Simplified Graph Neural Networks

Yukuo Cen

GNN Center, Zhipu AI

KEG, Tsinghua University

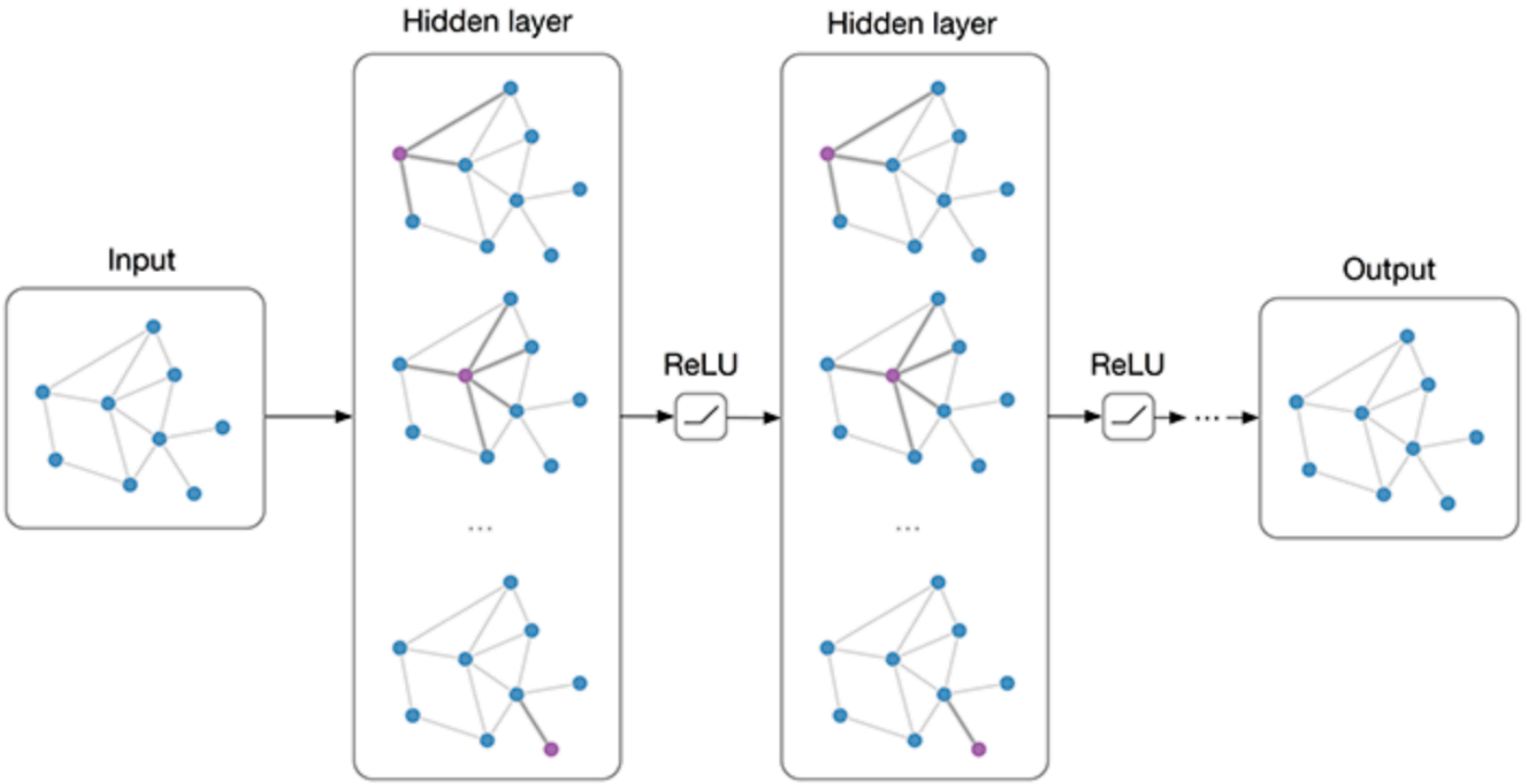
Advisors: Yuxiao Dong, Jie Tang

Course Link: https://cogdl.ai/gnn2022/

CogDL is publicly available at https://github.com/THUDM/cogdl

Review GCNs

- Layer-wise propagation: $f(H^{(l)}, A) = \sigma\left(AH^{(l)}W^{(l)}\right)$

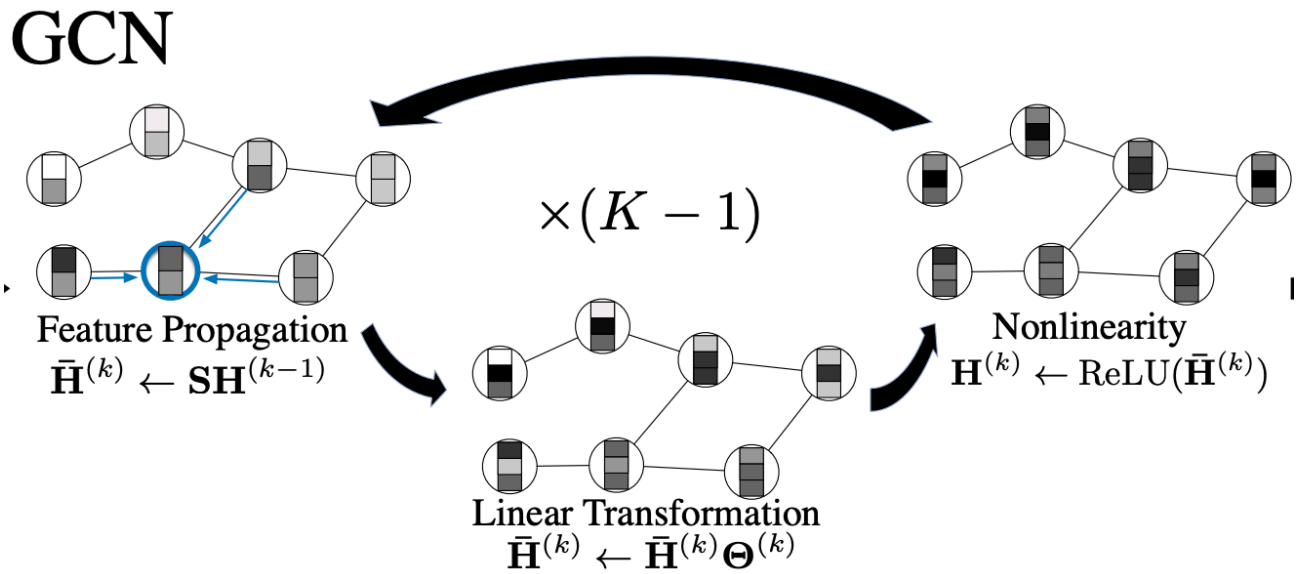


1. Kipf et al. Semi-supervised Classification with Graph Convolutional Networks. ICLR 2017

Review GCNs

- GCN $f(H^{(l)}, A) = \sigma\left(AH^{(l)}W^{(l)}\right)$
  - Feature propagation
  - Linear transformation
  - Nonlinearity

Question: How to train GCNs on large-scale graphs?



