Combinatorial Learning of Robust Deep Graph Matching: an Embedding based Approach

Runzhong Wang, Junchi Yan, *Member, IEEE,* and Xiaokang Yang, *Fellow, IEEE*

Abstract—Graph matching aims to establish node correspondence between two graphs, which has been a fundamental problem for its NP-hard nature. One practical consideration is the effective modeling of the affinity function in the presence of noise, such that the mathematically optimal matching result is also physically meaningful. This paper resorts to deep neural networks to learn the node and edge feature, as well as the affinity model for graph matching in an end-to-end fashion. The learning is supervised by combinatorial permutation loss over nodes. Specifically, the parameters belong to convolutional neural networks for image feature extraction, graph neural networks for node embedding that convert the structural (beyond second-order) information into node-wise features that leads to a linear assignment problem, as well as the affinity kernel between two graphs. Our approach enjoys flexibility in that the permutation loss is agnostic to the number of nodes, and the embedding model is shared among nodes such that the network can deal with varying numbers of nodes for both training and inference. Moreover, our network is class-agnostic. Experimental results on extensive benchmarks show its state-of-the-art performance. It bears some generalization capability across categories and datasets, and is capable for robust matching against outliers.

Index Terms—Graph Matching, Deep Learning, Graph Embedding, Combinatorial Optimization.

1 Introduction and Preliminaries

Graph matching (GM) aims to solve the problem of finding node correspondences over two or multiple graphs. It incorporates both node-wise unary similarity and edge-wise [1], [2] (or even higher-order [3], [4], [5]) similarity to establish a matching, in order to maximize the similarity between the matched graphs. By encoding the structural information in the objective, graph matching can often achieve more robust performance against disturbance. In contrast, the point based methods e.g. RANSAC [6] and iterative closet point (ICP) [7] do not explicitly account for such edge-to-edge information. For its expressiveness, graph matching has lied at the heart of many computer vision applications [8] such as action recognition, robotics, visual tracking, weak-perspective 3-D reconstruction. Refer to [9] for a more comprehensive literature review.

As a classic combinatorial problem, graph matching is known in general NP-hard [10], which has been addressed mostly by approximate techniques leading to inexact solutions. Consider the classic setting of two-graph matching between $\mathcal{G}_1$, $\mathcal{G}_2$, it can be generally expressed by the quadratic assignment programming (QAP) form:

$$\begin{aligned} &J(\mathbf{X}) = \mathrm{vec}(\mathbf{X})^\top \mathbf{K} \mathrm{vec}(\mathbf{X}), \\ &\mathbf{X} \in \{0,1\}^{N_1 \times N_2}, \quad \mathbf{X}\mathbf{1} = \mathbf{1}, \quad \mathbf{X}^\top \mathbf{1} \leq \mathbf{1} \end{aligned} \tag{1}$$

where $\mathbf{X}$ is a binary permutation matrix encoding the node correspondence, and the so-called affinity matrix $\mathbf{K} \in \mathbb{R}^{N_1 N_2 \times N_1 N_2}$ encoding the node-to-node and edge-to-edge affinity between two graphs, by its diagonal elements and off-diagonal ones, respectively. To calculate $\mathbf{K}$, a practical form is the Gaussian kernel by $\mathbf{K}_{ia,jb} = \exp\left(\frac{(\mathbf{f}_{ij} - \mathbf{f}_{ab})^2}{\sigma^2}\right)$ where $\mathbf{f}_{ij}$ denotes the feature vector of edge ij. Note the node similarity can also be encoded given the node index $ia = jb$ holds.

In particular, Eq. 1 is called Lawler's QAP [11], which can incorporate other more specific forms e.g. Koopmans-Beckmann's QAP [10]:

$$J(\mathbf{X}) = \mathrm{tr}(\mathbf{X}^\top \mathbf{F}_1 \mathbf{X} \mathbf{F}_2) + \mathrm{tr}(\mathbf{K}_p^\top \mathbf{X}) \tag{2}$$

where $\mathbf{F}_1 \in \mathbb{R}^{N_1 \times N_1}$, $\mathbf{F}_2 \in \mathbb{R}^{N_2 \times N_2}$ are weighted adjacency matrices of graph $\mathcal{G}_1$, $\mathcal{G}_2$ respectively. Matrix $\mathbf{K}_p$ denotes the node-to-node affinity. The connection to Lawler's QAP becomes clear by setting $\mathbf{K} = \mathbf{F}_2 \otimes \mathbf{F}_1$.

Another popular formulation is the so-called factorized graph matching model [12], which shows how to factorize affinity matrix $\mathbf{K}$ as a Kronecker product of smaller matrices. For concise, here we write the undirected version [12]:

$$\begin{aligned} &\mathbf{K} = (\mathbf{H}_2 \otimes \mathbf{H}_1)\mathrm{diag}(\mathrm{vec}(\mathbf{L}))(\mathbf{H}_j \otimes \mathbf{H}_i)^\top \\ \text{where} \quad &\mathbf{H}_k = [\mathbf{G}_k, \mathbf{I}_{n_k}] \in \{0,1\}^{n_k \times (m_k + n_k)}, \quad k = i, j \\ &\mathbf{L} = \begin{bmatrix} \mathbf{K}^q & -\mathbf{K}^q \mathbf{G}_j^\top \\ -\mathbf{G}_i \mathbf{K}^q & \mathbf{G}_i \mathbf{K}^q \mathbf{G}_j^\top + \mathbf{K}^p \end{bmatrix} \end{aligned} \tag{3}$$

where n_i and m_i is the number of nodes and edges in graph $\mathcal{G}_i$ respectively and $\otimes$ is the Kronecker product operation between matrices. $\mathbf{K}^p \in \mathbb{R}^{n_i \times n_j}$ denotes the node affinity matrix, and $\mathbf{K}^q \in \mathbb{R}^{m_i \times m_j}$ for the edge affinity matrix. The graph structure is specified by the node-edge incidence matrix $\mathbf{G} \in \mathbb{R}^{n \times m}$ such that the non-zero elements in each column of $\mathbf{G}$ indicate the starting and ending nodes in the corresponding edge. The factorization provides a taxonomy

• *R. Wang and J. Yan are with Department of Computer Science and Engineering, and MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, Shanghai, 200240, P.R. China.*
X. Yang is with MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, Shanghai, 200240, P.R. China.
E-mail: {runzhong.wang, yanjunchi, xkyang}@sjtu.edu.cn
Junchi Yan is the corresponding author.
Project homepage: http://thinklab.sjtu.edu.cn/IPCA_GM.html.

Manuscript received April 19, 2005; revised August 26, 2015.

for GM and reveals the connection among several methods. Readers are referred to [12] for greater details.

In parallel, recent efforts also try to explore higher-order affinity information which is beyond the second-order. Tensor marginalization is a popular technique as widely used by hypergraph matching works [5], [13], [14], [15]:

$$\begin{aligned} \mathbf{x}^* &= \arg\max(\mathbf{H} \otimes_1 \mathbf{x} \otimes_2 \mathbf{x} \dots \otimes_m \mathbf{x}) \quad s.t. \\ \mathbf{X}\mathbf{1} &= \mathbf{1}, \mathbf{X}^\top \mathbf{1} \leq \mathbf{1}, \mathbf{x} = \mathrm{vec}(\mathbf{X}) \in \{0,1\}^{N_1 N_2 \times 1} \end{aligned} \tag{4}$$

where m is the order of the affinity model, and $\mathbf{H}$ is the m-order affinity tensor which encodes hyperedge affinity. While $\otimes_k$ is the tensor product [3]. Readers are referred to Section 3.1 in [14] for details on tensor multiplication. In particular, these works all assume the affinity tensor is invariant regarding with the index of hyperedge pairs, to facilitate the optimization.

Meanwhile, there are works on multi-graph matching, aiming to establish node correspondence among multiple graphs. Two typical methodologies have been developed: i) transform the multi-graph problem into a two-graph matching QAP in each iteration [16], [17]; ii) obtain the putative matchings by first solving a two-graph matching problem for further smoothing in post-processing step [18], [19], [20], [21]. Compared with two graph matching, the advantage of joint matching of multiple graphs lie in that the information across graphs can be fused to resist outliers and noises.

Moreover, for all the cases of two-graph, hyper-graph and multi-graph matching, it has been a fundamental requirement for devising appropriate node/edge feature extractor and affinity model to transform the real-world matching task into an equivalent mathematical model, in the sense that the optimal solution corresponds to the meaningful and correct matching. However, the challenge lies in that the manually designated models (e.g. a Gaussian kernel with Euclid distance) may not be expressive enough to fit with the problem and data at hand, especially in the presence of outliers and noises.

Seeing these issues, recently a thread of works aim to learn the affinity model for graph matching, to improve the flexibility and model capacity for affinity modeling. The hope is that the bias introduced by the manually predefined affinity model can be effectively dismissed and the mathematically optimal solution (according to the affinity model and the corresponding objective) can truly reflect the meaningful correspondence. In deed, such learning based works, which are the focus of this paper, are somehow orthogonal to those focusing on devising effective solvers given the predefined affinity model [1], [2], [3], [14].

In summary, we have devised an end-to-end learnable supervised deep network for graph matching, which is known in general NP-hard. The presented work is an extended version[1] to the preliminary conference version [22] and it involves the following features:

i) We transform the graph matching problem to a linear assignment one by utilizing graph embedding network and specifically graph convolutional network, to extract the graph structures into node-wise feature vector i.e. graph embedding. The embedded node features are expected to contain structural information around the node such that even higher-order (beyond second-order) information can be incorporated in the matching procedure. In this way, the model circumvents the notoriously challenging QAP problem. Such a design also allows for different numbers of nodes in different graph pairs for training and testing. To our best knowledge, this is the first time for adopting a deep graph embedding network for learning graph matching.

ii) Combined with our embedding model, a Sinkhorn net based permutation loss for combinatorial optimization is developed. It applies Sinkhorn iteration on the input non-negative matrix to obtain a double-stochastic matrix as the soft matching, such that the cross-entropy loss which is often used for classification can be readily used as the loss for measuring the difference between the soft matching matrix and the ground truth. Such a combinatorial loss also allows for flexible handling of varying-sized graphs which is a persistent challenge in graph matching. For instance, in the structured support vector machine based learning model [23], the number of graphs has to be fixed for training a model. However, such condition can be unrealistic in practice. To our knowledge, this is the first work for adopting the above permutation loss for graph matching.

iii) Experimental results including ablation studies show the effectiveness of our devised components including the permutation loss, graph convolutional network based node embedding, convolutional network based node-wise image feature extraction layer, cross-graph affinity module, and the iterative cross-graph embedding component. In particular, our method outperforms the state-of-the-art peer method [24] based deep networks regarding with matching accuracy. Our method also outperforms [23] in accuracy while being more flexible as the method in [23] assumes fixed number of nodes for matching in both training and testing sets. Detailed study on transfer learning capability shows the robustness of our approach in the presence of outliers, when the training set and testing set are from different object categories, as well as from distinctive datasets.

The paper is organized as follows. Section 2 discusses the related work on graph matching for both learning-free and learning based methods. The main approach is presented in Section 3, which is evaluated in Section 4 with peer methods. Section 5 concludes this paper.

2 Related Work

Graph matching has been a long standing problem in vision and pattern recognition. There is a recent trend for learning of graph matching, which is the focus of this paper.

In this section, we first review the learning-free methods over the decades, and then discuss the recent line of research on learning based graph matching. Readers are referred to the survey [9] to take a detailed review. For learning, the basic idea is to model the feature extraction and representation of graph nodes and edges, as well as the affinity function

1. Compared with [22], the extensions include: i) a new iterative cross-graph embedding technique with the resulting method **IPCA-GM**; ii) an added treatment incorporating dummy nodes in Sinkhorn network against outliers with new experimental results; iii) comprehensive study on the behavior of our model on transfer learning across datasets, categories and with/without outliers, and the study on the learned CNN module and embedding module respectively; and iv) an extended literature review on graph matching that covers both learning-free and learning based approaches. These new results highlight the robustness of our approach in practice.

between graphs. Moreover, the solver can also be learned in different ways. Some related techniques as adopted in our approach are also discussed.

2.1 Learning-free Graph Matching

Over the decades, the majority line of research on graph matching are focused solving the constrained combinatorial optimization problem for graph matching, which assumes the affinity model is given. Along this direction, we review the related works in three aspects: i) two-graph matching, which is the classic setting for graph matching; ii) hypergraph matching namely higher-order graph matching, whereby the higher-order edge information is used in matching in contrast to the first case, in which only up to the second-order affinity is considered; iii) joint matching of multiple graphs and its online incremental matching setting.

2.1.1 Two-graph matching

Most existing works deal with the two-graph matching problem which is known NP-hard in general. Different (deterministic) optimization based techniques have been devised. The annealing based continuation method is adopted in early work [2] which addresses the second-order matching problem by iterative formulating and solving a linear assignment problem. Such strategy is widely adopted in followup works [1], [25] and the quadratic assignment model is explicitly formulated in [26] by introducing the so-called affinity matrix. Based on this compact writing, different relaxation techniques have been explored and a popular treatment is to relax the permutation matrix into a double-stochastic one which is in fact the convex hull in the continuous space [27]. Meanwhile factorization on the affinity matrix is also studied to reduce the space complexity for directly dealing with the affinity matrix, and a continuation method based on convex-concave relaxation [28] is developed. Meanwhile some (quasi-)discrete methods [29], [30] are also developed to avoid the truncation from continuous solution to the binary one which can incur unexpected accuracy loss. More recently, state-of-the-art graph matching solvers [31], [32], [33] are presented with high accuracy as well as heavy computational cost.

