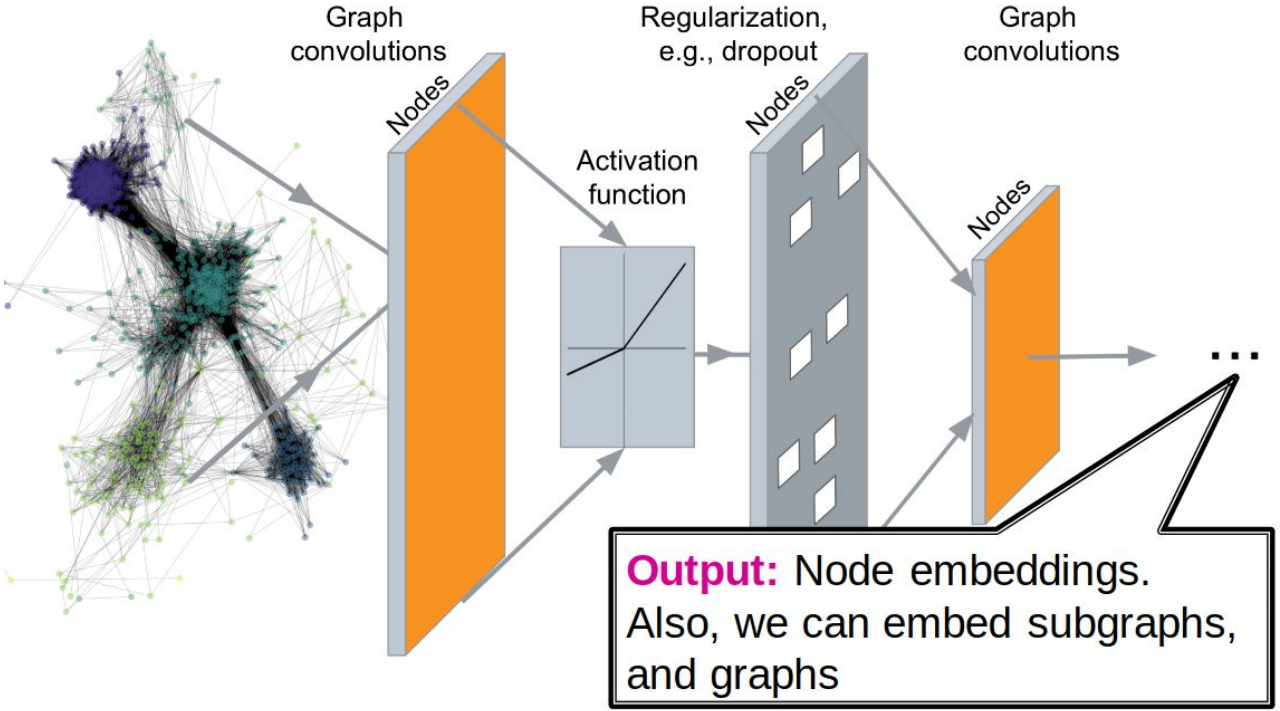
Advantages

- Beats baselines given significantly **less labeled** data
- Can make useful models of **various sizes**
- **Strong performance** with simple linear classifiers (logistic regression)

Limitations

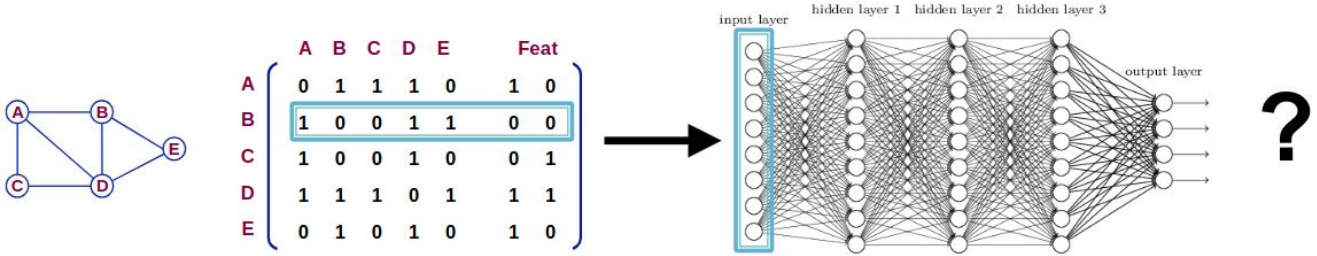
- $O(|V|d)$ parameters are needed
- Cannot generate embeddings for nodes that are not seen during training
- Do not incorporate node features

