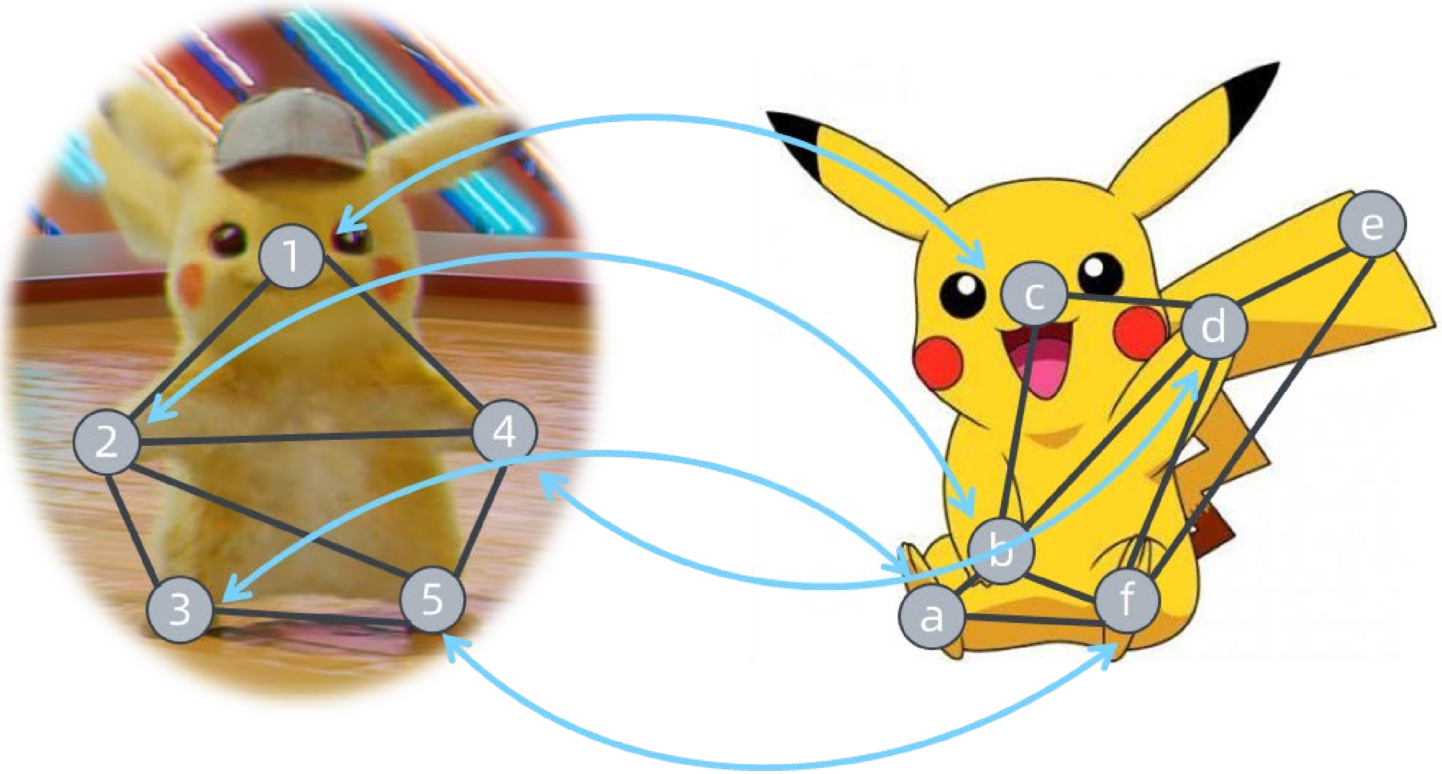
Deep Embedding Networks for Graph Matching

