Pattern Recognition

# Position-aware and Structure Embedding Networks for Deep Graph Matching

--Manuscript Draft--

| | |
|---|---|
| **Manuscript Number:** | PR-D-22-00983R2 |
| **Article Type:** | Full Length Article |
| **Section/Category:** | Structural pattern recognition, network and graph methods |
| **Keywords:** | Graph Matching; graph embedding; Deep Neural Network |
| **Corresponding Author:** | dongdong chen, Ph.D.<br>Shanghai Jiao Tong University School of Biomedical Engineering<br>Shanghai, CHINA |
| **First Author:** | dongdong chen, Ph.D. |
| **Order of Authors:** | dongdong chen, Ph.D. |
| | Yuxing Dai |
| | Lichi Zhang |
| | Zhihong Zhang |
| | Edwin R. Hancock |
| **Abstract:** | Graph matching refers to the process of establishing node correspondences based on edge-to-edge constraints between graph nodes. This can be formulated as a combinatorial optimization problem under node permutation and pairwise consistency constraints. The main challenge of graph matching is to effectively find the correct match while reducing the ambiguities produced by similar nodes and edges. In this paper, we present a novel end-to-end neural framework that converts graph matching to a linear assignment problem in a high-dimensional space. This is combined with relative position information at the node level, and high-order structural arrangement information at the subgraph level. By capturing the relative position attributes of nodes between different graphs and the subgraph structural arrangement attributes, we can improve the performance of graph matching tasks, and establish reliable node-to-node correspondences. Our method can be generalized to any graph embedding setting, which can be used as components to deal with various graph matching problems answered with deep learning methods. We validate our method on several real-world tasks, by providing ablation studies to evaluate the robustness and generalization capability across different categories. We also compare state-of-the-art alternatives to demonstrate performance. |
| **Suggested Reviewers:** | Jianjia Wang<br>Shanghai University<br>jianjiawang@shu.edu.cn |
| | Lu Bai<br>Central University of Finance and Economics<br>bailucs@cufe.edu.cn |

Title Page

# Position-aware and Structure Embedding Networks for Deep Graph Matching

Dongdong Chen (*PhD student*, Shanghai Jiao Tong Univ., China),

Yuxing Dai(Master student, Xiamen Univ., China),

Lichi Zhang (*PhD*, Shanghai Jiao Tong Univ., China),

Zhihong Zhang (PhD, Xiamen Univ., China),

Edwin R. Hancock (PhD, York Univ., UK)

May. 2022

Manuscript Draft Submitted to *Pattern Recognition*

Highlights

- This paper introduces the relative position information related to the matching of key points by constructing an anchor set for small graphs extracted from images, which is shown to be effective in graph matching experiments.
- This paper introduces subgraph arrangement structure information into the graph matching task and implements a hierarchical and comprehensive extraction of image features.
- The proposed position-aware node embedding module and subgraph-based structural embedding module can be added to existing deep learning graph matching methods, further boosting the graph matching performance.

## Comments from EiC:

Q1. Make sure your title is succinct and grammatical. It should ideally not exceed 10-15 words.

**Response:** Checked with no problem.

Q2. Make sure your conclusions reflect on the strengths and weaknesses of your work, how others in the field can benefit from it and thoroughly discus future work. The conclusions should be different in content from the abstract and be rather longer too.

**Response:** Have added a separate section detailing the limitations of the study as required by reviewer3 and checked with no problem.

Q3. Take a careful look at your bibliography and how you cite papers listed in it. Make sure it is current and cites recent work. Please cite a variety of different sources of literature. Please do not make excessive citation to arXiv papers, or papers from a single conference series. Do not cite large groups of papers without individually commenting on them. So we discourage " In prior work [1,2,3,4,5,6] …". Your bibliography should only exceptionally exceed about 40 items.

**Response:** Checked with no problem.

Q4. Make sure the revised version is relevant to the readership of Pattern Recognition. To this end, please make sure you cite RECENT work from the field of pattern recognition that will be relevant to our readership.

**Response:** Checked with no problem.

Q5. Do not exceed the page limits or violate the format, i.e. double spaced SINGLE column with a maximum of 35 pages for a regular paper and 40 pages for a review.

**Response:** Checked with no problem.

## Comments from R#3:

**Q1.** A separate section detailing the limitations of the study (theory/methods/results) may further strengthen the paper.

**Our Response:** Thank you for your suggestions. We have added a separate section detailing the limitations of the study. The changed contents of the revised manuscript are as follows:

**This work aims to address the issue of ambiguous matching that results from the absence of discriminability based on local structure and the semantic similarity of different nodes. We therefore focus on the feature extraction problem during the graph-matching task. However, we do not consider issues arising from the fact that the image itself may contain noise and other abnormalities. Our position-aware node embedding module and subgraph-based structural embedding module are adaptive plug-ins that can boost the performance of other methods too. However, this is at the expense of additional computing time consumption and memory usage.**

**Future work will aim to investigate the noisy image-matching task so that it can be made more practically applicable. In addition, and from a problem-solving perspective, we can consider how to deal with matching graphs that have more complicated structural relations. These include hypergraphs and directed graphs or using various relative position and subgraph relational strategies. (Correction appears on Page 31.)**

# Position-aware and Structure Embedding Networks for Deep Graph Matching

Dongdong Chen[a], Yuxing Dai[b], Lichi Zhang*[a], Zhihong Zhang[b], Edwin R. Hancock[c]

[a]*Shanghai Jiao Tong University, Shanghai, China*
[b]*Xiamen University, Xiamen, China*
[c]*University of York, York, UK*

**Abstract**

Graph matching refers to the process of establishing node correspondences based on edge-to-edge constraints between graph nodes. This can be formulated as a combinatorial optimization problem under node permutation and pairwise consistency constraints. The main challenge of graph matching is to effectively find the correct match while reducing the ambiguities produced by similar nodes and edges. In this paper, we present a novel end-to-end neural framework that converts graph matching to a linear assignment problem in a high-dimensional space. This is combined with relative position information at the node level, and high-order structural arrangement information at the subgraph level. By capturing the relative position attributes of nodes between different graphs and the subgraph structural arrangement attributes, we can improve the performance of graph matching tasks, and establish reliable node-to-node correspondences. Our method can be generalized to any graph embedding setting, which can be used as components to deal with various graph matching problems answered with deep learning methods. We validate our method on several real-world tasks, by providing ablation studies to evaluate the robustness and generalization capability across different categories. We also compare state-of-the-art alternatives to demonstrate performance.

*Keywords:* Graph Matching, Graph Embedding, Deep Neural Network

*Corresponding author: lichizhang@sjtu.edu.cn (Lichi Zhang)

## 1. Introduction

Graph matching aims to establish the correspondence between two or more graphs based on their structural information [1]. It is widely used in combinatorial optimization, machine learning and computer vision due to its natural representation and convenient coding of the relationship between abstract data. Specifically, many applications can be implemented using the graph matching technique, such as image registration in medical image analysis [2], linking user accounts in social network analysis [3], image extrapolation in computer vision [4], finding coherent motions and semantic regions in crowd scenes [5].

There are many methods and matching solvers developed for addressing the graph matching problem [6, 7]. These can be divided into different categories depending on the specific perspectives adopted. For example, based on the number of graphs included, there are graph-to-graph matching and multi-graph matching methods [8]. Since graph-to-graph matching is the basis of multi-graph matching, improving the matching ability between two graphs can also be extended to improve multi-graph matching. Therefore, this paper focuses on the matching of pairs of graphs. Based on matching content, they can be divided into a) structure information based graph matching and b) semantic information based graph matching [9]. Note that in graph matching, especially for image-generated graphs, semantic features based on structural information have been proved to be important for classifying nodes [10]. In this way, the graph matching problems studied in this paper are based on both node feature information and structural information from the graphs. On the other hand, graph matching methods can also be divided into a) learning-based methods and b) learning-free methods [11]. Notably, learning-free methods generally seek approximate solutions for a given fixed affinity model, which is usually cast in the form of simple parameters and is approximately solved by the joint similarity of individual entities together with their mutual relationships [12]. However, their time complexity is often too high to satisfy and solve large-scale

real-world problems.

In the case of deep learning-based graph matching solvers, the common feature of most methods is that they compromise in their evidence combining strategy in the sense that the resulting unclear combinatorial element would not be competitive in a purely combinatorial setup [13]. However, these methods can only calculate the similarity score of the entire graph, and heavily rely on inefficient global matching procedures [14]. Additionally, they only consider the embedding of local information for nodes in the graph, which tend to ambiguously match similar nodes in different regions of the graph in an inconsistent manner [15]. Fig. 1 illustrates an example of the main idea of our relative position-aware information embedding strategy. Here it is shown that it fails to match the cats ear positions correctly, since the node embedding information usually relies only on node semantic information, and the local structure information lacks effective discrimination ability. Therefore it is difficult to unambiguously distinguish the left and right ears of a cat. In this example, the relative position information is critical to graph matching, especially when both the semantic and structural information are very similar in the two regions of the graphs.

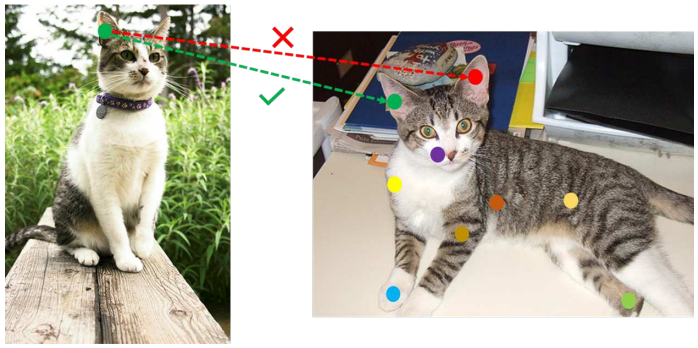

Figure 1: An exemplar failure case to establish the correct match of cat ears between two images, which are selected from the Pascal VOC dataset [16].

To address the above-mentioned challenges, here we propose a novel ap-

proach to solve the graph matching problem. We commence by utilizing node features extracted from the images to attribute the nodes, and design a position-aware node embedding algorithm to capture the relative position information of nodes in the graph. We further enhance the representation of local information of the nodes by merging a structural arrangement information representation of the subgraph in which the node is located. With the graph node-wise embedding to hand, we aim to obtain the required graph node permutations for node-to-node correspondence from raw pixel inputs. To our best knowledge, there are no methods that consider embedding both position and structural information for the graph matching problem.

The main contributions of our works are as follows:

1. Relative position information has not been considered in the existing process of graph matching, which hampers their applications in terms of reducing the ambiguity of matching accuracy. We propose to construct an anchor set for small graphs extracted from images, and fully consider the relative position information related to the matching of key points, which is also proved to be effective in experiments.