2.1.2 Hypergraph Matching

Beyond second-order edge information, hyperedges are also considered to improve the model robustness and expressiveness [3], [15], [34]. Though some improvements on accuracy have been achieved while the involved additional cost is not ignorable. To address this issue, the authors in [4] show a technique to decompose the higher-order (up to fourth order) model into second-order one such that the problem can be solved using classic second-order graph matching solvers. In another work [5], a discrete method is devised for more efficient hyperedge based matching.

2.1.3 Multiple graph Matching

Recently it receives more attention for joint matching of multiple graphs. This setting is especially pronounced in real-world scenarios when it is required to matching a batch of graphs. Also, accessing multiple graphs simultaneously further provides the opportunity to fuse the local information from each graph, which can be noisy or even fundamentally ambiguous and difficult to process independently. These works usually fall into two categories, the first one divides the matching into two stages [18], [20], [35]: first performing two-graph matching independently and then smoothing the pairwise matchings by global cycle-consistency. Here the concept of consistency refers to the correspondence loop closure across three or more graphs. Other works [16], [17] incorporate cycle-consistency into matching process over iterations, in addition with maximizing the affinity objective.

Graph data can arrive in a streaming way such that an incremental matching approach is welcomed to efficiently handle the new coming graphs and existing ones with matching results. The authors in [36] give one such solution by clustering the existing graphs into groups based on cluster-wise randomness and diversity, and incorporates the newly arrived graph by matching it with one of the clusters.

2.2 Learning for Graph Matching

2.2.1 Modeling and learning affinity

To address the challenges of mimicking the real-world setting of the graph matching problem, learning the node-wise and edge-wise affinity has been an effective way of utilizing training data, either in a supervised [23], [24], [37], unsupervised [38] or semi-supervised [38] way. This is in contrast to the aforementioned methods that use simple and predefined parametric models that often involve a Gaussian kernel in the Euclid space for feature distance calculation.

One shallow and unified treatment is to design an affinity function $\Phi(\mathcal{G}_1, \mathcal{G}_2, \pi)$ in the form [23]:

$$\Phi = [\cdots, s_v(a_u, a_{\pi(u)}), \cdots, s_e(a_{uv}, a_{\pi(u)\pi(v)}) \cdots]^\top$$

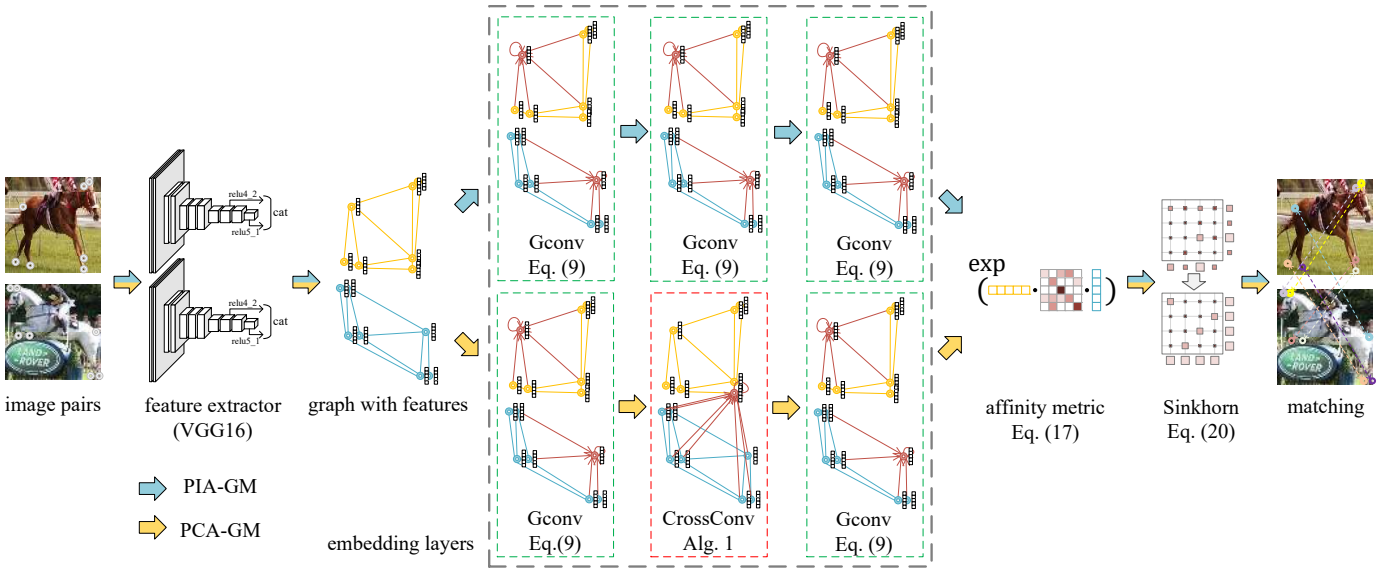
where π is the node mapping function from one graph to the other, and s_v denotes the similarity value between node u and its mapping in the other graph $\pi(v)$, which are with attributes (or features) a_u and $a_{\pi(v)}$ respectively. Similar notation holds for the edge similarity s_e between edge a_{uv} and $a_{\pi(u)\pi(v)}$. One can compute the accumulated similarity by re-weighting and adding up all the elements in $\Phi(\mathcal{G}_i, \mathcal{G}_j, \pi)$ with weighting column vector β, so that $S(\mathcal{G}_i, \mathcal{G}_j, \pi, \beta) = \beta^\top \Phi(\mathcal{G}_i, \mathcal{G}_j, \pi)$. As observed in [23], the above simple model can incorporate most previous shallow learning models [37], [38], [39]. Specifically, [37] uses a 60-dimensional model s_v for feature points and a binary similarity function s_e for edges. In [38], a multi-dimensional s_e is adopted to measure similarity while s_v is ignored. In [39], 2-dimensional s_v and s_e are devised to model appearance similarity, occlusion, and geometric compatibility.

Differing from the above non-deep learning models, the seminal work [24] shows how to integrate the feature extraction, affinity computing, spectral matching components into an end-to-end learnable pipeline via deep neural networks. The graph embedding is not considered in their approach and a regression based offset loss is used. In particular, we argue that a combinatorial loss can be a better choice which has been devised in the paper.

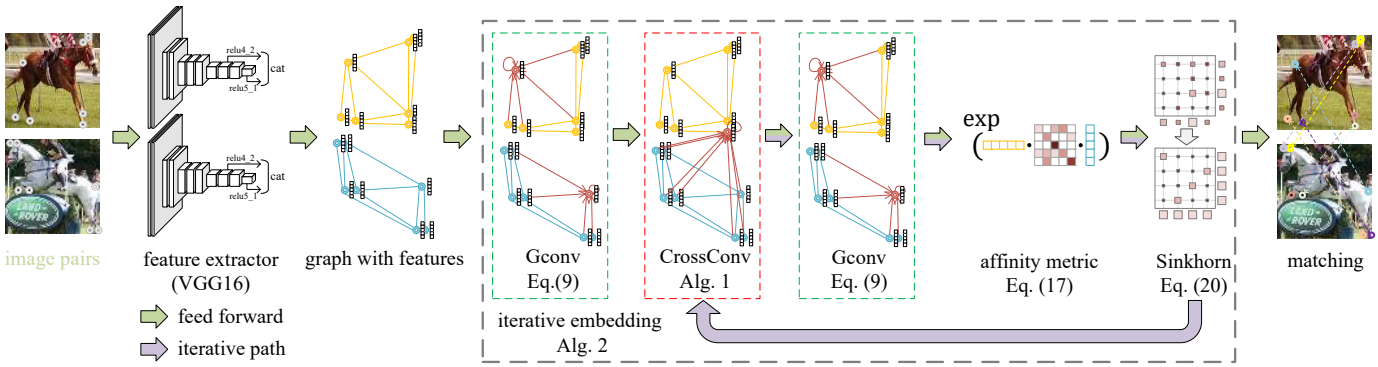
2.3 Graph Neural Networks and Embedding

Node embedding has recently been an active research area whose purpose is to embed the network structure information around the node into a vectorized feature. As such,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.3005590, IEEE Transactions on Pattern Analysis and Machine Intelligence



(a) Vanilla feed-forward methods PIA-GM and PCA-GM.



(b) Iterative cross-graph embedding method IPCA-GM.

Fig. 1. Overview of our (a) permutation loss based intra-graph affinity (**PIA-GM**), cross-graph affinity (**PCA-GM**) and (b) iterative cross-graph affinity (**IPCA-GM**) approaches for deep combinatorial learning of graph matching. The CNN features are extracted from image pairs followed by node embedding and Sinkhorn operation for matching. The CNN model, embedding model and affinity metric are all learnable in an end-to-end fashion.

traditional classification and regression based techniques can be readily reused on the graph-like data. One common technique refers to the so-called graph neural networks (GNN) [40]. In a GNN, node features are aggregated from its neighbors and often a same transfer function is shared among different nodes. As such, the output of GNN is invariant to permutations of graph elements. There are many followup variants such as the SNDE model [41] that is developed for deep node embedding by exploiting the first-order and second-order proximity jointly. Conversely there are some shallow embedding models which mostly only consider the structure rather than attribute of nodes and edges, including node2vec [42] inspired by skip-gram language model [43], DeepWalk [44] based on random walk, as well as LINE [45] which explicitly defines *first-order* proximity and *second-order* proximity and builds the corresponding heuristics models. These shallow models can be more scalable on large networks compared SNDE. Nevertheless, all these models cannot be directly used for end-to-end training in graph matching, neither the embedding model is designated for matching which calls for particular discrimination among nodes.

In this paper, we adopt the graph convolutional network (GCN) [46] modeling graph structure whose parameters are learnable in an end-to-end fashion. Moreover, its output is also invariant to the permutation of graph elements. Also, the model can allow for different numbers of nodes for each graph for training.

2.4 Learning of Combinatorial Optimization

There are emerging works on using deep neural networks to solve the combinatorial problems. One advantage is that the learned model is expected to better capture the specific structure of the problem at hand, instead of a general solver that is not tailored to the dataset. Moreover, the deep networks can be computational friendly to GPUs, leading to efficiency in solving optimization problem compared with pure CPU based pipeline. For instance, the NP-hard Travelling Salesman Problem (TSP) problem is explored in [47] via graph attention network based method to find a tour. In [48], another classic NP-hard graph coloring problem is addressed via deep reinforcement learning, which uncovers

new and effective heuristics for graph coloring. In [49] a deep leaning based approach is developed which involves permutation invariant objective functions to a set of nodes.

Recall graph matching bears the combinatorial nature and in general can be formulated as a quadratic assignment problem (QAP). In [50], a QAP solver is learned given predefined affinity matrix. While in our approach, the affinity model is part of the learning components hence the work [50] can be complementary to ours. On the other hand, as many traditional methods transform the QAP problem into linear assignment problem (LAP) in each iteration, learning of linear assignment can be of relevance. In particular, it is know that Sinkhorn algorithm [51] is the approximate and differentiable version of Hungarian algorithm [52]. Given predefined assignment cost, the LAP solver is learnt by the Sinkhorn Network in [53], whereby doubly-stochastic regulation is added on non-negative square matrix. Similarly, the Sinkhorn AutoEncoder [54] is devised to minimize Wasserstein distance in AutoEncoders. Reinforcement learning is adopted in [55] for learning a linear assignment solver. In DeepPermNet [56], the Sinkhorn layer is employed with a deep convolutional network, to solve a permutation prediction problem. However, DeepPermNet predicts a permutation matrix by fully-connected layers and is therefore not invariant to input permutation. Hence DeepPermNet need a predefined node permutation as reference which is unnatural for graph matching. Moreover, DeepPermNet is incapable to handle varying number of input nodes.

Our approach differs from the above methods in several ways. First our model computes node-wise similarity directly between two graphs, and the output of our network is invariant to input node permutations. Meanwhile, our model involves an affinity learning module to encode the structure affinity into node-wise embeddings. By doing so, graph matching is transformed into a linear assignment task which can be readily solved by the Sinkhorn layer, which can also be called permutation learning.

3 Proposed Approach

3.1 Notations

From dataset $\mathcal{D}$, we consider the matching between graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$, where the number of nodes $|\mathcal{V}_1| = N_1$, $|\mathcal{V}_2| = N_2$. Without loss of generality, we assume $N_1 \leq N_2$ for simplified notations. Graphs are indexed by subscript s and nodes are indexed by subscript i in the following content. The connectivity of graph s is represented by adjacency matrix $\mathbf{A}_s$.

In particular, for graphs built on images as mainly assumed in the paper, we note each node to a 2D pixel coordinate in image I_s denoted as $P_{si} = (x, y)$. While $\mathbf{h}_{si}^{(k)}$ represent the node embedding vector at keypoint i, layer k in graph s. $\mathbf{M} \in \mathbb{R}^{+N_1 \times N_2}$ is a non-negative affinity matrix encoding node-node similarity between two graphs. $\mathbf{S} \in [0,1]^{N_1 \times N_2}$ is a so-called doubly (sub-)stochastic matrix, which satisfies $\mathbf{S1} = \mathbf{1}$ and $\mathbf{S}^\top \mathbf{1} \leq \mathbf{1}$. $\mathbf{S}$ is denoted as the soft matching result of our model. Node-to-node matching is also represented by $\mathbf{X} \in \{0,1\}^{N_1 \times N_2}$, which is a discrete permutation matrix s.t. $\mathbf{X1} = \mathbf{1}, \mathbf{X}^\top \mathbf{1} \leq \mathbf{1}$.

