Graph-context Attention Networks for Size-varied Deep Graph Matching

Zheheng Jiang, Hossein Rahmani, Plamen Angelov, Sue Black, Bryan M. Williams
Lancaster University
Bailrigg, Lancaster, LA1 4WA, United Kingdom
[z.jiang11,h.rahmani,p.angelov,sue.black,b.williams6]@lancaster.ac.uk

Abstract

Deep learning for graph matching has received growing interest and developed rapidly in the past decade. Although recent deep graph matching methods have shown excellent performance on matching between graphs of equal size in the computer vision area, the size-varied graph matching problem, where the number of keypoints in the images of the same category may vary due to occlusion, is still an open and challenging problem. To tackle this, we firstly propose to formulate the combinatorial problem of graph matching as an Integer Linear Programming (ILP) problem, which is more flexible and efficient to facilitate comparing graphs of varied sizes. A novel Graph-context Attention Network (GCAN), which jointly capture intrinsic graph structure and cross-graph information for improving the discrimination of node features, is then proposed and trained to resolve this ILP problem with node correspondence supervision. We further show that the proposed GCAN model is efficient to resolve the graph-level matching problem and is able to automatically learn node-to-node similarity via graph-level matching. The proposed approach is evaluated on three public keypoint-matching datasets and one graph-matching dataset for blood vessel patterns, with experimental results showing its superior performance over existing state-of-the-art algorithms for keypoint and graph-level matching.

1. Introduction

Graph matching, which involves establishing correspondences of nodes between graph-structured data, has a wide range of real-world applications, such as objective keypoint matching [1], vein pattern matching [2] and 2D/3D shape matching [3]. Graph matching is a well-known general NP-hard problem, and is often formulated as a quadratic assignment problem (QAP) [4].

Recently, deep learning based graph matching methods have demonstrated superior performance over traditional methods. Research in deep graph matching typically focuses on two parts: developing graph embedding networks and resolving combinatorial problems. Generally, approaches to deep graph matching firstly involve extracting node features from a deep convolutional neural network (CNN). Since nodes with similar deep CNN features are difficult to distinguish, a graph embedding network is designed to model and refine node features, which is expected to capture intrinsic graph structure and cross-graph information for improving node differentiation. Driven by the popular graph convolutional neural networks (GCN) [5], GCNs and their variants are widely used as graph embedding methods [6–9]. In recent years, transformer based model [10–13], which use the attention mechanism to capture the relationships between different words or between different image portions, has received more attention due to their superior performance over traditional CNN models. Inspired by this, we propose a novel Graph-context Attention Network to jointly capture the relationships of nodes inside a graph and node-to-node relationships between graphs. In terms of the combinatorial problem, recent works formulate it as Lawler's quadratic assignment programming (QAP) problem [7, 14] or Koopmans-Beckmann's QAP problem [6, 9]. These approaches regard the node-to-node correspondence problem as a binary classification task. The combinatorial problem is typically tackled by performing the Hungarian algorithm on a square matrix of predicted classification scores. However these methods assume that the nodes V_2 in graph G_2 include all nodes V_1 in graph G_1, i.e. $V_1 \subseteq V_2$. If V_2 contain some outliers that are invisible in V_1, [7, 14] suggest adding dummy nodes to allow these outliers can be assigned in the Hungarian Algorithm. However, in a more general case, matching graphs may contain different numbers of nodes and the number of outliers in both graphs are unknown during testing. We observe that the Hungarian algorithm has difficulty in handling such size-varied graphs matching problem. Furthermore, the above mentioned methods require node correspondence supervision, in this paper we also explore graph matching without node-level labeling.

Our main contributions are summarized as follows:

1. To handle more general and more challenging graph

matching problems with sized-varied graphs, we propose to formulate the combinatorial problem as an Integer Linear Programming problem.

2. To obtain more distinguishable node features, We present a novel Graph-context Attention Network with attention mechanism, to capture and model intrinsic graph structure and cross-graph information.

3. We demonstrate that our network can be trained in either a supervised way, which is the typical approach to such problems and requires ground-truth node-to-node correspondence, or in a novel unsupervised way. In the supervised case, we show improved accuracy over the state-of-the-art using our ILP attention loss. In the unsupervised case, we avoid the typical requirement of manually-labelled node correspondence by enabling our network to learn the node-to-node similarity by reasoning on graph-level matching.

4. We evaluate the proposed approach on both size-equal and size-varied keypoint matching tasks with three public keypoint-matching datasets, and on a graph-level matching task with a vessel graph dataset. Our experimental results demonstrate that the proposed approach outperforms current state-of-the-art methods for all tasks. We also present an ablation study, which shows the effectiveness of each component of the proposed approach.

2. Related work

2.1. Graph Neural Networks

GNNs have drawn considerable attention due to their state-of-the-art performance on graph-related tasks. The basic idea of GNNs is to use message passing functions for nodes to aggregate feature messages from their neighbours in the graph. Popular graph convolutional networks [5] (GCN) compute node updates by aggregating information with a mean-pooling operation. GraphSAGE [15] concatenates the node feature and mean/max/LSTM pooled neighbourhood information to represent new node features. Graph Attention Networks [16] aggregate neighbourhood information by introducing trainable attention weights, which have achieved excellent performance on node classification. Recently, [17] have noted that previous models cannot distinguish nodes at symmetric/isomorphic positions in the network, and additionally computed the distances of the target node to randomly sampled anchor sets.