Runzhong Wang - Shanghai Jiao Tong University

R. Wang, J. Yan, X. Yang. "Learning Combinatorial Embedding Networks for Deep Graph Matching." In ICCV 2019.

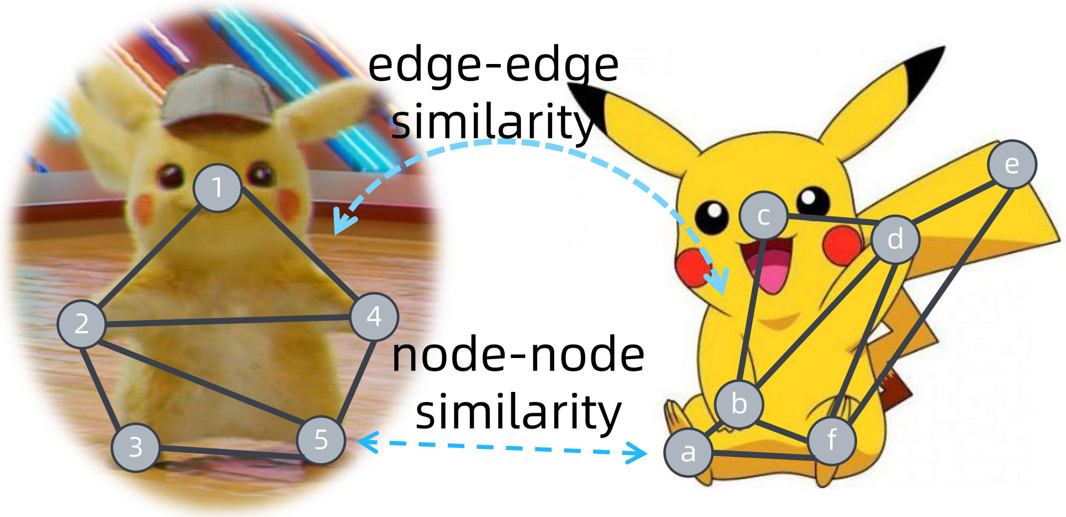
Seoul, Korea. 2019.10.30

Graph Matching



Graph matching finds node correspondence among multiple graphs.

Formulation via Affinity Maximization



node similarity

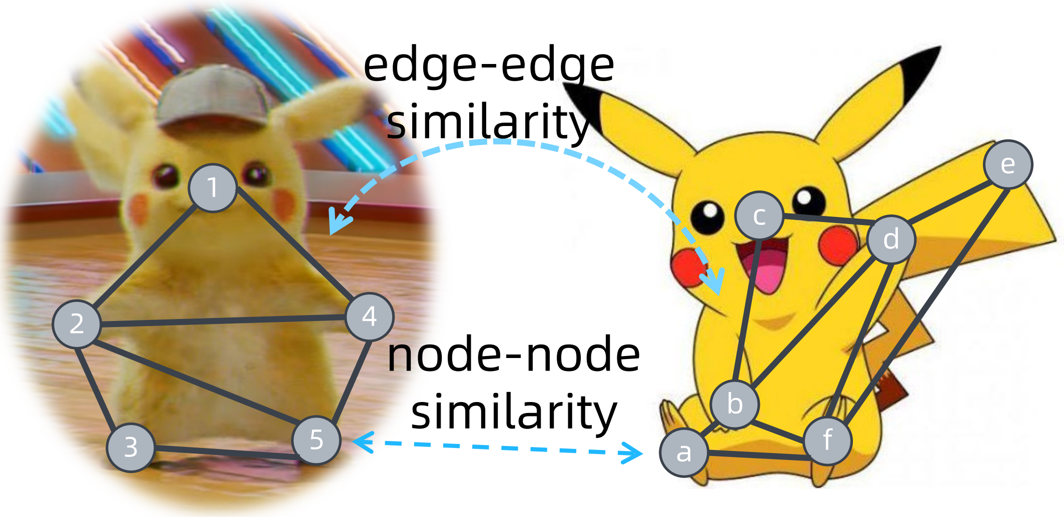
$$\max_{\mathbf{x}} \sum_{i_1 i_2} x_{i_1 i_2} \kappa^{p}_{i_1 i_2} + \sum_{\substack{(i_1,j_1)\in E_1 \\ (i_2,j_2)\in E_2}} x_{i_1 i_2} x_{j_1 j_2} \kappa^{q}_{c(i_1,j_1)c(i_2,j_2)}$$

