Graph Neural Networks

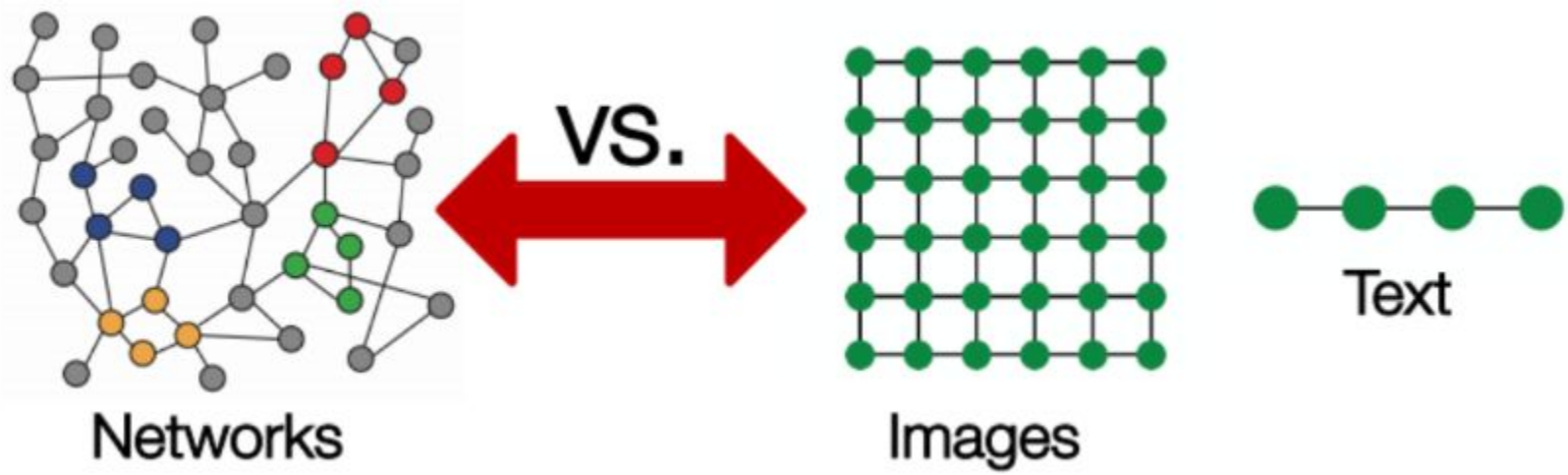
Yusupov Viacheslav

Content

- What is GNN?
- Applications
- Classification
- DeepWalk
- Graph Convolutional Network (GCN)

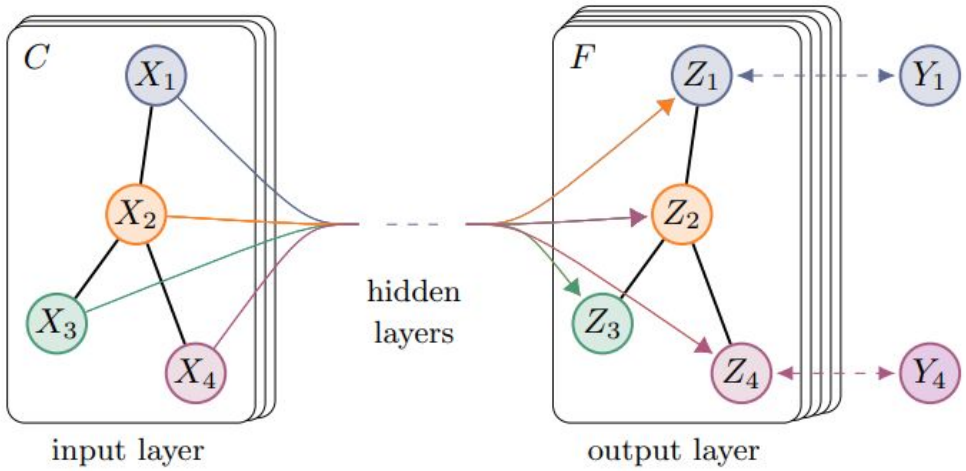
What is GNN?

Graph Neural Network (GNN) - a class of artificial neural networks for processing data that can be represented as graphs.

