NeurIPS | 2023

Graph Convolutional Kernel Machine versus Graph Convolutional Networks

Zhihao Wu[1], Zhao Zhang[2], Jicong Fan[1,3*]

[1]Shenzhen Research Institute of Big Data, Shenzhen, China

[2]Hefei University of Technology, Hefei, China

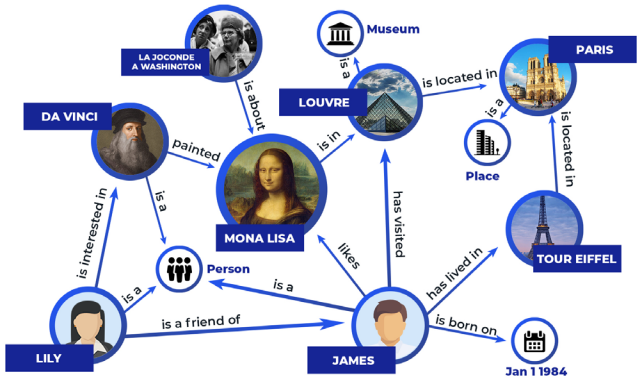
[3]The Chinese University of Hong Kong, Shenzhen, China

{zhihaowu1999,cszzhang}@gmail.com, fanjicong@cuhk.edu.cn

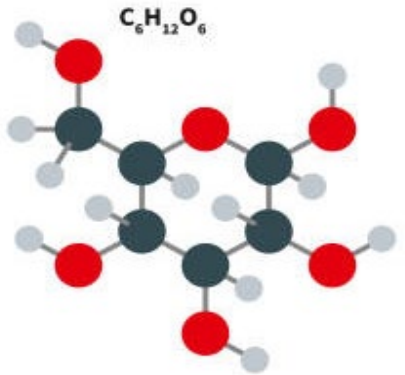
https://github.com/ZhihaoWu99/GCKM

Background

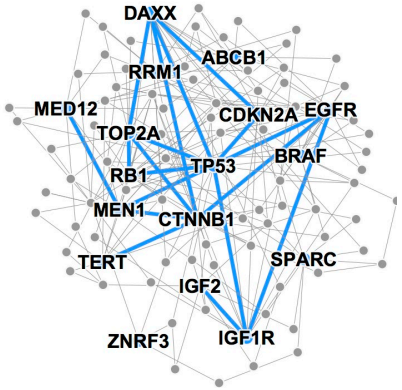
Social network



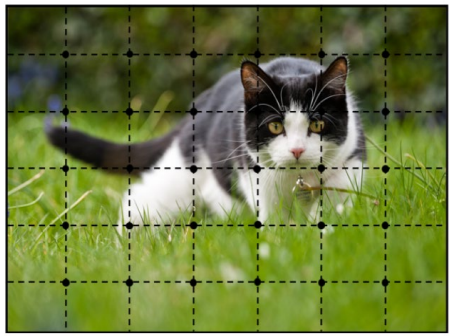
Knowledge graph



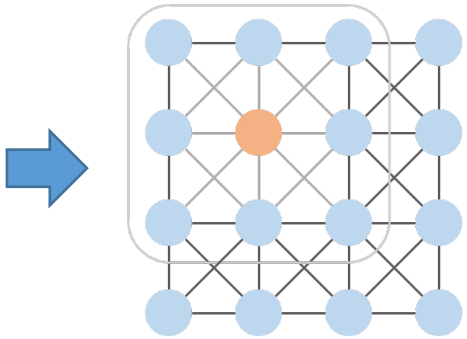
Molecule



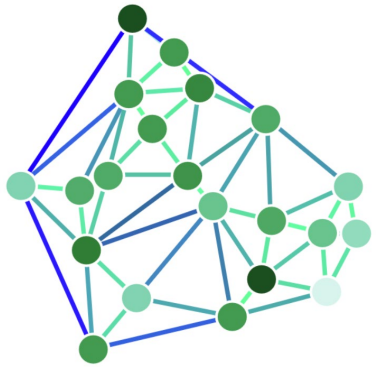
Disease pathway



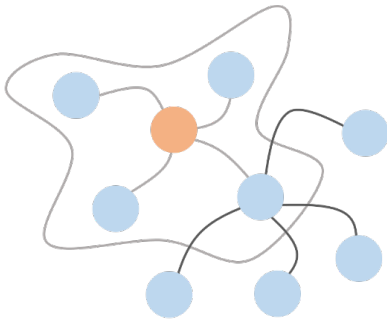