2. We propose a novel subgraph detection and graph representation method, and extract subgraph structural arrangement information for improved node embedding. Previous graph matching studies only match using node level and edge level information, but ignore the higher-order neighborhood structure information. Here we introduce subgraph arrangement structure information into the graph matching task, by capturing it using the subgraph structure. This arrangement procedure combines node and edge level information with relative position information. We can therefore implement a hierarchical and comprehensive extraction of image features, which further improves the matching accuracy.

3. We demonstrate in our experiments that our method outperforms alternative work on several widely-studied real-world datasets. In the ablation study, we also show the effectiveness of the introduced components of our new method of graph matching on these datasets. We show that our position-aware

node embedding module and subgraph-based structural embedding module can be added to existing deep learning graph matching methods, to further boost their classification performance.

The remainder of the paper is organized as follows: Section 2 provides a brief review of the related work on image feature extraction, graph embedding, and combinatorial optimization. Section 3 defines the specific graph matching problem that we intend to resolve in this work. Section 4 provides the details of our method, and the corresponding experiments to validate our new method are presented in Section 5. Finally, we conclude the paper and provide an extended discussion of potential future work in Section 6.

## 2. Related Work

In this section, we review existing deep learning solvers in dealing with graph matching problems.

### *2.1. Image Feature Extraction*

Feature extraction refers to the computation of higher-level features from the original pixels in an image. It can capture the differences between pixels of various object classes. Conventional feature extraction usually proceeds in an unsupervised manner, and no image category labels are used when extracting information from the pixels. Commonly used traditional features include GIST (Generalized Search Trees) [17], SIFT (Scale-Invariant Feature Transform) [18], and LBP (Local Binary Patterns) [19].

However, these hand-crafted feature extraction methods can not be optimized according to the training from images and their labels. However, they fail to provide more comprehensive feature information to better represent important image information. The method based on deep learning overcomes this shortcoming using a soft-coded feature extractor [20]. Recently, the CNN model has been developed and enhanced in detail. For example, Szegedy *et al.* [21] significantly increase the depth of the CNN in their GoogleNet with three different types of the convolution operation. He *et al.* [22] have proposed the

residual neural network (ResNet) which includes a cross-layer connection that passes information from the input across deeper layers. It also adds to the result of convolution by introducing shortcut connections to solve the problem of vanishing loss gradient.

In this paper, we use VGG16 [23] to extract the image features as graph nodes, which is also presently the most widely-used CNN model. It can adapt to different sizes of the convolutional kernel to capture more discriminative decision functions and with fewer parameters. It has been proved effective in deep visual feature extraction.

### *2.2. Graph Embedding*

At present, most studies related to graph embedding focus on static graphs, which can be divided into (a) spectral-based and (b) spatially-based methods.

The earliest graph embedding method provides local convolutions of graph structure data in the spectral frequency domain. The Fourier transform of the graph is used to transform the time domain data into a frequency domain signal. Then the frequency domain signal is convolved with local features. Finally, the frequency domain signal is transformed back into the time domain [24]. However, spectral methods can only solve the problem for undirected graphs. This means it is limited to satisfy the constraint of a symmetric transformation matrix.

Spatial approaches have an advantage over spectral approaches in that they can operate on problems where the graph structure varies significantly in the dataset being studied [25]. However, they generally require sophisticated data transformations to enable effective learning. For example, given the strong hypothesis that the existence of node-connected edges are based on sometimes ad-hoc and contrived or artificial settings, Zhang *et al.* [26] extracted local subgraphs around each target link. They use this to learn the functional mapping of subgraph patterns to infer the existence of links, and to automatically learn heuristic algorithms.

In this paper, we develop a novel graph embedding method to extract high-level information combined with the relative position of nodes, which can greatly assist graph matching.

### *2.3. Combinatorial Optimization*

Mathematically, the graph matching problem can be considered a combinatorial optimization problem, which is also an NP-Hard problem. Using mathematical methods to solve graph matching problems can be divided into relaxation optimization and matrix decomposition methods [27]. For the spectral relaxation optimization method, the computation speed is relatively fast, but the constraints on the obtained solution are often over relaxed.

With the rise of deep learning, the use of reinforcement learning and neural networks to solve combinatorial optimization problems has become a topical trend. The Sinkhorn network [28] can solve the linear assignment problem, by performing the doubly stochastic normalization operations of row normalization and column normalization on all non-negative matrices to minimize the cost. Furthermore, it is an approximate Hungarian algorithm [29] that can optimize polynomial complexity. Recently, Patrini *et al.* [30] proposed a Sinkhorn autoencoder to achieve the target of minimizing distance, and utilizing reinforcement learning to solve the combinatorial optimization problem. Furthermore, a novel and simple end-to-end training framework was proposed by Rolínek *et al.* [13], which incorporates the most advanced graph matching combinatorial solver.

In this paper, we develop a novel deep learning method to solve the combinatorial optimization problem. It can effectively reduce the dimensionality of the quadratic assignment problem for combinatorial optimization to a linear assignment problem.

## 3. Problem Definition and Notations

In this paper, we consider the graph $\mathcal{G} = (V, A, X)$ which is consist of a finite set of nodes $V$, an adjacency matrix $A$, and a set of attributes $X$ for the nodes,

which originate from images using various CNN models [23, 31]. The goal of graph matching is to establish a correspondence between two attributed graphs, which minimizes the sum of local and geometric costs of assignment between the vertices of the two graphs.

Let $\mathcal{G}_s = (V_s, A_s, X_s)$ be a source graph, with $|V_s| = n$, $A_s \in \{0,1\}^{n\times n}$, and feature matrix $X_s \in \mathbb{R}^{n\times F}$, where $F$ represents the $F$-dimensional feature vector of nodes in graph stacked to columns. The target graph to be matched is $\mathcal{G}_t = (V_t, A_t, X_t)$, with $|V_t| = m$, $A_t \in \{0,1\}^{m\times m}$, and feature matrix $X_t \in \mathbb{R}^{m\times F}$, whose w.l.o.g. $n \leq m$. We also construct a vector of length $nm$. The element $\mathbf{v} \in \{0,1\}^{nm\times 1}$ indicates the match of vertices in two graphs, where $\mathbf{v}_{i,j} = 1$ if vertex $i \in V_s$ is matched to vertex $j \in V_t$ and $\mathbf{v}_{i,j} = 0$ if otherwise. It is worth noting that all the vertex matches are subject to the one-to-one mapping constraints, i.e. $\sum_{j\in V_t} \mathbf{v}_{i,j} = 1\ \forall i \in V_s$ and $\sum_{i\in V_s} \mathbf{v}_{i,j} \leq 1\ \forall j \in V_t$.

Furthermore, we construct a square symmetric positive matrix $M \in \mathbb{R}^{nm\times nm}$ referred to as the affinity matrix, to encode the edge-to-edge affinity between two graphs by their off-diagonal elements. Specifically, $M_{ip;jq}$ measures how well edge $(i,j)$ in graph $\mathcal{G}_s$ matches with edge $(p,q)$ in graph $\mathcal{G}_t$ with $\{i! = p\} \cup \{j! = q\}$. The diagonal entries of the affinity matrix can also indicate the node-to-node affinity between two graphs. Therefore, the pairwise graph matching between $\mathcal{G}_s$ and $\mathcal{G}_t$ can be formulated as the edge-preserving, quadratic assignment programming (QAP) [27] problem:

$$\begin{aligned} &\underset{\mathbf{v}}{\operatorname{argmax}}\ \mathbf{v}^\top M \mathbf{v} \\ &\text{s.t.}\ \sum_{j\in V_t} \mathbf{v}_{i,j} = 1\ \forall i \in V_s, \sum_{i\in V_s} \mathbf{v}_{i,j} \leq 1\ \forall j \in V_t, \end{aligned} \tag{1}$$

where $\mathbf{v} \in \{0,1\}^{nm\times 1}$ and $M \in \mathbb{R}^{nm\times nm}$. In this work, we intend to resolve the graph matching problem based on the supervised matching of graphs, and we aim to learn an end-to-end model which can effectively extract graph information and match through given pair-wise ground-truth correspondences for a set of graphs, and generalize to unseen graph pairs.

## 4. Methods

In this section, we provide the details of the proposed end-to-end position-aware and structure-based graph matching model, with the overall pipeline shown in Fig. 2. Note that the deep feature extractor of the proposed method can be any CNN model, such as VGG16 [23], which is adapted to convert images into graphs with features. These graphs are the source and target graphs, which are the paired input of our model.

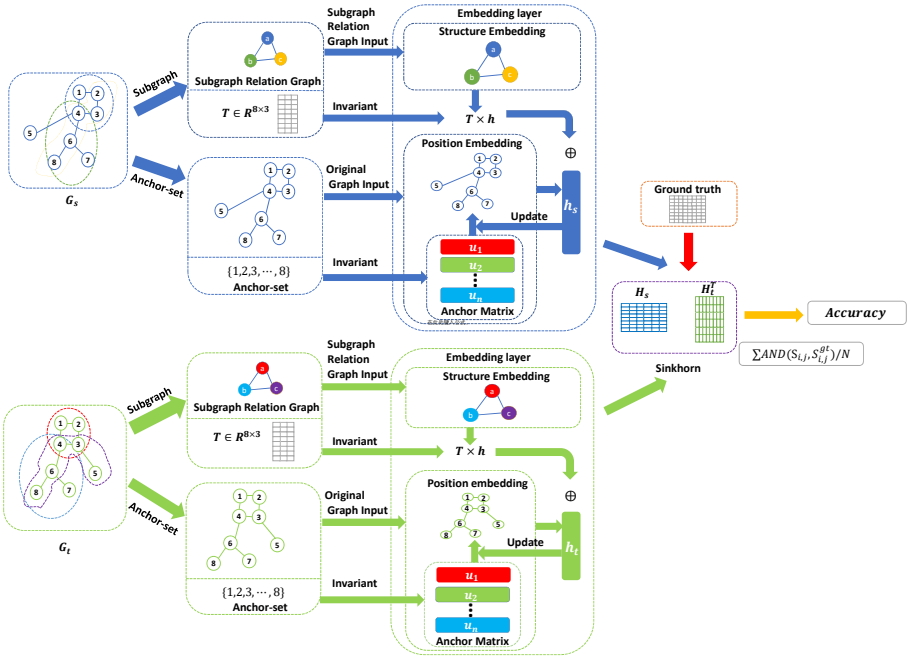


Figure 2: Overview of the end-to-end position and structure embedding networks for deep graph matching. The blue source graph $\mathcal{G}_s$ and the green target graph $\mathcal{G}_t$ are extracted with the node-wise graph feature representation at high-level through the two frameworks of position-aware node embedding and subgraph based structure embedding, which can be added to the existing deep learning graph matching methods. The graph feature is matched through matrix product and Sinkhorn function and is compared with the ground truth. The final output of the model is the matching accuracy of the pair of nodes between the source graph and the target graph.