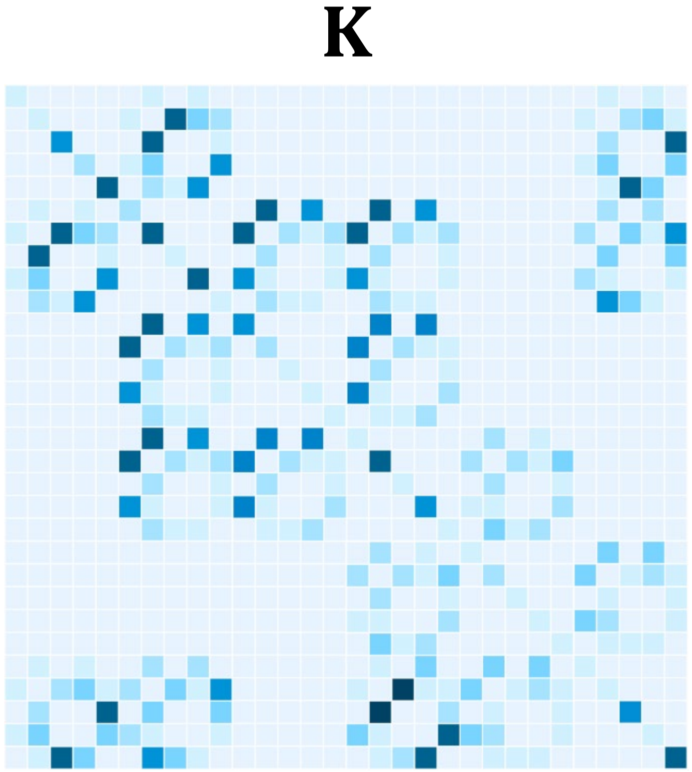
edge similarity

Graph matching incorporates both **first order (node-node)** and **second order (edge-edge)** similarities.

S. Gold and A. Rangarajan. "A graduated assignment algorithm for graph matching," IEEE Transaction on PAMI, 1996

Graph Matching via Affinity Matrix





Formulated as quadratic assignment problem (QAP):

$$\max_{\mathbf{X}} \ \mathrm{vec}(\mathbf{X})^T \mathbf{K} \ \mathrm{vec}(\mathbf{X})$$

$$\text{s.t.} \quad \mathbf{X} \in \{0,1\}^{5\times 6}, \mathbf{X}\mathbf{1} \leq \mathbf{1}, \mathbf{X}^{\mathrm{T}}\mathbf{1} \leq \mathbf{1}$$