2.2. Learning for graph matching

The above work only exploits intrinsic graph structures. For the graph matching task, recent work has focused considerably on developing advanced graph embedding networks, where cross-graph affinity information and matching consistency are encoded. For example, [6] combined a GCN with a cross-graph node embedding layer to refine the node embeddings provided by the backbone architecture. [14] further developed a matching aware embedding model, where the predicted soft assignment score is concatenated to node embeddings. To address the combinatorial problem in graph matching tasks, the problem is generally formulated by [6, 7, 9, 14] as Lawler's quadratic assignment programming (QAP) problem or Koopmans-Beckmann's QAP problem, which can be resolved using a Hungarian algorithm. Several combinatorial loss functions have been applied to train graph matching models in a supervised way. [6] demonstrated that the cross-entropy permutation loss has improved performance over pixel offset loss. [7] further improve their work by applying a Hungarian attention mechanism. A cost margin loss, which is more robust to distribution shifts than traditional Hamming distance loss, is proposed in [18] and is adopted by [8] as the training loss. Most recently, [9] applied focal loss to address the class imbalance issue. While most of the above mentioned models employ Hungarian algorithm to discretise predicted matching score, which we find is not efficient to handle size-varied graphs. In addition, they require node-level labelling for supervised training, there remains the open question of how to extend GNN models to graph matching without node-level labelling. In this paper, we also explore the computation of similarity scores between nodes by reasoning on graph-level matching.

3. Proposed Method

The overall framework of our Graph-context Attention Network is illustrated in Fig. 1. Given a pair of images with graphs to be matched, a CNN is firstly adopted to extract keypoint features via bi-linear interpolation followed by node positional encoding and channel-attention weight computing to support our GCA module. Our GCA module (as shown in Fig. 2) is designed not only to model the intrinsic graph structure for learning effective node representations, but also the similarity across the root nodes with k-hop neighbourhood between two graphs. There are two types of loss function designed for training the whole model in an end-to-end way. For node matching, integer linear programming (ILP) attention loss is introduced after aggregating and normalizing the matching scores of all GCA modules. For graph-level matching, a graph-level aggregator is firstly proposed to compute graph level representations, and then a margin-based pairwise loss is used to train the proposed model.

3.1. Problem Definition

Given two graphs $G_1 = (V_1, E_1, U_1)$ and $G_2 = (V_2, E_2, U_2)$, where V_1 and V_2 are the set of vertices, E_1 and E_2 are the set of edges, and U_1 and U_2 consist of pseudo-coordinates that locally define the spatial relationship between connected nodes, our model aims to produce

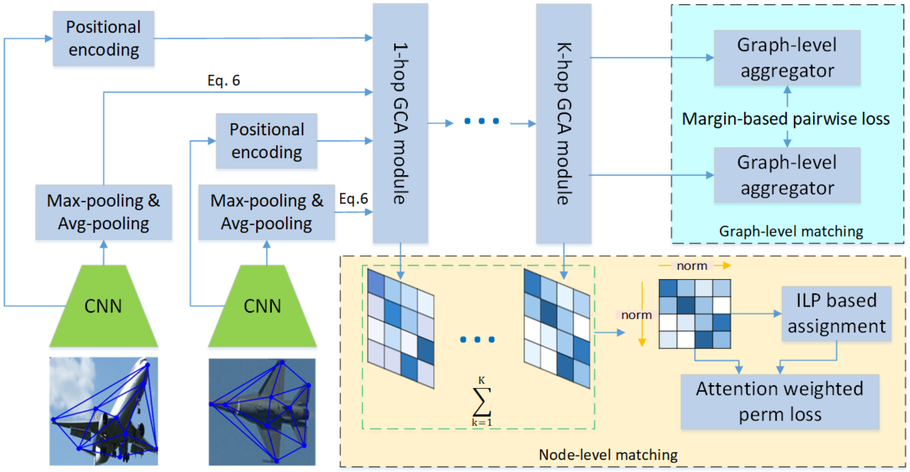


Figure 1: The overall framework of the proposed transformer-based graph matching framework.The node features are extracted from a CNN network followed by positional encoding and our Graph-context attention (GCA) module. The whole network can be either trained by our ILP attention loss for node-level matching or trained by margin-based pairwise loss for graph-level matching

the similarity score $S = \left\{ s_k \left(V_1^k, V_2^k \right) |_{k=1}^{K}, s\left(G_1, G_2\right) \right\}$, where $s_k \left(V_1^k, V_2^k \right)$ is the similarity score across the root nodes with k-hop neighbourhoods between two graphs and $s\left(G_1, G_2\right)$ is a graph-level similarity in vector space.

3.2. Positional Encoding

In most implementations of transformers for natural language processing [10, 11] and vision based tasks [12, 13], positional encoding/embedding is firstly introduced to capture positional information of each word in a sequence (one dimension) or of each patch in a grid (two dimensions). However, sequence or grid based positional encoding/embedding approaches are designed for regular data and can not be directly applied to irregular data such as a graph. Although GNNs [5, 15, 16, 19] are able to capture structural information of the graph, positional information of the node within the context of the graph structure is ignored [17]. To encode positional information in 2D spatial space, we define:

$$d_i^{pos} = d_i + \sum_{j \in \mathcal{N}_i} d_j \odot g(u(i,j)) \tag{1}$$

where d_i and d_j are the feature vectors of nodes i and j respectively, $\odot$ means element-wise product, $\mathcal{N}_i$ denotes the neighbours of node i and $g(.)$ is a kernel function which is dependent on coordinate $u(i,j) = (|x_i - x_j|, |y_i - y_j|) = (\Delta x_i, \Delta y_i) \in U_1, U_2$. For the choice of kernel function, we adopt the B-spline kernel proposed in [20], since it shows impressive performance in [8, 14]. Let $N^m_{vert,\delta}$ and $N^m_{hori,\delta'}$ denote B-spline bases of degree m along the vertical and horizontal in spatial space, with $\delta * \delta'$ kernel size. The kernel function is defined as:

$$g(u(i,j)) = \sum_{p \in \mathcal{P}} \omega_p \cdot N^m_{vert,p}(\Delta x_i) \cdot N^m_{hori,p}(\Delta y_i) \tag{2}$$

where, ω_p denotes a trainable weight vector for each element p, with the same dimension as d_i. $\mathcal{P}$ is the Cartesian product of B-spline bases $N^m_{vert,\delta}$ and $N^m_{hori,\delta'}$, consisting of $\delta * \delta'$ elements.