Images



Convolution

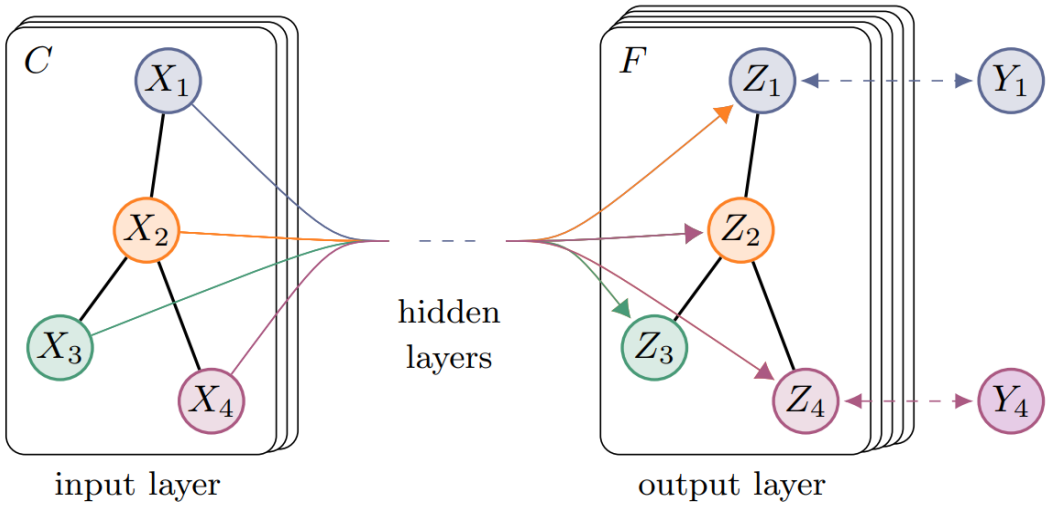


Network

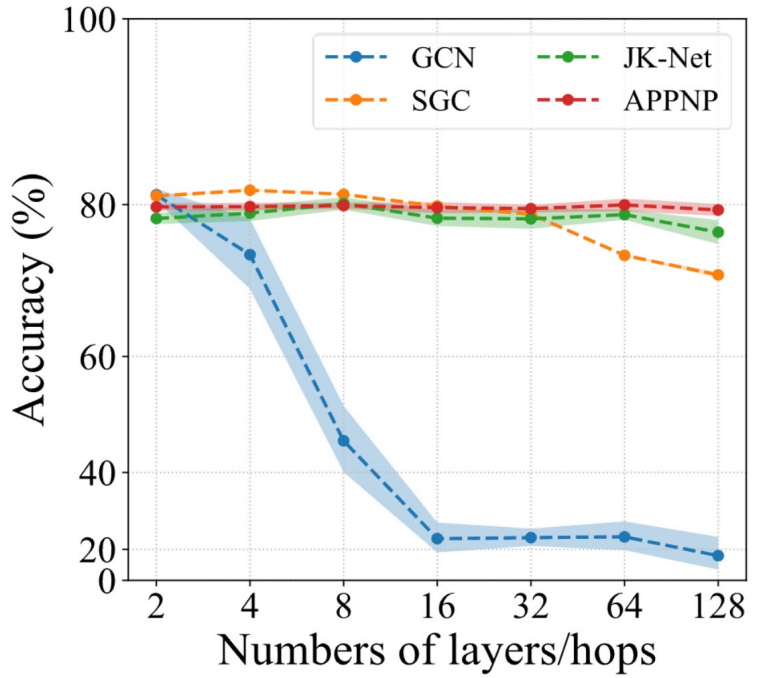


Graph Convolution

Motivation



$$\bar{\mathbf{H}}^{(l+1)} = \text{AGGREGATE}\left(\{\mathcal{G}; \mathbf{H}^{(l)}\}\right) = \hat{\mathbf{A}}\mathbf{H}^{(l)},$$
$$\mathbf{H}^{(l+1)} = \text{TRANSFORM}\left(\bar{\mathbf{H}}^{(l+1)}\right) = \sigma\left(\bar{\mathbf{H}}^{(l+1)}\mathbf{W}^{(l)}\right),$$

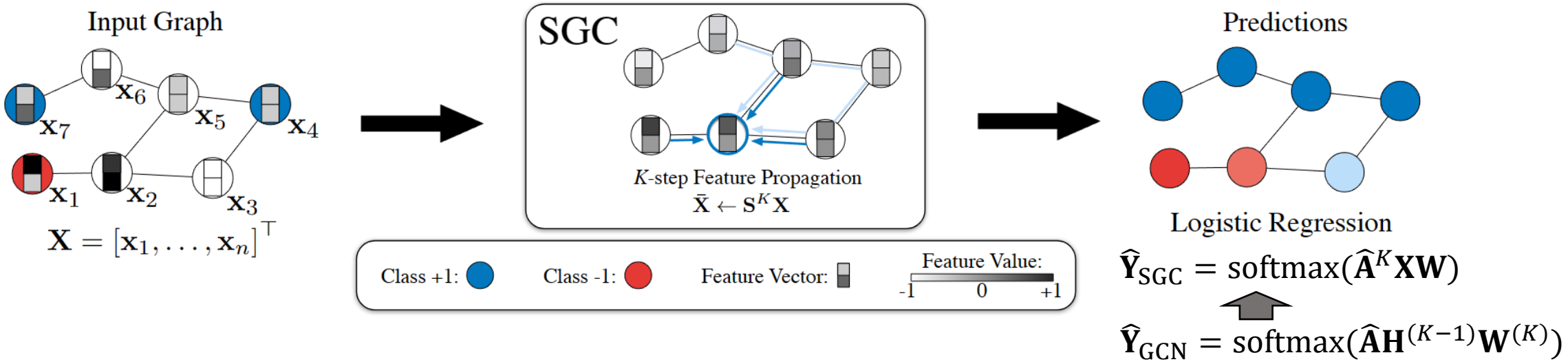


Deep or Simple?

- Over-smoothing issue
- benefited very limited from the deep GCNs
- Simple models can gain equal or even better performance

[1] Kipf, T. N.; and Welling, M. “Semi-supervised Classification with Graph Convolutional Networks.” ICLR 2017.