NP-Hard

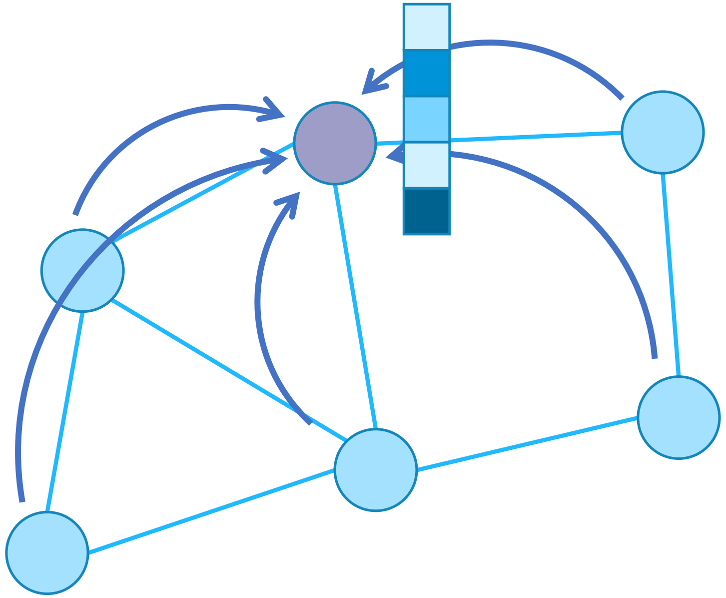
Diagonal elements – node similarity

+

Off-diagonal elements – edge similarity

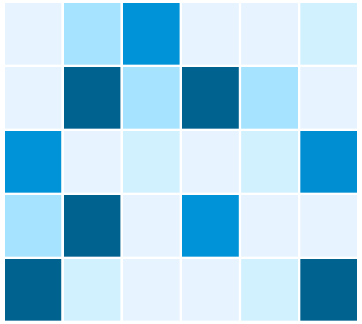
M. Leordeanu and M. Hebert. "A spectral technique for correspondence problems using pairwise constraints." in ICCV, 2005

Graph Embedding and Linear Matching



- Graph embedding embeds graph structure (nodes and edges) into node embedding vectors.
- Graph affinity can be evaluated between node embeddings.

- Linear similarity matrix:

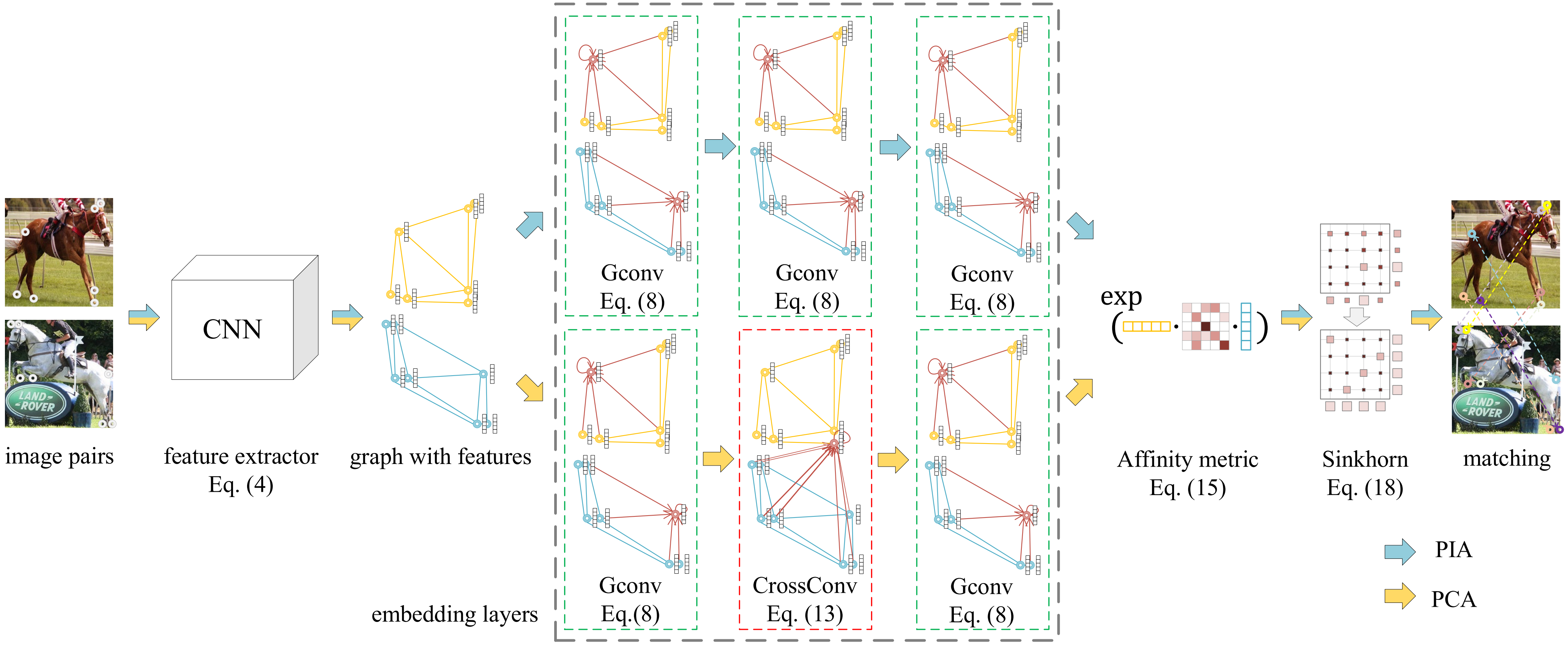


$\mathbf{M}$

- We simplify the NP-hard QAP into Linear Matching problem.