3.2 Approach Overview

We present three deep graph matching methods: i) **p**ermutation loss and **i**ntra-graph **a**ffinity based **g**raph **m**atching (**PIA-GM**), ii) **p**ermutation loss and **c**ross-graph **a**ffinity **g**raph **m**atching model (**PCA-GM**) and iii) **i**terative **p**ermutation loss and **c**ross-graph **a**ffinity based one (**IPCA-GM**). All models are built upon a deep network where both image feature and structure information are exploited, and a Sinkhorn network predicting permutation in a differentiable fashion allowing for gradient back propagation. Among them, PIA-GM and PCA-GM are vanilla feed-forward embedding networks where PCA-GM adopts an extra cross-graph component which aggregates cross-graph features. IPCA-GM further exploits an iterative update scheme for cross-graph aggregation. Figure 1(a) summarizes vanilla feed-forward methods PIA-GM and PCA-GM and Figure 1(b) shows the iterative update scheme IPCA-GM.

The proposed models are built on a CNN feature extractor, a graph embedding component, an affinity metric and a permutation prediction layer. From the given node (i.e. keypoint) positions, graph structures are built by Delaunay triangulation. CNN (VGG16 in our model) extracts image features as graph nodes, and they are aggregated through intra-graph and cross-graph embedding layers. The network predicts a permutation for node-to-node correspondence from raw pixel inputs.

3.3 Node Feature Extraction on Images

A CNN architecture is adopted to extract features from raw RGB images, where node features are constructed by bilinear interpolation on CNN's feature map. The extracted feature on the keypoint P_{si} of image I_s is:

$$\mathbf{h}_{si}^{(0)} = \mathrm{Interp}(P_{si}, \mathrm{CNN}(I_s)) \tag{5}$$

where $\mathrm{Interp}(P, X)$ bi-linearly interpolates on point P from feature map X. $\mathrm{CNN}(I)$ feeds image I into a CNN and outputs a feature tensor. Inspired by Siamese Network [57], the same CNN structure and weights are shared between two input images. We extract feature vectors from CNN layers at different depth for fusing both local texture and global semantic features. For fair comparison with [24], VGG16 pretrained with ImageNet on the classification task [58] is adopted as the CNN backbone.

3.4 Intra-graph Node Embedding

Researchers have shown that graph matching methods exploiting graph structure can reach a more robust matching [9] against point based methods [6], [7]. In PIA-GM, graph affinity is embedded by multiple embedding layers encoding higher-order information in graphs. We develop an intra-graph message passing scheme based on the popular embedding approach GCN [46], where features are effectively aggregated from adjacency nodes including a self-update path for node features:

$$\mathbf{m}_{si}^{(k)} = \frac{1}{|(i,j) \in \mathcal{E}_s|} \sum_{j:(i,j)\in\mathcal{E}_s} f_{msg}(\mathbf{h}_{sj}^{(k-1)}) \tag{6}$$

$$\mathbf{n}_{si}^{(k)} = f_{node}(\mathbf{h}_{si}^{(k-1)}) \tag{7}$$

$$\mathbf{h}_{si}^{(k)} = f_{update}(\mathbf{m}_{si}^{(k)}, \mathbf{n}_{si}^{(k)}) \tag{8}$$

Algorithm 1: Vanilla cross-graph node embedding (CrossEmb)

Input: $(k-1)$-th layer features $\{\mathbf{h}_{1i}^{(k-1)}, \mathbf{h}_{2j}^{(k-1)}\}_{i\in\mathcal{V}_1, j\in\mathcal{V}_2}$

// similarity prediction Eqs. 17, 20

build $\hat{\mathbf{M}}$ from $\{\mathbf{h}_{1i}^{(k-1)}, \mathbf{h}_{2j}^{(k-1)}\}$ by Eq. 17;

$\hat{\mathbf{S}} \leftarrow \text{Sinkhorn}(\hat{\mathbf{M}})$;

// cross-graph aggregation Eqs. 12, 13, 14

$\{\mathbf{h}_{1i}^{(k)}\} \leftarrow \text{CrossConv}(\hat{\mathbf{S}}, \{\mathbf{h}_{1i}^{(k-1)}\}_{i\in\mathcal{V}_1}, \{\mathbf{h}_{2j}^{(k-1)}\}_{j\in\mathcal{V}_2})$;

$\{\mathbf{h}_{2j}^{(k)}\} \leftarrow \text{CrossConv}(\hat{\mathbf{S}}^\top, \{\mathbf{h}_{2j}^{(k-1)}\}_{j\in\mathcal{V}_2}, \{\mathbf{h}_{1i}^{(k-1)}\}_{i\in\mathcal{V}_1})$;

Output: k-th layer features $\{\mathbf{h}_{1i}^{(k)}, \mathbf{h}_{2j}^{(k)}\}_{i\in\mathcal{V}_1, j\in\mathcal{V}_2}$

Here Eq. 6 updates feature along edges and f_{msg} denotes the message passing function. As a common practice, we normalize the aggregated features from adjacent nodes by the total number of adjacent nodes to avoid bias brought by varying numbers of neighbors owned by different nodes. Eq. 7 updates feature for each node via a self-passing function f_{node}. With f_{update}, Eq. 8 accumulates information from adjacent nodes and the node itself to update the state of node i. f_{msg}, f_{node}, f_{update} may take any differentiable form in implementation. In our models, we design single-layer neural networks with ReLU [59] activation for f_{msg}, f_{node}, and summation for f_{update}. Note Eqs. 6, 7, 8 are denoted as graph convolution (GConv) from layer $k-1$ to layer k:

$$\{\mathbf{h}_{si}^{(k)}\} = \text{GConv}(\mathbf{A}_s, \{\mathbf{h}_{si}^{(k-1)}\}), \quad i \in \mathcal{V}_s \tag{9}$$

Eq. 9 represents one node embedding layer. The connectivity of graph s is encoded by adjacency matrix $\mathbf{A}_s \in \{0,1\}^{N\times N}$. $\mathbf{h}_{si}^{(0)}$ is initialized by the CNN feature of node i in graph s.

3.5 Cross-graph Node Embedding

We improve from intra-graph embedding by a cross-graph aggregation step, where features are aggregated from the other graph with nodes containing similar features. The hope is to better fuse the information between two relevant graphs via joint embedding. First, we compute similarity based on graph features from shallower embedding layers and utilize a Sinkhorn network to predict a doubly-stochastic similarity matrix (see details in Section 3.7 and Section 3.8). The predicted cross-graph similarity matrix $\hat{\mathbf{S}}$ encodes the node-to-node similarity between two graphs.

$$\hat{\mathbf{M}}_{i,j} = f_{aff}(\mathbf{h}_{1i}^{(1)}, \mathbf{h}_{2j}^{(1)}), \quad i \in \mathcal{V}_1, j \in \mathcal{V}_2 \tag{10}$$

$$\hat{\mathbf{S}} = \text{Sinkhorn}(\hat{\mathbf{M}}) \tag{11}$$

where f_{aff} represents the affinity measure (see Eq. 17), and the superscript (1) means $\mathbf{h}_{1i}^{(1)}, \mathbf{h}_{2j}^{(1)}$ are from the output of first graph convolution layer.

We adopt a similar message passing scheme from intra-graph convolution introduced in Eq. 9, while the adjacency matrix is replaced by cross-graph similarity matrix $\hat{\mathbf{S}}$, and features are aggregated across two graphs. Note that $\hat{\mathbf{S}}$ is doubly-stochastic therefore we need no normalization for cross-graph aggregation. The cross-graph embedding is:

$$\mathbf{m}_{1i}^{(k)} = \sum_{j\in\mathcal{V}_2} \hat{\mathbf{S}}_{i,j} f_{msg\text{-}cross}(\mathbf{h}_{2j}^{(k-1)}) \tag{12}$$

$$\mathbf{n}_{1i}^{(k)} = f_{node\text{-}cross}(\mathbf{h}_{1i}^{(k-1)}) \tag{13}$$

$$\mathbf{h}_{1i}^{(k)} = f_{update\text{-}cross}(\mathbf{m}_{1i}^{(k)}, \mathbf{n}_{1i}^{(k)}) \tag{14}$$

Algorithm 2: Iterative cross-graph node embedding (IterCrossEmb)

Input: CNN features $\{\mathbf{h}_{1i}^{(0)}, \mathbf{h}_{2j}^{(0)}\}_{i\in\mathcal{V}_1, j\in\mathcal{V}_2}$; number of iterations K

// first intra-graph aggregation Eq. 6, 7, 8

$\{\mathbf{h}_{si}^{(1)}\} \leftarrow \text{GConv}_1(\mathbf{A}_s, \{\mathbf{h}_{si}^{(0)}\})$;

// Initialize $\hat{\mathbf{S}}^{(0)}$ as zero matrix

$\hat{\mathbf{S}}^{(0)} \leftarrow \mathbf{0}^{N\times N}$;

for $k \leftarrow \{1..K\}$ **do**

  // cross-graph aggregation Eq. 12, 13, 14

  $\{\mathbf{h}_{1i}^{(2)}\} \leftarrow \text{CrossConv}(\hat{\mathbf{S}}^{(k-1)}\{\mathbf{h}_{1i}^{(1)}\}, \{\mathbf{h}_{2j}^{(1)}\})$;

  $\{\mathbf{h}_{2j}^{(2)}\} \leftarrow \text{CrossConv}(\hat{\mathbf{S}}^{(k-1)\top}, \{\mathbf{h}_{2j}^{(1)}\}, \{\mathbf{h}_{1i}^{(1)}\})$;

  // second intra-graph aggregation Eq. 6, 7, 8

  $\{\mathbf{h}_{si}^{(3)}\} \leftarrow \text{GConv}_2(\mathbf{A}_s, \{\mathbf{h}_{si}^{(2)}\})$;

  // correspondence prediction Eq. 17, 20

  build $\hat{\mathbf{M}}$ from $\{\mathbf{h}_{1i}^{(3)}\}, \{\mathbf{h}_{2j}^{(3)}\}$ by Eq. 17;

  $\hat{\mathbf{S}}^{(k)} \leftarrow \text{Sinkhorn}(\hat{\mathbf{M}})$;

Output: embedding features $\{\mathbf{h}_{1i}^{(3)}, \mathbf{h}_{2j}^{(3)}\}_{i\in\mathcal{V}_1, j\in\mathcal{V}_2}$

where $f_{msg\text{-}cross}, f_{node\text{-}cross}$ are implemented as identity mapping, $f_{update\text{-}cross}$ is a concatenation of two input feature tensors followed by a linear layer. Introducing learnable parameters to $f_{msg\text{-}cross}, f_{node\text{-}cross}$ may result in training instability, as witnessed in experiments. For graph pairs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1), \mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$, the cross-graph aggregation scheme can be denoted as:

$$\begin{aligned}\{\mathbf{h}_{1i}^{(k+1)}\}_{i\in\mathcal{V}_1} &= \text{CrossConv}(\hat{\mathbf{S}}, \{\mathbf{h}_{1i}^{(k)}\}_{i\in\mathcal{V}_1}, \{\mathbf{h}_{2j}^{(k)}\}_{j\in\mathcal{V}_2}) \\ \{\mathbf{h}_{2j}^{(k+1)}\}_{j\in\mathcal{V}_2} &= \text{CrossConv}(\hat{\mathbf{S}}^\top, \{\mathbf{h}_{2j}^{(k)}\}_{j\in\mathcal{V}_2}, \{\mathbf{h}_{1i}^{(k)}\}_{i\in\mathcal{V}_1})\end{aligned} \tag{15}$$

where $\hat{\mathbf{S}}$ denotes the predicted correspondence from $\mathcal{G}_2$ to $\mathcal{G}_1$ and $\hat{\mathbf{S}}^\top$ denotes such relation from $\mathcal{G}_1$ to $\mathcal{G}_2$. $\hat{\mathbf{S}}$ can be regarded as an early-stage prediction on matching. We summarize this (vanilla) cross-graph embedding approach as $\text{CrossEmb}(\cdot)$ in Algorithm 1.

3.6 Iterative Cross-graph Node Embedding

The cross-graph embedding scheme introduced in Algorithm 1 considers the cross-graph matching relationship at shallower embedding layers, which is proved to advantage matching by experiments in [22]. However, Algorithm 1 only computes the matching by single forward pass, which is relatively simple and its prediction can be further improved. With the motivation that more precise cross-graph prediction will probably lead to better embedding feature and vice versa, we design and experiment an iterative update approach for a more accurate prediction of cross-graph similarity, where the cross-graph similarity matrix $\hat{\mathbf{S}}$ is predicted iteratively. The embedding layers (including intra-graph embedding layers) in our iterative cross-graph embedding design is summarized by $\text{IterCrossEmb}(\cdot)$ in Algorithm 2. In this alternative design, $\hat{\mathbf{S}}^{(0)}$ is initialized as zero matrix, and we iteratively predict $\hat{\mathbf{S}}^{(k)}$ from $\hat{\mathbf{S}}^{(k-1)}$ through both cross-graph embedding (L6 - L8) and intra-graph embedding (L10) layers. We predict a similarity matrix from embedding outputs (L12), and finally calculate a doubly-stochastic matching matrix $\hat{\mathbf{S}}$ by Sinkhorn algorithm

(L13). Note that the predicted matching matrix $\hat{\mathbf{S}}$ is regarded as the weighting matrix of cross-graph update paths. As will be shown in the experiment part, further improvement can be brought by this iterative cross-graph design.