SGC vs GCN

- SGC:
  - Removing the nonlinearities between GCN layers
  - More analysis on low-pass filtering

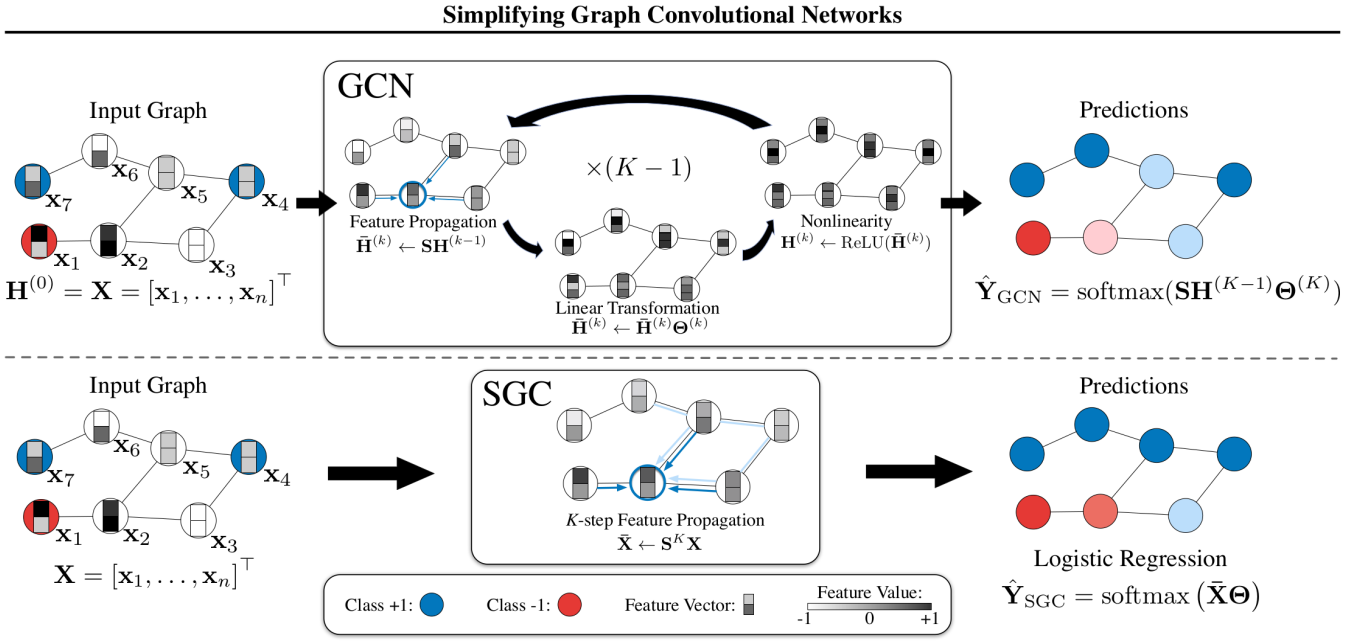


Figure 1. Schematic layout of a GCN v.s. a SGC. *Top row:* The GCN transforms the feature vectors repeatedly throughout K layers and then applies a linear classifier on the final representation. *Bottom row:* the SGC reduces the entire procedure to a simple feature propagation step followed by standard logistic regression.

Wu, Felix, et al. "Simplifying graph convolutional networks." International conference on machine learning. In ICLR 2019.

SGC

- Linearization
  - Hypothesize that the nonlinearity between GCN layers is not critical.
  - Therefore remove the nonlinear transition functions between each layer

$$\hat{\mathbf{Y}} = \mathrm{softmax}(\mathbf{S} \ldots \mathbf{S}\mathbf{S}\mathbf{X}^{(0)}\Theta^{(1)}\Theta^{(2)} \ldots \Theta^{(K)}) \tag{1}$$

$$\hat{\mathbf{Y}}_{\mathrm{SGC}} = \mathrm{softmax}(\mathbf{S}^{K}\mathbf{X}\Theta) \tag{2}$$

Where equation(2) is the simplified version of equation(1), which we refer to as Simple Graph Convolution (SGC)

Review Spectral Graph Convolutions

- Graph Laplacian

$$\mathbf{\Delta} = \mathbf{D} - \mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top \qquad \mathbf{\Delta}_{\text{sym}} = \mathbf{D}^{-1/2}\mathbf{\Delta}\mathbf{D}^{-1/2}$$

- Spectral graph convolution

$$\mathbf{g} * \mathbf{x} = \mathbf{U}\left((\mathbf{U}^\top \mathbf{g}) \odot (\mathbf{U}^\top \mathbf{x})\right) = \mathbf{U}\hat{\mathbf{G}}\mathbf{U}^\top \mathbf{x},$$

- Truncated by k-order polynomials

$$\mathbf{U}\hat{\mathbf{G}}\mathbf{U}^\top \mathbf{x} \approx \sum_{i=0}^{k} \theta_i \mathbf{\Delta}^i \mathbf{x} = \mathbf{U}\left(\sum_{i=0}^{k} \theta_i \mathbf{\Lambda}^i\right)\mathbf{U}^\top \mathbf{x},$$

- Simplifying via k=1

$$\mathbf{g} * \mathbf{x} = \theta(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{x}.$$

Spectral Analysis on SGC

- Low-Pass Filtering on SGC

$$\mathbf{S}_{\text{1-order}} = \mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} = 2\mathbf{I} - \mathbf{\Delta}_{\text{sym}}$$

$$\hat{g}_i = \hat{g}(\lambda_i) = (2 - \lambda_i)^K$$

$$\mathbf{S}_{\text{adj}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \qquad \tilde{\mathbf{S}}_{\text{adj}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$$

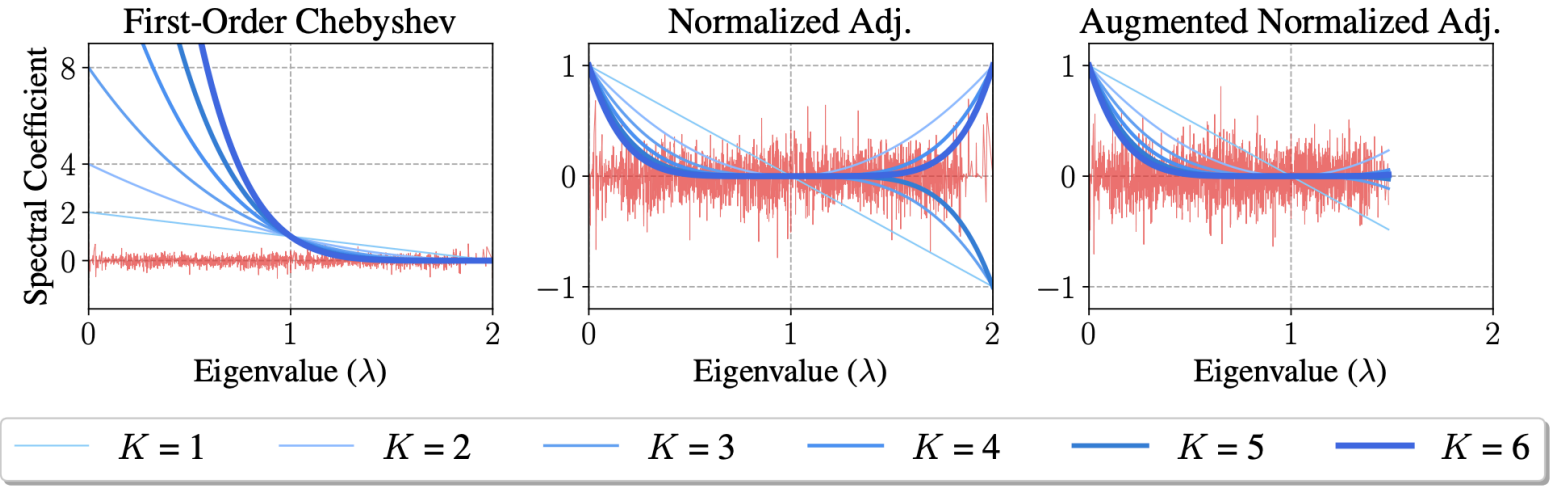


Figure 2. Feature (red) and filters (blue) spectral coefficients for different propagation matrices on Cora dataset (3rd feature).