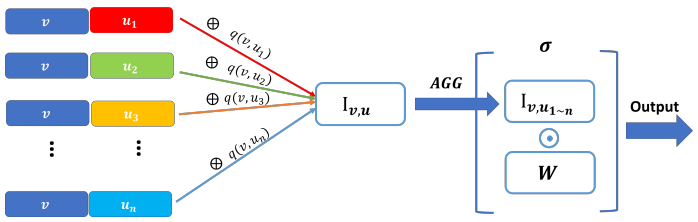


Figure 3: Procedure of Position Embedding.

The model consists of three main components, namely a) a position-aware node embedding module, b) a subgraph-based structure embedding module, and c) a graph feature matching module. The aim of the position-aware node embedding step, as described in Section 4.1, is to consider the information available from the relative structure. The second step is to iteratively combine the subgraph structure information described in Section 4.2, and the final part is the matching of two graphs with extracted graph representation described in Section 4.3.

### *4.1. Position-aware Node Embedding*

We establish our position-aware node embedding procedure with two key components: First, before the node embedding process, we propose a simple but effective strategy to construct an anchor-set for each graph which is used throughout the entire learning process. Here, we refer to the nodes used as reference position coordinates as *anchors*, and the set of these nodes as the *Anchor-set*. We aggregate the information of the node and each anchor in the anchor-set, in this way we replace the original graph embedding method that relies on message passing between local network neighborhoods. Second, we design an information aggregation mechanism for message aggregation between nodes and anchors in the anchor-set. With the addition of the position-aware node embedding, we can effectively eliminate the ambiguity-prone mismatching of similar nodes in different positions.

#### *4.1.1. Anchor-set Construction for Small Graph*

The position anchor framework is motivated by PGNN [32] which emphasizes large graphs (i.e. those consisting of at least thousands of nodes). However, in this paper, we further study the anchor mechanism and focus on small graphs constructed from images. Specifically, the anchor-sets first proposed for large graphs are completely randomly chosen subsets of all nodes from the graph. They are intended to sample multiple anchor-sets $P = \{P_{i,j}\}$, where the number of anchor-sets $|P| = \log^2 |V|$, $i = 1, 2, ..., \log |V|$, $j = 1, 2, ...c \log |V|$. Here, $|V|$ is the number of nodes in the graph and c is a hyper-parameter. Each node is sampled independently and randomly with a probability of $1/2^i$. The idea central to our method is that the anchor-sets can include each node or any of its one-hop neighbors. Large anchor sets have low position information with a large sampling probability. Small anchor sets have high position information with low sampling probability. The balance of these different sizes of anchor-sets may lead to efficient embeddings. However, this random sampling operation can result in the final anchor sets sampling almost all nodes in the graph. The proof is as follows:

*The sampling probability for each node in the anchor-set $P_{i,j}$ is $1/2^i$, and the number of anchors in each anchor-set $P_{i,j}$ is*

$$p_i = \frac{|V|}{2^i} \quad \forall i \in [1, \log |V|], \tag{2}$$

*where $|V|$ is the number of nodes in the graph.*

*Since the number of anchor points $p_i$ in the anchor-set $P_{i,j}$ is only related to element $i$, the construction of anchor-sets $P = \{P_{i,j}|(i \in [1, \log |V|], j \in [1, c \log |V|])\}$ is actually the repeated construction of anchor-sets $P = \{P_{i,j}|(i \in [1, \log |V|], j = 1)\}$ for $c \log |V|$ multiplied by the uniform sampling probability for each anchor in each anchor set construction step. Hence, we consider each anchor in a non-repetitive operation, i.e. the anchors in anchor-sets $P = \{P_i|(i \in [1, \log |V|])\}$. Therefore, the number of anchors p in the entire set of anchor-sets $P = \{P_i|(i \in [1, \log |V|])\}$ is given by:*

$$p = \sum_{i=1}^{log|V|} p_i = |V| - 1. \tag{3}$$

*The anchor sets obtained through fair non-repetitive sampling operation contain* $|V| - 1$ *anchors for a* $|V|$ *node graph and the sampling probability for each anchor is* $(|V| - 1)/|V|$*, which tends to unity for large graphs.*

Based on the derivation above, we propose an effective strategy to construct an anchor set $P$ consisting of all nodes in the graph, denoted by $P = V$, and serving as a stable reference for all nodes. According to our central idea in this paper, when all nodes in a graph become an anchor in an anchor-set, each node aggregates information with anchors based on a relative position relation. This is equivalent to each target node being subject to a relative position-based attention mechanism based on the global arrangement of nodes in the graph.

*4.1.2. Aggregation of Node Embedding*

With the anchor-set selected, we continue to compute the position-aware node embedding through the information aggregation between the nodes, in which each anchor cooperates with its relative position attributes. Here, we first compute the relative position coefficient $q_{v,u}$ between a pair of nodes $v$ and $u$ as:

$$q_{v,u} = \frac{e^{-d_{v,u}}}{\sum_{i \in V} e^{-d_{v,i}}}, \tag{4}$$

where $d_{v,u}$ is the shortest path distance between the two nodes. Considering the time complexity of calculating the shortest path, we stipulate that the maximum shortest path distance is not more than $r$, otherwise, the value is infinite. In this way, we can effectively eliminate the interference of anchors that are too far away from the node when the coefficient $q_{v,u}$ equals zero. We define the information aggregation function $I(v,u)^{(k)}$ for $k$-th layer between nodes $v$ and $u$ as:

$$I(v,u)^{(k)} = q_{v,u}\text{CONCAT}(h_v^{(k-1)}, h_u^{(k-1)}), \tag{5}$$

where $q_{v,u}$ is the relative position coefficient between node $u$ and $v$, $h_v^{(k-1)}$ and $h_u^{(k-1)}$ are hidden representations containing the position information in $(k-1)$-th layer. The message aggregation function CONCAT concatenates the hidden representations $h_v^{(k-1)}$ and $h_u^{(k-1)}$.

We further aggregate the information of each pair of nodes and the anchor with a non-linear transformation applied after the aggregation to achieve higher expressive power. The hidden representation passed to the next layer is:

$$h_v^{(k)} = \sigma(\text{AGG}(I(v,u)^{(k)} |\ \forall u \in V) W^{(k)}), \tag{6}$$

where AGG is a permutation-invariant function (e.g., the sum), and $W^{(k)}$ is a learnable weight vector for the $k$-th layer.

The general position-aware node embedding framework is summarized in Algorithm 1.

**Algorithm 1:** Position-aware node embedding framework.

**Input:** input original graph $\mathcal{G} = (V, A, X)$, anchor-set:$S = V$, relative position coefficient $q$, trainable weight in $k$-th layer $W^{(k)}$, network layer $k \in [1, K]$

**Output:** Position-aware node embedding $h_p$

$h_v = X_v$
**for** $k \in [1, K]$ **do**
  **for** $v \in V$ **do**
    $I(v,u)^{(k)} = q_{v,u}\text{CONCAT}(h_v^{(k-1)}, h_u^{(k-1)})$
    $h_v^{(k)} = \sigma(\text{AGG}(I(v,u)^{(k)})W^{(k)}),\ \forall u \in V$
  **end**
**end**
**return** $h_v^{(K)}$

### *4.2. Subgraph-based Structure Embedding*

In addition to the relative structural information, we further introduce subgraph structure embedding during the feature extraction and the information aggregation for each node. The subgraph-based structure embedding can be split into two steps: a) We first divide the original graph into different subgraphs according to the pre-defined structure. Each subgraph can be regarded as a new supernode, and the edges between new nodes are determined by the relationships between the nodes in each subgraph, thus we can construct a so-called **subgraph relation graph**. b) In the second step, we aggregate the structural arrangement information of the subgraphs in the subgraph relation graph and merge them with the relevant nodes in the original graph while concurrently aggregating the local information at the node level of the original graph.

#### *4.2.1. Subgraph Construction*

Here we retain the global structural arrangement information from the original graph to the greatest extent, and extract the key representative structural features for the subgraphs. Note that it is particularly important to select the nodes of the subgraph relation graph, i.e. pre-defined subgraphs in original graphs. In this paper, we define three different types of subgraph structures, namely a) paths, b) trees, and c) circuits. Combinations of these three basic subgraph types can be used to construct any required graph structure.

Since the search for all subgraphs in a graph is NP-hard, we propose a novel searching method to detect the $t$-hop neighbors of nodes to find subgraphs, and then save the search path in real-time to reduce the time complexity. The corresponding detection Algorithm 2 is designed to handle how to find the three different types of subgraph structures.

For tree structure detection, we first initialize two sets: $N^t(v)$ to store the $t$-hop neighbor of node $v$, where $t \in [1, D]$ and $D$ is the maximum neighbor depth of the graph, and $P_v(u)$ stores the path from node $v$ to node $u$. We traverse the adjacency matrix to find all edges, and store the 1-hop neighbor

**Algorithm 2:** The searching algorithm for the three different types of subgraph structure.

**Input:** input original graph $\mathcal{G} = (V, A, X)$, max neighbor depth $D$, min node leaf $\delta$

**Output:** output subgraph set $S$

initialize subgraph set $S = \{S_i\}$, neighbor set $\{N^t(v)\}$, path set $\{P_v(u)\}$, subgraph relation graph $\hat{\mathcal{G}}$ with $|\hat{V}|$ vertices
**for** $t \in [1, D]$ **do**
  **for** $v \in V$ **do**
    **if** $t == 1$ **then**
      **for** $u \in V$ **do**
        Add $u, v$ into $N^1(v), N^1(u)$
        Add $v, u$ into $P_v(u), P_u(v)$
        **if** $|N^1(v)| \geq \delta$ **then**
          add $(N^1(v) + [v])$ as Tree Subgraph into $S$
        **end**
      **end**
    **else**
      **for** $u \in N^i(v) | 2^i < t \leq 2^{i+1}$ **do**
        **for** $r \in N^{t-2^i}$ **do**
          **if** $P_v(r) \in \{P_v(u)\} and\ r \notin P_v(u)$ **then**
            Add $P_v(u)$ as Circuit Subgraph into $S$
          **else**
            Add $r, v$ into $N^t(v), P_v(r)$
            Add $\{P_v(u), P_u(r)\}$ to $P_v(r)$
            Add $P_v(r)$ to $P_r(v)$
          **end**
        **end**
      **end**
    **end**
  **end**
**end**
Add $P_v(u)$ as Path Subgraph into $S$
**return** $S$

set of each node. The tree-structural subgraph is selected for further processing if the number of leaf nodes exceeds the preset threshold $\delta$. For the path and circuit subgraphs, we continue the search for the $t$-hop neighbors of each node through the combination of several existing $t$-hop neighbors. If a path from node $v$ to node $u$ already exists in the set $P_v(u)$ and there is no repeating node in the path, we identify the path as a circuit subgraph. The searching progress for the three different types of subgraph structure is shown in Fig. 4.