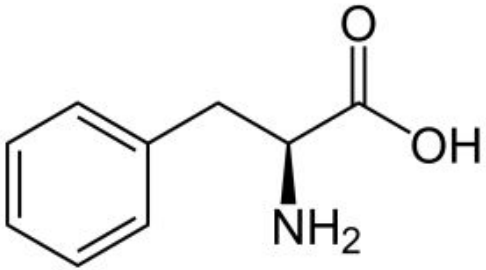


Types of GNNs

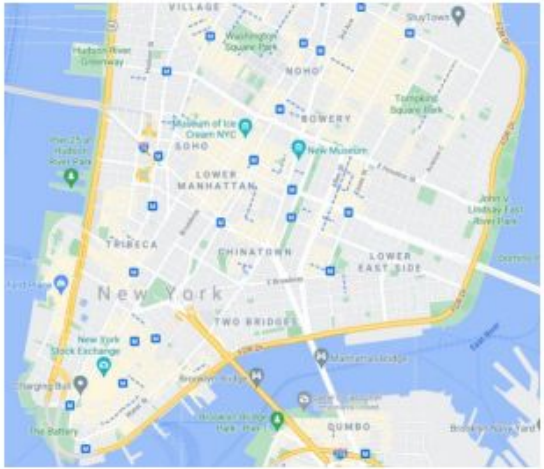
- Node Embeddings
- Random Walks
  - Node2vec
  - DeepWalk
- Convolutional Networks
  - GCN
  - GAT
  - GraphSAGE
  - etc.
- Transformers
  - Graphormer



Graphs in the World



phenylalanine

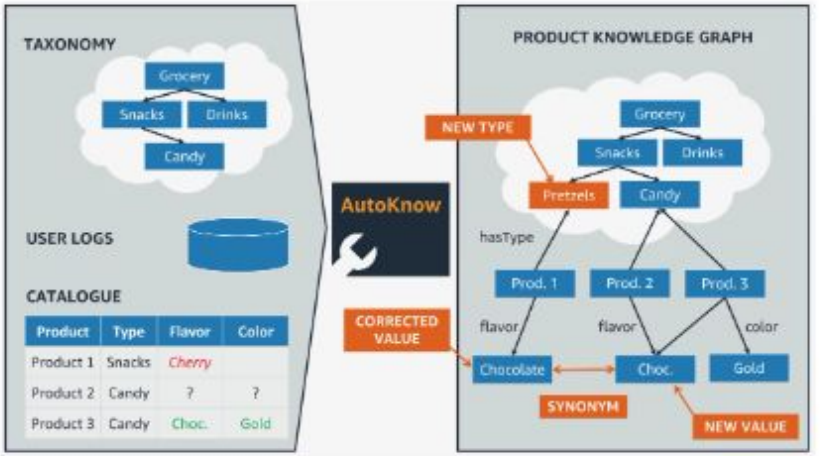


Map of Manhattan

Social Network

Recommender systems

- Uber
- Alibaba
- Pinterest
- Amazon
- etc.



The inputs to AutoKnow include an existing product taxonomy, user logs, and a product catalogue. AutoKnow automatically combines data from all three sources into a product graph, adding new product types to the taxonomy, adding new values for product attributes, correcting errors, and identifying synonyms.