SGC

- Efficiency

Simplifying Graph Convolutional Networks

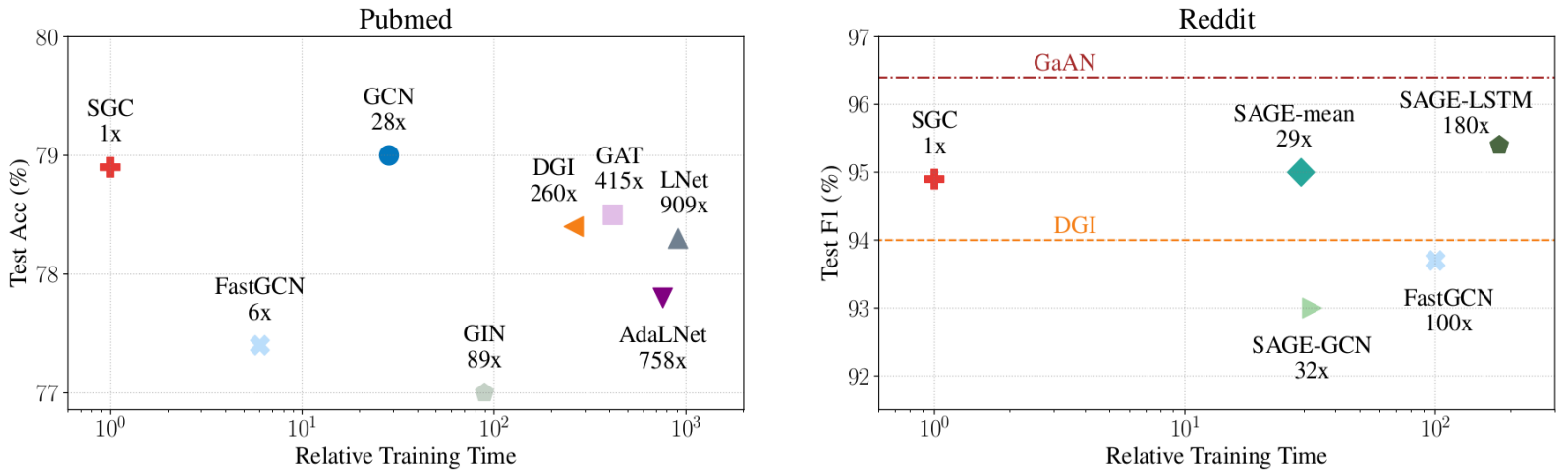


Figure 3. Performance over training time on Pubmed and Reddit. SGC is the fastest while achieving competitive performance. We are not able to benchmark the training time of GaAN and DGI on Reddit because the implementations are not released.

- SGC uses less computational resources and achieves relatively higher performance
- $S^K X$ part is computed in advance to save memory, hence only parameter matrix Θ needs to be learned during training process

SGC Results

- Performance
  - The performance of SGC can match the performance of GCN and state-of-the-art graph networks on citation networks.
  - In particular on Citeseer, SGC is about 1% better than GCN, and we reason this performance boost is caused by SGC having fewer parameters and therefore suffering less from overfitting

Table 2. Test accuracy (%) averaged over 10 runs on citation networks. †We remove the outliers (accuracy $< 75/65/75\%$) when calculating their statistics due to high variance.

	Cora	Citeseer	Pubmed
Numbers from literature:			
GCN	81.5	70.3	79.0
GAT	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3
GLN	81.2 ± 0.1	70.9 ± 0.1	78.9 ± 0.1
AGNN	83.1 ± 0.1	71.7 ± 0.1	79.9 ± 0.1
LNet	79.5 ± 1.8	66.2 ± 1.9	78.3 ± 0.3
AdaLNet	80.4 ± 1.1	68.7 ± 1.0	78.1 ± 0.4
DeepWalk	70.7 ± 0.6	51.4 ± 0.5	76.8 ± 0.6
DGI	82.3 ± 0.6	71.8 ± 0.7	76.8 ± 0.6
Our experiments:			
GCN	81.4 ± 0.4	70.9 ± 0.5	79.0 ± 0.4
GAT	83.3 ± 0.7	72.6 ± 0.6	78.5 ± 0.3
FastGCN	79.8 ± 0.3	68.8 ± 0.6	77.4 ± 0.3
GIN	77.6 ± 1.1	66.1 ± 0.9	77.0 ± 1.2
LNet	$80.2 \pm 3.0^{\dagger}$	67.3 ± 0.5	$78.3 \pm 0.6^{\dagger}$
AdaLNet	$81.9 \pm 1.9^{\dagger}$	$70.6 \pm 0.8^{\dagger}$	$77.8 \pm 0.7^{\dagger}$
DGI	82.5 ± 0.7	71.6 ± 0.7	78.4 ± 0.7
SGC	81.0 ± 0.0	71.9 ± 0.1	78.9 ± 0.0

SGC Results

- Performance

On Reddit, Table 3 shows that SGC outperforms the previous sampling-based GCN variants, SAGE-GCN and FastGCN by more than 1%.

Dataset	# Nodes	# Edges	Train/Dev/Test Nodes
Cora	2,708	5,429	140/500/1,000
Citeseer	3,327	4,732	120/500/1,000
Pubmed	19,717	44,338	60/500/1,000
Reddit	233K	11.6M	152K/24K/55K

Table 3. Test Micro F1 Score (%) averaged over 10 runs on Reddit. Performances of models are cited from their original papers. **OOM:** Out of memory.

Setting	Model	Test F1
Supervised	GaAN	96.4
	SAGE-mean	95.0
	SAGE-LSTM	95.4
	SAGE-GCN	93.0
	FastGCN	93.7
	GCN	**OOM**
Unsupervised	SAGE-mean	89.7
	SAGE-LSTM	90.7
	SAGE-GCN	90.8
	DGI	94.0
No Learning	Random-Init DGI	93.3
	SGC	94.9

More SGC Results

Text classification

Table 4. Test Accuracy (%) on text classification datasets. The numbers are averaged over 10 runs.

Dataset	Model	Test Acc. ↑	Time (seconds) ↓
20NG	GCN	87.9 ± 0.2	1205.1 ± 144.5
	SGC	88.5 ± 0.1	19.06 ± 0.15
R8	GCN	97.0 ± 0.2	129.6 ± 9.9
	SGC	97.2 ± 0.1	1.90 ± 0.03
R52	GCN	93.8 ± 0.2	245.0 ± 13.0
	SGC	94.0 ± 0.2	3.01 ± 0.01
Ohsumed	GCN	68.2 ± 0.4	252.4 ± 14.7
	SGC	68.5 ± 0.3	3.02 ± 0.02
MR	GCN	76.3 ± 0.3	16.1 ± 0.4
	SGC	75.9 ± 0.3	4.00 ± 0.04

User geolocation

Table 5. Test accuracy (%) within 161 miles on semi-supervised user geolocation. The numbers are averaged over 5 runs.

Dataset	Model	Acc.@161↑	Time ↓
GEOTEXT	GCN+H	60.6 ± 0.2	153.0s
	SGC	61.1 ± 0.1	5.6s
TWITTER-US	GCN+H	61.9 ± 0.2	9h 54m
	SGC	62.5 ± 0.1	4h 33m
TWITTER-WORLD	GCN+H	53.6 ± 0.2	2d 05h 17m
	SGC	54.1 ± 0.2	22h 53m

SIGN

- Motivation
  - Scale to very large graphs such as Twitter social networks
  - Combine graph convolutional filters of different types

- Overall results
  - Performs SOTA results on ogbn-papers100M (110 millions node + 1.5 billion edges)

Frasca, Fabrizio, et al. "Sign: Scalable inception graph neural networks." arXiv preprint arXiv:2004.11198 (2020).

SIGN

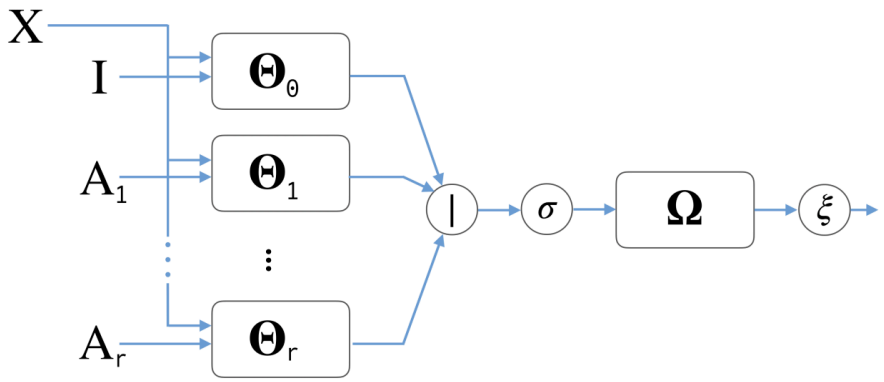


Figure 1: The *SIGN* architecture for r generic graph filtering operators. $\mathbf{\Theta}_k$ represents the k-th dense layer transforming node-wise features downstream the application of operator k, $|$ is the concatenation operation and $\mathbf{\Omega}$ refers to the dense layer used to compute final predictions.