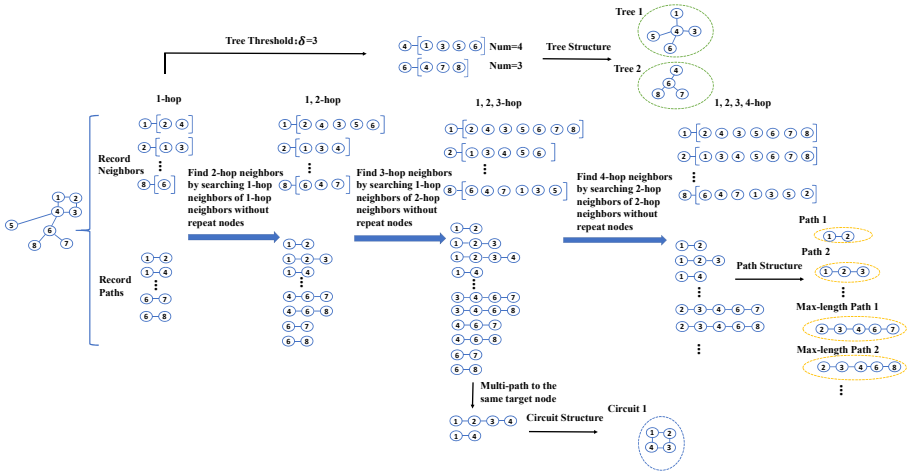


Figure 4: The searching progress for three different types of subgraph structure. We first record the 1-hop neighbor sets and select the tree structure if the number of leaf nodes exceeds the threshold. We find the $t$-hop neighbors of each node by searching the $(t-2^i)$-hop neighbor of $2^i$-hop neighbor, where $t \in [1, D]$ and $i = 0, 1, 2, ...$ If there are several paths to the same target node, the path is identified as a circuit subgraph, and the remaining paths in the path set are identified as path subgraphs.

After obtaining these three different types of different subgraphs, we construct the so-called **subgraph relation graph** to capture the internal relationship between different subgraph types. This process is shown in Fig. 5:

**Definition 1:** ***Subgraph Relation Graph.*** Let an undirected graph $\hat{\mathcal{G}} = (\hat{V}, \hat{A}, T)$ constructed from the original graph $\mathcal{G}$, in which the node set $\hat{V}$ contains $|\hat{V}|$ nodes, each denoting a different subgraph identified in the origi-

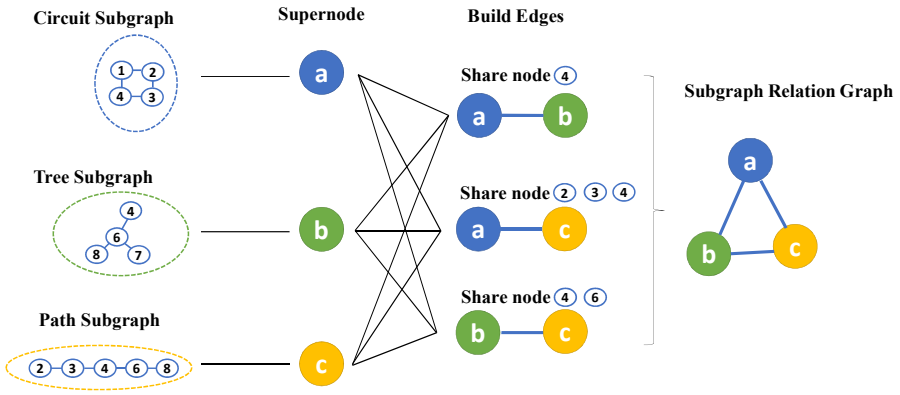


Figure 5: An example of the *Subgraph Relation Graph*.

nal graph. The adjacency matrix element of $\hat{A}$ equals to one if two nodes share the same node in the original graph. The transformation matrix $T \in \mathbb{R}^{|V|*|\hat{V}|}$ represents the matrix that transforms the original graph into the subgraph relation graph.

Let node $v_i$ belong to node-set $V$ of the source graph, and node $\hat{v_j}$ belong to node-set $\hat{V}$ in the target graph corresponding to subgraph $S_j$ in the source graph. In this way, the transformation matrix $T$ has the element:

$$T_{ij} = \begin{cases} 1 & v_i \in S_j, \\ 0 & \text{else .} \end{cases} \tag{7}$$

Algorithm 3 gives an overview of the subgraph relation graph construction procedure.

### *4.2.2. Merged Structure Embedding*

We augment the graph affinity embedding to better fuse the high-order information from the nodes by developing a merged structure message-passing scheme. We commence by aggregating the structural arrangement information of the subgraph relation graph at the subgraph-level with the classical embedding procedure:

**Algorithm 3:** Construction of the subgraph relation graph.

**Input:** input subgraph set $S$

**Output:** output subgraph relation graph $\hat{\mathcal{G}} = (\hat{V}, \hat{A}, T)$

Create a new subgraph set $\hat{S}$ by combining one of the longest path subgraphs, one of the tree subgraphs, and the circuit subgraphs from subgraph set $S$.

**for** *all* $S_i$ *in subgraph set* $\hat{S}$ **do**

  Assign $S_i$ to each node $\hat{v_i}$

  **if** $\exists v \in S_i \cap S_j$ **then**

    Add edge $\hat{A_{ij}}$ between node $\hat{v_i}$ and $\hat{v_j}$

  **end**

  **if** $v_i \in S_j$ **then**

    $T_{ij} = 1$

  **else**

    $T_{ij} = 0$

  **end**

**end**

**return** $\hat{\mathcal{G}}$

$$h_{\hat{v_i}}^{(k)} = \text{COMBINE}(h_{\hat{v_i}}^{(k-1)}, \text{AGG}\{h_{\hat{v_j}}^{(k-1)} | \hat{v_j} \in N(\hat{v_i})\}), \tag{8}$$

where $N(\hat{v_i})$ denotes the set of neighbors of node $\hat{v_i}$ (i.e. which corresponds to subgraph $S_i$ in the original graph) in the subgraph relation graph. Here, the functions AGG and COMBINE are sometimes folded into a single aggregation update. Specifically, AGG is a function that aggregates all neighbor features of the node $S_i$, and COMBINE updates the representation of node $S_i$ according to the output of AGG. We perform information aggregation and combination at the structural level. We then further merge them with the position-aware node embedding extracted from the original graph using the following procedure:

$$h_v^{(k)} = \mathrm{COMBINE_M}(h_v^{(k)}, \mathrm{AGG_M}(h_{\hat{v}_i}^{(k)} | \forall v \in S_i)), \tag{9}$$

where $h_{\hat{v}_i}^{(k)}$ denotes the hidden representation of subgraph node $S_i$ extracted using Eq. 8, $h_v^{(k)}$ represents the hidden node embedding combined with structure and position in the $k$-th layer. The function is the sum over all subgraph embeddings that contain the node $v$ and is computed as $AGG_M(h_S) = T * h_S$. We define as $\mathrm{COMBINE_M}(h_v, h_s) = \alpha h_s + \beta h_v$ where $\alpha$ and $\beta$ are two hyperparameters that can be used to balance the weight of position relationships and structural arrangement information.

The position-aware node embedding algorithm is summarized in Algorithm 4.

**Algorithm 4:** Merge structure embedding framework.

**Input:** input original graph $\mathcal{G} = (V, A, X)$, subgraph relation graph $\hat{\mathcal{G}} = (\hat{V}, \hat{A}, T)$, position-aware node embedding $h_p$, trainable weight $W$, network layer $k \in [1, K]$

**Output:** Subgraph-based Structure embedding $h_s$

$h_v = X_v$
**for** $k \in [1, K]$ **do**
  **for** $v \in V$ **do**
    $h_{\hat{v}_i}^{(k)} = \mathrm{COMBINE}(h_{\hat{v}_i}^{(k-1)}, \mathrm{AGG}\{h_{\hat{v}_j}^{(k-1)} | \hat{v_j} \in N(\hat{v}_i)\}))$
    $h_v^{(k)} = \mathrm{COMBINE_M}(h_v^{(k)}, \mathrm{AGG_M}(h_{\hat{v}_i}^{(k)} | \forall S_i \supset v))$
  **end**
**end**
**return** $h_v^{(K)}$

### *4.3. Graph Feature Matching*

With the proposed position-aware node embedding framework and subgraph-based structural embedding framework to hand, we encode each node with high-order graph structure information and position information and construct a

high-level embedding space. Such an embedding scheme allows us to simplify the second-order affinity matrix to a linear one.

With the final hidden representation of the source graph $H_s \in \mathbb{R}^{n*F'}$ and the target graph $H_t \in \mathbb{R}^{m*F'}$ to hand, we can obtain a soft correspondence between these graphs using the corresponding affinity matrix $\mathrm{H_{affinity}} = \exp\left(\mathrm{H_s \Gamma H_t^T}\right)$. Each element in the affinity matrix $\mathrm{H_{affinity}} \in \mathbb{R}^{\mathrm{n*m}}$ represents the affinity between a pair of nodes in two graphs considering node features, position-aware information, and high-order subgraph information in the graph, with $\Gamma$ as a learnable weight matrix. We further apply the Sinkhorn normalization [33] to obtain a rectangular doubly-stochastic correspondence matrix to fulfill the constraints on injectivity from the original graph $\mathcal{G}_s$ to the target graph $\mathcal{G}_t$. Here, the Sinkhorn operation works iteratively by iterative row-wise and column-wise normalization,

$$\begin{aligned} H^r_{\text{affinity}} &= H^c_{\text{affinity}} \oslash (H^c_{\text{affinity}} \mathbf{1_m} \mathbf{1_m}^T) \\ H^c_{\text{affinity}} &= H^r_{\text{affinity}} \oslash (\mathbf{1_n} \mathbf{1_n}^T H^r_{\text{affinity}}) \end{aligned} \tag{10}$$

where $\oslash$ indicates element-wise division, and $\mathbf{1_m} \in \mathbb{R}^{m*1}$, $\mathbf{1_n} \in \mathbb{R}^{n*1}$ are column vectors whose elements are all ones, $H^r_{\text{affinity}}$ and $H^c_{\text{affinity}}$ are the intermediate results of row normalization and column normalization respectively. We simplify these two iterative processes giving a continuous relaxation of the permutation matrix:

$$S = \mathrm{Sinkhorn}(\mathrm{H_{affinity}}) \tag{11}$$

We further employ the cross-entropy loss function as the permutation loss between the predicted permutation matrix and the ground truth:

$$L = -\sum_{i \in \mathcal{V}_s, j \in \mathcal{V}_t} \left( \mathbf{S}^{gt}_{i,j} \log \mathbf{S}_{i,j} + \left(1 - \mathbf{S}^{gt}_{i,j}\right) \log\left(1 - \mathbf{S}_{i,j}\right) \right), \tag{12}$$

where $\mathbf{S}^{gt}$ is the ground truth permutation matrix with each element as the ground truth node-to-node correspondence, and $S$ is the predicted permutation matrix given by Eq. 11. In this way, our cross-entropy loss can directly learn

the permutation loss based on linear allocation cost in an end-to-end fashion, no matter how the number of nodes and edges in the graph change.