Motivation



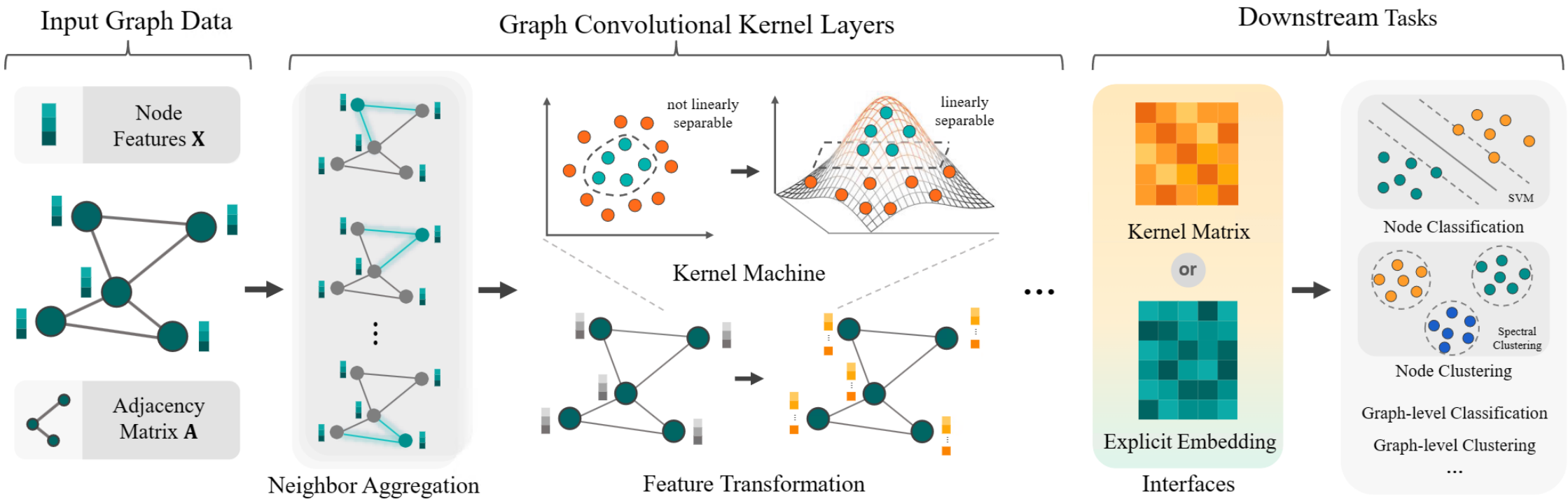
Towards a Better Simplified GCN

- Linear classifier limits the expressiveness of SGC
- Neighbor aggregation (feature propagation) is more crucial than the feature transformation
- A simple feature mapping is helpful and sufficient but should be expressive enough.

How to design a simplified model with simple yet expressive feature mapping?

[2] Wu, Felix, et al. "Simplifying graph convolutional networks." ICML 2019.

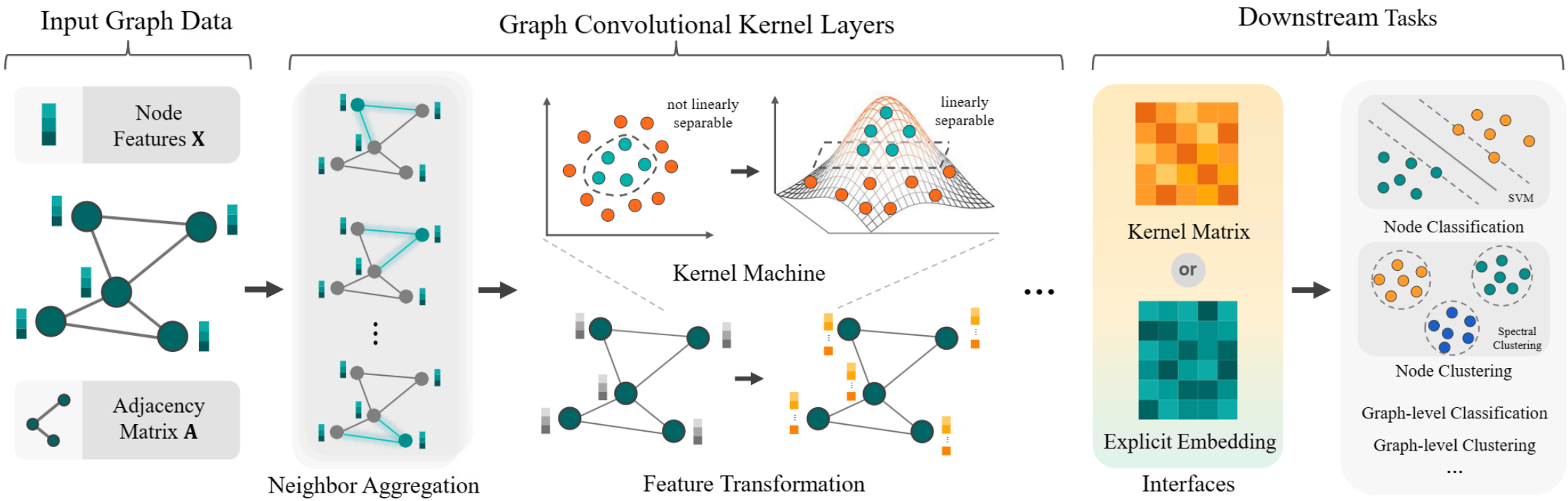
Graph Convolutional Kernel Machine (GCKM)



$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$$

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

Graph Convolutional Kernel Machine (GCKM)