$$\max_{\mathbf{X}} \quad \mathrm{tr}(\mathbf{X}^{\mathrm{T}} \mathbf{M})$$
$$\mathrm{s.t.} \quad \mathbf{X} \in \{0,1\}^{5\times 6}, \mathbf{X}\mathbf{1} \leq \mathbf{1}, \mathbf{X}^{\mathrm{T}}\mathbf{1} \leq \mathbf{1}$$

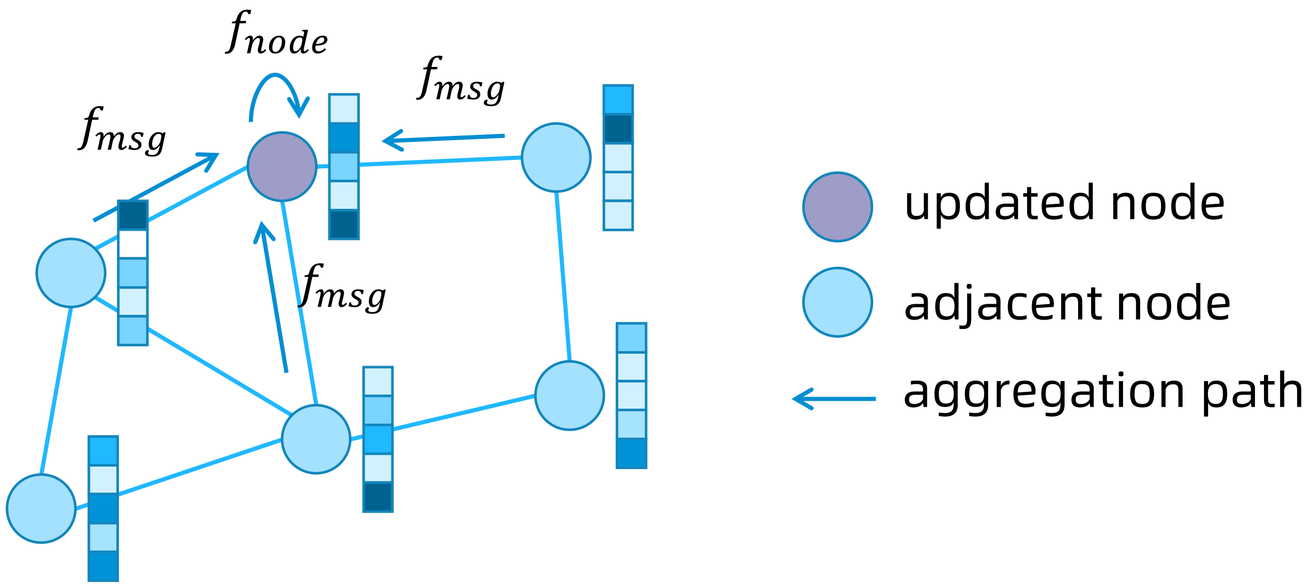
Our approach



- First end-to-end solution in graph matching incorporating **embedding**.
- NP-hard QAP simplified to **LAP (solved exactly)** thanks to embedding.
- Combinatorial **permutation loss** for supervision.

GConv

GConv is mainly inspired by **Graph Convolutional Network (GCN)**.
Feature aggregated from adjacent nodes.