## 5. Experiments

In this section, we verify our method in three different tasks: We first perform our experiments in real-world tasks of supervised keypoint matching in four different kinds of natural images. Then, we further demonstrate the effectiveness of each procedure in our framework in an ablation study. Finally, we conduct the experiment on transfer learning across different categories, to evaluate the robustness and generalization capability of our methods.

### *5.1. Datasets*

**Willow ObjectClass.** The Willow Object Class dataset [34] contains five object categories including faces, cars, ducks, motorbikes, and wine bottles. This dataset is a benchmark used to measure the ability of image classification and recognition. There are 9963 images in total, including 24640 labeled objects. In the experiment, we select a small subset for each category for training and use the remainder for testing. Each category has at least 40 images.

**Pascal VOC 2011.** The Pascal Visual Object Classes (VOC) 2011 dataset [16] contains 20 object sub-categories and four major categories in total, which are vehicles, indoors, animals, and people. Each image in the dataset has three annotations including category labels, attributes, locations, and bounding boxes. Note that most images in this dataset contain complicated component scenes.

**CUB-200-2011.** The Caltech-UCSD Birds-200-2011 (CUB-200-2011) dataset [35] contains 11788 images with 200 categories of birds. Each image has clear annotations containing 1 subcategory label, 15 part locations, 312 binary attributes, and 1 bounding box.

**IMC-PT-SparseGM.** The IMC-PT-SparseGM dataset [36] contains 16 categories and about 25061 images. This dataset gathers its data from 16 tourist attractions around the world. Compared with the datasets mentioned above, the number of images in this dataset is the largest.

*5.2. Alternative for Comparison*

The state-of-art benchmark methods used for comparison experiments are:

**GMN.** (Graph Matching Network, GMN) Zanfir *et al.* [37] developed the deep graph matching method employing an end-to-end deep learning framework. To extract image features, this framework adopts VGG16 [23] networks, with first-order and second-order features extracted from the shallower layer (relu4_2) and deeper layer (relu5_1) of VGG16, respectively. It also uses Delaunay triangulation to build graphs. Factorized Graph Matching (FGM) has been used to construct the affinity matrix efficiently. The method is based on supervised learning using a loss function on the offset loss. Although this method is relatively simple, it pioneered the use of deep learning to solve graph matching problems.

**PIA-GM/PCA-GM.** Permutation loss and intra-graph affinity-based graph matching (PIA-GM) and permutation loss and cross-graph affinity-based graph matching (PCA-GM) [38] replace the offset loss on GMN with the permutation loss. This can improve the learning process. The image preprocessing of PIA-GM is the same as GMN. It employs 3 GNN layers to perform intra-graph convolution. The feature dimension of the GNN layer is 2048. The main difference between PIA-GM and the original GMN architecture is that it introduces graph neural networks to learn the node embedding of the graph. The difference between PCA-GM and PIA-GM is that the former changes the method of graph convolution, from intra-graph convolution to cross-graph convolution. The aim is to achieve better extraction of information for similar nodes between two input graphs and better generation of the node embedding.

**IPCA-GM.** (Iterative Permutation loss and cross-graph affinity-based graph matching, IPCA-GM [39]). To better combine combinatorial optimization with a deep learning solution, IPCA-GM considers the iterative interaction between the Sinkhorn layer and the GCN layer to learn a better feature embedding. This approach fully considers the embedding information for nodes, and reduces the original two-dimensional QAP problem to a one-dimensional one, thus reducing the complexity and improving the learning efficiency.

**CIE-H.** (Channel-Independent Embedding and Hungarian attention, CIE-H [40]). This follows the CNN-GNN-metric-Sinkhorn pipeline proposed by PCA-GM and improves PCA-GM in two way: First, the edge attribute information is taken into account and combined with node embedding. This is used to produce a channel-independent embedding. Second, a Hungarian Attention module dynamically constructs a structured and sparsely connected layer. This considers the most significant matching pair during training.

**NGM/NHGM.** (Neural Graph Matching Network (NGM) and Neural Hyper Graph Matching Network, NHGM [11]). These are recognized as a learnable solution to Lawler's quadratic assignment problem (Lawler's QAP [41]). This first transforms Lawler's QAP into an association graph and founds a solution, which is equivalent to the vertex classification problem on the association graph. Finally, a graph neural network is used to solve the vertex classification problem. NGM considers the task of matching graph pairs, and can be regarded as Lawler's QAP solver using VGG16. On the other hand, NHGM is a hypergraph matching solver with VGG16. Both of these methods are based on supervised learning using permutation loss.

**GANN-GM/GANN-MGM.** (GANN-GM [8]) is self-supervised learning graduated assignment neural network for matching graph pairs. (GANN-MGM [8]) is a self-supervised learning graduated assignment neural network for multi-graph matching. Both were proposed by Wang *et al.* [8]. GANN-GM introduces a self-supervised learning framework by leveraging graph matching solvers to provide pseudo labels to train the neural network module in a deep graph matching pipeline. A general graph matching solver is proposed for various graph matching settings based on the classic Graduated Assignment (GA) algorithm.

### *5.3. Experimental Settings*

The experiments are conducted using two GeForce GTX 1080 Ti GPUs. We employ a batch size of 8 in training and evaluate our model in 2000 epochs for each iteration. In terms of experimental settings, we employ the Adam [42]

optimizer to train our models with a learning rate of $1 \times 10^{-4}$. To overcome the over-smoothing problem common to graph neural networks, which arises from the transmission of information between nodes caused by deepening the network layer. This results in the representations of different nodes becoming similar and causes ambiguities in matching. We adopt a two-layer graph embedding and restrict the degree of smoothing in our experiments. The details of the hyperparameter settings are summarized in Table 1.

Table 1: Summary of parameter settings for datasets under study.

| Datasets | Average Node Number | K | r | $\delta$ | $\alpha$ | $\beta$ |
|---|---|---|---|---|---|---|
| Pascal VOC | 9.07 | 2 | 1 | 2 | $1 \times 10^{-4}$ | $1 \times 10^{-5}$ |
| Willow ObjectClass | 10.00 | 2 | 1 | 2 | $1 \times 10^{-4}$ | $1 \times 10^{-4}$ |
| CUB 2011 | 12.00 | 2 | 1 | 2 | $1 \times 10^{-4}$ | $1 \times 10^{-5}$ |
| IMC-PT-SparseGM | 21.36 | 2 | 1 | 3 | $1 \times 10^{-4}$ | $1 \times 10^{-4}$ |

### *5.4. Performance Comparisons on Four Datasets*

In this experiment, the source graph $\mathcal{G}_s$ and target graph $\mathcal{G}_t$ are composed of key points extracted from two different images. Both two graphs maintain node consistency during the training and testing stages. We perform the same operations on four datasets to convert the images into graphs. First, we filter the outliers from the dataset. Next, we crop the images around their bounding boxes, which provide the pixel location of the object boundings in images. Their image sizes are further resized to $256 \times 256$. The preprocessed images are then fed to VGG16 [23] to extract the node features. The graphs are constructed by Delaunay triangulation [43] of the extracted feature points.

We evaluate the graph matching capacity of the proposed method using the matching accuracy, which is defined by $accuracy = \sum \mathrm{AND}(S_{i,j}, S_{i,j}^{gt})/N$ from [38, 39, 8]. Here, the AND operation is a logical operation. Note that $S_{i,j}$ is the element of the predicted permutation matrix representing the correspondence matching of node $i$ and node $j$ from two different graphs. Similarly, $S_{i,j}^{gt}$ is the correspondence of the ground truth between the two nodes, and $N$ is the number of matching node pairs. We use ground truth node correspondences (which are

set to one if two nodes are matched, and zero otherwise) for the image pairs, and perform an averaging operation to compute the final matching accuracy.

Table 2: Matching accuracy on the Willow ObjectClass database(%).

| Method | Car | Duck | Face | Motorbike | Winebottle | Mean accuracy |
|---|---|---|---|---|---|---|
| GMN | 67.90 | 76.70 | 99.80 | 69.20 | 83.10 | 79.34 |
| NGM | 84.20 | 77.60 | 99.40 | 76.80 | 88.30 | 85.30 |
| NHGM | 86.50 | 72.20 | 99.90 | 79.30 | 89.40 | 85.50 |
| CIE-H | 82.20 | 81.20 | **100.00** | **90.00** | **97.60** | 90.20 |
| PIA-GM | 88.60 | 87.00 | **100.00** | 70.30 | 87.80 | 86.74 |
| PCA-GM | 87.60 | 83.60 | **100.00** | 77.60 | 88.40 | 87.44 |
| IPCA-GM | 90.40 | 88.60 | **100.00** | 83.00 | 88.30 | 90.06 |
| PCA-GM+Position+Structure | **92.90** | **90.80** | **100.00** | 79.00 | 94.40 | **91.42** |
| IPCA-GM+Position+Structure | 92.70 | 88.70 | **100.00** | 80.20 | 94.90 | 91.30 |

The experimental results on the Willow ObjectClass, Pascal VOC, CUB and IMC-PT-SparseGM datasets are shown in Table 2, Table 3, Table 4, and Table 5, respectively. We also compare the proposed methods (i.e. PCA-GM+Position+Structure and IPCA-GM+Position+Structure) with the alternative methods introduced in Section 5.2. Specifically, our methods achieve the best performance in the classification accuracy of three different categories on the Willow ObjectClass dataset, while the CIE-H method outperforms on the Motorbike and Wine bottle classes. Note that the CIE-H method also achieves good results on the VOC and CUB datasets. However, it performs poorly on the IMC-PT-SparseGM dataset. The VOC dataset contains 20 object subcategories and four major categories. It is therefore much more complicated than the remaining datasets under study. As observed from Table 3, many methods perform better than others in one or several specific categories. Our method performs slightly lower than the highest average accuracy. For the CUB and IMC-PT-SparseGM datasets, our method achieves the highest mean accuracy performance compared with the alternatives. It can therefore be concluded that our model performs well on all four datasets under study, while the alternatives generally fail to perform well on at least one of the datasets. This also clearly demonstrates the advantage of our method in terms of robustness.