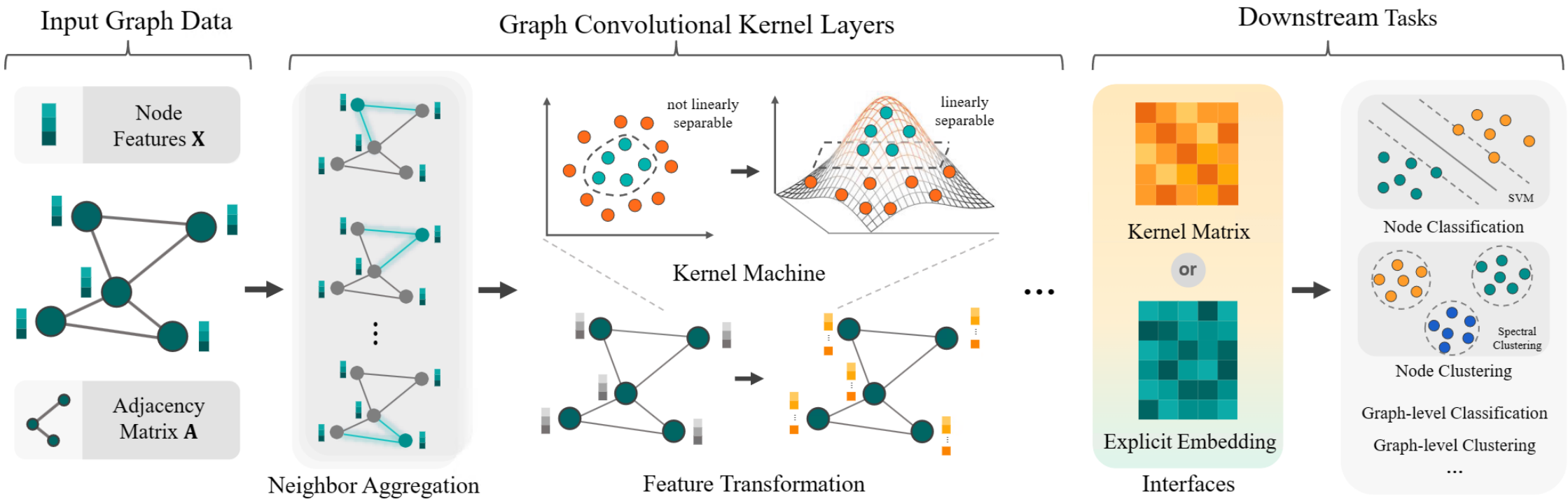


$$\bar{\mathbf{H}}^{(l+1)} = \text{AGGREGATE}\left(\{\mathcal{G}; \mathbf{H}^{(l)}\}\right) = \hat{\mathbf{A}}^q \mathbf{H}^{(l)},$$
$$\mathbf{H}^{(l+1)} = \text{TRANSFORM}\left(\bar{\mathbf{H}}^{(l+1)}\right) = \phi_{(l)}(\bar{\mathbf{H}}^{(l+1)}),$$

Implicit

$$\begin{cases} \bar{\mathbf{K}}^{(l+1)} = \hat{\mathbf{A}}^q \mathbf{K}^{(l)} (\hat{\mathbf{A}}^q)^\top, \\ \mathbf{K}^{(l+1)} = \exp\left(-\frac{\mathbf{d}_{\bar{\mathbf{K}}^{(l+1)}} \mathbf{1}_n^\top + \mathbf{1}_n \mathbf{d}_{\bar{\mathbf{K}}^{(l+1)}}^\top - 2\bar{\mathbf{K}}^{(l+1)}}{2\sigma_{l+1}^2}\right) \end{cases}$$

Graph Convolutional Kernel Machine (GCKM)

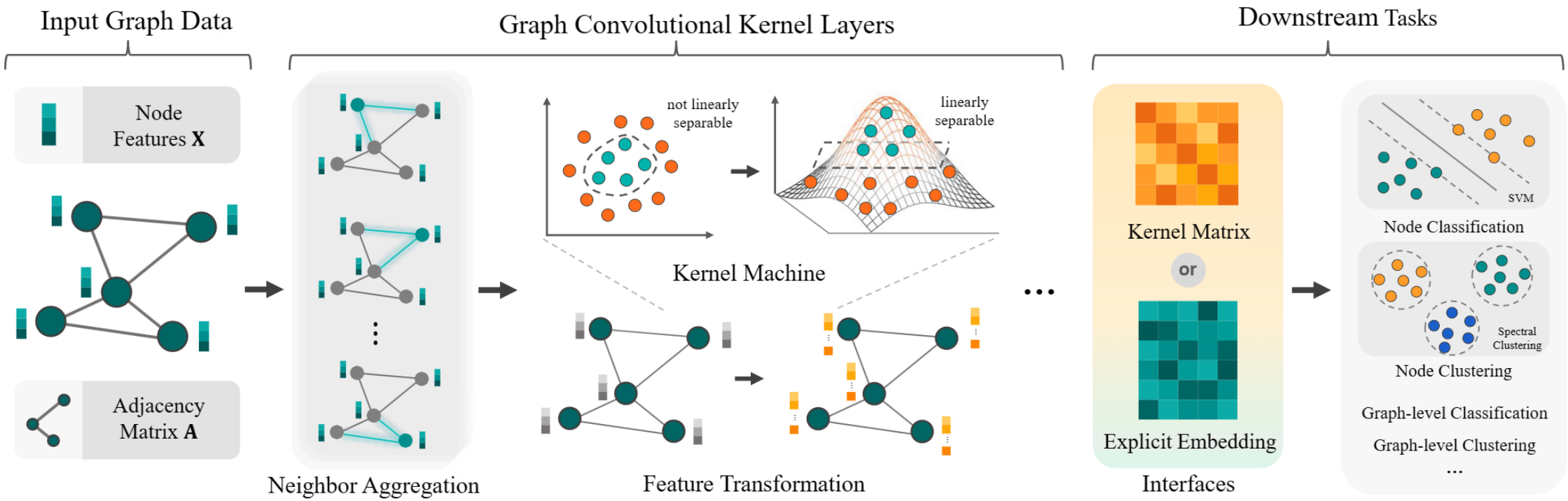


Explicit

$$\psi_{(l)}\left(\bar{\mathbf{z}}_i^{(l+1)}\right) = \sqrt{\frac{1}{D}}\left[\cos(\boldsymbol{\omega}_1^\top \bar{\mathbf{z}}_i^{(l+1)}), \ldots, \cos(\boldsymbol{\omega}_D^\top \bar{\mathbf{z}}_i^{(l+1)}), \sin(\boldsymbol{\omega}_1^\top \bar{\mathbf{z}}_i^{(l+1)}), \ldots, \sin(\boldsymbol{\omega}_D^\top \bar{\mathbf{z}}_i^{(l+1)})\right]^\top$$