3.7 Affinity Metric Learning

With the proposed embedding model, we encode the structural information between two graphs into node-to-node similarity in the embedding space. Such an embedding scheme allows to simplify the traditional second-order affinity matrix $\mathbf{K}$ in Eq. 1 into a linear one. Then the NP-hard problem in quadratic form is reduced into a linear one which is solvable in polynomial time. Let $\mathbf{h}_{1i}$ and $\mathbf{h}_{2j}$ be the feature of i and j from the first graph and the second graph, respectively:

$$\mathbf{M}_{i,j} = f_{\mathit{aff}}(\mathbf{h}_{1i}, \mathbf{h}_{2j}), \quad i \in \mathcal{V}_1, j \in \mathcal{V}_2 \tag{16}$$

The node-wise affinity matrix $\mathbf{M} \in \mathbb{R}^{+^{N_1 \times N_2}}$ encodes any node-to-node similarity between two graphs. $\mathbf{M}_{i,j}$ corresponds to the affinity score between node i in the first graph and node j in the second graph, considering node features and higher-order information inside and across graphs.

We assign f_{aff} as a weighted bi-linear function followed by an exponential function forcing all elements in the affinity matrix being positive[2].

$$f_{\mathit{aff}}(\mathbf{h}_{1i}, \mathbf{h}_{2j}) = \exp\left(\frac{\mathbf{h}_{1i}^{\top}\mathbf{A}\mathbf{h}_{2j}}{\tau}\right) \tag{17}$$

If the feature is a m-dimensional vector, i.e. $\forall i \in \mathcal{V}_1, j \in \mathcal{V}_2$ and $\mathbf{h}_{1i}, \mathbf{h}_{2j} \in \mathbb{R}^{m\times 1}$, $\mathbf{A} \in \mathbb{R}^{m\times m}$ contains learnable weights for this affinity function. τ is a regularization parameter controlling the discriminate level of the affinity function. For $\tau > 0$, with $\tau \to 0^+$, Eq. 17 becomes more discriminative, while there will be a higher chance of explosive gradient if τ is too small.

3.8 Sinkhorn Layer for Linear Assignment

With the node-wise affinity matrix in Eq. 17, Sinkhorn method is used for linear assignment, where the discrete assignment constraint is relaxed to doubly-stochastic matrix. Sinkhorn layer takes any non-negative square matrix as input and outputs a doubly-stochastic matrix as the predicted matching result. The doubly-stochastic matrix is considered as a continuous relaxation of permutation matrix. Sinkhorn networks have been shown effective in network based permutation prediction [53], [56]. We initialize $\mathbf{M}^{(0)} = \mathbf{M}$. For $\mathbf{M}^{(k-1)} \in \mathbb{R}^{+^{N\times N}}$, the Sinkhorn operator is:

$$\mathbf{M}^{(k)\prime} = \mathbf{M}^{(k-1)} \oslash (\mathbf{M}^{(k-1)}\mathbf{1}\mathbf{1}^{\top}) \tag{18}$$

$$\mathbf{M}^{(k)} = \mathbf{M}^{(k)\prime} \oslash (\mathbf{1}\mathbf{1}^{\top}\mathbf{M}^{(k)\prime}) \tag{19}$$

where $\oslash$ means element-wise division, and $\mathbf{1}$ is a column vector whose elements are all ones. Sinkhorn algorithm involves alternatively taking row-normalization (Eq. 18) and column-normalization (Eq. 19) till convergence. It is found about 10 iterations often leads to a satisfying result.

After passing the linear affinity matrix into Sinkhorn network we get a doubly-stochastic matrix $\mathbf{S}$, which is treated as our model's prediction in training. The Sinkhorn layer is summarized as:

$$\mathbf{S} = \mathrm{Sinkhorn}(\mathbf{M}) \tag{20}$$

If two graphs are of equal size i.e. $N_1 = N_2 = N$, then $M \in \mathbb{R}^{+^{N\times N}}$, the Sinkhorn algorithm works straight forward. If there exist outliers in graph 2, i.e. $N_1 < N_2$, $M \in \mathbb{R}^{+^{N_1\times N_2}}$ is no longer a square matrix and cannot be directly handled by Sinkhorn network. Under such circumstance, we add dummy nodes to $\mathcal{G}_1$ by padding M by zeros into a $N_2 \times N_2$ square matrix. After the Sinkhorn step, the padded zero elements are discarded in the resulting soft matching matrix $\mathbf{S} \in [0,1]^{N_1\times N_2}$. We further apply loss metric on it. Our treatment on outliers is summarized in L21 of Algorithm 3. Adopting dummy nodes is a common practice when matching graphs against outliers [1]. It is also a standard technique in linear programming.

Note Sinkhorn net is differentiable as its operation only involves matrix-vector multiplication and element-wise division, making it appealing to end-to-end pipelines. The backward gradient of Sinkhorn layer derived from [24] is:

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{M}^{(k)\prime}} =& \frac{\partial L}{\partial \mathbf{M}^{(k)}}[\mathbf{1}^{\top}\mathbf{M}^{(k)}]^{-1} \\ &- \mathbf{1}\mathrm{diag}\left([\mathbf{1}^{\top}\mathbf{M}^{(k)}]^{-2}\mathbf{M}^{(k)\top}\frac{\partial L}{\partial \mathbf{M}^{(k)}}\right)^{\top}\end{aligned} \tag{21}$$

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{M}^{(k-1)}} =& \frac{\partial L}{\partial \mathbf{M}^{(k)\prime}}[\mathbf{M}^{(k)\prime}\mathbf{1}]^{-1} \\ &- \mathrm{diag}\left([\mathbf{M}^{(k)\prime}\mathbf{1}]^{-2}\frac{\partial L}{\partial \mathbf{M}^{(k)\prime}}\mathbf{M}^{(k)\prime\top}\right)\mathbf{1}^{\top}\end{aligned} \tag{22}$$

where $[\,\cdot\,]$ builds a diagonal matrix from the given vector. While Sinkhorn can be efficiently implemented with automatic differentiation available in PyTorch [60]. During testing, we perform Hungarian algorithm [61] on $\mathbf{S}$ as a final discretization step to obtain a binary permutation matrix $\mathbf{X}$.

$$\mathbf{X} = \mathrm{Hungarian}(\mathbf{S}) \tag{23}$$

3.9 Permutation Cross-Entropy Loss

Our methods directly utilize node-to-node matching labels, i.e. permutation matrix, as the supervision for end-to-end training. As the matching result computed via Sinkhorn layer in Eq. 20 is a doubly-stochastic matrix, we propose a linear assignment based permutation loss to evaluate the difference between predicted doubly-stochastic matrix and ground truth permutation matrix for training.

Our models are trained end-to-end based on cross entropy loss. We take the ground truth permutation matrix $\mathbf{X}^{gt}$ provided by the dataset, and compute the cross entropy between $\mathbf{S}$ and $\mathbf{X}^{gt}$. We denote it as permutation loss, which is the supervision adopted to train most of our deep graph matching models L_{perm}:

$$-\sum_{i\in\mathcal{V}_1, j\in\mathcal{V}_2}\left(\mathbf{X}^{gt}_{i,j}\log\mathbf{S}_{i,j} + (1-\mathbf{X}^{gt}_{i,j})\log(1-\mathbf{S}_{i,j})\right) \tag{24}$$

2. We have also experimented other more flexible fully-connected and attention-like layers, while we empirically find the simple exponential function is more stable for learning.

Algorithm 3: Permutation and intra/cross/iterative-cross graph affinity learning for graph matching (PIA-GM/PCA-GM/IPCA-GM)

Input: Graph pairs from $\mathcal{D}$ with node correspondence; initial model weights $\mathbf{W}$; learning rate lr; number of cross-graph iterations K.

repeat
  // Draw image pair and ground truth from dataset, $s = 1, 2$ denoting two input images
  $\{(I_s, \{P_{si}\}_{i\in\mathcal{V}_s}, \mathbf{A}_s)|s = 1, 2\}, \mathbf{X}^{gt} \in \mathcal{D}$;
  // extract CNN features Eq. 5
  $\{h_{si}^{(0)}\} \leftarrow \mathrm{Interp}(\{P_{si}\}, \mathrm{CNN}(I_s))$;
  if *PIA-GM* **then**
    // intra-graph aggregation Eqs. 6, 7, 8
    for $k \leftarrow \{1, 2, 3\}$ **do**
      $\{\mathbf{h}_{si}^{(k)}\} \leftarrow \mathrm{GConv}_k(\mathbf{A}_s, \{\mathbf{h}_{si}^{(k-1)}\})$;
  if *PCA-GM* **then**
    // first intra-graph aggregation Eqs. 6, 7, 8
    $\{\mathbf{h}_{si}^{(1)}\} \leftarrow \mathrm{GConv}_1(\mathbf{A}_s, \{\mathbf{h}_{si}^{(0)}\})$;
    // cross-graph embedding Alg. 1
    $\{\mathbf{h}_{1i}^{(2)}, \mathbf{h}_{2j}^{(2)}\} \leftarrow \mathrm{CrossEmb}(\{\mathbf{h}_{1i}^{(1)}, \mathbf{h}_{2j}^{(1)}\})$;
    // second intra-graph aggregation Eqs. 6, 7, 8
    $\{\mathbf{h}_{si}^{(3)}\} \leftarrow \mathrm{GConv}_2(\mathbf{A}_s, \{\mathbf{h}_{si}^{(2)}\})$;
  if *IPCA-GM* **then**
    // iterative cross-graph embedding Alg. 2
    $\{\mathbf{h}_{1i}^{(3)}, \mathbf{h}_{2j}^{(3)}\} \leftarrow \mathrm{IterCrossEmb}(\{\mathbf{h}_{1i}^{(0)}, \mathbf{h}_{2j}^{(0)}\}, K)$
  // build $\mathbf{M} \in \mathbb{R}^{+N_2\times N_2}$ from $\{\mathbf{h}_{1i}^{(3)}\}, \{\mathbf{h}_{2j}^{(3)}\}$ Eq. 17
  $$\mathbf{M}_{i,j} = \begin{cases} f_{aff}(\mathbf{h}_{1i}, \mathbf{h}_{2j}), & \text{if } 1 \le i \le N_1 \\ 0, & \text{if } N_1 < i \le N_2 \end{cases};$$
  $\mathbf{S} \leftarrow \mathrm{Sinkhorn}(\mathbf{M})$;
  // end-to-end training based on Eq. 24
  $\mathbf{W} \leftarrow -lr \times \frac{\partial L_{perm}(\mathbf{S}, \mathbf{X}^{gt})}{\partial \mathbf{W}} + \mathbf{W}$;
until *convergence*;
Output: learned model weights $\mathbf{W}$.

Note GMN [24] applies "displacement loss" based on pixel offset. It computes an offset vector $\mathbf{d}$ from all matching candidates by a weighted summation. The offset loss is computed as the difference between predicted offset vector and ground truth offset vector.

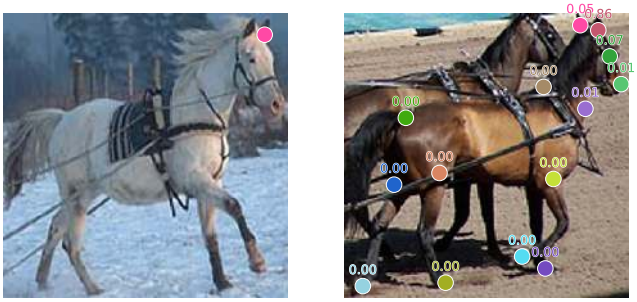
$$\mathbf{d}_i = \sum_{j\in V_2} (\mathbf{S}_{i,j} P_{2j}) - P_{1i} \tag{25}$$

$$L_{off} = \sum_{i\in V_1} \sqrt{||\mathbf{d}_i - \mathbf{d}_i^{gt}||^2 + \epsilon} \tag{26}$$

where $\{P_{1i}\}, \{P_{2j}\}$ are the keypoint coordinates in $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively. While ϵ is a small value added for numerical stability. In comparison, our cross entropy loss can directly learn a linear assignment cost based permutation loss in an end-to-end fashion.

3.10 Further Discussion

Our proposed deep graph matching methods PIA-GM, PCA-GM and IPCA-GM are summarized in Algorithm 3. Here we further discuss our network design including embedding layers, Sinkhorn network and permutation loss compared to peer methods.



$L_{perm} = 5.139, \quad L_{off} = 0.070$

Fig. 2. A failure case of offset loss: source image with keypoint (left) and target image with matching candidates (right), where numbers denote the probability of predicted matching. Ground truth matching nodes are colored in rose (only receives 0.05 probability by this poor prediction, where the model mistakenly match the right ear to the left ear). Offset loss is computed by a weighted sum among all candidates, resulting in a misleading low loss 0.070. In this example offset loss fails to distinguish between left/right ears and such information will not be learned under its supervision. Our permutation loss, on the contrary, issues a reasonably high loss 5.139. The underlying rationale is that the problem at hand is fundamentally a combinatorial problem rather than a regression task.