Table 3: Matching accuracy on the Pascal VOC database(%).

| Method | Aero | Bike | Bird | Boat | Bottle | Bus | Car | Cat | Chair | Cow | Table | Dog | Horse | Motorbike | Person | Plant | Sheep | Sofa | Train | TV | Mean accuracy |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GMN | 31.90 | 47.20 | 51.90 | 40.80 | 68.70 | 72.20 | 53.60 | 52.80 | 34.60 | 48.60 | **72.30** | 47.70 | 54.80 | 51.00 | 38.60 | 75.10 | 49.50 | 45.00 | 83.00 | 86.30 | 55.30 |
| NGM | 50.10 | 63.50 | 57.90 | 53.40 | 79.80 | 77.10 | 73.60 | 68.20 | 41.10 | 66.40 | 40.80 | 60.30 | 61.90 | 63.50 | 45.60 | 77.10 | 69.30 | 65.50 | 79.20 | 88.20 | 64.13 |
| NHGM | 52.40 | 62.20 | 58.30 | 55.70 | 78.70 | 77.70 | **74.40** | 70.70 | 42.00 | 64.60 | 53.80 | 61.00 | 61.90 | 60.80 | 46.80 | 79.10 | 66.80 | 55.10 | 80.90 | 88.70 | 64.58 |
| CIE-H | 49.94 | 63.13 | **70.65** | 52.98 | **82.43** | 75.36 | 67.66 | 72.30 | 42.35 | **66.88** | 69.90 | **69.52** | **70.74** | 61.96 | 46.67 | 85.04 | 70.00 | 61.75 | 80.23 | **91.78** | 67.56 |
| PIA-GM | 41.50 | 55.80 | 60.90 | 51.90 | 75.00 | 75.80 | 59.60 | 65.20 | 33.30 | 65.90 | 62.80 | 62.70 | 67.70 | 62.10 | 42.90 | 80.20 | 64.30 | 59.50 | 82.70 | 90.10 | 63.00 |
| PCA-GM | 40.90 | 55.00 | 65.80 | 47.90 | 76.90 | **77.90** | 63.50 | 67.40 | 33.70 | 65.50 | 63.60 | 61.30 | 68.90 | 62.80 | 44.90 | 77.50 | 67.40 | 57.50 | 86.70 | 90.90 | 63.80 |
| IPCA-GM | **53.78** | 66.22 | 67.14 | **61.20** | 80.39 | 75.27 | 72.55 | **72.52** | **44.55** | 65.24 | 54.30 | 67.24 | 67.90 | 64.21 | **47.93** | 84.35 | **70.79** | 63.98 | 83.80 | 90.83 | **67.70** |
| PCA-GM+Position+Structure | 49.34 | **68.51** | 63.32 | 56.37 | 81.34 | 75.81 | 65.8 | 70.93 | 42.90 | 66.15 | 70.46 | 64.21 | 66.78 | **65.05** | 42.31 | **85.06** | 67.60 | **67.25** | **86.87** | 90.06 | 67.31 |
| IPCA-GM+Position+Structure | 48.72 | 65.52 | 56.08 | 52.86 | 78.76 | 73.66 | 64.4 | 66.18 | 39.73 | 65.18 | 63.15 | 61.90 | 66.06 | 62.68 | 42.21 | 82.89 | 64.46 | 60.8 | 86.40 | 89.45 | 64.56 |

Table 4: Matching accuracy on the CUB database(%).

| Method | Mean accuracy |
|---|---|
| GANN-GM | 48.44 |
| GANN-MGM | 78.32 |
| CIE-H | 92.06 |
| PIA-GM | 88.56 |
| PCA-GM | 92.17 |
| IPCA-GM | 92.18 |
| PCA-GM+Position+Structure | 91.46 |
| IPCA-GM+Position+Structure | **92.34** |

Table 5: Matching accuracy on the IMC-PT-SparseGM (%).

| Method | Reichstag | Sacre_coeur | St_peters_square | Mean accuracy |
|---|---|---|---|---|
| GANN-GM | 76.02 | 44.15 | 50.49 | 56.89 |
| GANN-MGM | 67.41 | 42.72 | 44.42 | 51.52 |
| CIE-H | 42.24 | 28.47 | 30.78 | 33.83 |
| PIA-GM | 71.46 | 41.31 | 42.64 | 51.80 |
| PCA-GM | 69.38 | 39.86 | 42.10 | 50.40 |
| IPCA-GM | 72.96 | 43.80 | 44.93 | 53.89 |
| PCA-GM+Position+Structure | **96.28** | **75.93** | 81.66 | **84.63** |
| IPCA-GM+Position+Structure | 95.71 | 74.12 | **82.06** | 83.96 |

### *5.5. Ablation Study*

Here we conduct an ablation study to demonstrate that our novel method for embedding the positional and structural arrangement information can be effective in improving variants of the methods of the performance of graph matching. To this end, we construct three models based on the graph matching methods PCA-GM and IPCA-GM. The three variants are a) the original method combined with position-aware embedding, b) the original method combined with structure embedding, and c) the original method combined with complete position and structure embedding. Considering that our method has the alternative best performance on the IMC-PT-SparseGM data set compared with other methods, we conduct the ablation study on the IMC-PT-SparseGM dataset. The experimental results are shown in Table 6.

Table 6: Matching accuracy of ablation study on IMC-PT-SparseGM (%).

| Method | Reichstag | Sacre_coeur | St_peters_square | Mean accuracy |
|---|---|---|---|---|
| PCA-GM | 69.38 | 39.86 | 42.10 | 50.40 |
| PCA-GM+Position | 92.02 | 69.07 | 78.80 | 79.96 |
| PCA-GM+Structure | 93.83 | 70.44 | 78.78 | 81.02 |
| PCA-GM+Position+Structure | 96.49 | 72.01 | 83.00 | 83.83 |
| IPCA-GM | 72.96 | 43.80 | 44.93 | 53.89 |
| IPCA-GM+Position | 93.03 | 65.37 | 77.24 | 78.55 |
| IPCA-GM+Structure | 92.79 | 62.66 | 76.98 | 77.48 |
| IPCA-GM+Position+Structure | 95.71 | 74.12 | 82.06 | 83.96 |

The proposed position-aware node embedding and subgraph-based structure embedding greatly improve the classification accuracy of the original method in each category. Note that the IMC-PT-SparseGM dataset consists of tourist attractions, of which many (especially the buildings) are symmetrical, indicating that the semantic information of the key points is usually ambiguous. Therefore, the incorporation of position information for the graph matching methods can effectively assist the classification task. The high-order subgraph-based structural arrangement information is also valuable, since the global structural features of the tourist attractions can provide more comprehensive information than just local features alone.

### *5.6. Cross-category Generalization Study*

To verify the robustness and generalization capability of our methods on different types of categories, we construct the confusion matrix for the training and testing instances from the different categories. The experimental results are presented in Fig. 6 and Fig. 7. We first train each of the five classes separately for the Willow dataset. We then use the distinct training classes to validate the generalization capabilities for all five object classes.

The confusion matrix for the dataset illustrating transfer learning among the five object categories is given in Fig. 6. The models are trained on the object categories in the rows and then tested on the object categories in the columns. The diagonal of the matrix shows the classification ability of the method on each

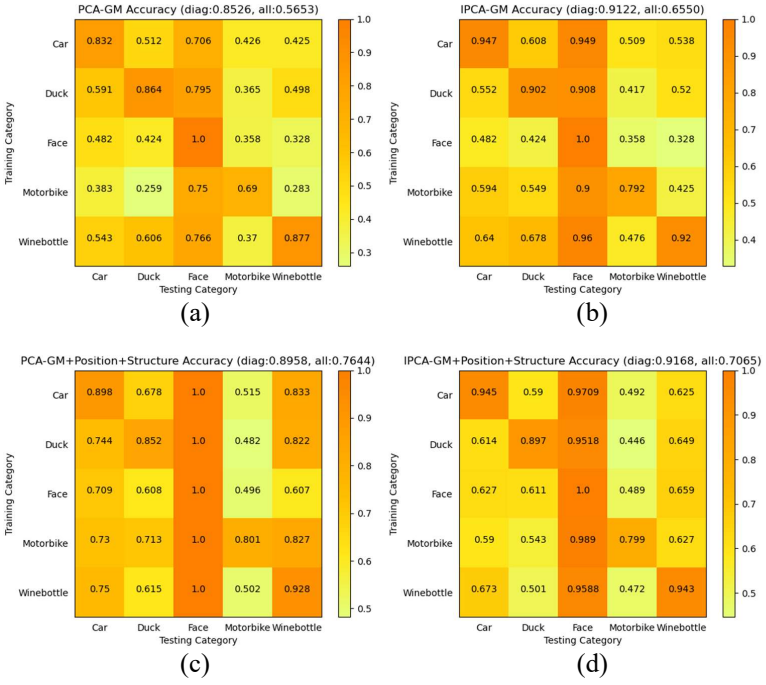


Figure 6: Cross-category generalization study represented by confusion matrix on Willow ObjectClass dataset. Each column of the matrix represents one category of images used for model training, while each row represents one class of testing samples. a) and b) show the matching accuracy of comparison methods in different category generalizations, respectively. c) and d) are obtained from the proposed methods.

object class. The darker the grid color in the figure, the higher the matching accuracy. We compare each element in Fig. 6 (a) with Fig. 6(c), and Fig. 6(b) with Fig. 6(d). The proposed graph matching method combined with position and structural arrangement information has improved the generalization ability for all classes compared with the original methods. Furthermore, the accuracy of the model in the face category reaches 100% in Fig. 6(c), and is also high in both Fig. 6(b) and Fig. 6(d). It may be related to the simplicity and lower noise in the face samples.

In Fig. 7, we re-color Fig. 6 by sorting all the matching results obtained by the four different methods from large to small. The larger the value, the

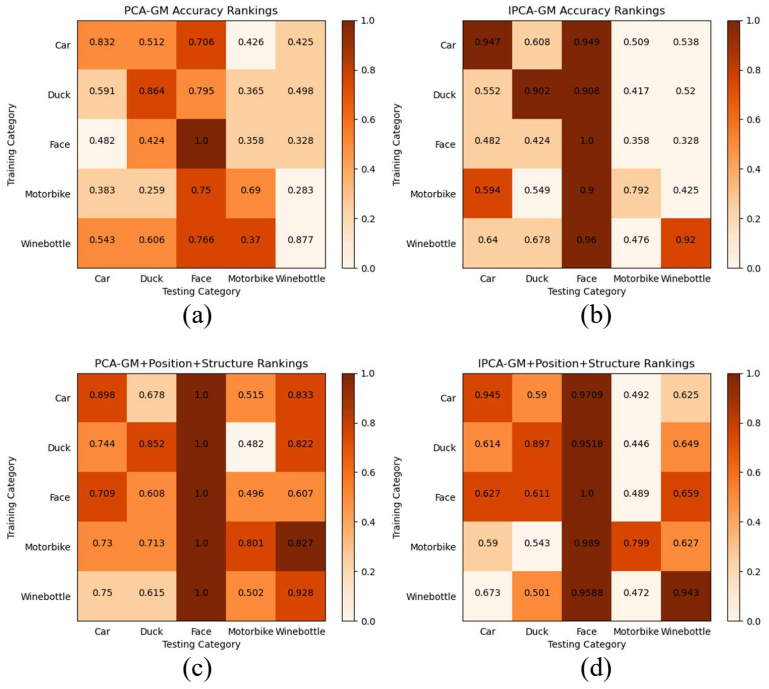


Figure 7: Cross-category generalization study method ranking on Willow ObjectClass dataset. The re-color figure from Fig. 6 by sorting all the matching results obtained by four different methods from large to small.

darker the color. This is also a reflection of the generalization ability of the model. Moreover, it gives the priority to the proposed position and structure framework based on higher accuracy scores in the confusion matrix. The generalization ability of the models provided in this study is substantially greater than the alternative methods due to the addition of position information and the correlation connection of the subgraph structure. Furthermore, IPCA-GM+Position+Structure has greater diagonal accuracy, indicating that it is more suitable for the training category.