CREDIT: STACY REILLY

Optimization

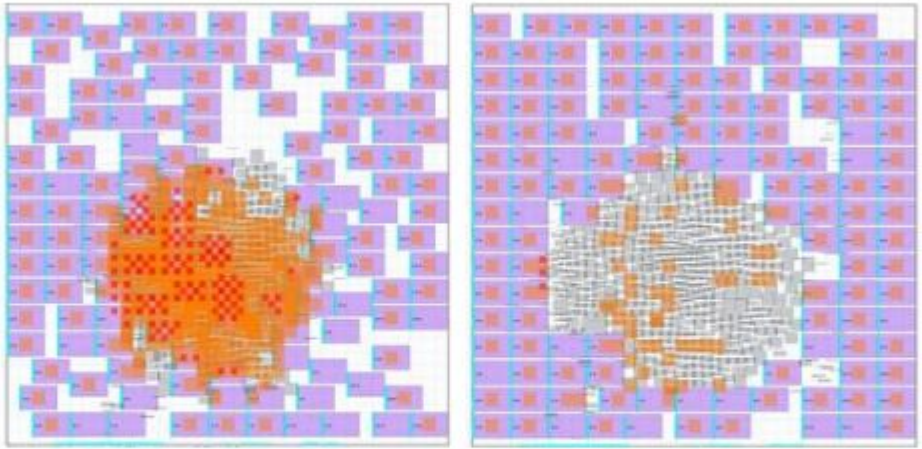
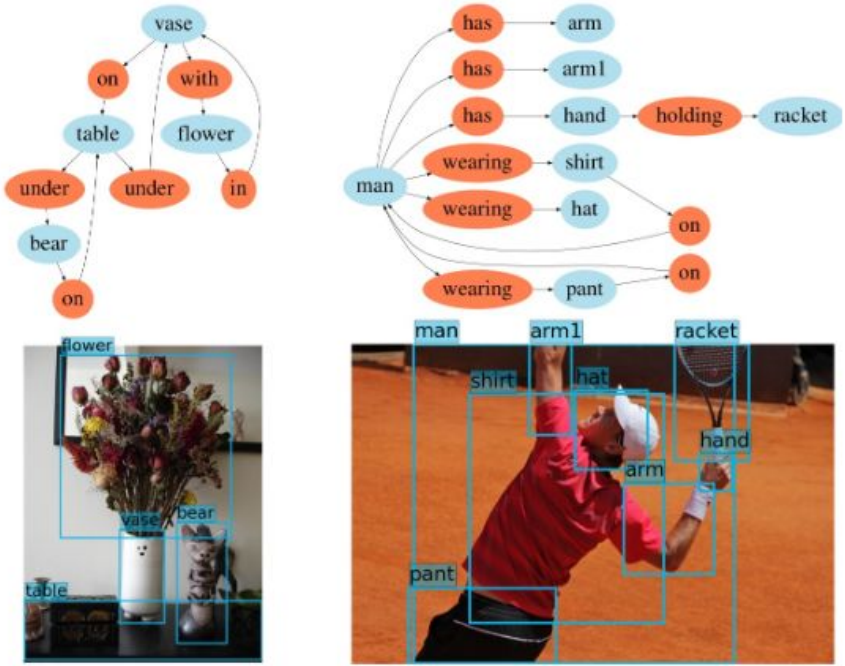


Figure 6. Visualization of placements. On the left, zero-shot placements from the pre-trained policy and on the right, placements from the finetuned policy are shown. The zero-shot policy placements are generated at inference time on a previously unseen chip. The pre-trained policy network (with no fine-tuning) places the standard cells in the center of the canvas surrounded by macros, which is already quite close to the optimal arrangement and in line with the intuitions of physical design experts.

https://arxiv.org/pdf/2004.10746.pdf

Computer Vision



https://cs.stanford.edu/~danfei/scene-graph/

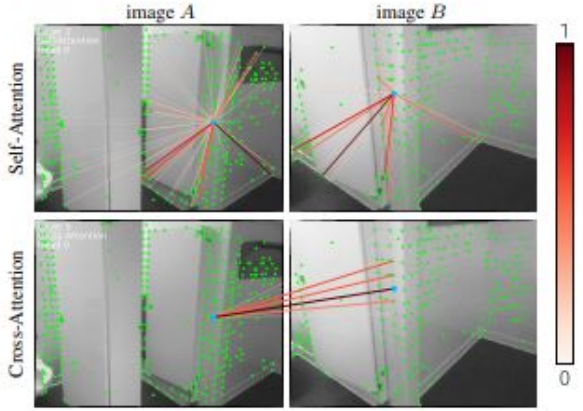
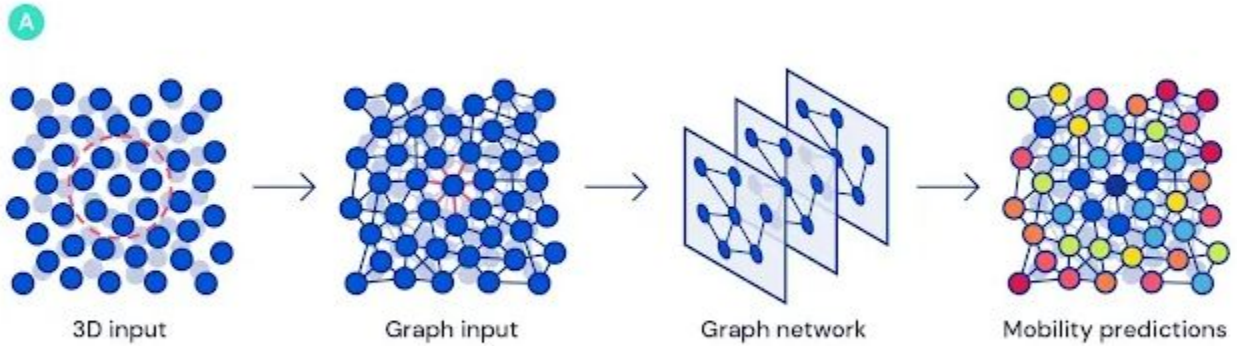


