

Semi-Supervised Classification with Graph Convolutional Networks

Thomas N.Kipf Max Welling

University of Amsterdam

Canadian Institute for Advanced Research(CIFAR)





Max Welling

Professor Machine Learning, [University of Amsterdam](#)
在 uva.nl 的电子邮件经过验证

[Machine Learning](#) [Artificial Intelligence](#) [Statistics](#)

关注


创建我的个人资料

标题	引用次数	年份
Auto-encoding variational bayes DP Kingma, M Welling arXiv preprint arXiv:1312.6114	1837	2013
Unsupervised learning of models for recognition M Weber, M Welling, P Perona European conference on computer vision, 18-32	830	2000
On smoothing and inference for topic models A Asuncion, M Welling, P Smyth, YW Teh Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial ...	485	2009
A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation YW Teh, D Newman, M Welling Advances in neural information processing systems, 1353-1360	462	2007
Fast collapsed gibbs sampling for latent dirichlet allocation I Porteous, D Newman, A Ihler, A Asuncion, P Smyth, M Welling Proceedings of the 14th ACM SIGKDD international conference on Knowledge ...	449	2008
Semi-supervised learning with deep generative models DP Kingma, S Mohamed, DJ Rezende, M Welling Advances in Neural Information Processing Systems, 3581-3589	445	2014
Bayesian learning via stochastic gradient Langevin dynamics M Welling, YW Teh Proceedings of the 28th International Conference on Machine Learning (ICML ...	429	2011
Exponential family harmoniums with an application to information retrieval M Welling, M Rosen-Zvi, GE Hinton Advances in neural information processing systems, 1481-1488	420	2005
Distributed algorithms for topic models D Newman, A Asuncion, P Smyth, M Welling Journal of Machine Learning Research 10 (Aug), 1801-1828	332	2009
Towards automatic discovery of object categories M Weber, M Welling, P Perona Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference ...	297	2000
Distributed inference for latent dirichlet allocation	244	2008

引用次数 [查看全部](#)



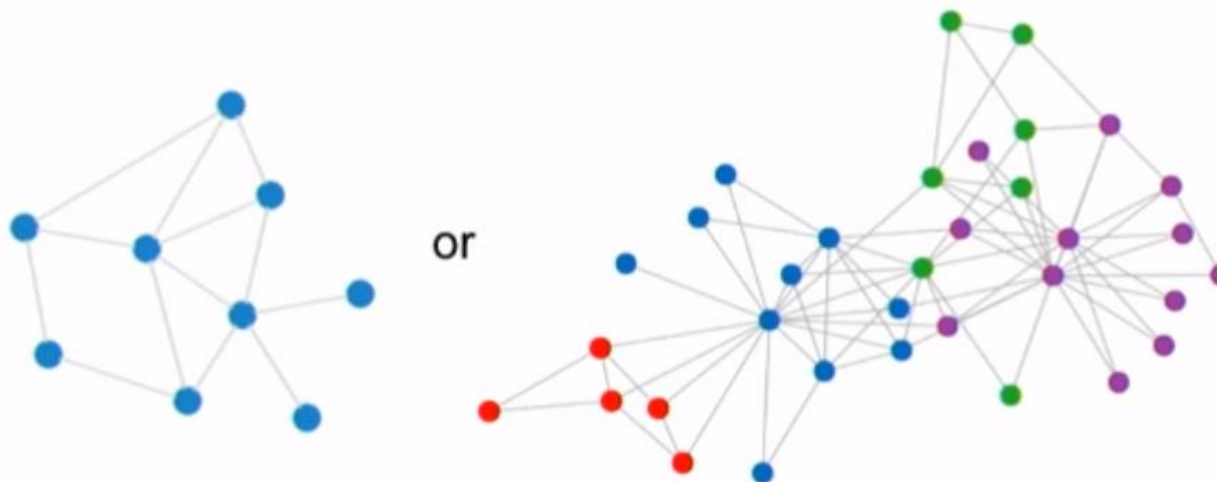
合著作者

 **Paul Newman**
Professor University of Oxford >

Outline

- Introduction
- GCN
- Experiments
- Conclusions

Graph-structured data



Real-world example:

- Social networks
- World-wide-web
- Protein-interaction networks
- Citation networks
- Knowledge graphs
-