Deep Graph Encoders



Naive Approach

- Join adjacency matrix and features
- Feed them into a deep neural net:

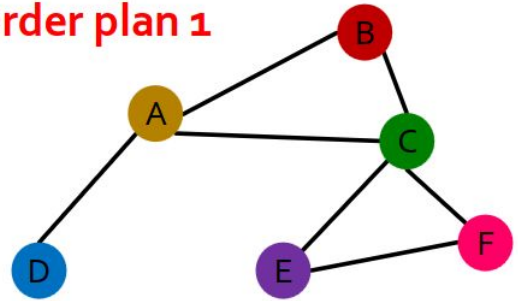


- Issues with this idea:
  - $O(|V|)$ parameters
  - Not applicable to graphs of different sizes
  - Sensitive to node ordering

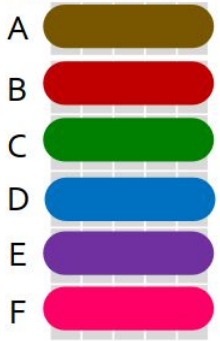
Naive Approach

- **Graph does not have a canonical order of the nodes!**

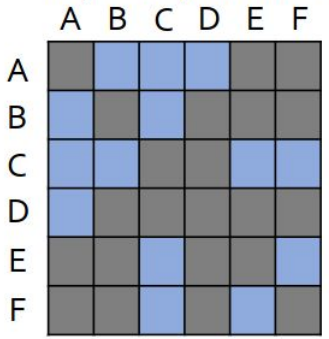
Order plan 1



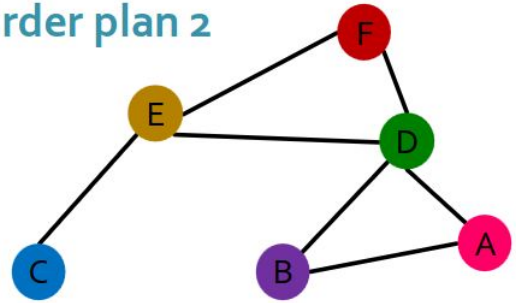
Node features X_1



Adjacency matrix A_1

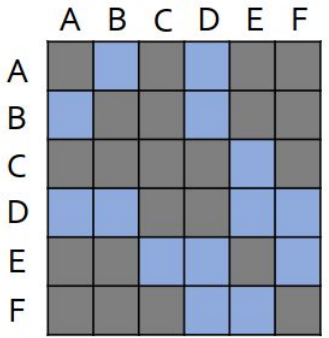


Order plan 2



Node features X_2

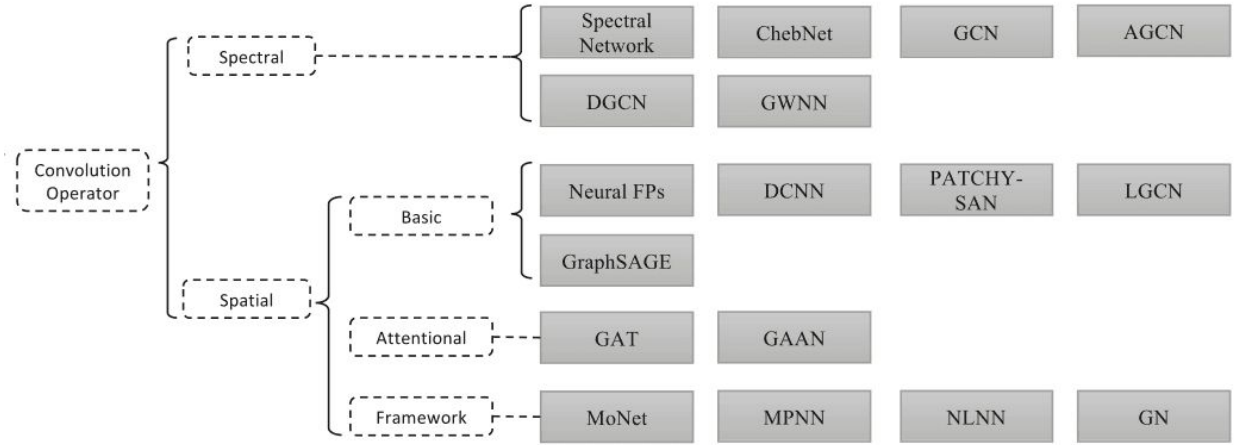
Adjacency matrix A_2



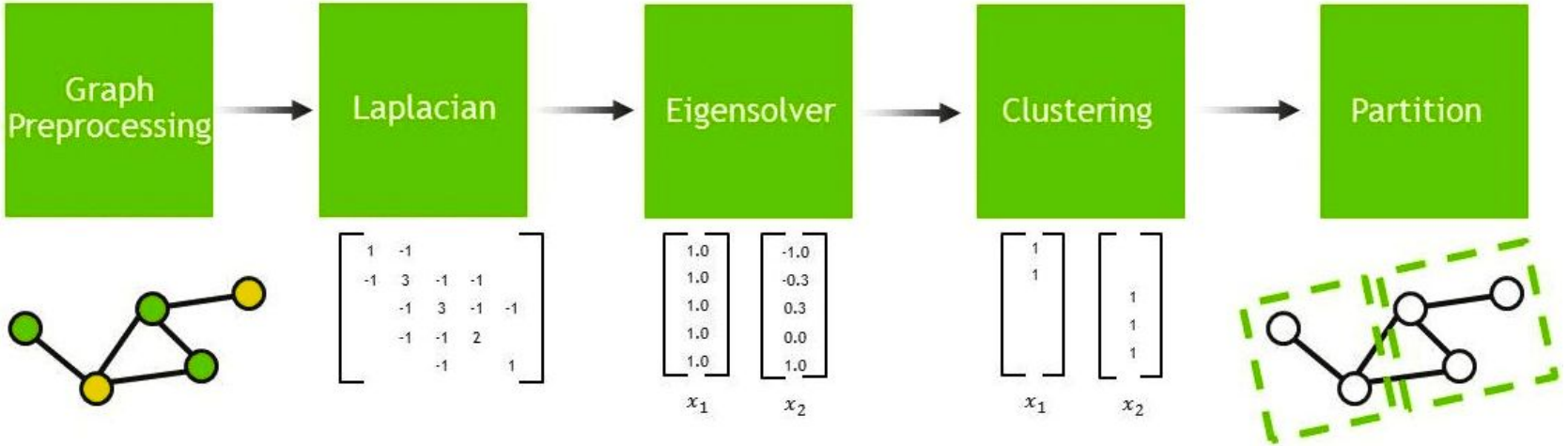
Spectral vs Spatial Convolutions

- Spectral approach is based on graph laplacian and its eigenvectors
- Spatial approach is based on message-passing paradigm

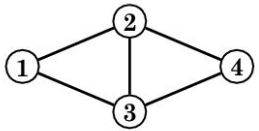
- Spectral approach works better on large graphs, while spatial approach better finds local structures inside a graph



Spectral Convolution



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



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

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

Spectral Convolution: GCN Layer

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) .$$

Here, $\tilde{A} = A + I_N$ is the adjacency matrix of the undirected graph $\mathcal{G}$ with added self-connections. I_N is the identity matrix, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ and $W^{(l)}$ is a layer-specific trainable weight matrix. $\sigma(\cdot)$ denotes an activation function, such as the $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$. $H^{(l)} \in \mathbb{R}^{N \times D}$ is the matrix of activations in the l^{th} layer; $H^{(0)} = X$. Let's define graph contraction for 2 signals as

$$g_\theta \star x = U g_\theta U^\top x \,,$$ U is the matrix of eigenvectors of the normalized graph Laplacian $L = I_N - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$

$$g_{\theta'} \star x \approx \theta'_0 x + \theta'_1 \left(L - I_N\right) x = \theta'_0 x - \theta'_1 D^{-\frac{1}{2}}AD^{-\frac{1}{2}}x \,,$$

$$g_\theta \star x \approx \theta \left(I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}\right) x \,,$$

$$Z = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\Theta \,.$$ *renormalization trick*: $I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \to \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$,

Why GCN

Table 3: Comparison of propagation models.