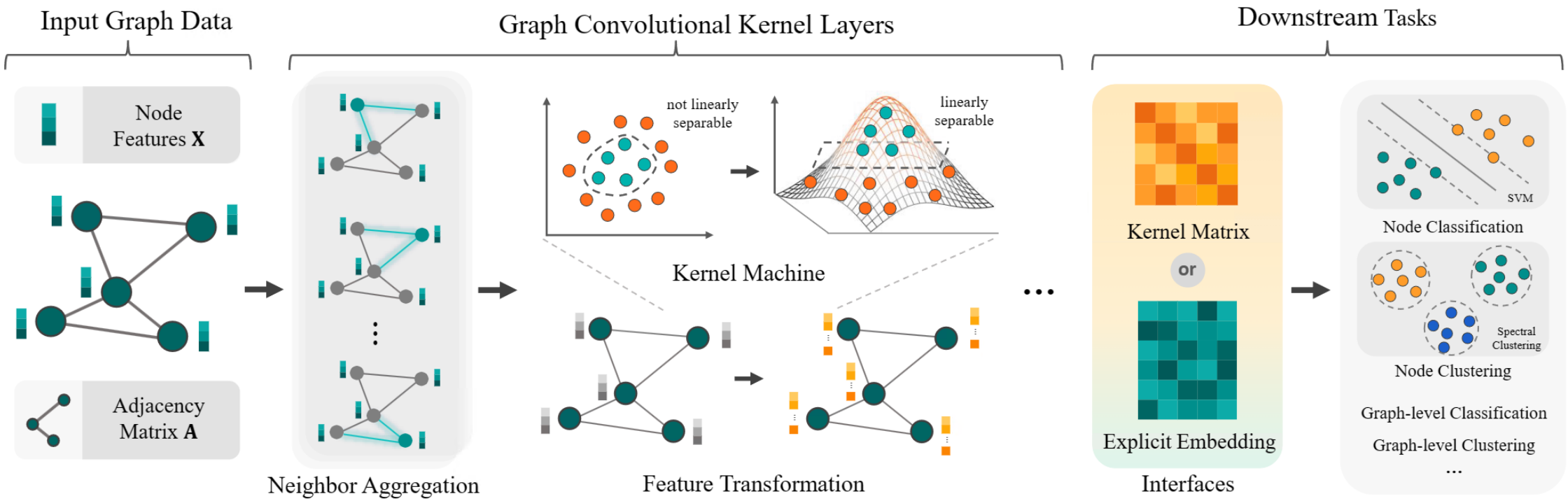
where $\{\boldsymbol{\omega}_1^{(l)}, \boldsymbol{\omega}_2^{(l)}, \ldots, \boldsymbol{\omega}_D^{(l)}\}$ are sampled from $p_{\mathrm{RBF}}(\boldsymbol{\omega}) = \mathcal{N}\left(0, \frac{1}{\sigma}\mathbf{I}\right)$

Graph Convolutional Kernel Machine (GCKM)



$$\mathcal{K}_{\mathbb{G}}^{(L)} = \left[\mathcal{K}_{gg'}^{(L)}\right]_{(g,g')\in[N]\times[N]} = \begin{bmatrix} \mathbf{1}_{n_1}^\top & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{1}_{n_N}^\top \end{bmatrix} \begin{bmatrix} \mathbf{K}_{11}^{(L)} & \cdots & \mathbf{K}_{1N}^{(L)} \\ \vdots & \ddots & \vdots \\ \mathbf{K}_{N1}^{(L)} & \cdots & \mathbf{K}_{NN}^{(L)} \end{bmatrix} \begin{bmatrix} \mathbf{1}_{n_1} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{1}_{n_N} \end{bmatrix}$$

Graph Convolutional Kernel Machine (GCKM)



Proposition 1. *Suppose* $k^{(L)}$ *is a Gaussian kernel with* $0 < \sigma_L < \infty$. *Then* $\mathcal{K}_{\mathbb{G}}^{(L)}$ *is positive definite.*

$\mathcal{K}_{\mathbb{G}}^{(L)}$ can be applied to various graph-level learning tasks such as graph classification and clustering.

Theorem 1. *Denote the index set of the support vectors as* $\mathcal{V} = \{i : 1 \le i \le n, \hat{c}_i \neq 0\}$. *For any* $0 < \delta < 1$, *it holds with probability at least* $1 - \delta$ *over a set of* n *samples* $S \sim \mathcal{D}$ *that*

$$\mathcal{L}_{\mathcal{D}}(\hat{\mathbf{c}}, \mathbf{K}^{(L)}) \le \mathcal{L}_S(\hat{\mathbf{c}}, \mathbf{K}^{(L)}) + O\left(\frac{\eta \ln n + \ln(1/\delta)}{n} + \sqrt{\frac{\eta \ln n + \ln(1/\delta)}{n} \cdot \mathcal{L}_S(\hat{\mathbf{c}}, \mathbf{K}^{(L)})}\right), \tag{11}$$

where $\eta = \sum_{i\in\mathcal{V}} \hat{c}_i^2 + \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}\setminus i, y_i = y_j} \hat{c}_i \hat{c}_j K_{ij}^{(L)} - \sum_{i\in\mathcal{V}} \sum_{j\in\mathcal{V}, y_i \neq y_j} \hat{c}_i \hat{c}_j K_{ij}^{(L)} \le \frac{\lambda^2 |\mathcal{V}|^2}{n^2}$.

$$K_{ij}^{(L)} = \exp\left(- \|(\hat{\mathbf{A}}^q)_i \mathbf{H}^{(L-1)} - (\hat{\mathbf{A}}^q)_j \mathbf{H}^{(L-1)}\|^2 / (2\sigma_L^2)\right)$$

Given a fixed λ, the graph convolution reduces the number of support vectors $|\mathcal{V}|$ and has minor influence on the training error $\mathcal{L}_S(\hat{\mathbf{c}}, \mathbf{K}^{(L)})$, which eventually reduces the upper bound of test error. The fundamental reason is that incorporating the graph structure significantly improved the quality of the kernel matrix $\mathbf{K}^{(L)}$, in which the overall within-class similarity becomes much larger than the between-class similarity.