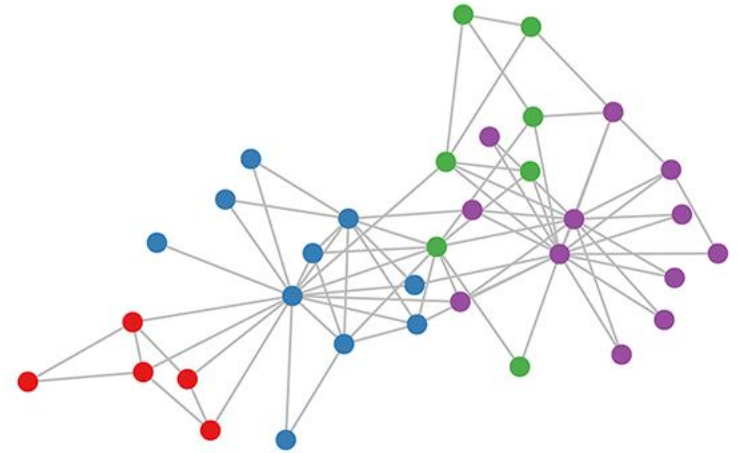
Semi-supervised classification on graphs

- **Setting:**

A small number of labeled nodes and
large number of unlabeled nodes

- **Task:**

Predict node label of unlabeled nodes



- **Standard approach:**

– Graph-based regularization (“smoothness constraints”) [Zhu et al., 2003]

$$\mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_{\text{reg}}, \quad \text{with} \quad \mathcal{L}_{\text{reg}} = \sum_{i,j} A_{ij} \|f(X_i) - f(X_j)\|^2$$

Assumption: connected nodes likely to share same label

Semi-supervised classification on graphs

Embedding-based approaches

Two-step pipeline:

- 1) Get embedding for nodes
- 2) Train classifier on node embedding

Examples:

- DeepWalk [Perozzi et al., 2014]
- node2vec [Grover & Leskovec, 2016]

Problem: Embeddings are not optimized for classification task!

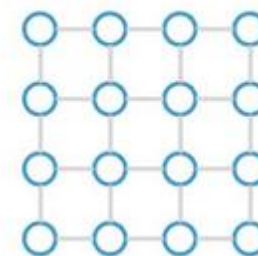
Solution: Train graph-based classifier end-to-end using GCN

Convolutional network

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved
Feature

2D grid

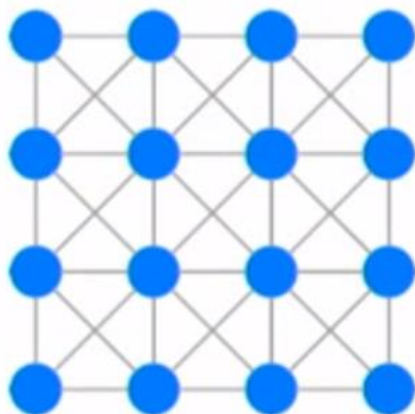


1D grid

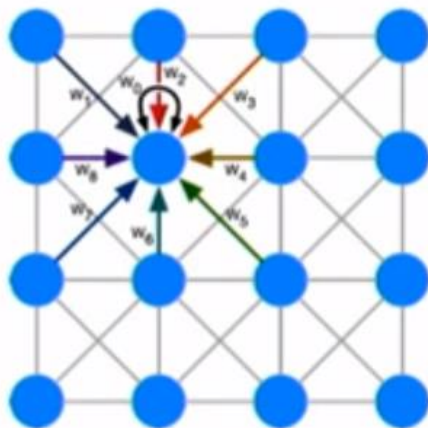
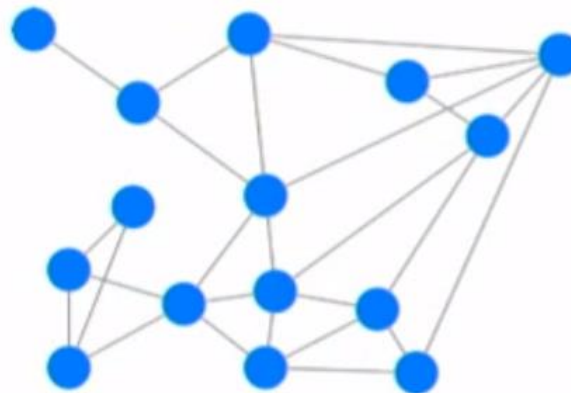
Convolutional network is a useful tool on **Euclidean Structure graphs**