3.10.1 Predefined affinity vs. embedding

Existing algorithms in graph matching aims to model second-order [1], [26] and higher-order [3], [5] graph affinity feature with an explicitly pre-defined affinity matrix or tensor. An $N_1N_2 \times N_1N_2$ affinity matrix can be adopted to encode node-wise and edge-wise affinity information. Optimization techniques are applied to compute the matching result via graph affinity maximization.

In contrast, we resort to the node embedding technique with two merits. First, the space occupation of the affinity matrix is reduced to $N_1 \times N_2$. Second, the embedding layers can implicitly model higher-order feature in graphs, while only the second-order edge information can be encoded by the affinity matrix $\mathbf{K}$ in Eq. 1.

3.10.2 Sinkhorn net vs. spectral matching

Spectral matching (SM) [26] is adopted by GMN [24]. The SM solver actually computes the leading eigenvector of $\mathbf{K}$ and is capable for back propagation. In contrast, we adopt the Sinkhorn net instead. In fact, the input of Sinkhorn is of complexity $O(N_1N_2)$ while the input size is of $O(N_1^2N_2^2)$ for spectral matching. Meanwhile, it is observed that SM takes more iterations to converge. We empirically observe there exists an optimal value for the number of iterations, as heavy iteration may bring negative effect to the accuracy of back propagation. In fact, spectral matching is designed for graph matching while Sinkhorn net is for linear assignment, which is relaxed from NP-hard graph matching owing to the embedding layers.

3.10.3 Pixel offset loss vs. permutation loss

The method GMN [24] adopts a pixel offset loss function named "displacement loss". The loss firstly computes a weighted sum from all matching candidates, and then compute the so-called offset vector from the source image ($\mathcal{G}_1$) to the target image ($\mathcal{G}_2$). Under the supervision of offset loss, GMN learns to minimize the difference between

predicted offset vector and ground truth offset vector on training samples. In comparison, based on the Sinkhorn net, a permutation loss is computed where the cross entropy between predicted matching and ground truth matching is evaluated. Such permutation loss directly takes the ground truth matching relationship as supervision, and utilize such information for end-to-end training. The performance gap between offset loss and permutation loss becomes more significant with the existence of outliers, as will be shown in the experiments. Permutation loss models the combinatorial nature of graph matching problems.

Figure 2 presents a real failure case produced by using the offset loss. In this case, the model makes a poor prediction, but the offset loss is unreasonably low. In contrast, the permutation loss provides correct information. Experiments also reveal that our permutation loss models surpass models trained with offset in matching accuracy.

4 Experiments

We report the performance on both synthetic data and real-image datasets for semantic point matching. State-of-the-art peer methods including both deep learning [24] and shallow learning [23] are compared in our experiment. We further show that our methods enjoy some robustness against outliers, different datasets and different categories. Detailed results are also provided for further insights on our proposed methods. Experiments are conducted on our workstation with i9-7920X@2.90GHz and quad RTX2080Ti GPU. We also provide a project homepage: http://thinklab.sjtu.edu.cn/IPCA_GM.html.

4.1 Evaluation Metrics and Peer Methods

We evaluate the matching accuracy between two given graphs $\mathcal{G}_1, \mathcal{G}_2$, whose sizes are of N_1 and N_2, respectively. We evaluate the models' performance under two configurations: 1) $N_1 = N_2$ so that there exists no outliers between two graphs. Each node in one graph is labeled to another node in the other graph, i.e. a bijection between $\mathcal{G}_1$ and $\mathcal{G}_2$; and 2) $N_1 \leq N_2$ as we allow outliers exist in $\mathcal{G}_2$. The matching problem becomes more challenging and it is more appealing to real world applications. The model predicts a node-to-node correspondence between two graphs. Such correspondence is represented by a permutation matrix $\mathbf{X}$.

We evaluate the matching accuracy computed from the permutation matrix, by the number of node pairs correctly matched averaged by the total number of ground truth node pairs. For settings with and without outliers, given a predicted permutation matrix $\mathbf{X} \in \{0,1\}^{N_1 \times N_2}$ and a ground truth permutation $\mathbf{X}^{gt} \in \{0,1\}^{N_1 \times N_2}$, the matching accuracy in category $\mathcal{C}$ is computed by:

$$\text{acc} = \frac{\sum_{\mathcal{C}} \langle \mathbf{X}, \mathbf{X}^{gt} \rangle}{\sum_{\mathcal{C}} \langle \mathbf{X}^{gt}, \mathbf{X}^{gt} \rangle} \tag{27}$$

where $\langle \cdot, \cdot \rangle$ denotes inner product and graph pairs are sampled from the given category $\mathcal{C}$.

When matching on graphs from real-world images, each node i of graph s corresponds to a keypoint with position P_{si}. In our experiments, P_{si} is taken from the dataset's ground truth annotation and fed into the network.

The evaluation involves the following peer methods:

GMN. Graph Matching Network (GMN) [24] is the seminal model built on VGG16 [62] network to extract image features. GMN extracts shallower (relu4_2) and deeper (relu5_1) CNN features as node and edge features, respectively. Graph matching is tackled by spectral matching (SM) [26], which is an unlearnable graph matching solver. This model is agnostic to object categories, meaning an universal model is learned for all instance classes. $\mathcal{G}_1$ and $\mathcal{G}_2$ are constructed by Delaunay triangulation and as fully-connected, respectively. GMN is the first end-to-end pipeline for deep graph matching which incorporates deep CNNs. Note there exists a major difference that their loss function is an offset based loss given by Eq. 26. We follow [24] and re-implement GMN with PyTorch as their code is not publicly available at the time when the paper is written. Also, we modify the the raw GMN model by replacing the regression based node position output with the permutation matrix for node matching, in order to make GMN consistent with the graph matching evaluation protocol i.e. matching accuracy.

HARG-SSVM. It is a graph matching learning method based on structured SVM [23]. We adopt it as the baseline for shallow graph matching learning methods. HARG-SSVM is class-specific, where it learns a graph model for each object category. We use the source code released by the authors upon their approval. The experimental setting in [23] assumes that keypoint position of the target graph $\mathcal{G}_2$ is unknown, and they adopt a Hessian detector [63] to produce candidate positions. In our setting, however, all candidate positions are known to the model and we slightly modify the originally released code. From all candidate points found by the Hessian detector, we assign the nearest neighbor from ground truth position as matching candidate. We discovered such practice is originally taken during the training process of HARG-SSVM. In experiments, graphs are constructed with hand-crafted features namely HARG.

PIA/PCA/IPCA-GM. We build our models on VGG16 [62] as the CNN backbone, and features are extracted from relu4_2 and relu5_1 in line with [24]. We concatenate these features to fuse both local and global information. In PIA-GM, we build 3 intra-embedding layers on top of VGG16 while in PCA-GM, the embedding module contains sequentially 1 intra layer, 1 cross layer and 1 intra layer. IPCA-GM is built in line with PCA-GM, while the cross-graph similarity matrix is updated iteratively. All methods contain an affinity mapping as Eq. 17 after the embedding layers. The output feature dimension of all embedding layers is 2048. Models are trained by permutation loss Eq. 24. Two input graphs are both constructed by Delaunay triangulation and we set $\tau = 0.05$ for PIA-GM and PCA-GM, $\tau = 0.005$ for IPCA-GM in Eq. 17. Our models are implemented by PyTorch.

GMN-PL & PIA/PCA-GM-OL. GMN-PL and PIA/PCA-GM-OL are variants from the original GMN [24] and our PIA/PCA-GM, respectively. We build GMN-PL by simply changing the offset loss in GMN to permutation loss. While in PIA/PCA-GM-OL we switch the permutation loss to offset loss, leaving all other components unchanged.

For natural image tests, we randomly draw image pairs from the dataset, and build two graphs from the given

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.3005590, IEEE Transactions on Pattern Analysis and Machine Intelligence

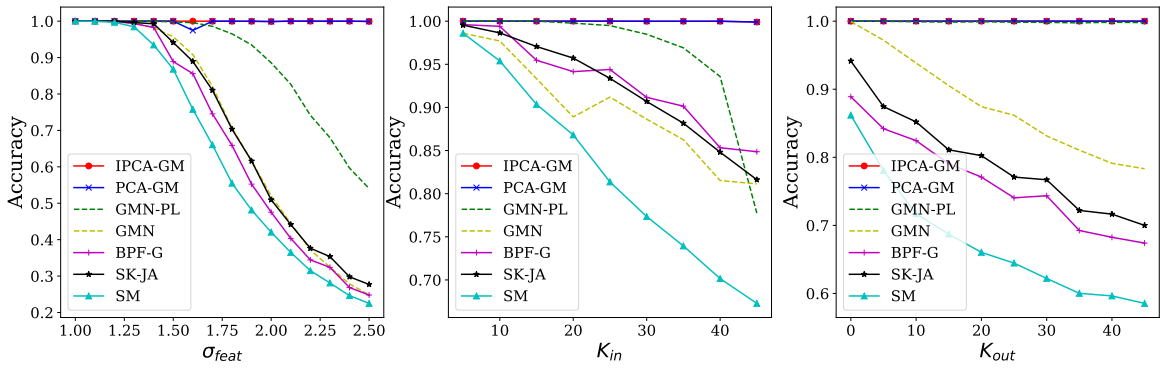


Fig. 3. Performance comparison of different methods on synthetic dataset with noise on feature vectors, inlier numbers and outlier numbers, as well as using different affinity models and losses. Default setting: $K_{in} = 20, K_{out} = 0, \sigma_{feat} = 1.5, \sigma_{coo} = 10$.

TABLE 1
Matching accuracy (%) on Pascal VOC Keypoint, without outliers (white) and with outliers (gray). Note after replacing the offset loss by permutation loss, GMN-PL outperforms GMN [24] almost in all categories. PIA-GM surpasses GMN baselines, while both PCA-GM and IPCA-GM further boost the matching accuracy, by utilizing (vanilla) cross-graph embedding and iterative cross-graph embedding, respectively. Results with white background are obtained without outliers, and the gray background are with outliers. Note that here we report improved results for PCA-GM compared to [22] by changing the hyper-parameter τ from 0.005 to 0.05.

method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
GMN [24]	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-GM	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-GM	**51.2**	61.3	61.6	58.4	78.8	73.9	68.5	71.1	40.1	63.3	45.1	64.4	66.4	62.2	45.1	79.1	68.4	60.0	80.3	**91.9**	64.6
IPCA-GM	51.0	**64.9**	**68.4**	**60.5**	**80.2**	74.7	**71.0**	**73.5**	**42.2**	**68.5**	48.9	**69.3**	67.6	**64.8**	**48.6**	**84.2**	**69.8**	**62.0**	79.3	89.3	**66.9**
GMN [24]	34.2	55.0	46.4	39.6	77.0	60.5	46.9	54.5	31.7	51.0	48.0	48.0	48.5	50.8	28.8	73.8	49.8	38.3	69.4	83.9	51.8
GMN-PL	38.9	56.1	47.9	41.0	79.1	66.5	49.0	57.9	33.7	54.4	43.7	49.5	53.5	55.4	31.2	76.6	53.0	37.8	71.3	86.4	54.1
PIA-GM	43.8	60.6	51.5	43.5	75.4	**70.6**	**58.9**	62.0	35.3	54.4	44.3	57.1	56.1	**58.6**	40.0	76.5	**60.1**	36.5	76.1	86.3	57.4
PCA-GM	**44.6**	63.6	53.7	45.9	78.0	69.5	52.7	63.1	37.6	56.4	44.4	**58.3**	56.2	57.5	39.0	**80.1**	59.6	40.2	69.4	**87.1**	57.8
IPCA-GM	44.5	**63.9**	**54.6**	**47.6**	**79.9**	69.8	54.7	**64.4**	**37.9**	**59.4**	**55.6**	57.5	**57.5**	57.4	**40.2**	**80.1**	60.0	**41.2**	**71.4**	86.9	**59.2**

keypoints. Graph structures are agnostic to the model, and graphs are constructed according to different methods (see discussions above). The CNN model is pretrained on ImageNet [58] classification task with 21,841 subcategories and 14 million images, downloaded from PyTorch model zoo.

4.2 Synthetic Keypoint Matching

We first evaluate models on generated synthetic graphs, where the protocol is built following [1]. Graphs are generated with a given inlier keypoint number K_{in}. Each inlier is assigned with a 1024-dimensional random vector simulating CNN feature (512 dimensions for relu4_2 and 512 dimensions for relu5_1), and a 2D position in $\mathcal{U}(0, 256)$. Keypoint features are under the same Gaussian distribution $\mathcal{N}(\mu_f, \sigma^2_{feat})$, where the Gaussian center μ_f is sampled from uniform distribution: $\mu_f \sim \mathcal{U}(-1, 1)$. During training and testing, we randomly sample keypoint features and blur keypoint coordinates by a random affine transform $\begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix}$ with $s \sim \mathcal{U}(0.8, 1.2), \theta \sim \mathcal{U}(-60, 60), \{t_x, t_y\} \sim \mathcal{U}(-50, 50)$, followed by an additive random noise $\mathcal{N}(0, \sigma^2_{coo})$. For outlier points, their features are randomly sampled from $\mathcal{U}(-1, 1)$ and their positions are from $\mathcal{U}(0, 256)$. Note that we adopt no CNN feature extractor during synthetic experiments, and we compare only graph modeling approaches and loss metrics. The matching accuracy of IPCA-GM, PCA-GM, GMN-PL, GMN and unlearning peer methods SM [26], BPF-G [32], SK-JA [33] are evaluated with respect to K_{in}, σ_{feat} and K_{out}. For each trial, we generate 300 random graphs (200 for training and 100 for testing) from the same distribution. We conduct 10 trials under each setting and report the averaged accuracy. All generated samples are cached and shared among all methods. Figure 3 contains our experimental result showing the robustness of our PCA-GM against feature deformation and complicated graph structure. Note that compared to GMN-PL supervised by permutation loss, the performance of GMN supervised by offset loss degenerates significantly when the number of outlier increases. The learning based methods also surpass learning-free baselines [26], [32], [33].