## 6. Conclusion

In this paper, we propose a novel deep learning framework for graph matching, which combines a relative position-aware node embedding and subgraph-

based structure embedding into node-wise embedding between graphs. The experiments include comparisons with alternatives, an ablation study and the cross-category generalization study to demonstrate the robustness and effectiveness of the proposed method with its embedding modules. Specifically, the matching accuracy on several real-world datasets compared with peer methods demonstrate the state-of-the-art performance of our method.

In future work, we intend to employ different relative positions and subgraph relational strategies to further improve the graph matching performance. Besides, from a problem-solving perspective, we can further evaluate the performance of the proposed method in multi-graph matching and unsupervised learning fields.

## References


[1] S. Gold, A. Rangarajan, A graduated assignment algorithm for graph matching, IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (4) (1996) 377–388.

[2] R. W. So, A. C. Chung, A novel learning-based dissimilarity metric for rigid and non-rigid medical image registration by using bhattacharyya distances, Pattern Recognition 62 (2017) 161–174.

[3] R. Chaker, Z. Al Aghbari, I. N. Junejo, Social network model for crowd anomaly detection and localization, Pattern Recognition 61 (2017) 266–281.

[4] M. Wang, Y.-K. Lai, Y. Liang, R. R. Martin, S.-M. Hu, Biggerpicture: data-driven image extrapolation using graph matching, ACM Transactions on Graphics 33 (6) (2014).

[5] W. Wang, W. Lin, Y. Chen, J. Wu, J. Wang, B. Sheng, Finding coherent motions and semantic regions in crowd scenes: A diffusion and clustering approach, in: European Conference on Computer Vision, Springer, 2014, pp. 756–771.

[6] B. Jiang, H. Zhao, J. Tang, B. Luo, A sparse nonnegative matrix factorization technique for graph matching problems, Pattern Recognition 47 (2) (2014) 736–747.

[7] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, A. J. Smola, Learning graph matching, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (6) (2009) 1048–1058.

[8] R. Wang, J. Yan, X. Yang, Graduated assignment for joint multi-graph matching and clustering with application to unsupervised graph matching network learning, Advances in Neural Information Processing Systems 33 (2020) 19908–19919.

[9] H. Zhang, Z. Huang, X. Lin, Z. Lin, W. Zhang, Y. Zhang, Efficient and high-quality seeded graph matching: Employing higher-order structural information, ACM Transactions on Knowledge Discovery from Data (TKDD) 15 (3) (2021) 1–31.

[10] K. Henderson, B. Gallagher, L. Li, L. Akoglu, T. Eliassi-Rad, H. Tong, C. Faloutsos, It's who you know: graph mining using recursive structural features, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2011, pp. 663–671.

[11] R. Wang, J. Yan, 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) 1–1.

[12] S. Berretti, A. Del Bimbo, E. Vicario, Efficient matching and indexing of graph models in content-based retrieval, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (10) (2001) 1089–1105.

[13] M. Rolínek, P. Swoboda, D. Zietlow, A. Paulus, V. Musil, G. Martius, Deep graph matching via blackbox differentiation of combinatorial solvers, in: European Conference on Computer Vision, Springer, 2020, pp. 407–424.

[14] R. M. Cesar Jr, E. Bengoetxea, I. Bloch, P. Larrañaga, Inexact graph matching for model-based recognition: Evaluation and comparison of optimization algorithms, Pattern Recognition 38 (11) (2005) 2099–2113.

[15] A. Zanfir, C. Sminchisescu, Deep learning of graph matching, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2684–2693.

[16] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge, International Journal of Computer Vision 88 (2) (2010) 303–338.

[17] A. Friedman, Framing pictures: The role of knowledge in automatized encoding and memory for gist., Journal of Experimental Psychology: General 108 (3) (1979) 316.

[18] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110.

[19] T. Ojala, M. Pietikainen, T. Maenpaa, Multiresolution gray-scale and rotation invariant texture classification with local binary patterns, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (7) (2002) 971–987.

[20] F. Cen, X. Zhao, W. Li, G. Wang, Deep feature augmentation for occluded image classification, Pattern Recognition 111 (2021) 107737.

[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.

[22] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[23] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: International Conference on Learning Representations, 2014.

[24] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations, 2017.

[25] Y. Yang, Y. Qi, Image super-resolution via channel attention and spatial graph convolutional network, Pattern Recognition 112 (2021) 107798.

[26] M. Zhang, Y. Chen, Link prediction based on graph neural networks, Advances in Neural Information Processing Systems 31 (2018) 5165–5175.

[27] F. Zhou, F. De la Torre, Factorized graph matching, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 127–134.

[28] P. A. Knight, The sinkhorn–knopp algorithm: convergence and applications, SIAM Journal on Matrix Analysis and Applications 30 (1) (2008) 261–275.

[29] H. W. Kuhn, The hungarian method for the assignment problem, Naval Research Logistics Quarterly 2 (1-2) (1955) 83–97.

[30] G. Patrini, R. van den Berg, P. Forre, M. Carioni, S. Bhargav, M. Welling, T. Genewein, F. Nielsen, Sinkhorn autoencoders, in: Uncertainty in Artificial Intelligence, 2020, pp. 733–743.

[31] W. Fang, F. Zhang, V. S. Sheng, Y. Ding, A method for improving cnn-based image recognition using dcgan, Computers, Materials and Continua 57 (1) (2018) 167–178.

[32] J. You, R. Ying, J. Leskovec, Position-aware graph neural networks, in: International Conference on Machine Learning, 2019, pp. 7134–7143.

[33] R. Sinkhorn, P. Knopp, Concerning nonnegative matrices and doubly stochastic matrices, Pacific Journal of Mathematics 21 (2) (1967) 343–348.

[34] M. Cho, K. Alahari, J. Ponce, Learning graphs to match, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 25–32.

[35] C. Wah, S. Branson, P. Welinder, P. Perona, S. Belongie, The Caltech-UCSD Birds-200-2011 Dataset, Tech. Rep. CNS-TR-2011-001, California Institute of Technology (2011).

[36] Y. Jin, D. Mishkin, A. Mishchuk, J. Matas, P. Fua, K. M. Yi, E. Trulls, Image matching across wide baselines: From paper to practice, International Journal of Computer Vision (2021) 517–547.

[37] A. Zanfir, C. Sminchisescu, Deep learning of graph matching, in: IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2684–2693.

[38] R. Wang, J. Yan, X. Yang, Learning combinatorial embedding networks for deep graph matching, in: IEEE International Conference on Computer Vision, 2019, pp. 3056–3065.

[39] R. Wang, J. Yan, X. Yang, Combinatorial learning of robust deep graph matching: an embedding based approach, IEEE Transactions on Pattern Analysis and Machine Intelligence (2020) 1–1.

[40] T. Yu, R. Wang, J. Yan, B. Li, Learning deep graph matching with channel-independent embedding and hungarian attention, in: International Conference on Learning Representations, 2019.

[41] E. L. Lawler, The quadratic assignment problem, Management Science 9 (4) (1963) 586–599.

[42] I. K. M. Jais, A. R. Ismail, S. Q. Nisa, Adam optimization algorithm for wide and deep neural network, Knowledge Engineering and Data Science 2 (1) (2019) 41–46.

[43] D.-T. Lee, B. J. Schachter, Two algorithms for constructing a delaunay triangulation, International Journal of Computer and Information Sciences 9 (3) (1980) 219–242.

**Dongdong Chen** received his B. Sc. degree (2018) and M.Sc. degree (2021) from Xiamen University. He is currently a PhD student at Shanghai Jiao Tong University, Shanghai, China. His research interests include pattern recognition, complex networks, thermodynamic and quantum statistics, especially in graph and network analysis.

**Yuxing Dai** is now a postgraduate student at the information school of Xiamen University, China. His research interests include data mining, machine learning, and network representation learning.

**Lichi Zhang** received his B.Sc. degree (2008) from Beijing University of Posts and Telecommunications, and the PhD degree in computer science from the University of York, UK. He is now an associate professor at School of Biomedical Engineering, Shanghai Jiao Tong University, Shanghai, China. His research interests mainly include medical image analysis, computer vision, machine learning, and pattern recognition.

**Zhihong Zhang** received his BSc degree (1st class Hons.) in computer science from the University of Ulster, UK, in 2009 and the PhD degree in computer science from the University of York, UK, in 2013. He won the K. M. Stott prize for best thesis from the University of York in 2013. He is now an associate professor at the software school of Xiamen University, China. His research interests are wide-reaching but mainly involve the areas of pattern recognition and machine learning, particularly problems involving graphs and networks.

**Edwin R. Hancock** holds a BSc degree in physics (1977), a PhD degree in high-energy physics (1981) and a D.Sc. degree (2008) from the University of Durham, and a doctorate Honoris Causa from the University of Alicante in 2015. He is currently Emeritus Professor in the Department of Computer Science at the University of York, and also Adjunct Professor and Principal Investigator - Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University. He became a fellow of the International Association for Pattern Recognition in 2000, the Institute of Physics in 2007 and the IEEE in 2016. He is currently Editor-in-Chief of the journal Pattern Recognition, and was founding Editor-in-Chief of IET Computer Vision from 2006 until 2012. He has also been a member of the editorial boards of the journals IEEE Transactions on Pattern Analysis and Machine Intelligence, Pattern Recognition, Computer Vision and Image Understanding, Image and Vision Computing,

and the International Journal of Complex Networks. He was Conference Chair in 1994 and Programme Chair in 2016 for the British Machine Vision Conference, Track Chair for ICPR (2004 and 2016) and Area Chair for ECCV (2006) and CVPR (2008 and 2014). In 1997 he jointly established the EMMCVPR workshop series with Marcello Pelillo. He was awarded a Royal Society Wolfson Research Merit Award in 2009, was named BMVA Distinguished Fellow in 2016 and received the IAPR Pierre Devijver Award in 2018. He was a Governing Board Member of the IAPR (2006-2016), and was Second Vice President of the Association (2016-2018). He was a member of the Computer Science and Information Technology sub-panel for the UK REF 2014 and will also be a panelist for REF 2021. He has published about 200 journal papers and 650 refereed conference papers.