Generalizing networks to arbitrarily structure graphs is a challenging problem

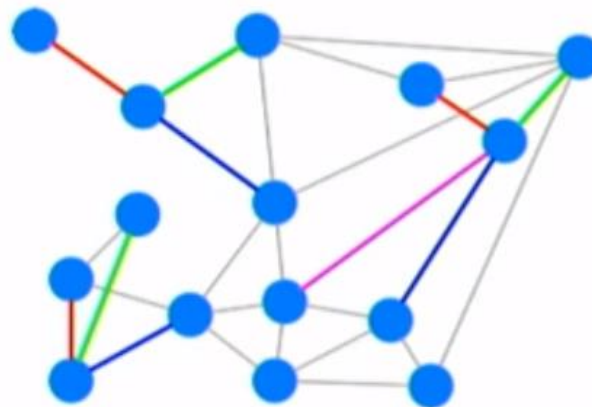
An example



vs.



vs.



Outline

- Introduction
- GCN
- Experiments
- Conclusions

GCNs Part I: Definitions

The goal is to learn a function of features on a graph $G = (V, E)$ which takes as input:

- A $N \times D$ **feature matrix** X (N: number of nodes, D: number of features)
- A representative description of the graph structure in matrix form; typically in the form of **adjacency matrix** A

and produces a node-level output Z (an $N \times F$ feature matrix where F is the number of output features per node)

Every neural network layer can be written as $H^{(l+1)} = f(H^{(l)}, A)$

where $H^{(0)} = X$ and $H^{(L)} = Z$

GCNs Part II: Two tricks

$$f(H^{(l)}, A) = \sigma \left(AH^{(l)}W^{(l)} \right)$$

Multiplication with A means for every node, we sum up all the feature vectors of all neighboring nodes but the node itself

Solution: simply add the identity matrix to A

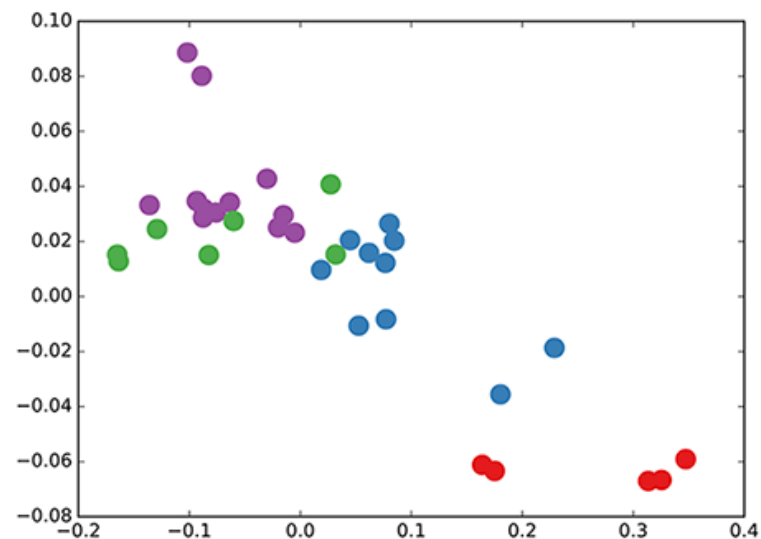
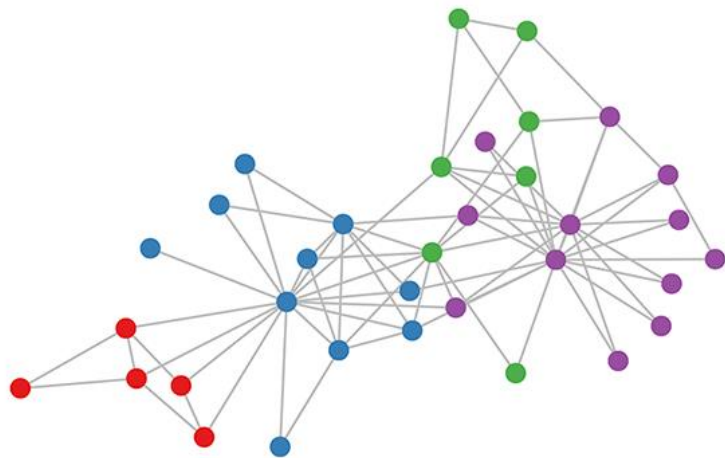
A is not normalized and therefore the multiplication with A will change the scale of the feature vectors