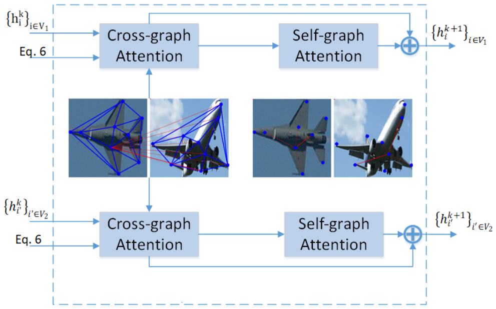


Figure 2: Our Graph-context Attention Module consists of two major components: the cross-graph attention layer and self-graph attention layer.

3.3. Graph-context Attention Module

Here, we introduce GCA modules, where the correspondence is established through preserving the structure similarity across two graphs. Each GCA Module consists of a cross-attention layer and a self-attention layer defined as follows.

Cross-attention layer In this layer, an attention mechanism is used to firstly measure node-to-node similarity between graphs, and then refine node features by aggregating this cross-graph information. We compute node-to-node similarity between graphs as below:

$$a_{i'\to i} = \frac{exp\left(\left\langle \alpha_h \odot h_i^{(k)}, \alpha_h \odot h_{i'}^{(k)} \right\rangle\right)}{\sum_{\hat{i}'} exp\left(\left\langle \alpha_h \odot h_i^{(k)}, \alpha_h \odot h_{\hat{i}'}^{(k)} \right\rangle\right)} \tag{3}$$

where $\langle \cdot, \cdot \rangle$ is the inner product, α_h is a channel-attention weight vector for exploiting the inter-channel relationship of features, inspired by the recent popular channel attention module [21]. We compute α_h as below:

$$\alpha_h = tanh\left(F_{avg} * W_c + F_{pool} * W_c\right) \tag{4}$$

where F_{avg} and F_{pool} denote average-pooled features and max-pooled features computed from the last convolutional layer of the CNN model. $W_c \in \mathbb{R}^{D_F \times D_h}$ is a trainable weight matrix, and D_F and D_h are the dimension of the average-/ max-pooled feature F and node feature h respectively. $tanh(\cdot)$ is the hyperbolic tangent function.

To aggregate this cross-graph information, we use $a_{i'\to i}$ as the attention coefficient and define:

$$f_{cr}\left(h_i^{(k)}, \left\{h_{i'}^{(k)}\right\}_{i'\in V'}\right) = h_i^{(k)} \left\| \sum_{i'} a_{i'\to i}\left(h_i^{(k)} - h_{i'}^{(k)}\right)\right. \tag{5}$$

where if $i \in V = V_1$ then $i^{'} \in V^{'} = V_2$, or if $i \in V = V_2$ then $i^{'} \in V^{'} = V_1$. $\|$ is the concatenation operation, $\left\{h_i^{(k)}\right\}_{i\in V}$ denotes a set of node representations before the cross-attention layer.

Self-attention layer To encode the graph structure, Graph Convolutional Networks are usually used in graph matching tasks [1, 6, 9, 22]. However, such models assign equal importance to neighbourhoods during message passing from neighbourhoods to the center node, and ignores the relationship between the features of neighbourhoods. In contrast, we utilize an attention mechanism to learn the importance of neighbourhoods and propose a self-attention layer:

$$f_{se}\left(h_i^{cr}, \left\{h_j^{(k)}\right\}_{j\in\mathcal{N}_i}\right) = \overset{K}{\underset{k=1}{\|}} \sum_{j\in\mathcal{N}_i} a_{j\to i}^{k} h_j^{cr} * W_s \tag{6}$$

where h_i^{cr} and h_j^{cr} are node features after the cross-attention layer, $W_s \in \mathbb{R}^{D_{cr}\times D_{se}}$ is a trainable weight matrix, and D_{cr} and D_{se} respectively denote the dimension of node feature F after the cross-attention layer and after the self-attention layer, $a_{j\to i}$ is a normalized attention coefficient computed by the k_{th} attention mechanism, defined as:

$$a_{j\to i} = \frac{exp\left(\left\langle h_i^{cr} * W_s, h_j^{cr} * W_s \right\rangle\right)}{\sum_{\hat{j}} exp\left(\left\langle h_i^{cr} * W_s, h_{\hat{j}}^{cr} * W_s \right\rangle\right)} \tag{7}$$

With the above definition, each GCA module can be formulated as:

$$h_i^{(k+1)} = f_{se}\left(h_i^{cr}, \left\{h_j^{(k)}\right\}_{j\in\mathcal{N}_i}\right) + h_i^{cr} \tag{8}$$

This architecture has a nice property that it is able to preserve node-to-node similarity to the next GCA module, and meanwhile, the discrimination of node features is improved through self-attention layer.