- $I, A_1, \ldots, A_r$ denote different operators
- Θ denotes learnable parameters, used to perform linear transformation

SIGN

Overall formulation of the SIGN

$$\begin{aligned}\mathbf{Z} &= \sigma\left(\left[\mathbf{X}\boldsymbol{\Theta}_0, \mathbf{A}_1\mathbf{X}\boldsymbol{\Theta}_1, \ldots, \mathbf{A}_r\mathbf{X}\boldsymbol{\Theta}_r\right]\right) \\ \mathbf{Y} &= \xi\left(\mathbf{Z}\boldsymbol{\Omega}\right),\end{aligned}$$

- Inspired by CNN inception-like module, different operators $I, A_1, \ldots, A_r$ can capture different graph features
- $A_1X \ldots A_rX$ can be pre-computed, which tremendously reduce the computational complexity

Generailization to GCNs

Table 2: By appropriate configuration, *SIGN* inception layer is able to replicate some popular graph convolutional layers. α represents the learnable parameter of a PReLU activation.

	$\mathbf{B}_1, \ldots, \mathbf{B}_r$	α	$\mathbf{\Theta}_0, \ldots, \mathbf{\Theta}_r$	$\mathbf{\Omega}$
ChebNet [15]	$\mathbf{\Delta}, \ldots, \mathbf{\Delta}^r$	1	$\mathbf{\Theta}_0, \ldots, \mathbf{\Theta}_r$	$[\mathbf{I}, \ldots, \mathbf{I}]^\top$
GCN [34]	$r = 1,\ \tilde{\mathbf{A}}$	1	$\mathbf{0}, \mathbf{\Theta}$	$[\mathbf{0}, \mathbf{I}]^\top$
S-GCN [59]	$r = 1,\ \tilde{\mathbf{A}}^L$	1	$\mathbf{0}, \mathbf{\Theta}$	$[\mathbf{0}, \mathbf{I}]^\top$

- In particular, by setting the σ non-linearity to PReLU , ChebNet, GCN, and S-GCN can be automatically learnt if suitable diffusion operator B and activation ξ are used.

Datasets used by SIGN

- Seven datasets with different scales

Table 3: Summary of (s)ingle and (m)ulti-label dataset statistics. `Wikipedia` is used, with random features, for timing purposes only.

	n	$\lvert\mathcal{E}\rvert$	*Avg. Deg.*	d	Classes	Train / Val / Test
`Wikipedia`	12,150,976	378,142,420	62	100	2(s)	100% / — / 100%
`ogbn-papers100M`	111,059,956	1,615,685,872	30	128	172(s)	78% / 8% / 14%
`ogbn-products`	2,449,029	61,859,140	51	100	47(s)	10% / 2% / 88%
`Reddit`	232,965	11,606,919	50	602	41(s)	66% / 10% / 24%
`Yelp`	716,847	6,977,410	10	300	100(m)	75% / 10% / 15%
`Flickr`	89,250	899,756	10	500	7(s)	50% / 25% / 25%
`PPI`	14,755	225,270	15	50	121(m)	66% / 12% / 22%

SIGN results

- Performance of SIGN and other scalable methods on the inductive node classification task.

Table 5: Micro-averaged F1 scores. For *SIGN*, we show the best performing configurations. The top three performance scores are highlighted as: **First**, **Second**, **Third**.

	Reddit	Flickr	PPI	Yelp
GCN [34]	0.933±0.000	0.492±0.003	0.515±0.006	0.378±0.001
FastGCN [11]	0.924±0.001	**0.504±0.001**	0.513±0.032	0.265±0.053
Stochastic-GCN [12]	**0.964±0.001**	0.482±0.003	**0.963±0.010**	**0.640±0.002**
AS-GCN [30]	0.958±0.001	**0.504±0.002**	0.687±0.012	—
GraphSAGE [24]	0.953±0.001	0.501±0.013	0.637±0.006	**0.634±0.006**
ClusterGCN [13]	0.954±0.001	0.481±0.005	0.875±0.004	0.609±0.005
GraphSAINT [64]	**0.966±0.001**	**0.511±0.001**	**0.981±0.004**	**0.653±0.003**
S-GCN [59]	0.949±0.000	0.502±0.001	0.892±0.015	0.358±0.006
SIGN	**0.968±0.000**	**0.514±0.001**	**0.970±0.003**	0.631±0.003
(p, s, t)	$(4, 2, 0)$	$(4, 0, 1)$	$(2, 0, 1)$	$(2, 0, 1)$

SIGN results on ogbn datasets

Table 6: Performance on `ogbn-products`. *SIGN(p,s,t)* refers to a configuration using p, s, and t powers of simple, PPR-based, and triangle-based adjacency matrices. The top three performance scores are highlighted as: **First**, **Second**, **Third**.

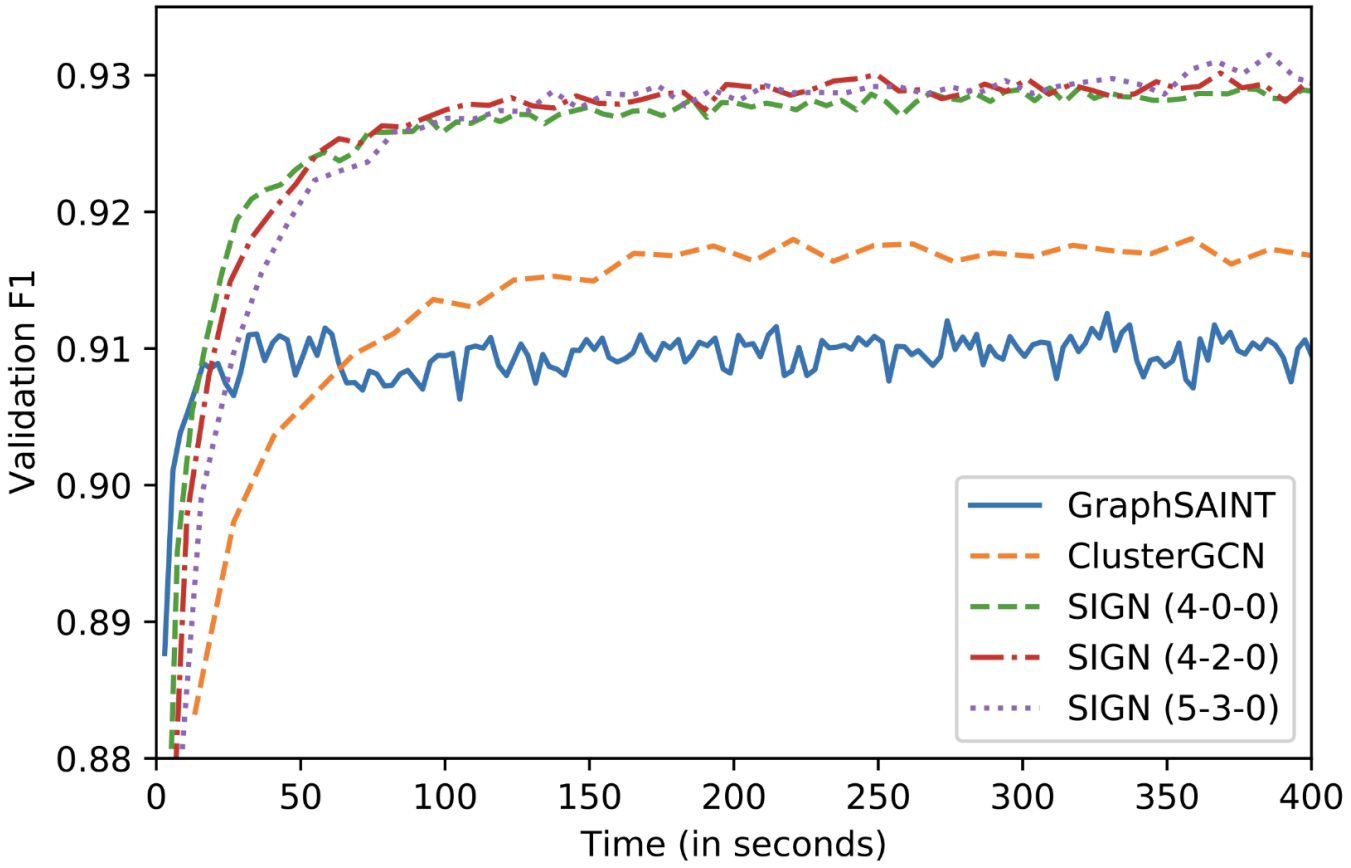
	Training	*Validation*	*Test*
MLP	84.03±0.93	75.54±0.14	61.06±0.08
Node2Vec	93.39±0.10	90.32±0.06	72.49±0.10
S-GCN(L=5)	92.54±0.09	91.38±0.07	74.87±0.25
ClusterGCN	93.75±0.13	92.12±0.09	**78.97±0.33**
GraphSAINT	92.71±0.14	91.62±0.08	**79.08±0.24**
SIGN(3,0,0)	96.21±0.31	**92.99±0.05**	76.52±0.14
SIGN(3,0,1)	**96.46±0.29**	92.93±0.04	75.73±0.20
SIGN(3,3,0)	**96.87±0.23**	**93.02±0.04**	77.13±0.10
SIGN(5,0,0)	95.99±0.69	92.98±0.18	76.83±0.39
SIGN(5,3,0)	**96.92±0.46**	**93.10±0.08**	**77.60±0.13**