Description		Propagation model	Citeseer	Cora	Pubmed
Chebyshev filter (Eq. 5)	$K = 3$	$\sum_{k=0}^{K} T_k(\tilde{L})X\Theta_k$	69.8	79.5	74.4
	$K = 2$		69.6	81.2	73.8
1^{st}-order model (Eq. 6)		$X\Theta_0 + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta_1$	68.3	80.0	77.5
Single parameter (Eq. 7)		$(I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})X\Theta$	69.3	79.2	77.4
Renormalization trick (Eq. 8)		$\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\Theta$	**70.3**	**81.5**	**79.0**
1^{st}-order term only		$D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta$	68.7	80.5	77.8
Multi-layer perceptron		$X\Theta$	46.5	55.1	71.4

- More layers for fixed computational budget
- Avoids overfitting on local neighbourhood structure

Simple GCN Model

$$Z = f(X, A) = \mathrm{softmax}\Big(\hat{A}\ \mathrm{ReLU}\Big(\hat{A}XW^{(0)}\Big)\,W^{(1)}\Big) \qquad \hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$$

$W^{(0)} \in \mathbb{R}^{C\times H}$ is an input-to-hidden weight matrix for a hidden layer with H feature maps

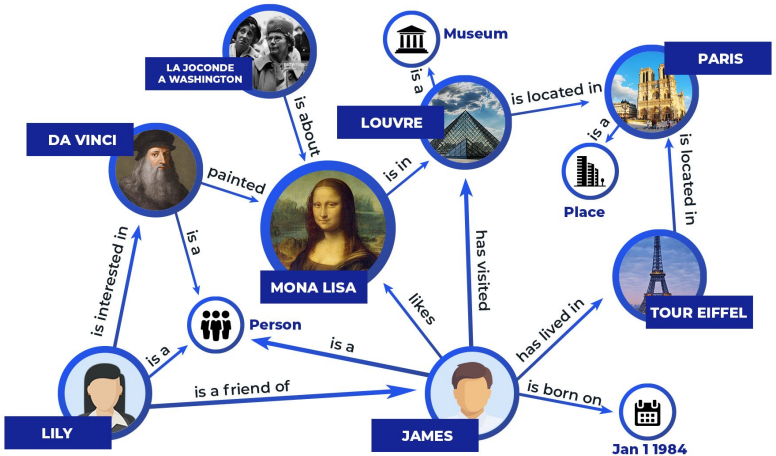
$W^{(1)} \in \mathbb{R}^{H\times F}$ is a hidden-to-output weight matrix.