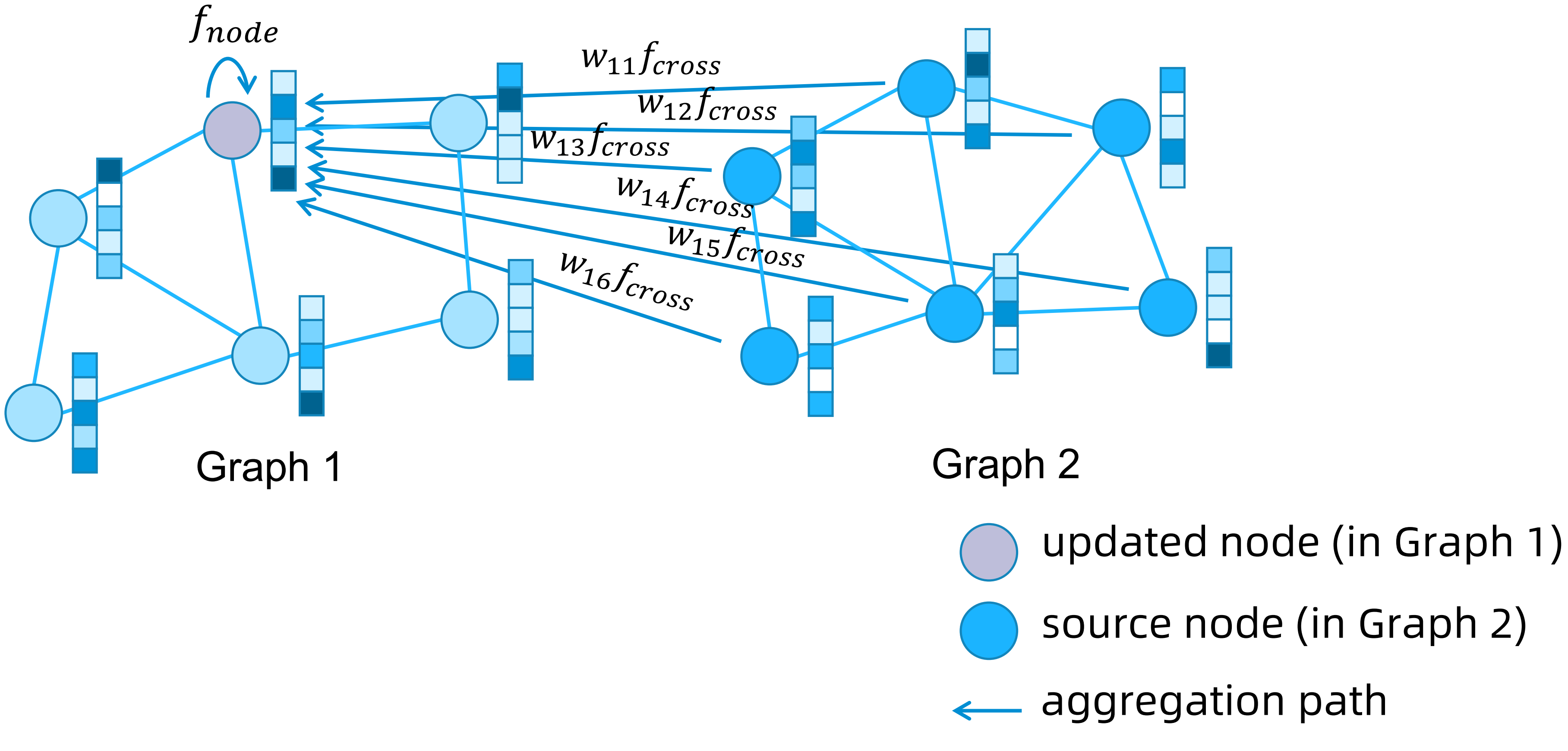


f_{node} and f_{msg} are neural networks and weight-sharing among all nodes.