Table 8: Results on `ogbn-papers100M`, the largest public graph dataset with over 110 million nodes. *SIGN(p,d,f)* refers to a configuration using p, d, and f powers of simple undirected, directed and directed-transposed adjacency matrices. The top three performance scores are highlighted as: **First**, **Second**, **Third**.

	Training	*Validation*	*Test*
MLP	54.84±0.43	49.60±0.29	47.24±0.31
Node2Vec	—	55.60±0.23	58.07±0.28
S-GCN(L=3)	67.54±0.43	66.48±0.20	63.29±0.19
SIGN(3,0,0)	**70.18±0.37**	**67.57±0.14**	**64.28±0.14**
SIGN(3,1,1)	**72.24±0.32**	**67.76±0.09**	**64.39±0.18**
SIGN(3,3,3)	**73.94±0.72**	**68.6±0.04**	**65.11±0.14**

Efficiency of SIGN

- Up to two orders of magnitude faster than ClusterGCN and GraphSAINT at inference time, while also being significantly faster at training time



SAGN

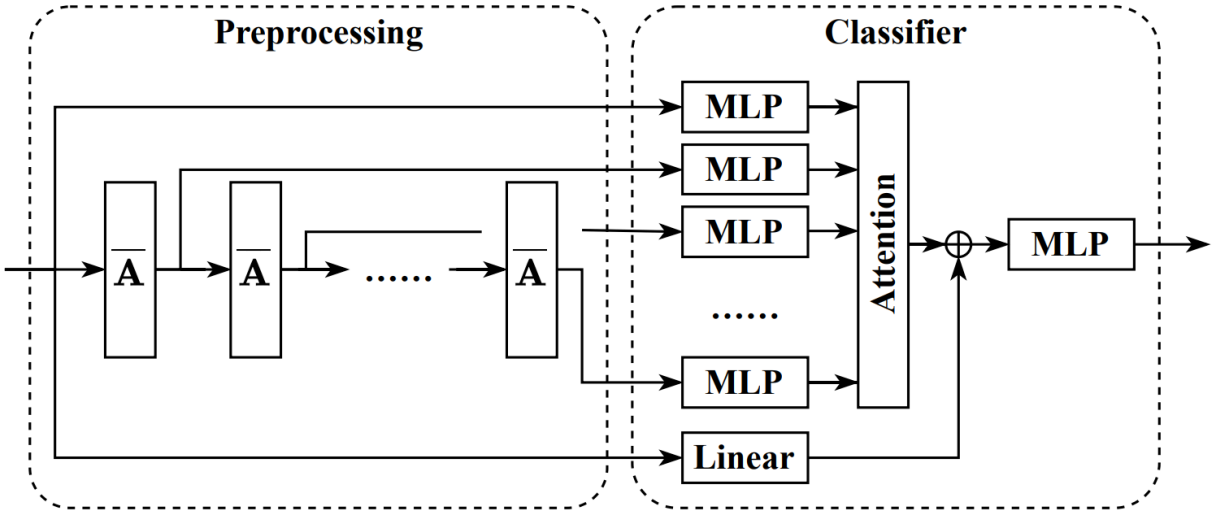


Figure 2: Architecture of SAGN. The multi-hop encoders and post encoder are depicted in form of MLP. The bottom "linear" refers to the residual linear layer. The symbol ⊕ represents the operation of summing integrated representation and residual term.

- SAGN uses attention mechanism to replace the simple concatenation in SIGN, which enables the model to capture more important information

Sun, Chuxiong, Hongming Gu, and Jie Hu. "Scalable and adaptive graph neural networks with self-label-enhanced training." arXiv preprint arXiv:2104.09376 (2021).

Datasets used by SAGN

- four inductive datasets + three ogbn datasets

Table 1: Summary of datasets. For classes, there are single (s) label classification task and multiple (m) label classification task.

Dataset	Nodes	Edges	Classes	Train/Val/Test	Setting
Reddit	232,965	11,606,919	41(s)	66% / 10% / 24%	Inductive
Yelp	716,847	6,977,410	100(m)	75% / 10% / 15%	Inductive
Flickr	89,250	899,756	7(s)	50% / 25% / 25%	Inductive
PPI	14,755	225,270	121(m)	66% / 12% / 22%	Inductive
ogbn-products	2,449,029	61,859,140	47(s)	10% / 2% / 88%	Transductive
ogbn-papers100M	111,059,956	1,615,685,872	172(s)	78% / 8% / 14%	Transductive
ogbn-mag	1,939,743	21,111,007	349(s)	85% / 9% / 6%	Transductive

SAGN Results on inductive datasets

Table 2: Results on inductive datasets. The means and standard deviations of micro-F1 score over 10 runs are reported. The best results are highlighted in **bold** fonts.

Method	Reddit	Flickr	PPI	Yelp
GCN [1]	93.3±0.0%	49.2±0.3%	51.5±0.6%	37.8±0.1%
FastGCN [16]	92.4±0.1%	50.4±0.1%	51.3±3.2%	26.5±5.3%
Stochastic-GCN [17]	96.4±0.1%	48.2±0.3%	96.3±1.0%	64.0±0.2%
AS-GCN [15]	95.8±0.1%	50.4±0.2%	68.7±1.2%	—
GraphSAGE [13]	95.3±0.1%	50.1±1.3%	63.7±0.6%	63.4±0.6%
ClusterGCN [19]	95.4±0.1%	48.1±0.5%	87.5±0.4%	60.9±0.5%
GraphSaint [20]	96.6±0.1%	51.1±0.1%	**98.1±0.4%**	**65.3±0.3%**
SGC [21]	94.9±0.0%	50.2±0.1%	89.2±1.5%	35.8±0.6%
SIGN [22]	96.8±0.0%	51.4±0.1%	97.0±0.3%	63.1±0.3%
SAGN	96.9±0.0%	51.4±1.2%	97.9±0.1%	**65.3±0.1%**
SAGN+1-SLE	**97.1±0.0%**	54.3±0.5%	98.0±0.1%	**65.3±0.1%**
SAGN+2-SLE	**97.1±0.0%**	**54.6±0.4%**	98.0±0.1%	**65.3±0.1%**

SAGN Results on ogbn datasets

Table 3: Results on transductive datasets. The means and standard deviations of validation and test accuracies over 10 runs are reported. The best results are highlighted in **bold** fonts.