Graph-level aggregator The proposed GCAN can be trained for node matching problem in a supervised way given ground-truth node-to-node correspondence as discussed in section 3.3.1. To apply the GCAN for graph-level matching problem or learning node-to-node similarity by reasoning on graph-level matching, a graph-level aggregator, which takes the set of node representations $\left\{h_i^{(K)}\right\}_{i\in V}$ of K_{th} GCA module as input and computes a graph-level presentation, is then proposed as below:

$$h_G = \sum_{i\in V} \sigma\left(\alpha_G \odot h_i^{(K)} * W_{G'}\right) * W_G \tag{9}$$

where $W_{G'}$ and W_G are trainable weight matrix. α_G is a channel-attention weight vector similar to Eq.4, but uses average-pooled features and max-pooled features computed from the last GCA module. After the graph representations, h_{G_1} and h_{G_2} are computed for the pair graph G_1 and G_2, their similarity can be computed in the vector space by using the Euclidean, cosine or Hamming similarities.

3.3.1 Learning

The proposed model takes a pair of graphs with node features as input and can be either trained for node-level matching problem or graph-level matching problem. For node matching, the proposed model can be trained in a supervised way by utilizing the ground-truth node-to-node correspondence as [6, 9, 14]. While many previous works [6, 9, 14, 23] consider the node matching problem as a bijective matching problem, i.e. node matching between two graphs with the same number of nodes (see Fig. 3), which can be easily resolved by the Hungarian algorithm. In such cases, the invisible and occluded nodes are filtered during training and testing. To target a more flexible assignment problem, we formulate it as an Integer Linear Programming problem, which is suitable for not only bijective node matching but also non-injective node matching. To address the assignment issue of invisible and occluded nodes, we introduce a placeholder for them and define $m \in M_0^*, n \in N_0^*$, where m and n are indexes of source nodes and target nodes. 0 is a placeholder for a 'dummy' node. We define one possible solution to this assignment problem as:

$$\theta \in \Theta = \{a_n^m | a_n^m \in \{0,1\} \quad m \in M_0^*, n \in N_0^*\} \tag{10}$$

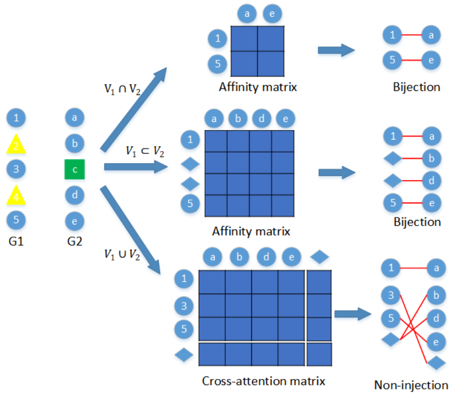


Figure 3: Description of node matching. All nodes of the two graphs are shown in the left. Blue solid circles denote visible nodes. Yellow triangles and green squares denote the invisible nodes in G1 and G2 respectively. The top and middle branch show how the most recent work [6, 9, 14, 23] addresses the bijective matching problem, where keypoints are filtered by intersection or inclusion filtering. Then the Hungarian algorithm is used to resolve the node assignment problem in the square affinity matrix. The bottom branch shows the more general node matching problem without filtering that is addressed by our approach. We consider all keypoints for matching by adding row and column dummy nodes, which allow unmatched nodes to be assigned.

where Θ consists of all possible source-to-target assignments. The score of each possible assignment is defined as:

$$\mathcal{S}(\theta) = \prod_{n \in N_0^*, m \in M_0^*} \mathcal{S}\left(a_n^m\right)^{a_n^m} \tag{11}$$

where $\mathcal{S}\left(a_n^m\right)$ is the aggregated and normalized score of all GCA modules.

We rewrite the assignment problem as a minimization:

$$\begin{aligned} \mathcal{L}_t &= \min_{\theta \in \Theta} -\left(\log\left(\mathcal{S}(\theta)\right)\right) \\ &= \min \sum_{n \in N_0^*, m \in M_0^*} -\left(\log \mathcal{S}\left(a_n^m\right) \cdot a_n^m\right) \end{aligned} \tag{12}$$

$$s.t. \quad \forall m \in M^*, \sum_{n \in N_0^*} a_n^m = 1 \tag{13}$$

$$\forall n \in N^*, \sum_{m \in M_0^*} a_n^m = 1 \tag{14}$$

The constraints (13), (14) ensure that each source node is uniquely assigned to a target node or dummy node and each target node is uniquely assigned to a source node or dummy node. The branch-and-bound algorithm [24] is then applied to find optimal solution.

To train the model, due to the sparsity of the ground-truth node-to-node correspondence, the cross-entropy loss used by previous approaches tends to back-propagate more gradients of negative samples than positive samples resulting in prediction bias towards the negative class. Focal-loss, used by [9], is able to deal with such class imbalance by introducing class weights, but the class weights are user-defined and heavily dependant on experimental experience. To resolve this problem, we propose an ILP attention loss:

$$\begin{aligned} \mathcal{L}_{perm} = &- \sum_{\substack{i \in V_1 \\ i' \in V_2}} \max\left\{a_{i,i'}, Y_{i,i'}\right\}\left(Y_{i,i'} \log \bar{Y}_{i,i'}\right. \\ &\left. + \left(1 - Y_{i,i'}\right) \log\left(1 - \bar{Y}_{i,i'}\right)\right) \end{aligned} \tag{15}$$

where $Y_{i,i'}$ and $\bar{Y}_{i,i'}$ are the ground truth and prediction respectively, $a_{i,i'}$ is the ILP assignment result. If two graphs have different number of nodes, then $i \in V_1 = M_0^*$ and $i' \in V_2 = N_0^*$. Otherwise $i \in V_1 = M^*$ and $i' \in V_2 = N^*$. Note that in this loss function, node matching is regarded as a binary classification task where 1 and 0 mean matched and unmatched respectively. By introducing $a_{i,i'}$, the proposed model will focus on minimizing loss at positive samples and high ILP responses.