Solution: Normalizing A such that all rows sum to one, i.e. $D^{-1}A$. In practice, we use $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$

$$f(H^{(l)}, A) = \sigma \left(\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} H^{(l)} W^{(l)} \right) \quad \text{where} \quad \hat{A} = A + I,$$

GCNs Part III: Embedding the karate club network

- Take a 3-layer GCN with randomly initialized weights
- Simply insert the adjacency matrix and $X = I$



GCNs Part IV: Theory(1)

Weisfeiler-Lehman algorithm:

For all nodes v_i :

- Get features $\{h_{v_j}\}$ of neighboring nodes $\{v_i\}$
- update node feature $h_{v_i} \leftarrow \text{hash}(\sum h_{v_j})$

Repeat for k steps or until convergence

For GCN:

$$h_{v_i}^{(l+1)} = \sigma \left(\sum_j \frac{1}{c_{ij}} h_{v_j}^{(l)} W^{(l)} \right)$$

The propagation rule can be interpreted as a differentiable and parameterized variant of hash function

GCNs Part IV: Theory(2)

- Spectral Graph Theory:

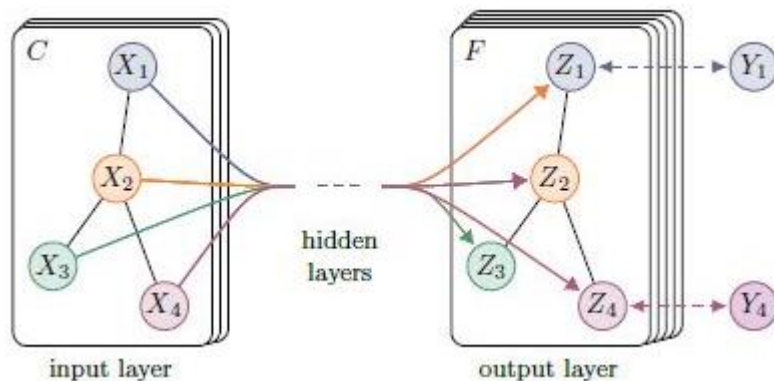
$$g_\theta \star x = U g_\theta U^\top x$$

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

- Chebyshev polynomials and other approximations:

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

GCNs Part VI: Semi-supervised



- Calculate $\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$
- Get the result: $Z = f(X, A) = \text{softmax}\left(\hat{A} \text{ReLU}\left(\hat{A} X W^{(0)}\right) W^{(1)}\right)$
- Use cross entropy as the loss function:

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

Outline

- Introduction
- GCN
- Experiments
- Conclusions

Experiments

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

Comparison of propagation models

Description	Propagation model	Citeseer	Cora	Pubmed
Chebyshev filter (Eq. 5)	$K = 3$	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