Method	ogbn-products		ogbn-papers100M	
	Validation	Test	Validation	Test
MLP	75.54±0.14%	61.06±0.08%	49.60±0.29%	47.24±0.31%
Node2Vec [46, 49]	90.32±0.06%	72.49±0.10%	58.07±0.28%	55.60±0.23%
GCN [1, 46]	92.00±0.03%	75.64±0.21%	—	—
GraphSAGE [13, 46]	92.24±0.07%	78.50±0.14%	—	—
NeighborSampling [13, 46]	91.70±0.09%	78.70±0.36%	—	—
ClusterGCN [19]	92.12±0.09%	78.97±0.33%	—	—
GraphSaint [20]	—	80.27±0.26%	—	—
SIGN [22, 54]	92.86±0.02%	80.52±0.13%	69.32±0.06%	65.68±0.06%
SAGN	93.09±0.04%	81.20±0.07%	70.34±0.99%	66.75±0.84%
SAGN+1-SE	92.54±0.04%	82.23±0.09%	70.79±0.12%	67.21±0.12%
SAGN+2-SE	92.33±0.03%	82.50±0.13%	70.89±0.12%	67.30±0.15%
UniMP	93.08±0.17%	82.56±0.31%	71.72±0.05%	67.36±0.10%
MLP+C&S	91.47±0.09%	84.18±0.07%	—	—
SAGN+0-SLE	93.27±0.04%	83.29±0.18%	71.06±0.08%	67.55±0.15%
SAGN+1-SLE	93.06±0.07%	84.18±0.14%	71.23±0.10%	67.77±0.15%
SAGN+2-SLE	92.87±0.03%	**84.28±0.14%**	71.31±0.10%	**68.00±0.15%**

SAGN Results on ogbn-mag

Table 7: Results on ogbn-mag. Validation and test accuracies with means and standard deviations (%) are reported. The best results in homogeneous and heterogeneous settings are highlighted in **bold** fonts.

Method	Validation	Test
MLP	26.26±0.16%	26.92±0.26%
GCN	29.53±0.22%	30.43±0.25%
GraphSAGE	30.70±0.19%	31.53±0.15%
SIGN	40.68±0.10%	40.46±0.12%
SAGN+0-SLE	42.13±0.51%	40.51±0.83%
SAGN+1-SLE	43.85±0.49%	42.18±0.61%
SAGN+2-SLE	44.27±0.30%	**42.75±0.38%**
MetaPath2Vec	35.06±0.17%	35.44±0.36%
R-GCN	40.84±0.41%	39.77±0.46%
NeighborSampling	47.61±0.68%	46.78±0.67%
ClusterGCN	38.40±0.31%	37.32±0.37%
GraphSaint	48.37±0.26%	47.51±0.22%
NARS	53.72±0.09%	52.40±0.16%
SAGN+TransE+0-SLE	49.42±0.17%	47.95±0.25%
SAGN+TransE+1-SLE	51.23±0.16%	49.70±0.17%
SAGN+TransE+2-SLE	51.80±0.15%	50.29±0.16%
NARS_SAGN+0-SLE	54.12±0.15%	52.32±0.25%
NARS_SAGN+1-SLE	55.52±0.16%	53.95±0.14%
NARS_SAGN+2-SLE	55.91±0.17%	**54.40±0.15%**

Runtime and Efficiency of SAGN

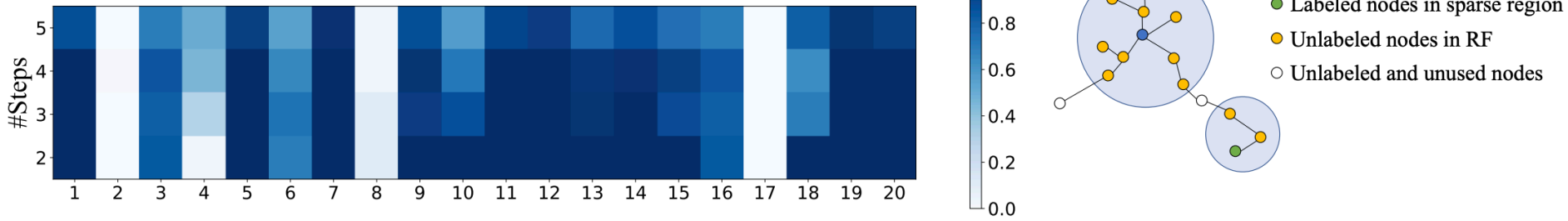
Table 5: Runtime experiments on ogbn-products. The runtimes (seconds) with means and standard deviations, memory costs (Mb) and parameter numbers over 10 runs are reported. SIGN+SLE is not reported due to out-of-memory (OOM) error.

Method	Training	Inference	Memory	Parameters
MLP	0.67±0.04s	7.78±0.25s	10832Mb	669743
SIGN	1.27±0.10s	8.68±0.64s	15558Mb	3489847
SAGN	1.12±0.01s	8.43±0.22s	13948Mb	2233391
MLP+0-SLE	0.76±0.03s	7.85±0.30s	11958Mb	1181278
SAGN+0-SLE	1.47±0.01s	8.35±0.33s	14894Mb	2810462

- SIGN and SAGN have very close runtime in both training and inference process, as expected in complexity analysis.
- SAGN has 10% smaller memory cost and 36% less parameters compared with SIGN

GAMLP

- Observation:



(a) Inconsistent optimal steps.

(b) Inconsistent RF expansion speed.

Figure 1: (Left) Test accuracy of SGC on 20 randomly sampled nodes of Citeseer. The X-axis is the node id, and Y-axis is the propagation steps. The color from white to blue represents the ratio of being predicted correctly in 50 different runs. (Right) The node in the dense region has a larger RF within two iterations of propagation.

- Different nodes may need different propagation steps!
- Use attention!

Zhang, Wentao, et al. “Graph attention multi-layer perceptron.” *In KDD 2022.*

GAMLP

- Combine feature and label information:
  - Node-wise feature propagation $\mathbf{X}^{(k)} \leftarrow \hat{\mathbf{A}}\mathbf{X}^{(k-1)}, \ \forall k = 1, \ldots, K$
  - Node-wise label propagation $\mathbf{Y}^{(l)} \leftarrow \hat{\mathbf{A}}\mathbf{Y}^{(l-1)}, \ \forall l = 1, \ldots, L$

$$\mathbf{H}_{\mathbf{X}} = \sum_{k=0}^{K} \mathbf{W}_k \mathbf{X}^{(k)} \qquad \mathbf{H}_{\mathbf{Y}} = \sum_{l=0}^{L} \hat{\mathbf{W}}_l \hat{\mathbf{Y}}^{(l)}$$

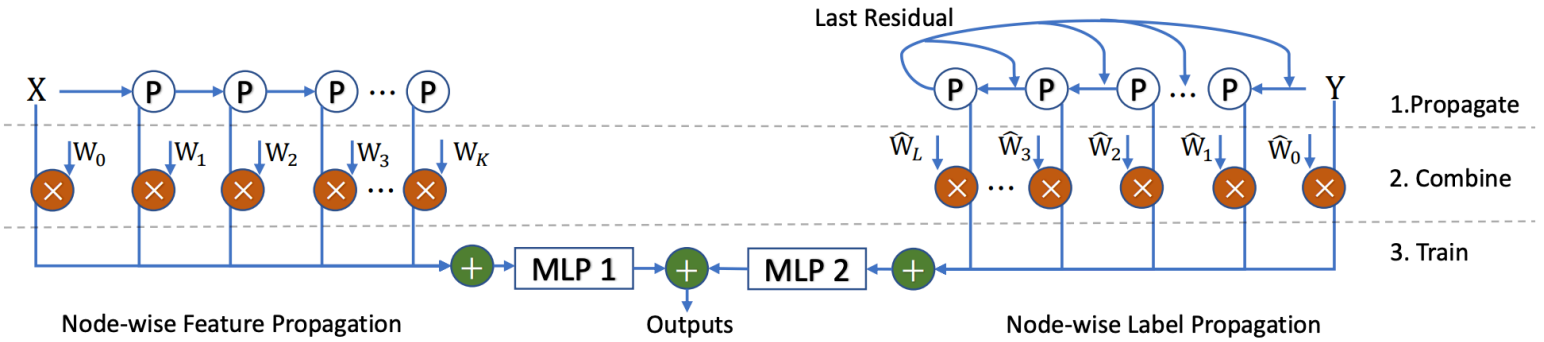


Figure 2: Overview of the proposed GAMLP, including (1) feature and label propagation, (2) combine the propagated features and labels with RF attention, and (3) MLP training. Note that both the feature and label propagation can be pre-processed.