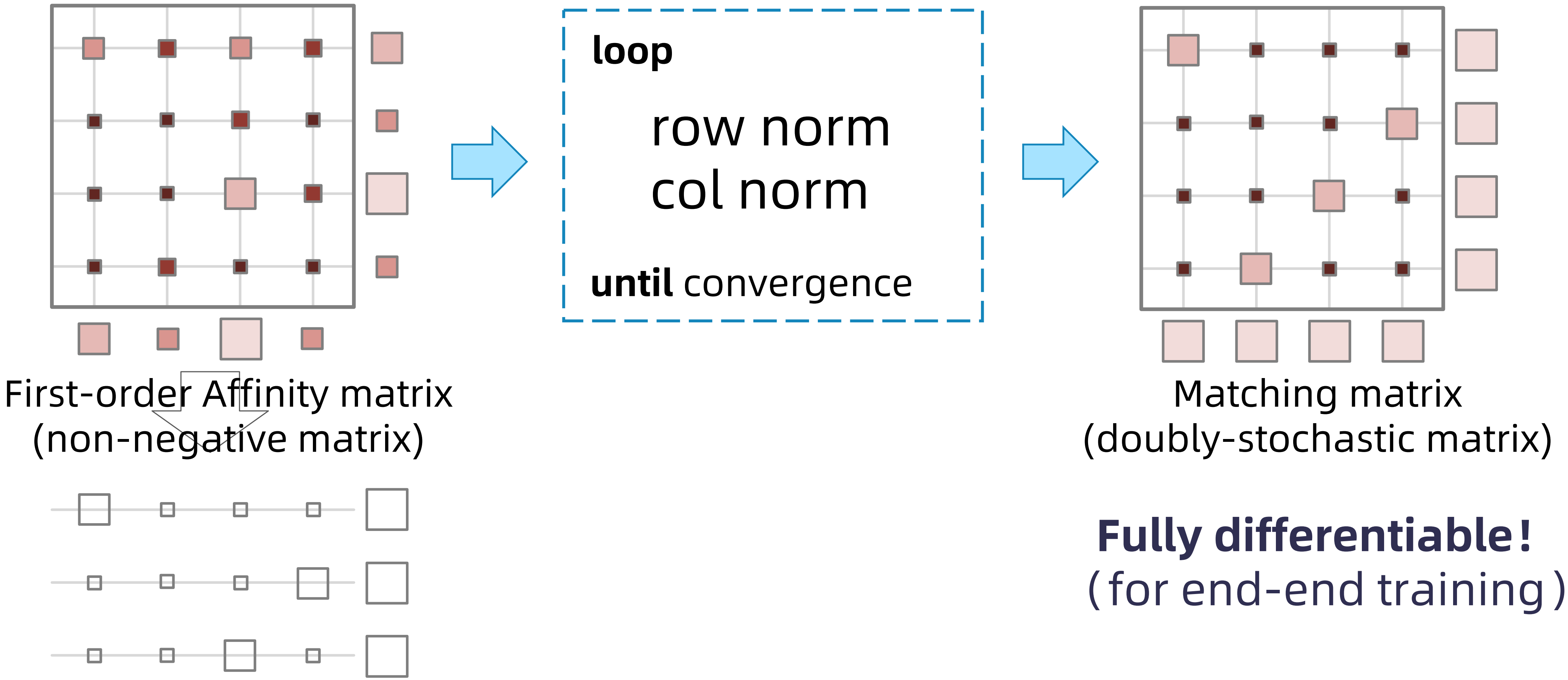
Kipf and Welling, "Semi-Supervised Classification with Graph Convolutional Networks." ICLR 2017.

CrossConv

Features are aggregated from nodes with similar features across graphs.



Sinkhorn



Fully differentiable!
(for end-end training)

Permutation Loss

0.3	0.5	0.1	0.1
0.2	0.1	0.6	0.1
0.4	0.1	0.2	0.3
0.1	0.3	0.1	0.5

prediction S

cross-entropy

1.	0	0	0
0	0	1.	0
0	1.	0	0
0	0	0	1.