For the case that node labels are not given, the proposed model can still compute similarity scores between nodes through our cross-graph attention layer. To train the model with graph-level matching, we use Euclidean similarity and follow margin-based pairwise loss:

$$\begin{aligned} \mathcal{L}_{graph} = &Y_G \left\| h_{G_1} - h_{G_2} \right\| + \left(1 - Y_G\right) \max\{ \\ &0, \lambda - \left\| h_{G_1} - h_{G_2} \right\| \} \end{aligned} \tag{16}$$

where Y_G is a binary flag equal to 0 for a negative pair and to 1 for a positive pair, $\lambda > 0$ is a margin parameter.

4. Experiments

In this section, we evaluate the proposed graph network on two types of task: (i) keypoint matching and (ii) graph-level matching. We compare our graph network with existing state-of-the-art methods using three public key point matching datasets and one vessel graph dataset. The results demonstrate that our graph network consistently outperform all other approaches.

Algorithm 1 Training algorithm for node matching.

```
Input: Graph pairs $G_1$, $G_2$; image pairs $I_1$, $I_2$; ground-truth node-to-node correspondence $Y^*$
Extract and keypoint features from CNN via bi-linear mapping.
Concatenate keypoint features extracted from CNN to represent node features $\{d_i\}_{i \in V}$ in our GCAN.
Compute average-pooled features $F_{avg}$ and max-pooled features $F_{pool}$ from the last convolutional layer of the CNN.
for epoch $e \leq E$ do
    for attention module $k < K$ do
        $h_i^{cr} \leftarrow f_{cr}\left(h_i^{(k)}, \left\{h_{i'}^{(k)}\right\}_{i' \in V'}\right)$ (Eq. 5)
        $\mathcal{S}_{i,i'}^k \leftarrow a_{i' \rightarrow i}$
        $h_i^{se} \leftarrow f_{se}\left(h_i^{cr}, \left\{h_j^{(k)}\right\}_{j \in \mathcal{N}_i}\right)$ (Eq. 9)
        $h_i^{k+1} \leftarrow h_i^{cr} + h_i^{se}$ (Eq. 8)
    end for
    Perform row and column normalization on $\mathcal{S}$
    Solve $\underset{\{a_{i,i'}\}}{\operatorname{argmin}} \sum_{i,i'} -\left(\log \mathcal{S}\left(a_{i,i'}\right) \cdot a_{i,i'}\right)$ using branch-and-bound algorithm
    Update model weights using Eq. 15
end for
```

4.1. Implementation details

Our PyTorch implementation of GCAN involves 3 GCA modules, which achieve the best performance in our experiment. We employ Adam [25] as an optimizer with an initial learning rate of 2×10^{-5} for VGG16 and 2×10^{-3} for other models. All experiments were performed using a Dell Precision 5820 Workstation with a NVidia GeForce RTX 2080 Ti GPU, 32GB RAM and an Intel(R) Xeon(R) W-2245 CPU. Our codes is available on github: https://github.com/ZhehengJiang/GCAN.

4.2. Keypoint matching

Datasets We evaluate the proposed method for keypoint matching on three public datasets: Willow ObjectClass dataset [27], Pascal VOC dataset with Berkeley annotations [28] and SPair-71k [29]. The **Pascal VOC dataset** consists of 20 classes of instances with labeled keypoint locations. This dataset is considerably challenging since image instances may vary in scale, pose and illumination, affecting the numbers of available keypoints and the number of inliers ranges from 6 to 23. We follow [14] to use 7,020 annotated images for training and 1,682 for testing. The **SPair-71k** is a larger dataset collected from Pascal VOC 2012 and Pascal 3D+, containing 70,958 image pairs with higher image quality and richer keypoint annotations. Following [8], we use 58,724 image pairs for training and 12,234 for testing. For the input image pairs, we follow [14] to construct the reference graph by Delaunay triangulation, and the target graph by full connection. The **Willow ObjectClass** dataset is a less challenging one than previous two datasets, which contains five classes (car, duck, face, motorbike and winebottle) from Caltech-256 and Pascal VOC 2007. Each class consists of at least 40 images with different instances and each of them is annotated with 10 distinctive image keypoints. Following the default setting in [14], the first 20 images from each class are used for training and the rest are for testing.

Baseline methods and metrics We compare the proposed method against 8 state-of-the-art methods, GMN [1], PCA-GM [6], IPCA-GM [26], CIE-H [7], LCS [23], BBGM [8] and NGM-v2 [14]. To compare the proposed ILP attention loss with previous work, we use focal loss (FL) [9], binary cross-entropy (BCE) [14], cost margin (CM) [8], pixel offset regression (POR) [1], Hungarian Attention Loss (HAL) [7]. For fair comparison, we follow previous work [1, 8, 14] and extract node features from relu4_2 and relu5_1 of VGG16 [30] via bilinear mapping. The matching accuracy is computed as the number of correctly matched keypoint pairs averaged by the number of all true matched keypoint pairs.