4.3 Pascal VOC Keypoints Matching

We experiment on large-scale real image matching dataset with Pascal VOC [64] with additional keypoint annotation from [65], namely Pascal VOC Keypoint. It contains 20 classes of instances with ground truth keypoint positions. Following [24], we filter the original dataset into 7,020 training images and 1,682 testing images. Note that the number of training samples grows combinatorially w.r.t. number of images since we can iterate over all combinations of image pairs. During experiment, we randomly draw image pairs from the same category in the dataset. All instances are cropped around its bounding box and resized to 256×256 before passing to the network. We perform experiment

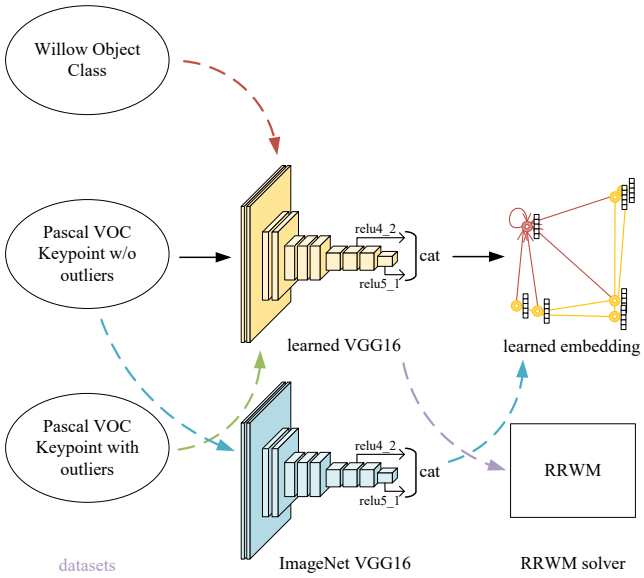
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.3005590, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 2
Matching accuracy (%) for transfer learning test in the presence of outliers on Pascal VOC Keypoint. "Transfer w/o outliers" means directly transfer learned weights from outlier-free setting, and "finetune with outliers" means finetuning the weights with outliers. The last column reports results trained from scratch with outliers. Note '–' means the model fails to *converge* under this setting.

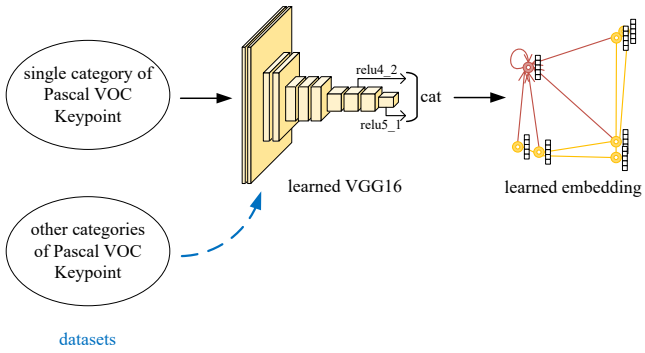
method	transfer w/o outliers	finetune with outliers	training with outliers
GMN [24]	51.7	51.8	51.3
GMN-PL	53.2	54.1	50.2
PIA-GM	55.2	57.4	–
PCA-GM	**56.0**	57.8	**57.6**
IPCA-GM	54.1	**59.2**	–

under two settings on Pascal VOC Keypoint: 1) A simplified setting $N_1 = N_2$ without outliers where we leave only the keypoints co-existent in two images (i.e. inliers) to be matched; 2) A more challenging setting $N_1 \leq N_2$ where $\mathcal{G}_1$ contains only inliers, but outlier keypoints are not filtered in $\mathcal{G}_2$. Problems with $N_1 < 3$ is not considered in both settings since they are relatively easy to solve. We regard Pascal VOC Keypoint a challenging dataset under both setting as objects vary greatly in their pose, texture and illumination, and the number of labeled keypoints ranges from 6 to 23.

We test on Pascal VOC Keypoint [65] and report performance on 20 Pascal categories. We compare GMN, GMN-PL, PIA-GM, PCA-GM, IPCA-GM and present detailed experimental results in Table 1 with (white background) and without (gray background) outliers. Our proposed methods PIA-GM, PCA-GM, IPCA-GM surpass competing methods in most categories, especially the mean accuracy over 20 categories, and they are robust with the existence of outliers (see detailed discussion in Section 4.4.1). With dual RTX2080Ti GPUs, PCA-GM runs at around 18 pairs per second and IPCA-GM runs at about 16 during training. The result shows the superiority of the combinatorial permutation loss over offset loss in training, embedding and Sinkhorn over fixed SM [26] in affinity modeling, and cross-graph embedding, especially iterative cross-graph embedding over intra-graph embedding in the embedding module.



(a) Solid arrows represent the baseline model. Green dashed arrow represents the inlier/outlier transfer setting for Section 4.4.1 and Table 2. Red dashed arrow represents the cross-dataset transfer protocol for Section 4.4.2 and Table 3. Blue and purple arrows stand for generalization study for CNN and embedding modules in Section 4.4.4 and Table 4.



(b) Cross-category learning test in Section 4.4.3 and Figure 5.

Fig. 4. Illustration of transfer learning experiments for our methods.

4.4 Transfer Learning Experiments

As summarized in Figure 4, the robustness and generalization capability of our methods are tested by the following aspects: transfer learning by using outlier-free training data and the testing data from the same dataset (but different samples) added with outliers (Figure 4(a) green), transfer learning across different datasets (Figure 4(a) red), generalization capability when training and testing instances are from different categories (Figure 4(b)) and finally the generalization behavior of learned CNN and embedding modules (Figure 4(a) blue and purple).

4.4.1 *Transfer learning in the presence of outliers*

We experiment the transfer learning capability of our proposed methods against the existence of outliers. As illustrated by the green dashed arrow in Figure 4(a), we firstly pretrain models on Pascal VOC Keypoint without outliers, and directly apply them on the dataset with outliers. These models are later finetuned on Pascal VOC Keypoint with outliers. Experimental result reported in Table 2 shows the superiority of our proposed methods in transfer learning against outliers. Direct learning with outliers seems challenging for IPCA-GM, and one possible reason is the predicted cross-graph similarity matrix may be sub-stochastic since there are outliers (the summations of columns may be less equal to 1), misleading the iterative cross-graph update scheme at early training stage. PIA-GM also fails to converge when directly trained with outliers, which may be caused by its poor model capacity. In contrast, transfer learning from outlier-free models will result in better performance and more stable training. The outlier results in Table 1 are based on transfer learning.

4.4.2 *Cross-dataset transfer with Willow ObjectClass*

Knowledge transfer between different datasets is experimented on Willow ObjectClass dataset, which is a real image matching dataset collected by [23]. It contains 5 instance categories collected from Caltech-256 (face, duck and wine bottle) and Pascal VOC 2007 (car and motorbike), while each

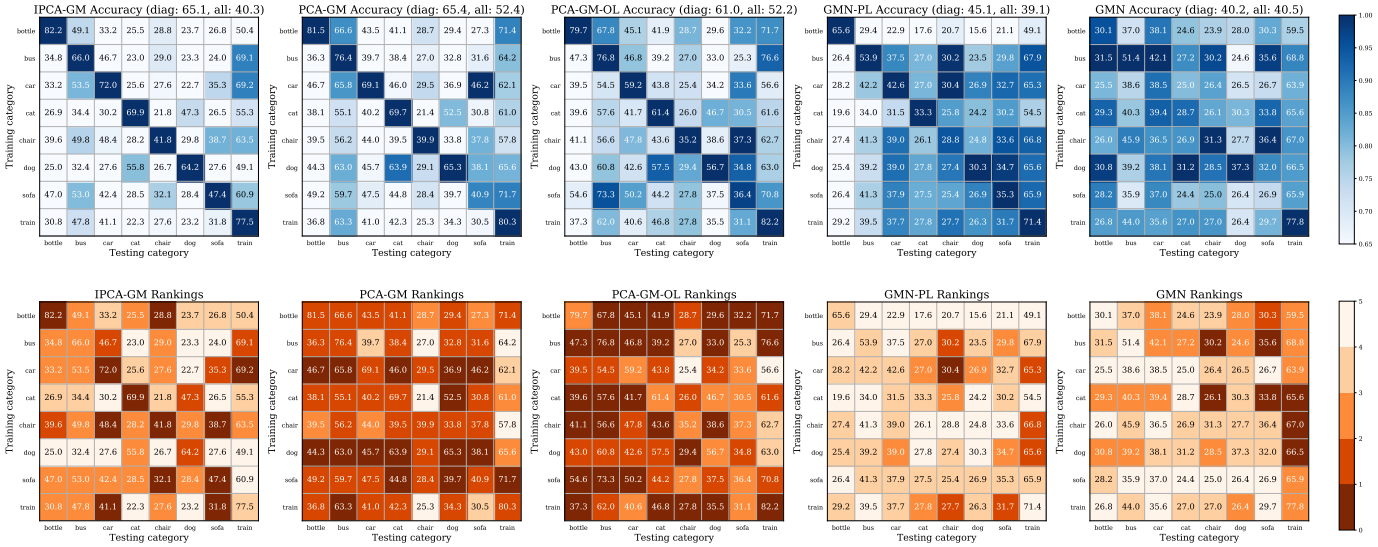


Fig. 5. Transfer learning among eight categories from Pascal VOC Keypoint represented by confusion matrix. Models are trained on categories on the y-axis, and testing results are reported on categories on the x-axis. Note that accuracy does not degenerate much for our embedding models between similar categories (such as cat and dog). Numbers in cells are the corresponding accuracy. Here we plot two lines of confusion matrices in parallel for better illustration. For blue matrices, the color map stands for accuracy normalized by the highest accuracy on this column, and they do not denote the absolute value of accuracy among different categories and matrices. For the orange matrices, we plot the ranking of accuracy on this current cell among all 5 confusion matrices. Darker color corresponds to higher ranking in accuracy. We also report accuracy for elements in diagonal and overall for each confusion matrix, as shown in brackets on the top of blue matrices (best viewed in color and zoom in for better view).

TABLE 3
Matching accuracy (%) for transfer learning test across datasets on Willow ObjectClass. GMN-VOC means model trained on Pascal VOC Keypoint, likewise for Willow ObjectClass.

method	face	m-bike	car	duck	w-bottle
HARG-SSVM [23]	91.2	44.4	58.4	55.2	66.6
GMN-VOC [24]	98.1	65.0	72.9	74.3	70.5
PCA-GM-VOC	**100.0**	69.8	78.6	82.4	95.1
IPCA-GM-VOC	**100.0**	67.1	73.3	82.1	91.7
GMN-Willow [24]	99.3	71.4	74.3	82.8	76.7
PCA-GM-Willow	**100.0**	76.7	84.0	**93.5**	**96.9**
IPCA-GM-Willow	**100.0**	**77.7**	**90.2**	84.9	95.2

category contains at least 40 images. All instances from the same category are carefully aligned in their pose, and in the same category all instances have 10 visible keypoints. Further more, it lacks scale, background and illumination changes and we therefore consider it an easier dataset compared to Pascal VOC Keypoint. We resize images to 256×256 before passing to CNN. Since every object from the same category shares the same number of keypoints, we do not perform outlier test on Willow ObjectClass dataset.

We follow the source code released by the authors of [23], with which HARG-SSVM is trained and evaluated. Following the red dashed arrow in Figure 4(a), cross-dataset generalization capability is evaluated for other competing methods. For deep graph matching models, their weights are firstly initialized on a slightly modified Pascal VOC Keypoint dataset, from which we remove all VOC 2007 car and motorbike images. These transferred models are denoted as GMN-VOC, PCA-GM-VOC and IPCA-GM-VOC. We further fine-tune them on the Willow dataset, namely GMN-Willow, PCA-GM-Willow and IPCA-GM-Willow, respectively. Note that HARG-SSVM is class-specific so that it learns a specific model for each class, while GMN, PCA-GM and IPCA-GM are class-agnostic and a unified model is learned for all classes. We only report cross-dataset transfer learning result because Willow ObjectClass is too small to train a deep graph matching model from scratch. Table 3 demonstrates our proposed PCA-GM and IPCA-GM showing superior transfer learning capability across different datasets, and outperform all competing methods in all categories of Willow ObjectCalss dataset.

4.4.3 *Cross-category generalization capability*

To testify the generalization behavior of our model among different categories, we train IPCA-GM, PCA-GM, PCA-GM-OL, GMN-PL, GMN on eight arbitrarily selected categories in Pascal VOC Keypoint and report testing result on each category as shown in Figure 5. The experimental setup is illustrated by Figure 4(b). We follow the train/test split provided by the benchmark for each category. Cross-category test result is plotted via confusion matrix (y-axis stands for training category and x-axis stands for testing category) with blue color denoting relative accuracy and orange color denoting the ranking among 5 models on each certain cell. Our learned embedding models generalize soundly to unseen similar categories, such as between cat and dog. It shows that embedding based models generalize better to unseen categories on off-diagonal cells, while the permutation loss offers better supervision on diagonal elements. We notice that IPCA-GM ranks comparatively on diagonal cells with PCA-GM, while PCA-GM and PCA-GM-OL generalize better on off-diagonal cells. A possible explanation may be IPCA-GM is with higher model capacity, therefore it fits better on its training category and achieves higher accuracy on diagonal elements in confusion matrix.