Figure 4: **Visualizing self- and cross-attention.** Attentional aggregation builds a dynamic graph between keypoints. Weights α_{ij} are shown as rays. Self-attention (top) can attend anywhere in the same image, e.g. distinctive locations, and is thus not restricted to nearby locations. Cross-attention (bottom) attends to locations in the other image, such as potential matches that have a similar appearance.

https://arxiv.org/pdf/1911.11763.pdf

Physics and Chemistry



https://deepmind.google/discover/blog/towards-understanding-glasses-with-graph-neural-networks/

Pharmaceuticals

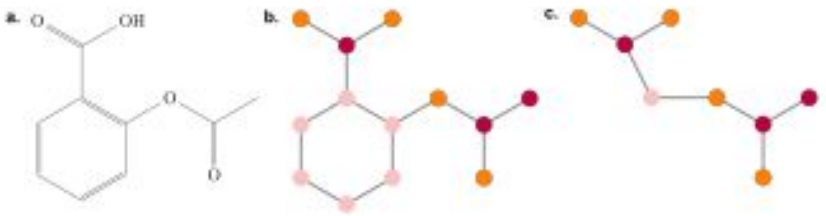


Figure 6: Illustration of **a.** the molecule aspirin, **b.** its basic graph representation, and **c.** the associated junction tree representation. Colours on the node correspond to atom types.

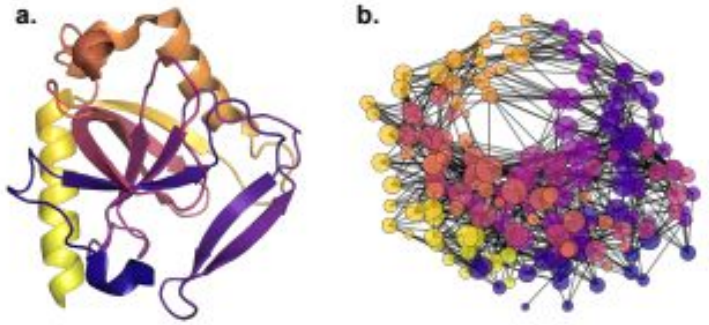
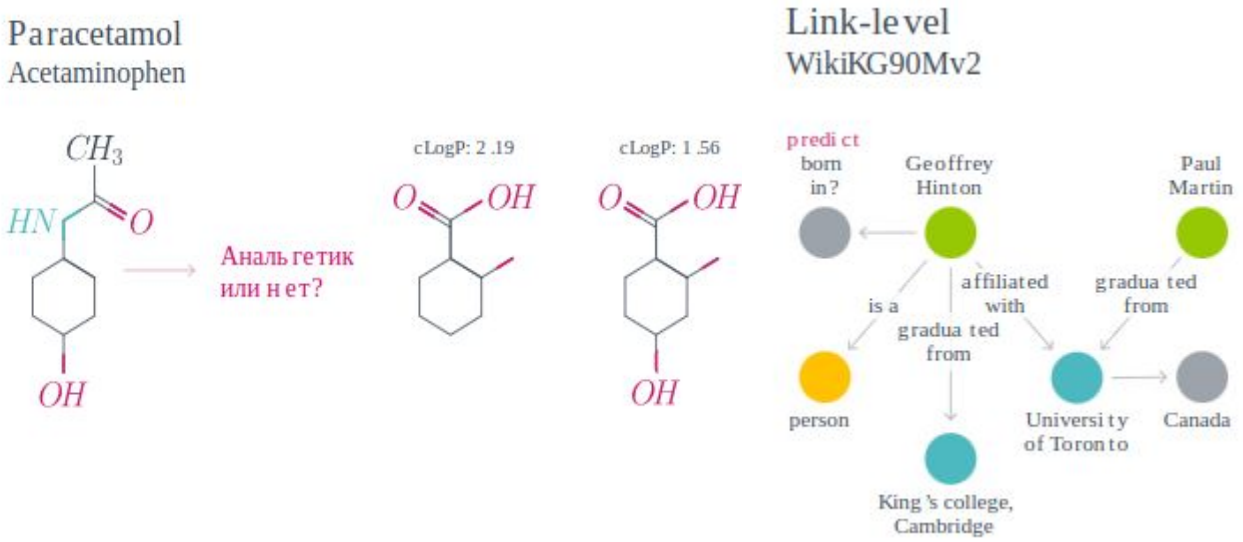