Results We firstly follow the most common experimental setting, where intersection filtering is applied to generate graphs with the equal size, and report matching accuracy of the 20 classes and average accuracy on Pascal VOC dataset in Tab. 1. We can see that the proposed approach outperform the other state-of-the-art approaches for most classes and particularly in terms of mean accuracy. Note that BBGM, NGM-v2 and NHGM-v2 also adopt SplineCNN to refine the features, but our approach achieves 1.9% higher accuracy than the best one of them due to our more efficient GCAN and ILP Attention loss function. The matching accuracy of the proposed approach against the current state-of-the-art on the SPair-71k and Willow ObjectClass datasets respectively is reported in Supplementary. Our approach achieves the best performance in both tables, demonstrating its generality. To demonstrate the effectiveness of the proposed loss function, we replace our ILP_AL with different types loss function while keeping our GCAN unchanged, and report their accuracy in Tab. 2 (full table can be found in Supplementary). Specifically, our ILP_AL achieves the best performance and has more than 1% improvement of mean accuracy over BCE (used in CIE-H) and HAL (used in NGM-v2). Comparing the accuracy of BCE and HAL reported in Tab. 2 against the accuracy of CIE-H and NGM-v2 reported in Tab. 1, we can see that the proposed GCAN surpasses CIE-H and NGM-v2 even trained by the same loss function. In the more general setting, matching graphs may contain different numbers of nodes, which is closer to the real-world scenario. To evaluate the systems on such size-varied graph problem, since matching accuracy ignores

Table 1: Matching accuracy (%) on Pascal VOC Keypoint with intersection filtering.

method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbkie	person	plant	sheep	sofa	train	tv	mean
GMN [1]	41.6	59.6	60.3	48.0	79.2	70.2	67.4	64.9	39.2	61.3	66.9	59.8	61.1	59.8	37.2	78.2	68.0	49.9	84.2	91.4	62.4
PCA-GM [6]	49.8	61.9	65.3	57.2	78.8	75.6	64.7	69.7	41.6	63.4	50.7	67.1	66.7	61.6	44.5	81.2	67.8	59.2	78.5	90.4	64.8
IPCA-GM [26]	53.8	66.2	67.1	61.2	80.4	75.3	72.6	72.5	44.6	65.2	54.3	67.2	67.9	64.2	47.9	84.4	70.8	64.0	83.8	90.8	67.7
CIE-H [7]	49.9	63.1	70.7	53.0	82.4	75.4	67.7	72.3	42.4	66.9	69.9	69.5	70.7	62.0	46.7	85.0	70.0	61.8	80.2	91.8	67.6
LCS [23]	46.9	58.0	63.6	69.9	87.8	79.8	71.8	60.3	44.8	64.3	79.4	57.5	64.4	57.6	52.4	96.1	62.9	65.8	94.4	92.0	68.5
BBGM [8]	61.9	71.1	79.7	79.0	87.4	94.0	89.5	80.2	56.8	79.1	64.6	78.9	76.2	75.1	65.2	98.2	77.3	77.0	94.9	93.9	79.0
NGM [6]	50.1	63.5	57.9	53.4	79.8	77.1	73.6	68.2	41.1	66.4	40.8	60.3	61.9	63.5	45.6	77.1	69.3	65.5	79.2	88.2	64.1
NGM-v2 [14]	61.8	71.2	77.6	78.8	87.3	93.6	87.7	79.8	55.4	77.8	89.5	78.8	80.1	**79.2**	62.6	97.7	77.7	75.7	96.7	93.2	80.1
NHGM-v2 [14]	59.9	**71.5**	77.2	79.0	87.7	94.6	89.0	**81.8**	60.0	**81.3**	87.0	78.1	76.5	77.5	64.4	**98.7**	**77.8**	75.4	97.9	92.8	80.4
ours	**63.4**	71.2	**80.1**	**81.1**	**90.4**	**95.5**	**89.5**	80.4	**65.3**	80.8	**89.9**	**81.4**	**80.6**	78.1	**67.7**	98.2	77.5	**82.6**	**98.4**	**93.4**	**82.3**

Table 2: Matching accuracy (%) of different loss functions on Pascal VOC Keypoint with intersection filtering.

loss	FL [9]	CM [8]	POR [1]	BCE [14]	HAL [7]	ILP_AL
mean acc	77.7	40.2	78.7	81.3	81.2	**82.3**

false positives, we report mean F1-Score instead in Tab. 3. We show in Fig. 4-left the distribution of the test set used in Tab. 3, which has size difference $\leq$ 14; only 29% of graph pairs are the same size. Fig. 4-right shows the significant advantage of the proposed method for handling varying size differences. As the size difference increases, the F1 score of Hungarian+NGMv2 decreases, demonstrating inability to handle large size disparities. Hungarian+GCAN improves on this considerably while ILP+GCAN gives consistently high performance. We show in Fig. 5 that 82% of graph pairs in our test set (16445/20000) have at least one invisible node, and our performance remains strong as the total number of invisible nodes increases.

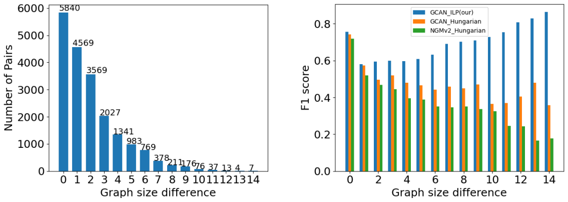


Figure 4: L: size distribution. R: impact of size-variance.

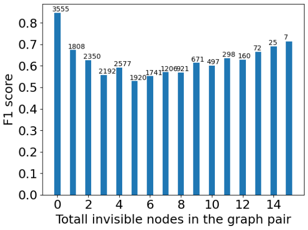
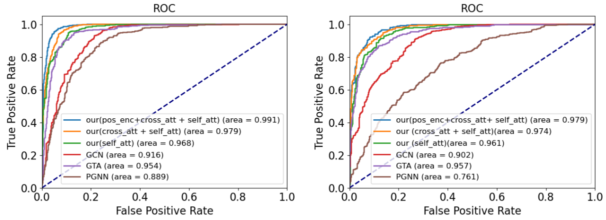


Figure 5: Distribution of invisible nodes, showing its impact on the performance of our approach.