TABLE 4
Matching accuracy (%) on Pascal VOC Keypoint with different module configurations. For column "CNN", "DGM" indicates the CNN module learned with PCA-GM and "ImgNet" means the CNN is only pretrained on ImageNet. For column "graph", "Emb" corresponds to learned cross-graph embedding in PCA-GM while "RRWM" means the existing RRWM solver [1]. The first row is identical to PCA-GM baseline in Table 1.

CNN	graph	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
DGM	Embed	**51.2**	61.3	**61.6**	**58.4**	**78.8**	**73.9**	**68.5**	**71.1**	40.1	**63.3**	45.1	**64.4**	**66.4**	**62.2**	**45.1**	**79.1**	**68.4**	**60.0**	**80.3**	**91.9**	**64.6**
DGM	RRWM	44.0	**61.4**	53.5	51.7	72.1	67.7	64.5	59.6	**41.0**	55.8	**53.9**	58.6	60.0	57.2	41.3	78.8	61.8	45.2	74.2	88.1	59.5
ImgNet	Embed	36.5	46.9	51.9	43.0	42.7	52.0	53.4	57.7	33.5	47.5	44.9	54.7	54.6	48.1	28.0	61.8	57.2	39.9	72.6	58.3	49.3
ImgNet	RRWM	29.9	44.7	41.4	41.6	34.7	44.9	45.2	38.8	27.0	34.3	36.7	37.7	41.4	41.7	24.9	57.2	41.2	30.7	68.3	53.3	40.8

4.4.4 Generalization of CNN and embedding modules

Our deep graph matching pipeline generally learns image feature (with CNN) and graph feature (with embedding) simultaneously. We conduct further study on how they generalize with other modules and how do they couple with each other. In this experiment, we decompose the learned PCA-GM into two parts: a CNN backbone (denoted as "DGM") tuned by the matching dataset at hand, and the embedding component, and plug them into existing graph matching solver RRWM [1] and a vanilla VGG16 pretrained on ImageNet classification, respectively. In Figure 4(a), blue arrows represent "ImgNet+Embed" and the purple arrow represents "DGM+RRWM". Experimental result in Table 4 shows that the learned CNN module generalizes soundly after replacing the embedding module by RRWM, especially on categories such as *bike*, *chair* and *table*. In comparison, the learned embedding module seems more tightly coupled with the learned CNN. However, the "ImgNet+Embed" configuration outperforms "ImgNet+RRWM" in all categories, and in some categories "ImgNet+Embed" performs comparatively with "DGM+RRWM". Therefore, the learned embedding module is still meaningful when transferred to different CNN backbones.

4.5 Discussion and Further Study

4.5.1 Model design details

There are few hyper-parameters for tuning, including number of embedding layers, regularization factor τ of affinity metric and the number of cross-graph iterations in IPCA-GM. The parameter configuration is determined by averaged accuracy on Pascal VOC Keypoint dataset without outliers, and applied to all datasets under various settings.

For the number of embedding layers, we test PCA-GM and PIA-GM with varying configurations, and find PIA-GM is insensitive to the number of layers while PCA-GM best performs with 3 layers, as shown in Table 5. Introducing more than one cross-graph layers will also make the model fail to converge. A possible explanation to this phenomenon is deep graph convolutional nets may suffer from over-smoothing [66], especially for our cross-graph convolution. We keep all networks with 3 embedding layers for fair comparison. For hyperparameter setting, we perform grid search on $\tau = \{0.5, 0.05, 0.005, 0.0005\}$ and number of cross-graph iterations $= \{2, 3, 4, 5\}$, and we adopt the best-performing $\tau = 0.005$, with 3 iterations for IPCA-GM and $\tau = 0.05$ for PCA-GM and PIA-GM under all settings. We regard such parameter setting a suitable choice for various scenes and graph sizes, as Pascal VOC Keypoint dataset contains 20 categories with various graph sizes. As shown in Table 6, more complicated IPCA-GM seems more sensitive to parameter τ compared to PCA-GM and PIA-GM, since it is crucial to select a suitable discriminative level of affinity metric at early iterations in IPCA-GM. And there will be a higher chance of explosive gradient if τ is too small, causing PIA-GM and IPCA-GM fail to converge with $\tau = 0.0005$.

TABLE 5
Matching accuracy (%) of PIA-GM and PCA-GM by number of embedding layers on Pascal VOC Keypoint with no outliers.

# of embedding layers	2	3	4	5	6
PIA-GM	60.5	**63.0**	**63.0**	62.9	62.7
PCA-GM	62.5	**64.6**	49.5	47.6	47.7

TABLE 6
Matching accuracy (%) of PIA-GM, PCA-GM and PCA-GM by different τ on Pascal VOC Keypoint with no outliers. '–' means the model fails to *converge* under this setting.

τ	0.5	0.05	0.005	0.0005
PIA-GM	61.8	**63.0**	62.5	-
PCA-GM	63.4	**64.6**	63.8	63.1
IPCA-GM	55.1	65.0	**66.9**	-

TABLE 7
Ablation study on proposed components using the Pascal VOC Keypoint as benchmark. Tick denotes the feature is activated. For finetuned CNN feature it means it is fine-tuned using the graph matching training data, otherwise the CNN is pretrained by ImageNet.

finetuned CNN feature	intra-graph embedding	cross-graph embedding	iterative embeding	affinity metric	accuracy
✓	✓	✓	✓	✓	**66.9**
✓	✓	✓	✓	×	66.3
✓	✓	✓	×	×	64.2
✓	✓	×	×	×	62.1
✓	×	×	×	×	54.8
×	×	×	×	×	41.9

TABLE 8
Matching accuracy (%) by number of iterations for iterative cross-graph affinity component design in the **IPCA-GM** method on Pascal VOC Keypoint. Here '–' means the model fails to *converge* under this setting.

# of iterations	1	2	3	4	5
IPCA-GM (Alg. 2)	63.1	65.3	**66.9**	64.2	–
Intra_1-IPCA-GM	64.6	64.1	63.8	63.6	63.0

4.5.2 IPCA-GM components

We conduct ablation study by enabling/disabling different IPCA-GM components and report results in Table 7. The ablation study reveals that all our components have positive effect on matching accuracy. We initialize VGG16 by pretrained weights on ImageNet classification, embedding layers by random weights from uniform distribution, and

TABLE 9
Matching accuracy (%) on Pascal VOC Keypoint without outliers tested with varying numbers of iterations during testing. The model weights are all from the learned IPCA-GM model reported in Table 1, trained with 3 iterations.

# iters	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
3	**51.0**	64.9	**68.4**	**60.5**	80.2	74.7	**71.0**	**73.5**	**42.2**	**68.5**	48.9	**69.3**	67.6	**64.8**	**48.6**	**84.2**	69.8	**62.0**	79.3	89.3	**66.9**
5	50.0	64.8	65.6	60.1	80.3	75.6	69.7	72.9	41.7	66.7	**52.5**	67.2	67.6	62.7	46.2	82.5	71.3	61.0	79.6	**90.6**	66.4
10	49.9	64.9	65.3	59.8	**80.4**	**75.8**	69.7	73.1	41.9	66.7	**52.5**	67.1	**67.7**	62.6	46.4	82.5	71.3	60.1	79.5	90.5	66.4
20	50.0	**65.0**	65.2	59.8	**80.4**	**75.8**	69.7	73.1	41.9	66.7	**52.5**	67.1	67.6	62.6	46.2	82.5	**71.4**	59.3	**79.7**	90.5	66.4

the weighting matrix of affinity function by identity matrix disturbed by uniform random noise.

4.5.3 Cross-graph component design

The the cross-graph matrix in iterative cross-graph in Algorithm 2, is initialized $\hat{\mathbf{S}}^{(0)}$ as a zero matrix. We experiment a family of models where $\hat{\mathbf{S}}^{(0)}$ is initialized by the output of first intra-graph embedding layer $\mathbf{h}_{1i}^{(1)}, \mathbf{h}_{2j}^{(1)}$ namely "Intra$_1$-IPCA-GM", and we find they do not perform comparatively compared to our zero-initialization design. We also test with $f_{msg\text{-}cross}, f_{msg\text{-}cross}$ in Eqs. 12, 13 implemented by single fully-connect layer followed by ReLU activation, however the model fails to converge, which may be due to too much complexity in the cross-graph layer. As shown in Table 8, too many cross-graph iterations (e.g. $4, 5$) will degenerate the model's performance, even causing the model fail to converge. A possible explanation would be the heavy iteration adopted (including iterative cross-graph update and Sinkhorn iterations) may cause instability for backward gradient computation. We empirically find 3 iterations for IPCA-GM introduced in Algorithm 2 best performs among all methods, therefore we stick to this design in this paper.

The convergence property of the iterative cross-graph embedding in IPCA-GM is also studied, and in experiment the predicted cross-graph similarity matrix $\hat{\mathbf{S}}^{(k)}$ gradually converges when the number of iterations k grows. By adopting the IPCA-GM model weights reported in Table 1, we report its test result with 5, 10 and 20 cross-graph iterations in Table 9. Interestingly, the performance does not vary much compared to the original configuration, therefore we stick with 3 iterations for its cost-efficiency.

Compared to Algorithm 2, the iterative update scheme discussed in Section 4.5 of [22] can be improved. It will become identical to Algorithm 2 if we assign $\{\mathbf{h}_{1i}^{(3)}\}, \{\mathbf{h}_{2j}^{(3)}\}$ to $\{\mathbf{h}_{1i}^{(1)}\}, \{\mathbf{h}_{2j}^{(1)}\}$ after L13 inside the loop. Such an update scheme assumes features in the first and the last embedding layers are in the same feature space, which violates different feature mapping functions in different embedding layers. Therefore, the iterative update scheme in Section 4.5 of [22] performs poorly in experiments.

Recently, another differentiable replacement of Sinkhorn module for linear assignment problem is proposed by [67]. However, it seems to suffer from poor convergence speed compared to Sinkhorn algorithm under our deep graph matching setting. We empirically find [67] needs around 200 inner loops and 100 outer loops to converge. In contrast, Sinkhorn algorithm requires only 10 iterations. Thus we stick to Sinkhorn algorithm in our model design.

5 Conclusion and Future Work

In this paper, we have presented a novel deep graph matching pipeline, which parameterizes the graph matching affinity with deep CNN and novel embedding layers. To model the arbitrary transformation between two graphs, a permutation loss is proposed as the learning objective. We demonstrate our methods achieve state-of-the-art matching accuracy, robustness against outliers, and generalization capability across different datasets and different categories with extensive experimental results, including an ablation study on proposed components and the comparison with peer methods. Future work may explore the semi-supervised and unsupervised settings by incorporating the cycle consistency over multiple graphs.

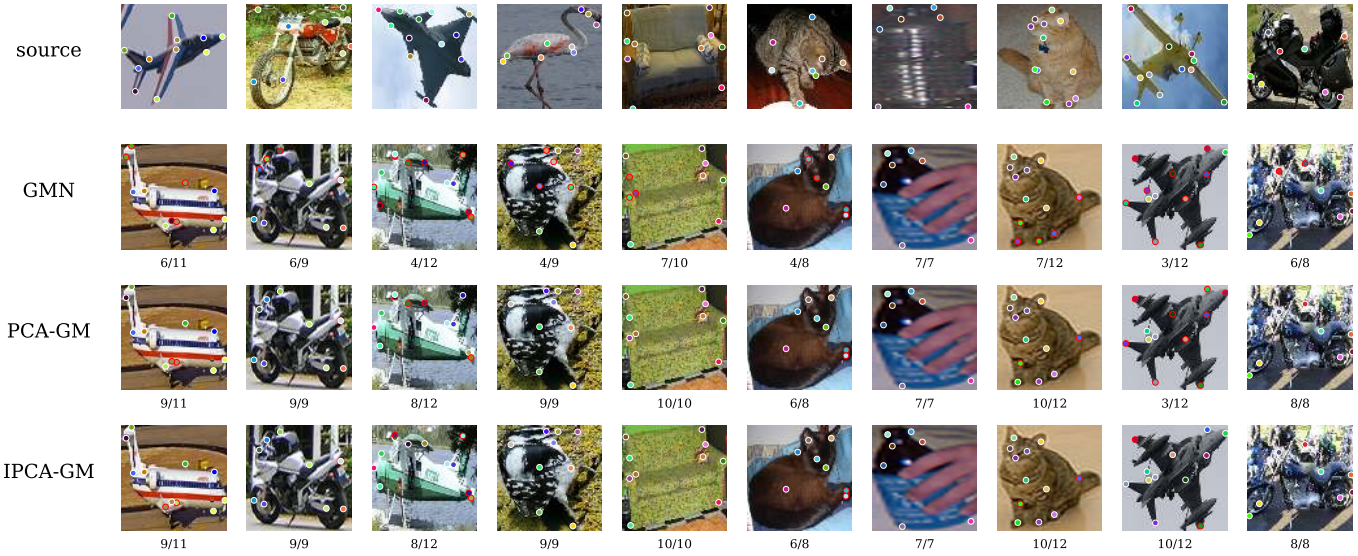
Acknowledgments

The work is partially supported by National Key Research and Development Program of China (2018AAA0100704, 2016YFB1001003), NSFC (61972250, U19B2035), and the Open Project Program of the National Laboratory of Pattern Recognition (NLPR).