Table 9: The values of the training error (TE) and the number of support vectors $|\mathcal{V}|$. Note that λ is the regularization parameter of SVM, $\mathbf{I}_n$ and $\mathbf{1}_n$ are $n \times n$ identity matrix and all-ones matrix respectively.

		$\mathbf{I}_n$		$\mathbf{1}_{n\times n} - \mathbf{I}_n$		$\hat{\mathbf{A}}^q$	
		TE	$\lvert\mathcal{V}\rvert$	TE	$\lvert\mathcal{V}\rvert$	TE	$\lvert\mathcal{V}\rvert$
Cora	$\lambda = 1$	0.01	80	0.50	32	0.20	40
	$\lambda = 10$	0.00	72	0.50	32	0.00	30
	$\lambda = 50$	0.00	72	0.50	32	0.00	24
	$\lambda = 100$	0.00	72	0.50	32	0.00	24
	$\lambda = 1000$	0.00	72	0.50	32	0.00	24
Citeseer	$\lambda = 1$	0.02	120	0.50	62	0.20	120
	$\lambda = 10$	0.00	112	0.50	62	0.00	105
	$\lambda = 50$	0.00	111	0.50	62	0.00	102
	$\lambda = 100$	0.00	111	0.50	62	0.00	102
	$\lambda = 1000$	0.00	111	0.50	62	0.00	102
Pubmed	$\lambda = 1$	0.12	1206	0.48	802	0.18	1366
	$\lambda = 10$	0.05	755	0.48	802	0.10	755
	$\lambda = 50$	0.02	617	0.48	794	0.07	571
	$\lambda = 100$	0.01	574	0.48	802	0.07	508
	$\lambda = 1000$	0.00	548	0.48	804	0.11	396

Assumption 1. *The aggregation step with graph $\mathcal{G}$ increases the inner product between the kernel feature maps of samples in the same class and reduces or does not change the inner product between the kernel feature maps of samples in different classes.*

Theorem 2. *Given a graph $\mathcal{G}$ that satisfies Assumption 1, let φ and $\varphi_{\mathcal{G}}$ be the kernel feature maps without and with aggregation on graph $\mathcal{G}$ respectively. The corresponding negative Lagrangian dual objectives (to minimize) are denoted as $\mathcal{L}(\mathbf{c}) := \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j q_{ij} - \sum_{i=1}^{n} c_i$, where $q_{ij} = y_i y_j \varphi(\mathbf{x}_i)^\top \varphi(\mathbf{x}_j)$ and $\mathcal{L}_{\mathcal{G}}(\mathbf{c}) := \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j q^{\mathcal{G}}_{ij} - \sum_{i=1}^{n} c_i$, where $q^{\mathcal{G}}_{ij} = y_i y_j \varphi_{\mathcal{G}}(\mathbf{x}_i)^\top \varphi_{\mathcal{G}}(\mathbf{x}_j)$. We have*

$$\mathcal{L}_{\mathcal{G}}(\mathbf{c}) \geq \mathcal{L}(\mathbf{c}) + \mathcal{R}(\mathbf{c}), \tag{12}$$

where $\mathcal{R}(\mathbf{c}) \geq 0$ is a regularization term inducing sparsity in $\mathbf{c}$.

Semi-supervised Node Classification

Table 1: Accuracy (mean% and standard deviation%) of all methods, note that the best results are highlighted in **red** and the second-best results highlighted are in **blue**.