Figure 7: Illustration of **a.** a protein (PDB accession: 3EIY) and **b.** its graph representation derived based on intramolecular distance with cut-off threshold set at 10Å.

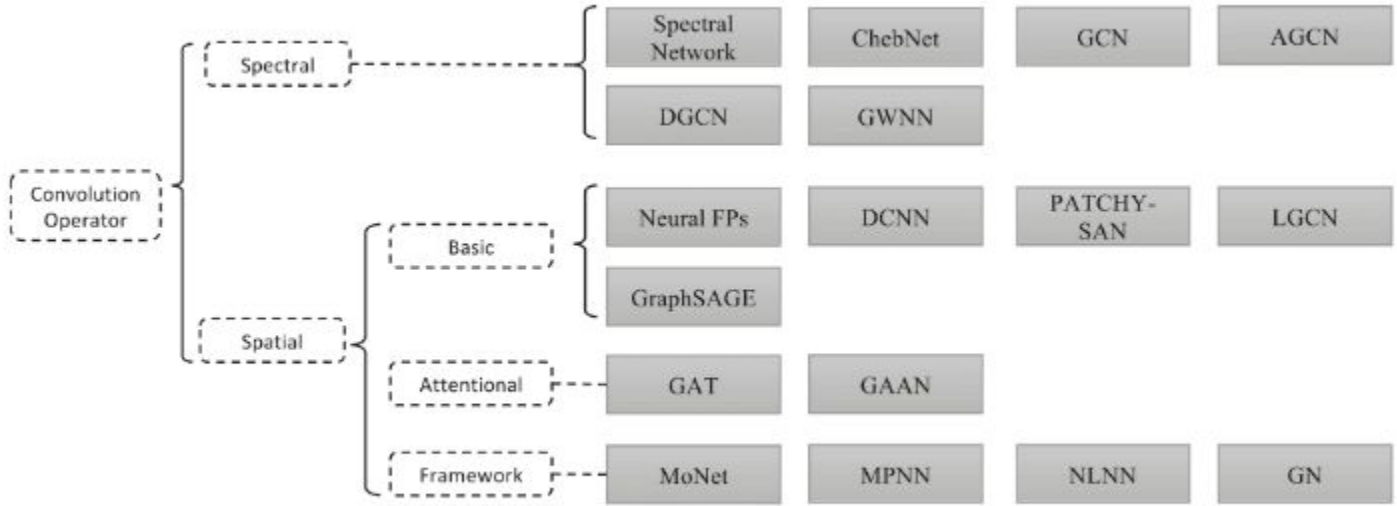
https://arxiv.org/pdf/2012.05716.pdf

Classification

- Graph level
- Node level
- Edge level



GNN Classification



GNN Classification

- The spatial paradigm is based on the message passing algorithm between the vertices of the graph.

- The spectral paradigm relies on the analysis of the signal diffusion process inside the graph and analyzes the matrices describing the graph – the adjacency matrix and the matrix, which is called the Laplacian of the graph.

DeepWalk

- Model input: An undirected graph with partially classified nodes.
- Model output: Classes of unclassified nodes.
- Parameters:
  - t - number of random walks
  - w - size of window
  - d - embedding size
  - **γ** - walks per vertex

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**
    $\mathcal{W}_{v_i} = RandomWalk(G, v_i, t)$
    SkipGram(Φ, $\mathcal{W}_{v_i}$, w)
  end for
end for