As shown in Tab. 4, we conduct four ablation experiments on the Pascal VOC dataset to quantitatively evaluate the effectiveness of each component in the proposed architecture. All experiments use our proposed ILP attention loss which has shown superior performance in the previous experiments. The baseline in experiment A is a basic VGG16 network, and a inner-product similarity is used to measure the similarity between nodes. Each major component of our Graph-context Attention Network, i.e. attention, positional encoding and self attention, is evaluated in experiments B, C and D respectively. We can observe that cross-attention is critical for graph matching. Positional encoding and self-attention further improve the accuracy considerably.



(a) Pascal VOC

(b) Hand vessel

Figure 6: ROC curves of different state-of-the-art graphic models evaluated on Pascal VOC and hand vessel datasets

4.3. Graph-level matching

Dataset To evaluate the proposed method for graph-level matching, we use the Pascal VOC dataset and the Bosphoros vessel graph dataset [31], which contains 1575 near-infrared images of the dorsal hand from 100 subjects. For the Pascal VOC dataset, we adopt the same data partitioning as in the Keypoint matching experiment. The vessel graph dataset is split to 600 images from 50 people for training and 600 images from another group of 50 people for testing. We then generate training/testing positive and negative pairs by randomly picking image pairs with the same/different class. Moreover, we also evaluate the node matching accuracy on Pascal VOC dataset without any node correspondence groundtruth for training.

Table 3: F1 score (%) on Pascal VOC Keypoint with size-varied graphs.

method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbkie	person	plant	sheep	sofa	train	tv	mean F1
BBGM [8]	42.7	70.9	57.5	46.6	85.8	64.1	51.0	63.8	42.4	63.7	47.9	61.5	63.4	69.0	46.1	94.2	57.4	39.0	78.0	82.7	61.4
qc-DGM_1 [9]	30.1	59.1	48.6	40.0	79.7	51.6	32.4	55.4	26.1	52.1	47.0	50.1	56.8	59.9	27.6	90.4	50.9	33.1	71.3	78.8	52.0
qc-DGM_2 [9]	30.9	59.8	48.8	40.5	79.6	51.7	32.5	55.8	27.5	52.1	48.0	50.7	57.3	60.3	28.1	90.8	51.0	35.5	71.5	79.9	52.6
NHGM-v2 [14]	41.3	66.3	49.1	40.2	88.9	58.0	43.0	52.1	41.2	54.7	38.7	50.2	52.8	61.5	41.7	91.4	41.1	43.7	66.5	74.7	54.8
ours	**45.0**	**66.7**	**60.6**	**49.7**	**89.7**	**66.3**	**65.2**	64.9	**45.5**	**66.9**	**54.4**	**63.1**	**62.5**	**63.5**	**55.0**	**96.1**	**63.5**	**49.7**	**80.6**	**83.6**	**64.6**

Table 4: Ablations study of the proposed architecture.

Exp.	Setup	Accuracy
A	baseline+inner-product similarity	58.1
B	A+cross-attention	70.2
C	B+positional encoding	79.4
D	C+self-attention	82.3

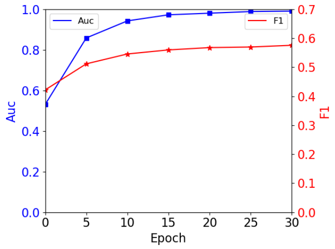


Figure 7: Auc/F1 vs. training epoch on Pascal VOC dataset.

Baseline methods and metrics In these experiments, we explore two questions: (i) Is GCAN able to facilitate graph-level matching? (ii) Without ground truth node-level labeling, can GCAN reason node matching by learning graph similarity. To address the first question, we compare our model against several state-of-the-art graphic models, such as GCN [5], GTA [16] and PGNN [17], which have shown good performance on graph-level classification. Their performance are evaluated using pair AUC, i.e. area under the ROC curve for classifying pairs of graphs as similar or not. To investigate the second question, we only use graph-level training loss (Eq. 16) to train our GCAN and then report its node-level matching accuracy on Pascal VOC dataset.

Results Fig. 6 shows the ROC curves and AUC of the different SOA graphic models evaluated on the Pascal VOC and Bosphorus [31] datasets. We firstly explore the ability of our GCAN for node-level matching without node correspondence supervision, and show AUC/F1 vs. training epoch on Pascal VOC in Fig. 7. Both AUC and F1 ascend during training, supporting our claim that node-to-node similarity can be learned by reasoning on graph-level matching without node correspondence supervision. From Fig. 6, we can also observe that our model built with cross-attention and self-attention consistently outperforms our model with self-attention only and competing models.

4.4. Conclusion

This paper has presented a novel neural graph matching network that produces more discriminative node features by capturing intrinsic graph structure and cross-graph information. To address the size-varied graph matching problem, we formulate it as an Integer Linear Programming Problem. The proposed neural graph matching network trained by our ILP attention is shown to outperform other competing methods, particularly for size-varied graph matching problem. Furthermore, we extend the proposed network to learn none-to-node matching with graph-level training loss. Our experiments and ablation studies on three public keypoint-matching datasets and the vessel graph dataset demonstrate the state-of-the-art performance of our method.