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

Datasets

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

Citation Networks

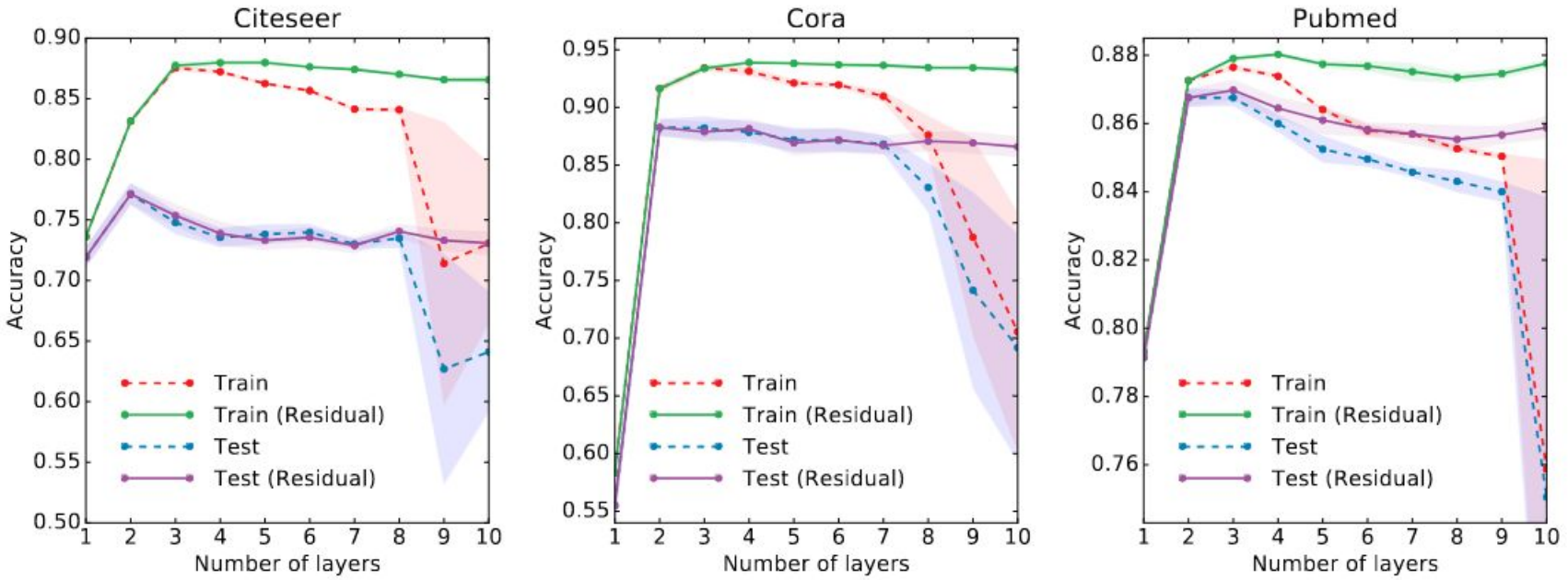


Results

Table 2: Summary of results in terms of classification accuracy (in percent).

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

Effects on model depth



Advantages & Limitations

Advantages

- One step realisation
- Time-effective & great quality

Limitations

- Does not support edge features
- Limited to undirected graphs

GraphSAGE

A **transductive** approach

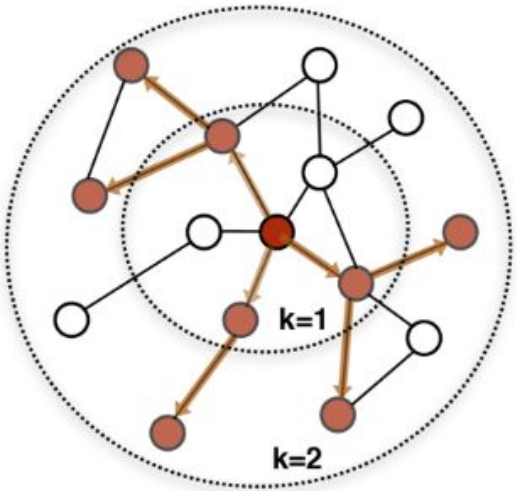
- Optimizes embeddings for each node in a single fixed graph

An **inductive** approach:

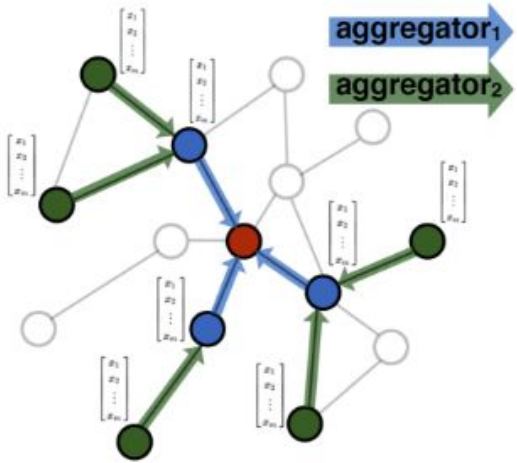
- Creates embeddings for previously unseen nodes/new subgraphs
- Generalizes over graphs with the same form of features
- Should learn both the node's local role as well as its global position

In GraphSAGE, instead of training individual embeddings, we learn a set of functions that **sample(SA)** and **aggregate(GE)** features from the node's local neighbourhood. At test, we use our system to generate embeddings

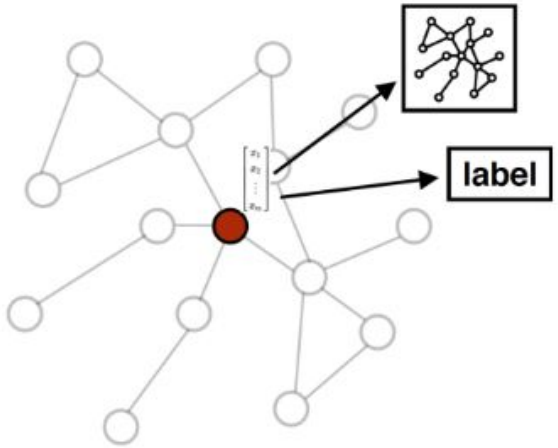
Spatial Convolution: GraphSAGE Layer



1. Sample neighborhood



2. Aggregate feature information from neighbors



3. Predict graph context and label using aggregated information