DeepWalk

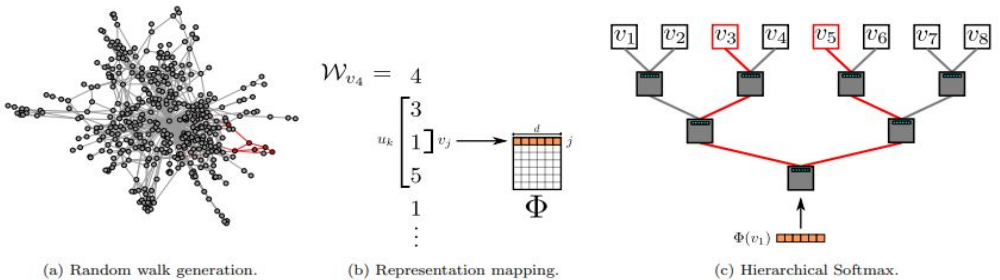
SkipGram

Algorithm 2 SkipGram(Φ, W_{v_i}, w)

1: **for each** $v_j \in W_{v_i}$ **do**
2: **for each** $u_k \in W_{v_i}[j-w : j+w]$ **do**
3: $J(\Phi) = -\log \Pr(u_k \mid \Phi(v_j))$
4: $\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$
5: **end for**
6: **end for**

Hierarchical Softmax

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



(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 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 $\{v_3, v_5\}$.

Experiments

	% 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

Experiments

	% Labeled Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-F1(%)	DeepWalk	**32.4**	**34.6**	**35.9**	**36.7**	**37.2**	**37.7**	**38.1**	**38.3**	**38.5**	**38.7**
	SpectralClustering	27.43	30.11	31.63	32.69	33.31	33.95	34.46	34.81	35.14	35.41
	EdgeCluster	25.75	28.53	29.14	30.31	30.85	31.53	31.75	31.76	32.19	32.84
	Modularity	22.75	25.29	27.3	27.6	28.05	29.33	29.43	28.89	29.17	29.2
	wvRN	17.7	14.43	15.72	20.97	19.83	19.42	19.22	21.25	22.51	22.73
	Majority	16.34	16.31	16.34	16.46	16.65	16.44	16.38	16.62	16.67	16.71
Macro-F1(%)	DeepWalk	**14.0**	17.3	**19.6**	**21.1**	**22.1**	**22.9**	**23.6**	**24.1**	**24.6**	**25.0**
	SpectralClustering	13.84	**17.49**	19.44	20.75	21.60	22.36	23.01	23.36	23.82	24.05
	EdgeCluster	10.52	14.10	15.91	16.72	18.01	18.54	19.54	20.18	20.78	20.85
	Modularity	10.21	13.37	15.24	15.11	16.14	16.64	17.02	17.1	17.14	17.12
	wvRN	1.53	2.46	2.91	3.47	4.95	5.56	5.82	6.59	8.00	7.26
	Majority	0.45	0.44	0.45	0.46	0.47	0.44	0.45	0.47	0.47	0.47

Table 3: Multi-label classification results in Flickr

Experiments

	% 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

Graph Convolutional Network (GCN)

Model Input:

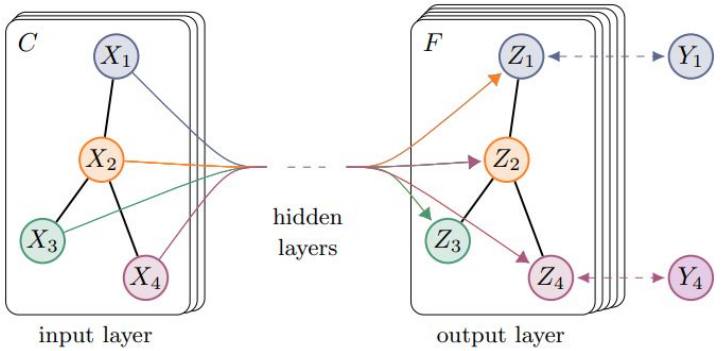
A in R^{nxn} - Adjacency Matrix

X in R^{nxd} - Matrix of Features

Model Output:

Z in R^{nxc} - Matrix of class probabilities

Optimizer: Adam



Graph Convolutional Network (GCN)

Graph Convolutions

$$g_\theta \star x = U g_\theta U^\top x$$

$$L = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U \Lambda U^\top$$

$$g_{\theta'} \star x \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{L}) x \,,$$

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

GCN Layer

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

$$\tilde{A} = A + I_N$$

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

A - Adjacency matrix of a Graph

H - Output of a layer

W - Weights of a layer

Experiments

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

Experiments

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

Thanks for your attention