Node-adaptive Attention Mechanisms

- JK Attention

$$\widetilde{\mathbf{X}}_i^{(l)} = \mathbf{X}_i^{(l)} \parallel \mathbf{E}_i, \quad \widetilde{w}_i(l) = \delta(\widetilde{\mathbf{X}}_i^{(l)} \cdot s), \quad w_i(l) = e^{\widetilde{w}_i(l)} / \sum_{k=0}^{K} e^{\widetilde{w}_i(k)}.$$

$$\mathbf{E}_i = \mathrm{MLP}(\mathbf{X}_i^{(1)} \parallel \mathbf{X}_i^{(2)} \parallel \ldots \parallel \mathbf{X}_i^{(K)})$$

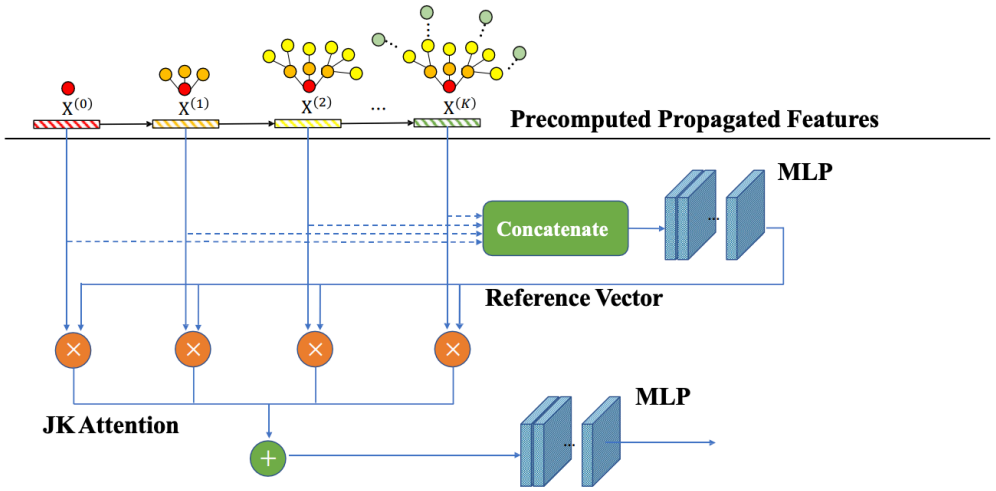


Figure 5: The architecture of GAMLP with JK Attention.

Node-adaptive Attention Mechanisms

- Recursive Attention

$$\widetilde{\mathbf{X}}_i^{(l)} = \mathbf{X}_i^{(l)} \,\|\, \sum_{k=0}^{l-1} w_i(k)\mathbf{X}_i^{(k)}, \quad w_i(k) = e^{\widetilde{w}_i(k)} / \sum_{j=0}^{l-1} e^{\widetilde{w}_i(j)},$$

$$\widetilde{w}_i(l) = \delta(\widetilde{\mathbf{X}}_i^{(l)} \cdot s)$$

Using the recursive method, when calculating the weight assigned to the node features propagated after L steps, the fused node features of the previous (L-1) steps are considered as the reference vector

Complexity Analysis

Type	Method	Pre-processing	Training	Memory
Sampling	GraphSAGE	-	$O(k^K nf^2)$	$O(bk^K f + Kf^2)$
	FastGCN	-	$O(kKnf^2)$	$O(bkKf + Kf^2)$
	Cluster-GCN	$O(m)$	$O(Pmf + Pnf^2)$	$O(bKf + Kf^2)$
Graph-wise propagation	SGC	$O(Kmf)$	$O(nf^2)$	$O(bf + f^2)$
Layer-wise propagation	SIGN	$O(Kmf)$	$O(Pnf^2)$	$O(bLf + Pf^2)$
	S^2GC	$O(Kmf)$	$O(nf^2)$	$O(bf + f^2)$
	GBP	$O(Knf + K\frac{\sqrt{m \lg n}}{\varepsilon})$	$O(Pnf^2)$	$O(bf + Pf^2)$
Node-wise propagation	GAMLP	$O(Kmf + Lmc)$	$O(Pnf^2 + Qnc^2)$	$O(bf + Pf^2 + Qc^2)$

n, m, c, and f are the number of nodes, edges, classes, and feature dimensions, respectively. b is the batch size, and k refers to the number of sampled nodes. K and L corresponds to the number of times we aggregate features and labels. Besides, P and Q are the number of layers in MLP classifiers trained with features and labels.

GAMLP Results

- Experiment on Small datasets

Table 2: Transductive performance on citation networks.

Methods	Cora	Citeseer	PubMed
GCN	81.8±0.5	70.8±0.5	79.3±0.7
GAT	83.0±0.7	72.5±0.7	79.0±0.3
JK-Net	81.8±0.5	70.7±0.7	78.8±0.7
ResGCN	82.2±0.6	70.8±0.7	78.3±0.6
APPNP	83.3±0.5	71.8±0.5	80.1±0.2
AP-GCN	83.4±0.3	71.3±0.5	79.7±0.3
SGC	81.0±0.2	71.3±0.5	78.9±0.5
SIGN	82.1±0.3	72.4±0.8	79.5±0.5
S^2GC	82.7±0.3	73.0±0.2	79.9±0.3
GBP	83.9±0.7	72.9±0.5	80.6±0.4
GAMLP(JK)	**84.3±0.8**	**74.6±0.4**	80.7±0.4
GAMLP(R)	83.9±0.6	73.9±0.6	**80.8±0.5**

Table 3: Transductive performance on the co-authorship and co-purchase graphs.

Methods	Amazon Computer	Amazon Photo	Coauthor CS	Coauthor Physics
GCN	82.4±0.4	91.2±0.6	90.7±0.2	92.7±1.1
GAT	80.1±0.6	90.8±1.0	87.4±0.2	90.2±1.4
JK-Net	82.0±0.6	91.9±0.7	89.5±0.6	92.5±0.4
ResGCN	81.1±0.7	91.3±0.9	87.9±0.6	92.2±1.5
APPNP	81.7±0.3	91.4±0.3	92.1±0.4	92.8±0.9
AP-GCN	83.7±0.6	92.1±0.3	91.6±0.7	93.1±0.9
SGC	82.2±0.9	91.6±0.7	90.3±0.5	91.7±1.1
SIGN	83.1±0.8	91.7±0.7	91.9±0.3	92.8±0.8
S^2GC	83.1±0.7	91.6±0.6	91.6±0.6	93.1±0.8
GBP	83.5±0.8	92.1±0.8	92.3±0.4	93.3±0.7
GAMLP(JK)	**84.5±0.7**	**92.8±0.7**	92.6±0.5	**93.6±1.0**
GAMLP(R)	84.2±0.5	92.6±0.8	**92.8±0.7**	93.2±1.0

GAMLP Results

- Experiments on ogbn datasets

Table 4: Performance comparison on ogbn-products.

Methods	Val Accuracy	Test Accuracy
GCN	92.00±0.03	75.64±0.21
SGC	92.13±0.02	75.87±0.14
GraphSAGE	92.24±0.07	78.50±0.14
GraphSAINT	92.52±0.13	80.27±0.26
GBP	92.82±0.10	80.48±0.05
SIGN	92.99±0.04	80.52±0.16
DeeperGCN	92.38±0.09	80.98±0.20
UniMP	93.08±0.17	82.56±0.31
SAGN	93.09±0.04	81.20±0.07
SAGN+0-SLE	93.27±0.04	83.29±0.18
GAMLP(JK)	93.19±0.03	83.54±0.25
GAMLP(R)	93.11±0.05	**83.59±0.09**

Table 5: Performance comparison on ogbn-papers100M.