Broader Impact

Our proposed approach directly addresses a general graph matching problem, where graph pairs may contain different numbers of nodes. This paper goes beyond object identification and comparison by node-to-node correspondence into real world applications by not requiring prior knowledge of corresponding nodes. This leads to many potential new applications such as pattern matching for person identification, with the potential for perpetrator identification. As with all biometric approaches, there is potential for misuse. This work can lead to new developments in 3D and 4D reconstruction as well as shape matching, tracking and identification with wider applications in many areas including security, pharmaceutical manufacturing, and astronomy.

Limitations

While the proposed approach achieves impressive speeds for image comparison and is able to handle the understudied size-varied graph matching problem, the complexity of much larger graphs that are not typically associated with this problem would result in increased complexity and training time. It would be prudent to address this issue in future research for other applications.

Acknowledgments

The work in this publication is supported by funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 787768).

References

[1] A. Zanfir and C. Sminchisescu, “Deep learning of graph matching,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2684–2693, 2018. 1, 4, 6, 7

[2] D. Zhong, H. Shao, and X. Du, “A hand-based multi-biometrics via deep hashing network and biometric graph matching,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 12, pp. 3140–3150, 2019. 1

[3] X. Bai, *Graph-Based Methods in Computer Vision: Developments and Applications: Developments and Applications*. IGI global, 2012. 1

[4] E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido, “A survey for the quadratic assignment problem,” *European journal of operational research*, vol. 176, no. 2, pp. 657–690, 2007. 1

[5] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *arXiv preprint arXiv:1609.02907*, 2016. 1, 2, 3, 8

[6] R. Wang, J. Yan, and X. Yang, “Learning combinatorial embedding networks for deep graph matching,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3056–3065, 2019. 1, 2, 4, 5, 6, 7

[7] T. Yu, R. Wang, J. Yan, and B. Li, “Learning deep graph matching with channel-independent embedding and hungarian attention,” in *International conference on learning representations*, 2019. 1, 2, 6, 7

[8] M. Rolínek, P. Swoboda, D. Zietlow, A. Paulus, V. Musil, and G. Martius, “Deep graph matching via blackbox differentiation of combinatorial solvers,” in *European Conference on Computer Vision*, pp. 407–424, Springer, 2020. 1, 2, 3, 6, 7, 8

[9] Q. Gao, F. Wang, N. Xue, J.-G. Yu, and G.-S. Xia, “Deep graph matching under quadratic constraint,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5069–5078, 2021. 1, 2, 4, 5, 6, 7, 8

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, pp. 5998–6008, 2017. 1, 3

[11] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, “Transformer-xl: Attentive language models beyond a fixed-length context,” *arXiv preprint arXiv:1901.02860*, 2019. 1, 3

[12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv preprint arXiv:2010.11929*, 2020. 1, 3

[13] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, “Training data-efficient image transformers & distillation through attention,” in *International Conference on Machine Learning*, pp. 10347–10357, PMLR, 2021. 1, 3

[14] R. Wang, J. Yan, and X. Yang, “Neural graph matching network: Learning lawler’s quadratic assignment problem with extension to hypergraph and multiple-graph matching,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 1, 2, 3, 4, 5, 6, 7, 8

[15] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1025–1035, 2017. 2, 3

[16] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” *arXiv preprint arXiv:1710.10903*, 2017. 2, 3, 8

[17] J. You, R. Ying, and J. Leskovec, “Position-aware graph neural networks,” in *International Conference on Machine Learning*, pp. 7134–7143, PMLR, 2019. 2, 3, 8

[18] M. Rolínek, V. Musil, A. Paulus, M. Vlastelica, C. Michaelis, and G. Martius, “Optimizing rank-based metrics with blackbox differentiation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7620–7630, 2020. 2

[19] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?,” *arXiv preprint arXiv:1810.00826*, 2018. 3

[20] M. Fey, J. E. Lenssen, F. Weichert, and H. Müller, “Splinecnn: Fast geometric deep learning with continuous b-spline kernels,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 869–877, 2018. 3

[21] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, “Cbam: Convolutional block attention module,” in *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018. 4

[22] A. Nowak, S. Villar, A. S. Bandeira, and J. Bruna, “Revised note on learning quadratic assignment with graph neural networks,” in *2018 IEEE Data Science Workshop (DSW)*, pp. 1–5, IEEE, 2018. 4

[23] T. Wang, H. Liu, Y. Li, Y. Jin, X. Hou, and H. Ling, “Learning combinatorial solver for graph matching,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 7568–7577, 2020. 4, 5, 6, 7

[24] P. M. Narendra and K. Fukunaga, “A branch and bound algorithm for feature subset selection,” *IEEE Transactions on computers*, vol. 26, no. 09, pp. 917–922, 1977. 5

[25] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014. 6

[26] R. Wang, J. Yan, and X. Yang, “Combinatorial learning of robust deep graph matching: an embedding based approach,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 6, 7

[27] M. Cho, K. Alahari, and J. Ponce, “Learning graphs to match,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 25–32, 2013. 6

[28] L. Bourdev and J. Malik, "Poselets: Body part detectors trained using 3d human pose annotations," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 1365–1372, IEEE, 2009. 6

[29] J. Min, J. Lee, J. Ponce, and M. Cho, "Spair-71k: A large-scale benchmark for semantic correspondence," *arXiv preprint arXiv:1908.10543*, 2019. 6

[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 6

[31] A. Yüksel, L. Akarun, and B. Sankur, "Biometric identification through hand vein patterns," in *2010 International Workshop on Emerging Techniques and Challenges for Hand-Based Biometrics*, pp. 1–6, IEEE, 2010. 7, 8