Training time per epoch

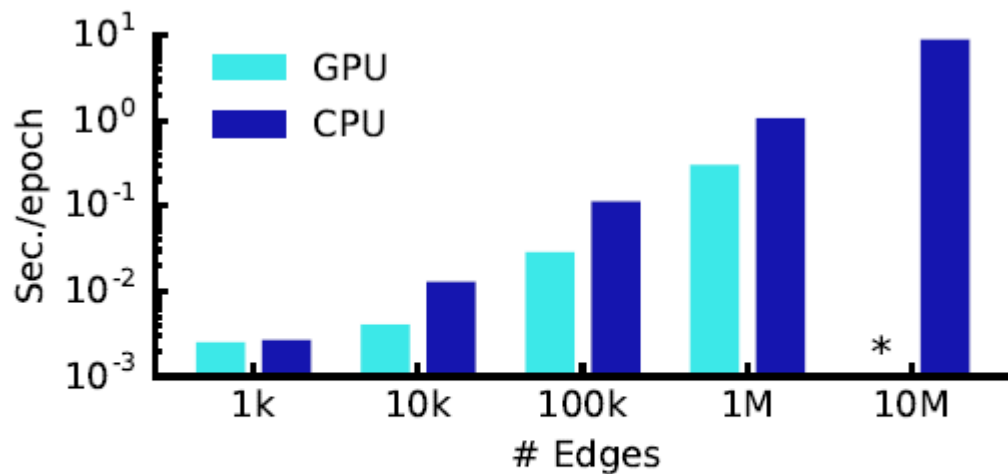
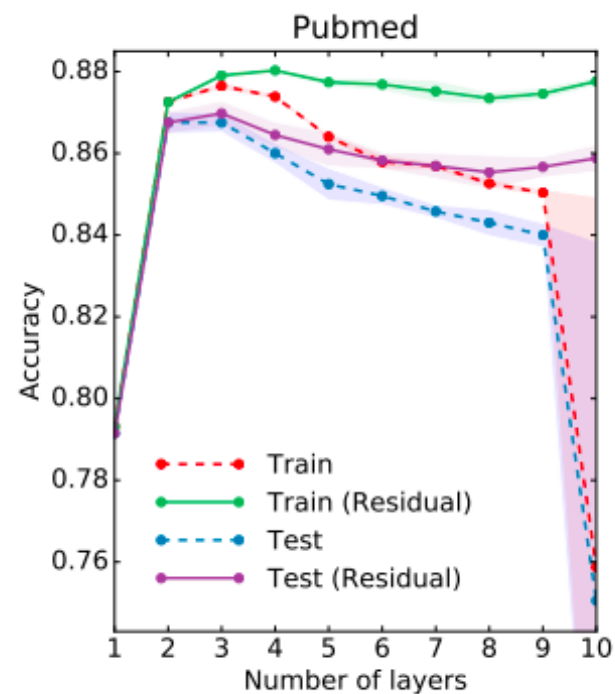
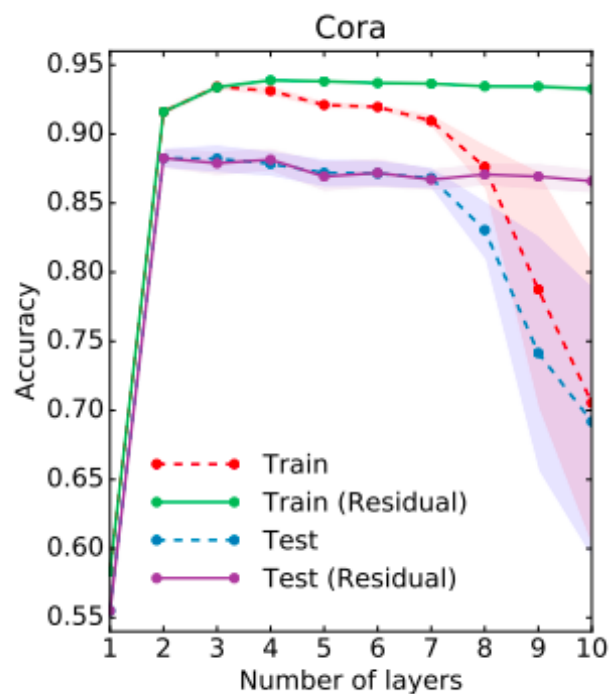
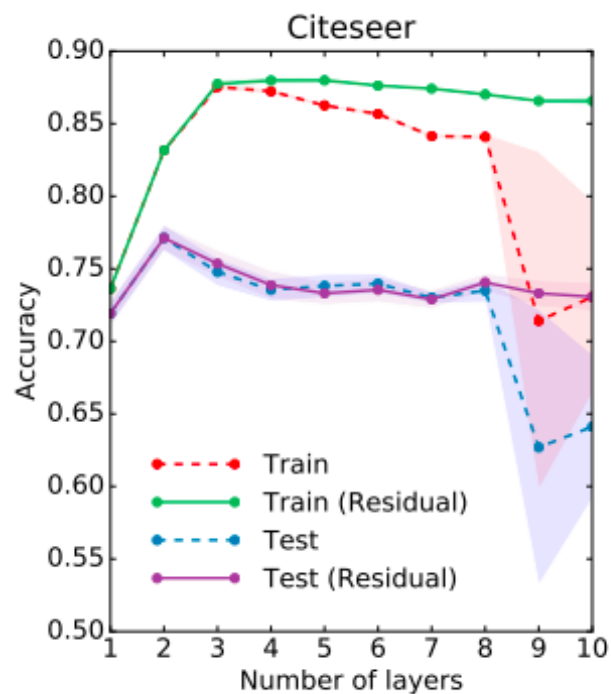


Figure 2: Wall-clock time per epoch for random graphs. (*) indicates out-of-memory error.

Influence of model depth



Outline

- Introduction
- GCN
- Experiments
- Conclusions

Conclusions

- Proposed a novel approach for SSC on graph-structured data
- Discussion:
 - Memory requirement
 - Directed edges and edge features
 - Limiting assumptions: $\tilde{A} = A + \lambda I_N$