Methods	Val Accuracy	Test Accuracy
SGC	66.48±0.20	63.29±0.19
SIGN	69.32±0.06	65.68±0.06
SIGN-XL	69.84±0.06	66.06±0.19
SAGN	70.34±0.99	66.75±0.84
SAGN+0-SLE	71.06±0.08	67.55±0.15
GAMLP(JK)	71.92±0.04	**68.07±0.10**
GAMLP(R)	71.21±0.03	67.46±0.02

GAMLP Results

- Experiments on inductive datasets

Table 6: Performance comparison on three inductive datasets.

Methods	PPI	Flickr	Reddit
SGC	65.7±0.01	50.2±0.12	94.9±0.00
GraphSAGE	61.2±0.05	50.1±0.13	95.4±0.01
Cluster-GCN	99.2±0.04	48.1±0.05	95.7±0.00
GraphSAINT	99.4±0.03	51.1±0.10	96.6±0.01
GAMLP(JK)	**99.82±0.01**	**54.12±0.01**	**97.04±0.01**
GAMLP(R)	99.66±0.01	53.12±0.00	96.62±0.01

GAMLP Results

- Performance on Sparse Graphs
  - Edge sparsity or label sparsity

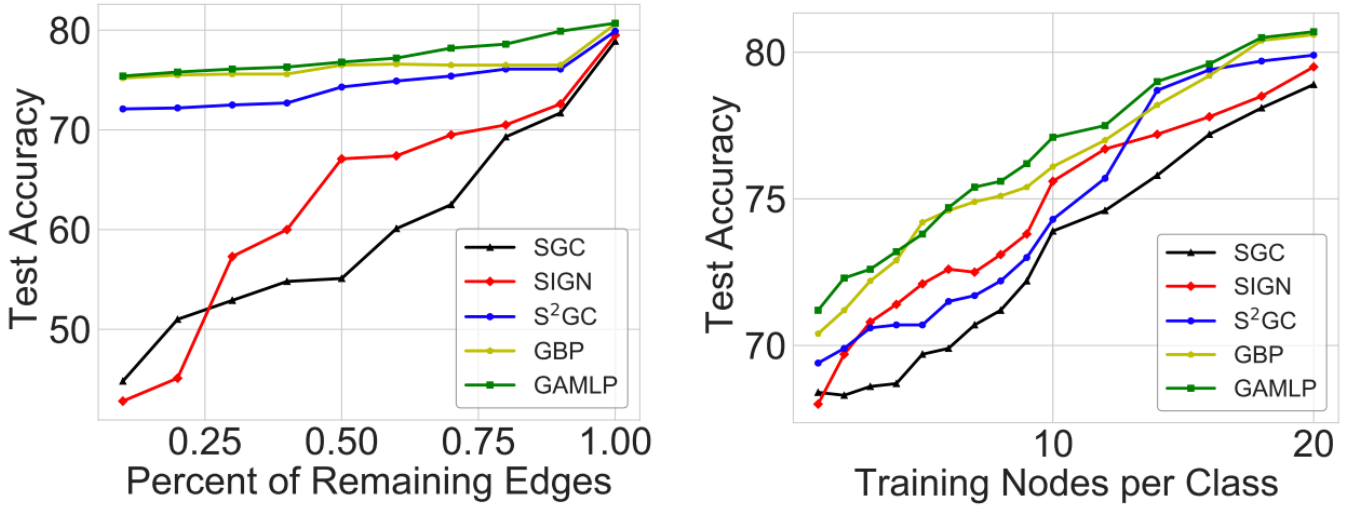


Figure 3: Test accuracy on PubMed dataset under different levels of label and edge sparsity.

Applications on Tencent

Table 10: Efficiency and accuracy comparison on the Tencent video classification.

	SGC	S^2GC	GBP	SIGN	GAMLP(R)	GAMLP (JK)	GCN	APPNP	AP-GCN	JK-Net	ResGCN	GAT
Training Time	1.0	1.2	1.3	3.2	6.1	7.4	33.1	77.8	112.3	112.8	132.3	372.4
Test Accuracy	45.2±0.3	46.6±0.6	46.9±0.7	46.3±0.5	47.8±0.4	**48.1**±0.6	45.9±0.4	46.7±0.6	46.9±0.7	47.2±0.3	45.8±0.5	46.8±0.7

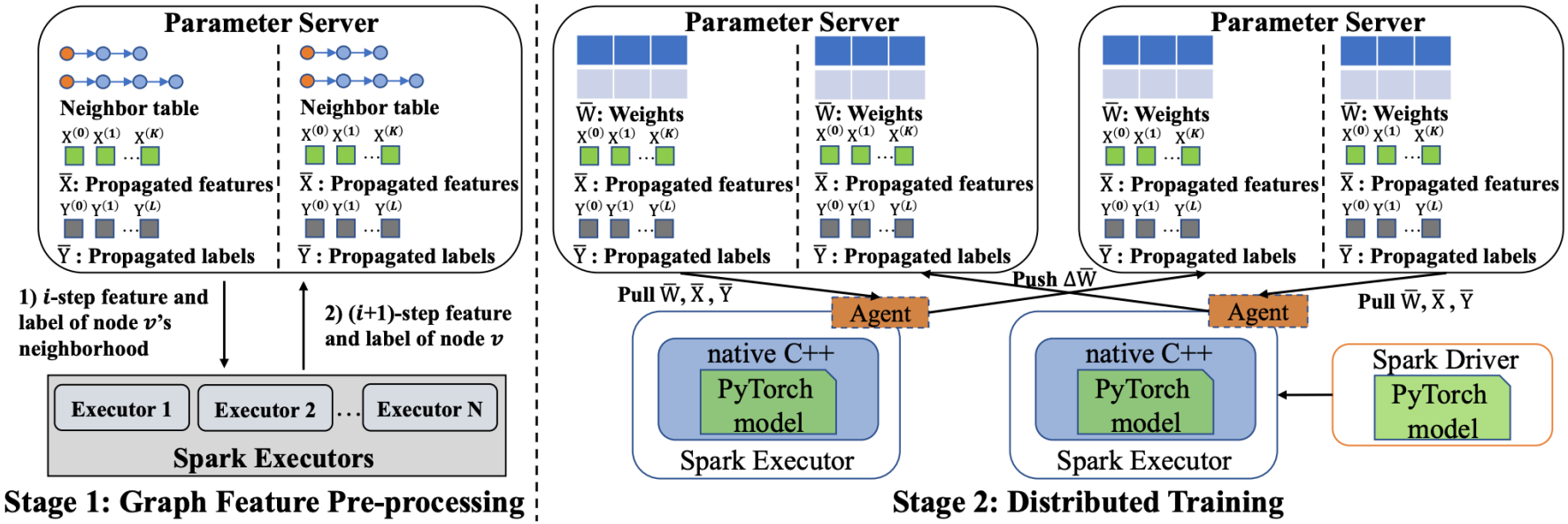


Figure 4: An overview of GAMLP deployed in Tencent.

Summary

- SGC
  - Remove the linearity (scale to large datasets)
  - Low-pass filtering analysis
- SIGN
  - Combine filters of different types and sizes
- SAGN + GAMLP
  - Learnable attention weights between multiple hops

Homework 4: SIGN Implementation

- Experiments on SIGN:
  - Due by 7th Aug.
  - Implement SIGN model based on CogDL
  - Test SIGN model on the cora dataset
  - Give the analysis of the results
- Find the homework material from the course website: https://cogdl.ai/gnn2022/

- Bonus: summarize these simplified GNNs on: https://discuss.cogdl.ai/t/topic/67.

Thank you !

Collaborators:

Zhenyu Hou, Yuxiao Dong, Jie Tang, et al. (**THU**)

Qingfei Zhao, Xinije Zhang, Peng Zhang, et al. (**Zhipu AI**)

Hongxiao Yang, Chang Zhou, et al. (**Alibaba**)

Yang Yang (**ZJU**)

Yukuo Cen, KEG, Tsinghua U.

https://github.com/THUDM/cogdl

Online Discussion Forum https://discuss.cogdl.ai/