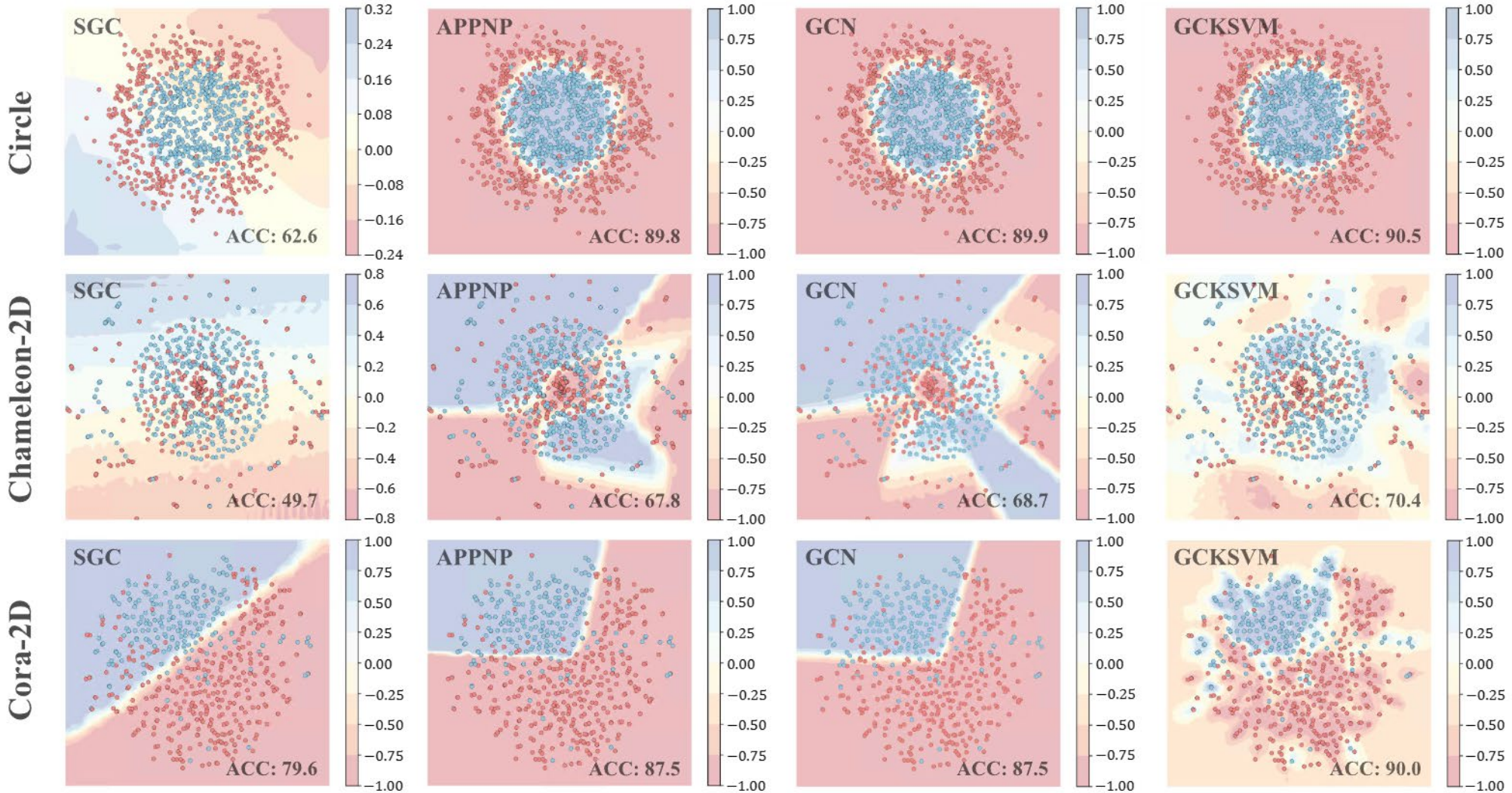
	Standard Split			Random Split		
	Cora	Citeseer	Pubmed	Cora	Citeseer	Pubmed
Chebyshev	75.0 (0.7)	63.8 (0.6)	74.7 (0.8)	76.1 (1.5)	63.4 (3.8)	75.4 (2.8)
GraphSAGE	76.9 (0.5)	63.3 (0.7)	74.8 (0.2)	76.3 (0.9)	63.2 (1.1)	73.9 (1.8)
GAT	82.0 (0.5)	69.8 (0.4)	77.5 (0.2)	79.2 (1.4)	65.8 (2.9)	78.1 (1.7)
GCN	81.7 (0.7)	70.5 (0.5)	78.4 (0.4)	80.6 (1.4)	69.1 (1.8)	77.4 (2.1)
SGC	81.3 (0.0)	68.4 (0.1)	78.6 (0.1)	79.6 (1.3)	68.0 (1.7)	76.6 (2.4)
APPNP	79.5 (0.6)	70.0 (0.8)	78.8 (1.3)	80.1 (1.8)	**69.2 (2.1)**	77.7 (2.0)
JKNet	77.9 (0.8)	72.3 (0.1)	**80.1 (0.2)**	78.0 (1.7)	66.9 (2.3)	77.0 (1.7)
DAGNN	**82.6 (1.1)**	71.0 (0.5)	**80.0 (1.0)**	**81.8 (1.2)**	68.4 (1.4)	78.8 (1.4)
AdaGCN	77.1 (0.1)	69.4 (0.2)	78.0 (0.1)	76.7 (1.8)	67.0 (1.2)	77.3 (1.3)
AMGCN	81.3 (0.4)	70.4 (0.2)	75.5 (0.9)	80.9 (1.2)	68.1 (1.4)	74.2 (2.1)
DefGCN	77.8 (1.0)	67.5 (1.7)	77.9 (0.6)	78.4 (2.3)	67.8 (2.6)	77.1 (1.7)
GCKSVM-E	82.4 (0.3)	**72.4 (0.4)**	79.1 (0.4)	80.8 (0.8)	68.5 (1.3)	**79.2 (1.2)**
GCKSVM	**82.4 (0.0)**	**72.3 (0.0)**	79.8 (0.0)	**83.3 (0.8)**	**71.9 (1.0)**	**80.9 (0.5)**

Graph Classification

Table 2: Graph classification accuracy (mean% and standard deviation%) of all methods, note that the best results are highlighted in **red** and the second-best results highlighted are in **blue**.

	IMDB-B	IMDB-M	COLLAB	MUTAG	PROTEINS	PTC
WL subtree	73.8 (3.9)	50.9 (3.8)	78.9 (1.9)	**90.4 (5.7)**	75.0 (3.1)	59.9 (4.3)
DCNN	49.1	33.5	52.1	67	61.3	56.6
PATCHYSAN	71.0 (2.2)	45.2 (2.8)	72.6 (2.2)	**92.6 (4.2)**	75.9 (2.8)	60.0 (4.8)
DGCNN	70	47.8	73.7	85.8	75.5	58.6
AWL	74.5 (5.9)	51.5 (3.6)	73.9 (1.9)	87.9 (9.8)	—	—
MLP	73.7 (3.7)	52.3 (3.1)	79.2 (2.3)	84.0 (6.1)	76.0 (3.2)	**66.6 (6.9)**
GIN	**75.1 (5.1)**	**52.3 (2.8)**	**80.2 (1.9)**	89.4 (5.6)	**76.2 (2.8)**	64.6 (7.0)
GCN	74.0 (3.4)	51.9 (3.8)	79.0 (1.8)	85.6 (5.8)	**76.0 (3.2)**	64.2 (4.3)
GraphSAGE	72.3 (5.3)	50.9 (2.2)	—	—	75.9 (3.2)	63.9 (7.7)
GCKSVM	**75.4 (2.4)**	**53.9 (2.8)**	**81.7 (1.5)**	88.7 (7.6)	74.5 (3.9)	**67.7 (5.4)**