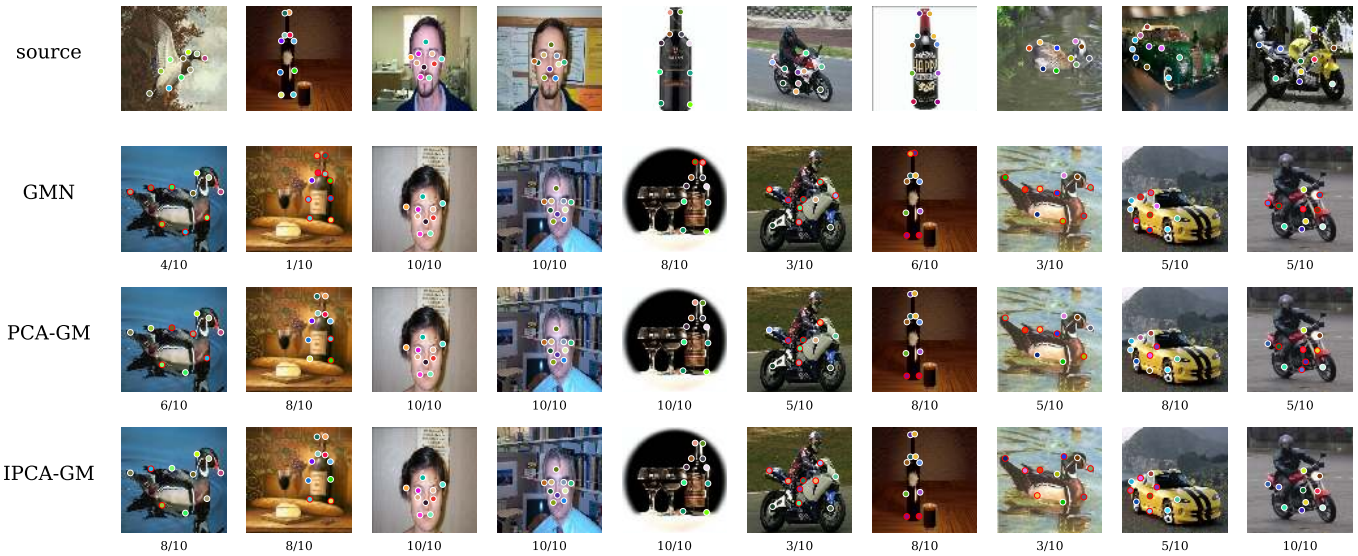
References

[1] M. Cho, J. Lee, and K. M. Lee, "Reweighted random walks for graph matching," in *ECCV*, 2010.
[2] S. Gold and A. Rangarajan, "A graduated assignment algorithm for graph matching," *TPAMI*, 1996.
[3] J. L. Lee, M. Cho, and K. M. Lee, "Hyper-graph matching via reweighted randomwalks," in *CVPR*, 2011.
[4] Q. N. Ngoc, A. Gautier, and M. Hein, "A flexible tensor block coordinate ascent scheme for hypergraph matching," in *CVPR*, 2015.
[5] J. Yan, C. Zhang, H. Zha, W. Liu, X. Yang, and S. M. Chu, "Discrete hyper-graph matching," in *CVPR*, 2015.
[6] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, 1981.
[7] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *IJCV*, 1994.
[8] M. Vento and P. Foggia, "Graph matching techniques for computer vision," *Graph-Based Methods in Computer Vision: Developments and Applications*, 2012.
[9] J. Yan, X.-C. Yin, W. Lin, C. Deng, H. Zha, and X. Yang, "A short survey of recent advances in graph matching," in *ICMR*, 2016.
[10] E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido, "A survey for the quadratic assignment problem," *EJOR*, 2007.
[11] E. L. Lawler, "The quadratic assignment problem," *Management Science*, 1963.
[12] F. Zhou and F. Torre, "Factorized graph matching," *IEEE PAMI*, 2016.
[13] M. Chertok and Y. Keller, "Efficient high order matching," *TPAMI*, 2010.
[14] O. Duchenne, F. Bach, K. In-So, and J. Ponce, "A tensor-based algorithm for high-order graph matching," *PAMI*, 2011.
[15] R. Zass and A. Shashua, "Probabilistic graph and hypergraph matching," in *CVPR*, 2008.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2020.3005590, IEEE Transactions on Pattern Analysis and Machine Intelligence



(a) Visualization of matching result on Pascal VOC Keypoint dataset (correct matching #/total matching #).



(b) Visualization of matching result on Willow ObjectClass dataset (correct matching #/total matching #).

Fig. 6. Visualization on (a) Pascal VOC Keypoint and (b) Willow ObjectClass. On each column, from top to the bottom, are source image, GMN result, PCA-GM result and IPCA-GM result, respectively. Keypoints with the same color represent the predicted matching pairs. White outer circles represent correct matching, and red ones denote the wrong matching (best viewed in color and zoom in for better view).

[16] J. Yan, M. Cho, H. Zha, X. Yang, and S. Chu, "Multi-graph matching via affinity optimization with graduated consistency regularization," *TPAMI*, 2016.

[17] J. Yan, J. Wang, H. Zha, X. Yang, and S. Chu, "Consistency-driven alternating optimization for multigraph matching: A unified approach," *IEEE Transactions on Image Processing*, vol. 24, no. 3, pp. 994–1009, 2015.

[18] Y. Chen, L. Guibas, and Q. Huang, "Near-optimal joint object matching via convex relaxation," in *ICML*, 2014.

[19] Q. Huang and L. Guibas, "Consistent shape maps via semidefinite programming," in *Proc. Eurographics Symposium on Geometry Processing (SGP)*, 2013.

[20] D. Pachauri, R. Kondor, and S. Vikas, "Solving the multi-way matching problem by permutation synchronization," in *NIPS*, 2013.

[21] Q. Wang, X. Zhou, and K. Daniilidis, "Multi-image semantic matching by mining consistent features," in *CVPR*, 2018.

[22] R. Wang, J. Yan, and X. Yang, "Learning combinatorial embedding networks for deep graph matching," in *ICCV*, 2019.

[23] M. Cho, K. Alahari, and J. Ponce, "Learning graphs to match," in *ICCV*, 2013.

[24] A. Zanfir and C. Sminchisescu, "Deep learning of graph matching," in *CVPR*, 2018.

[25] Y. Tian, J. Yan, H. Zhang, Y. Zhang, X. Yang, and H. Zha, "On the convergence of graph matching: Graduated assignment revisited," in *ECCV*, 2012.

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

[27] T. Cour, P. Srinivasan, and J. Shi, "Balanced graph matching," in *NIPS*, 2006.

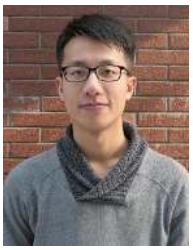
[28] F. Zhou and F. D. Torre, "Factorized graph matching," in *CVPR*, 2012.

[29] M. Leordeanu, M. Hebert, and R. Sukthankar, "An integer projected fixed point method for graph matching and map inference," in *NIPS*, 2009.

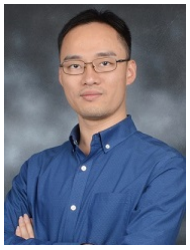
[30] K. Adamczewski, Y. Suh, and K. Lee, "Discrete tabu search for graph matching," in *ICCV*, 2015.

[31] F.-D. Wang, N. Xue, Y. Zhang, G.-S. Xia, and M. Pelillo, "A

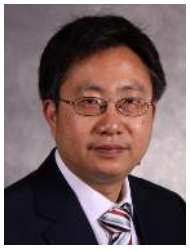
functional representation for graph matching," *IEEE transactions on pattern analysis and machine intelligence*, 2019.
[32] T. Wang, H. Ling, C. Lang, and S. Feng, "Graph matching with adaptive and branching path following," *TPAMI*, 2017.
[33] Y. Kushinsky, H. Maron, N. Dym, and Y. Lipman, "Sinkhorn algorithm for lifted assignment problems," *SIAM Journal on Imaging Sciences*, vol. 12, no. 2, pp. 716–735, 2019.
[34] M. Leordeanu, A. Zanfir, and C. Sminchisescu, "Semi-supervised learning and optimization for hypergraph matching," in *ICCV*, 2011.
[35] X. Zhou, M. Zhu, and K. Daniilidis, "Multi-image matching via fast alternating minimization," in *ICCV*, 2015.
[36] T. Yu, J. Yan, W. Liu, and B. Li, "Incremental multi-graph matching via diversity and randomness based graph clustering," in *ECCV*, 2018.
[37] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, and A. J. Smola, "Learning graph matching," *TPAMI*, 2009.
[38] M. Leordeanu, R. Sukthankar, and M. Hebert, "Unsupervised learning for graph matching," *IJCV*, 2012.
[39] L. Torresani, V. Kolmogorov, and C. Rother, "Feature correspondence via graph matching: Models and global optimization," in *ECCV*, 2008.
[40] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *TNN*, 2009.
[41] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *SIGKDD*, 2016.
[42] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *SIGKDD*, 2016.
[43] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *Computer Science*, 2013.
[44] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *SIGKDD*, 2014.
[45] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *WWW*, 2015.
[46] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *ICLR*, 2017.
[47] W. Kool and M. Welling, "Attention solves your tsp," *arXiv:1803.08475*, 2018.
[48] J. Huang, M. Patwary, and G. Diamos, "Coloring big graphs with alphagozero," *arXiv:1902.10162*, 2019.
[49] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," in *NIPS*, 2017.
[50] A. Nowak, S. Villar, A. S. Bandeira, and J. Bruna, "Revised note on learning quadratic assignment with graph neural networks," in *DSW*, 2018.
[51] R. Sinkhorn, "A relationship between arbitrary positive matrices and doubly stochastic matrices," *AoMS*, 1964.
[52] G. Mena, D. Belanger, G. Muñoz, and J. Snoek, "Sinkhorn networks: Using optimal transport techniques to learn permutations," *NIPS Workshop in Optimal Transport and Machine Learning*, 2017.
[53] R. P. Adams and R. S. Zemel, "Ranking via sinkhorn propagation," *arXiv:1106.1925*, 2011.
[54] G. Patrini, M. Carioni, P. Forre, S. Bhargav, M. Welling, R. v. d. Berg, T. Genewein, and F. Nielsen, "Sinkhorn autoencoders," *arXiv:1810.01118*, 2018.
[55] P. Emami and S. Ranka, "Learning permutations with sinkhorn policy gradient," *arXiv:1805.07010*, 2018.
[56] R. Santa Cruz, B. Fernando, A. Cherian, and S. Gould, "Visual permutation learning," *TPAMI*, 2018.
[57] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," in *NIPS*, 1994.
[58] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.
[59] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *AISTATS*, 2011.
[60] F. Serratosa, A. Solé-Ribalta, and X. Cortés, "Automatic learning of edit costs based on interactive and adaptive graph recognition," in *GbR*, 2011.
[61] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, 1955.
[62] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2014.
[63] K. Mikolajczyk and C. Schmid, "Scale & affine invariant interest point detectors," *IJCV*, 2004.
[64] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *IJCV*, 2010.
[65] L. Bourdev and J. Malik, "Poselets: Body part detectors trained using 3d human pose annotations," in *ICCV*, 2009.
[66] G. Li, M. Muller, A. Thabet, and B. Ghanem, "Deepgcns: Can gcns go as deep as cnns?" in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.
[67] X. Zeng, R. Liao, L. Gu, Y. Xiong, S. Fidler, and R. Urtasun, "Dmm-net: Differentiable mask-matching network for video object segmentation," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.



Runzhong Wang is currently a PhD Candidate with Department of Computer Science and Engineering, Shanghai Jiao Tong University. He is sponsored by Wen-Tsun Wu Honorary Doctoral Scholarship, AI Institute, Shanghai Jiao Tong University. He obtained B.E. in Electronic Engineering from Shanghai Jiao Tong University. He serves a reviewer for CVPR 2020. His research interests include machine learning on computer vision and combinatorial optimization.



Junchi Yan (M'10) is currently an Associate Professor with Department of Computer Science and Engineering, and AI Institute, Shanghai Jiao Tong University, China. Before that, he was a Senior Research Staff Member and Principal Scientist with IBM Research – China, where he started his career in April 2011. He obtained the Ph.D. in Electrical Engineering from Shanghai Jiao Tong University. He received the ACM China Doctoral Dissertation Nomination Award and China Computer Federation Doctoral Dissertation Award for his work on graph matching. His research interests include machine learning and computer vision. He serves as an Associate Editor for IEEE ACCESS, Area Chair for CVPR 2021, ICPR 2020, and Senior PC for CIKM 2019. He is a Member of IEEE.



Xiaokang Yang (M'00-SM'04-F'19) received the B. S. degree from Xiamen University, in 1994, the M. S. degree from Chinese Academy of Sciences in 1997, and the Ph.D. degree from Shanghai Jiao Tong University in 2000. He is currently a Distinguished Professor of School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. His research interests include visual signal processing and communication, media analysis and retrieval, and pattern recognition. He serves as an Associate Editor of IEEE Transactions on Multimedia and an Associate Editor of IEEE Signal Processing Letters. Prof. Yang is also a Fellow of IEEE.