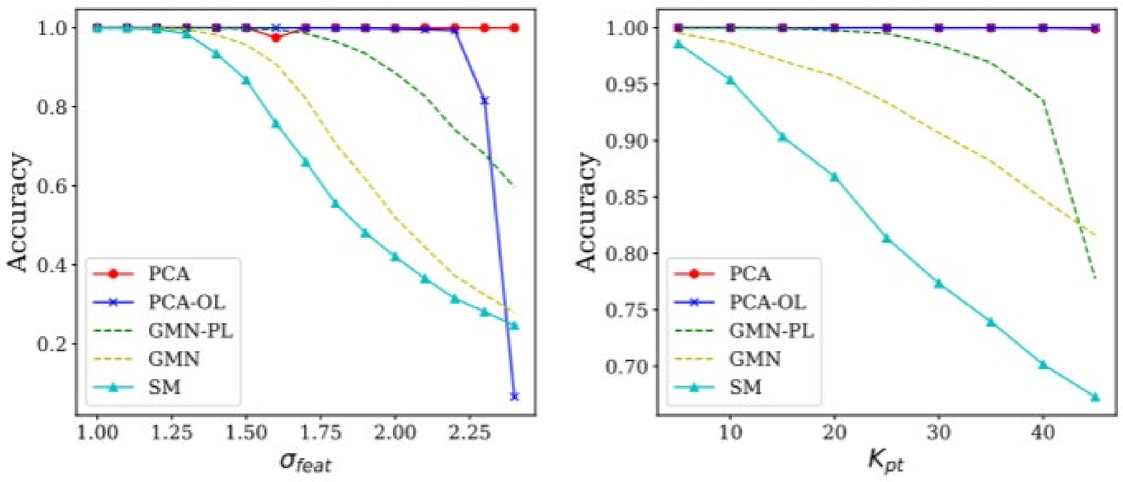
ground truth S^{gt}

$$L_{perm} = \frac{1}{N} \sum_{ij} \left[S_{ij}^{gt} \log S_{ij} + \left(1 - S_{ij}^{gt}\right) \log\left(1 - S_{ij}\right) \right]$$

It meets the combinatorial nature of graph matching.

State-of-the-art Performance on

- Synthetic Data



- Real Image Datasets

method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
GMN	31.9	47.2	51.9	40.8	68.7	72.2	53.6	52.8	34.6	48.6	**72.3**	47.7	54.8	51.0	38.6	75.1	49.5	45.0	83.0	86.3	55.3
GMN-PL	31.1	46.2	58.2	45.9	70.6	76.4	61.2	61.7	**35.5**	53.7	58.9	57.5	56.9	49.3	34.1	77.5	57.1	53.6	83.2	88.6	57.9
PIA-OL	39.7	**57.7**	58.6	47.2	74.0	74.5	62.1	66.6	33.6	61.7	65.4	58.0	67.1	58.9	41.9	77.7	64.7	50.5	81.8	89.9	61.6
PIA	**41.5**	55.8	60.9	**51.9**	75.0	75.8	59.6	65.2	33.3	**65.9**	62.8	**62.7**	67.7	62.1	42.9	**80.2**	64.3	**59.5**	82.7	90.1	63.0
PCA	40.9	55.0	**65.8**	47.9	**76.9**	**77.9**	**63.5**	**67.4**	33.7	65.5	63.6	61.3	**68.9**	**62.8**	**44.9**	77.5	**67.4**	57.5	**86.7**	**90.9**	**63.8**

method	face	m-bike	car	duck	w-bottle
HARG-SSVM [6]	91.2	44.4	58.4	55.2	66.6
GMN-VOC [45]	98.1	65.0	72.9	74.3	70.5
GMN-Willow [45]	99.3	71.4	74.3	82.8	76.7
PCA-VOC	**100.0**	69.8	78.6	82.4	95.1
PCA-Willow	**100.0**	**76.7**	**84.0**	**93.5**	**96.9**

Thank you!

↓ Code on GitHub ↓

- Paper: https://arxiv.org/abs/1904.00597
- Code: https://github.com/Thinklab-SJTU/PCA-GM

- Contact
  - Runzhong Wang:
    runzhong.wang@sjtu.edu.cn
  - Prof. Junchi Yan:
    yanjunchi@sjtu.edu.cn

- Homepage of our lab:
  http://thinklab.sjtu.edu.cn