Decision Boundary

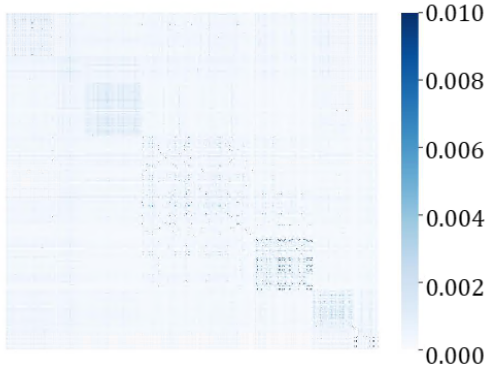


Node Clustering

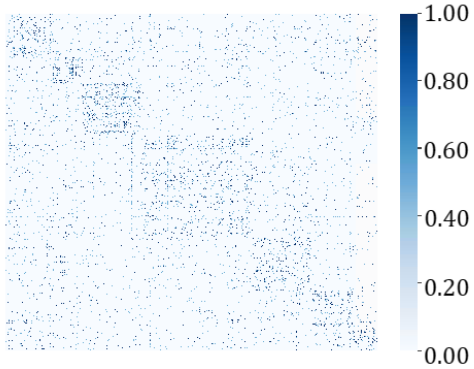
Table 3: ACC, NMI, and ARI of all methods, note that the best results are highlighted in **red** and the second-best results highlighted are in **blue**.

	Cora			Citeseer			Pubmed		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
K-means	49.20	32.10	22.90	54.00	30.50	27.80	59.50	31.50	28.10
SC	36.70	12.60	3.10	23.80	5.50	1.00	52.80	9.70	6.20
GAE	59.60	42.90	34.70	40.80	17.60	12.40	67.20	27.70	27.90
VGAE	50.20	32.90	25.40	46.70	26.00	20.50	63.00	22.90	21.30
ARGA	64.00	44.90	35.20	35.20	35.00	34.10	66.80	30.50	29.50
ARVGA	64.00	45.00	37.40	54.40	26.10	24.50	69.00	29.00	30.60
DGI	55.40	41.10	32.70	51.40	31.50	32.60	58.90	27.70	31.50
MVGRL	73.20	56.20	51.90	68.10	43.20	43.40	69.30	**34.40**	32.30
GALA	**74.59**	**57.67**	**53.15**	69.32	**44.11**	44.60	69.39	32.73	32.14
DFCN	64.07	48.24	39.17	69.50	43.90	**45.50**	69.32	32.19	31.55
S^3GC	74.20	**58.80**	**54.40**	68.80	**44.10**	44.80	**71.30**	**33.30**	**34.50**
GCKSC	**74.30**	54.95	52.89	**71.27**	43.57	**46.51**	**71.31**	32.24	**34.21**

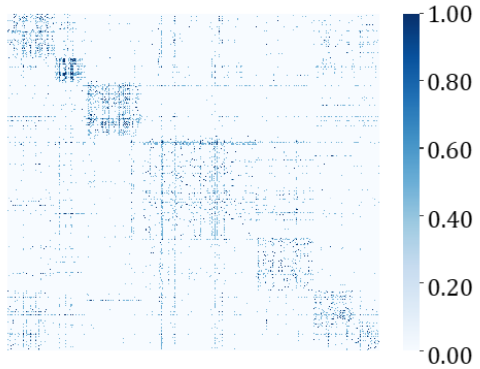
Visualization of Adjacency Matrix & Runtime



(a) Adjacency Matrix (36.7)



(b) Kernel Matrix (63.2)



(c) GCKM Matrix (74.3)

Figure 4: Visualizations of the adjacency matrix $\hat{\mathbf{A}}^q$, kernel matrix $[k(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1}^n$, and GCKM matrix $\mathbf{K}^{(2)}$ on Cora, with corresponding best-tuned clustering accuracy.

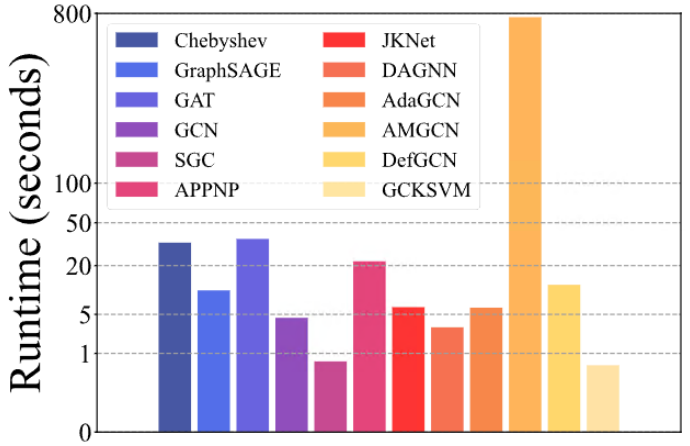


Figure 5: Runtimes of all methods.

THANKS!

Zhihao Wu[1], Zhao Zhang[2], Jicong Fan[1,3*],

[1]Shenzhen Research Institute of Big Data, Shenzhen, China

[2]Hefei University of Technology, Hefei, China

[3]The Chinese University of Hong Kong, Shenzhen, China

{zhihaowu1999,cszzhang}@gmail.com, fanjicong@cuhk.edu.cn

https://github.com/ZhihaoWu99/GCKM