# Declaration of Interest Statement

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Authors: Dongdong Chen, Yuxing Dai, Lichi Zhang, Zhihong Zhang and Edwin R. Hancock.

# Position-aware and Structure Embedding Networks for Deep Graph Matching

Dongdong Chen[a], Yuxing Dai[b], Lichi Zhang*[a], Zhihong Zhang[b], Edwin R. Hancock[c]

[a]*Shanghai Jiao Tong University, Shanghai, 20030, Shanghai, China*
[b]*Xiamen University, Xiamen, 361005, Fujian, China*
[c]*University of York, York, YO105DD, York, UK*

**Abstract**

Graph matching refers to the process of establishing node correspondences based on edge-to-edge constraints between graph nodes. This can be formulated as a combinatorial optimization problem under node permutation and pairwise consistency constraints. The main challenge of graph matching is to effectively find the correct match while reducing the ambiguities produced by similar nodes and edges. In this paper, we present a novel end-to-end neural framework that converts graph matching to a linear assignment problem in a high-dimensional space. This is combined with relative position information at the node level, and high-order structural arrangement information at the subgraph level. By capturing the relative position attributes of nodes between different graphs and the subgraph structural arrangement attributes, we can improve the performance of graph matching tasks, and establish reliable node-to-node correspondences. Our method can be generalized to any graph embedding setting, which can be used as components to deal with various graph matching problems answered with deep learning methods. We validate our method on several real-world tasks, by providing ablation studies to evaluate the generalization capability across different categories. We also compare state-of-the-art alternatives to demonstrate performance.

*Keywords:*
Graph Matching, Graph Embedding, Deep Neural Network

*Corresponding author: lichizhang@sjtu.edu.cn (Lichi Zhang)

## 1. Introduction

Graph matching aims to establish the correspondence between two or more graphs based on their structural information [1]. It is widely used in combinatorial optimization, machine learning and computer vision due to its natural representation and convenient coding of the relationship between abstract data. Specifically, many applications can be implemented using the graph matching technique, such as image registration in medical image analysis, linking user accounts in social network analysis, image extrapolation in computer vision, finding coherent motions and semantic regions in crowd scenes [2].

There are many methods and matching solvers developed for addressing the graph matching problem [3, 4]. These can be divided into different categories depending on the specific perspectives adopted. For example, based on the number of graphs included, there are graph-to-graph matching and multi-graph matching methods [5]. Since graph-to-graph matching is the basis of multi-graph matching, improving the matching ability between two graphs can also be extended to improve multi-graph matching. Therefore, this paper focuses on the matching of pairs of graphs. Based on matching content, they can be divided into a) structure information-based graph matching and b) semantic information-based graph matching [6]. Note that in graph matching, especially for image-generated graphs, semantic features based on structural information have been proved to be important for classifying nodes [7]. In this way, the graph matching problems studied in this paper are based on both node feature information and structural information from the graphs. On the other hand, graph matching methods can also be divided into a) learning-based methods and b) learning-free methods [8]. Notably, learning-free methods generally seek approximate solutions for a given fixed affinity model, which is usually cast in the form of simple parameters and is approximately solved by the joint similarity of individual entities together with their mutual relationships [9]. However, their time complexity is often too high to satisfy and solve large-scale real-world problems.

In the case of deep learning-based graph matching solvers, the common feature of most methods is that they compromise in their evidence-combining strategy in the sense that the resulting unclear combinatorial element would not be competitive in a purely combinatorial setup [10]. However, these methods can only calculate the similarity score of the entire graph, and heavily rely on inefficient global matching procedures. Additionally, they only con-

sider the embedding of local information for nodes in the graph, which tend to ambiguously match similar nodes in different regions of the graph in an inconsistent manner [10]. Fig. 1 illustrates an example of the main idea of our relative position-aware information embedding strategy. Here it is shown that it fails to match the cats' ear positions correctly, since the node embedding information usually relies only on node semantic information, and the local structure information lacks effective discrimination ability. Therefore it is difficult to unambiguously distinguish the left and right ears of a cat. In this example, the relative position information is critical to graph matching, especially when both the semantic and structural information are very similar in the two regions of the graphs.

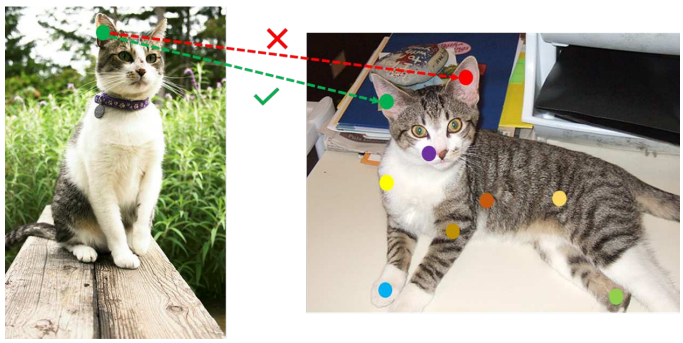

Figure 1: An exemplar failure case to establish the correct match of cat ears between two images, which are selected from the Pascal VOC dataset [11].

To address the above-mentioned challenges, here we propose a novel approach to solve the graph matching problem. We commence by utilizing node features extracted from the images to attribute the nodes, and design a position-aware node embedding algorithm to capture the relative position information of nodes in the graph. We further enhance the representation of local information of the nodes by merging a structural arrangement information representation of the subgraph in which the node is located. With the graph node-wise embedding to hand, we aim to obtain the required graph node permutations for node-to-node correspondence from raw pixel inputs. To our best knowledge, there are no methods that consider embedding both position and structural information for the graph matching problem.

The main contributions of our works are as follows:

1. Relative position information has not been considered in the existing

process of graph matching, which hampers their applications in terms of reducing the ambiguity of matching accuracy. We propose to construct an anchor set for small graphs extracted from images, and fully consider the relative position information related to the matching of keypoints, which is also proved to be effective in experiments.

2. We propose a novel subgraph detection and graph representation method, and extract subgraph structural arrangement information for improved node embedding. Previous graph matching studies only match using node level and edge level information, but ignore the higher-order neighborhood structure information. Here we introduce subgraph arrangement structure information into the graph matching task, by capturing it using the subgraph structure. This arrangement procedure combines node and edge level information with relative position information. We can therefore implement a hierarchical and comprehensive extraction of image features, which further improves the matching accuracy.

3. We demonstrate in our experiments that our method outperforms alternative work on several widely-studied real-world datasets. In the ablation study, we also show the effectiveness of the introduced components of our new method of graph matching on these datasets. We show that our position-aware node embedding module and subgraph-based structural embedding module can be added to existing deep learning graph matching methods, to further boost their classification performance.

The remainder of the paper is organized as follows: Section 2 provides a brief review of the related work on image feature extraction, graph embedding, and combinatorial optimization. Section 3 defines the specific graph matching problem that we intend to resolve in this work. Section 4 provides the details of our method, and the corresponding experiments to validate our new method are presented in Section 5. Finally, we conclude the paper and provide an extended discussion of potential future work in Section 7.

## 2. Related Work

In this section, we review existing deep learning solvers in dealing with graph matching problems.

### *2.1. Image Feature Extraction*

Feature extraction refers to the computation of higher-level features from the original pixels in an image. It can capture the differences between pixels

of various object classes. Conventional feature extraction usually proceeds in an unsupervised manner, and no image category labels are used when extracting information from the pixels. Commonly used traditional features include GIST (Generalized Search Trees), SIFT (Scale-Invariant Feature Transform), and LBP (Local Binary Patterns) [12].

However, these hand-crafted feature extraction methods can not be optimized according to the training from images and their labels. However, they fail to provide more comprehensive feature information to better represent important image information. The method based on deep learning overcomes this shortcoming using a soft-coded feature extractor [13]. Recently, the CNN model has been developed and enhanced in detail. For example, Szegedy *et al.* [14] significantly increase the depth of the CNN in their GoogleNet with three different types of convolution operation. He *et al.* [15] have proposed the residual neural network (ResNet) which includes a cross-layer connection that passes information from the input across deeper layers. It also adds to the result of convolution by introducing shortcut connections to solve the problem of vanishing loss gradient.

In this paper, we use VGG-16 [16] as the backbone architecture for image feature extraction, and which also closely follows the previous graph-matching works described in [5, 8, 10, 17]. This guarantees equity in the comparisons performed in our experiments. It can adapt to different sizes of the convolutional kernel to capture more discriminative decision functions with fewer parameters. It has been proved effective in deep visual feature extraction.

### *2.2. Graph Embedding*

At present, most studies related to graph embedding focus on static graphs, which can be divided into (a) spectral-based and (b) spatially-based methods.

#### *2.2.1. Spectral Methods for Graph Embedding*

The earliest graph embedding method provides local convolutions of graph structure data in the spectral frequency domain. The Fourier transform of the graph is used to transform the time domain data into a frequency domain signal. Then the frequency domain signal is convolved with local features. Finally, the frequency domain signal is transformed back into the time domain.

Although spectral frequency-based methods can effectively avoid the cross-influence of data in the time domain, it needs to calculate the Laplace matrix eigendecomposition and a representation of the graph structure corresponding to the sample of the graph structure. This makes it difficult to apply in real scenes due to the large computational overheads. To solve the problem of high computational complexity, Defferrard *et al.* [18] proposed to use Chebyshev polynomials instead of the exact calculation of graph eigenvalues and eigenvectors. In addition, Kifp and Welling [19] simplified the algorithm with first-order polynomials and used it to solve the task of semi-supervised node classification.

In general, spectral methods have suffered from the following three defects: a) The generalization ability to new unseen and variable graph structures is poor. If any node or edge in the graph changes, the spectral domain of the graph will also change rapidly, so that the filter learned from the original graph can not be well matched to the new graph. b) The alignment of Fourier patterns is difficult. It is challenging to align graphs with different typologies in the spectral domain, especially when the difference between graphs is large. Such differences in the spectral domain will be magnified, making it harder to align the Fourier modes. c) Spectral methods can only solve the problem for undirected graphs. This means it is limited to satisfying the constraint of a symmetric transformation matrix.

*2.2.2. Spatial Methods for Graph Embedding*

Spatial approaches have an advantage over spectral approaches in that they can operate on problems where the graph structure varies significantly in the dataset being studied [20]. However, they generally require sophisticated data transformations to enable effective learning. For example, given the strong hypothesis that the existence of node-connected edges is based on sometimes ad-hoc and contrived or artificial settings, Zhang *et al.* [21] extracted local subgraphs around each target link. They use this to learn the functional mapping of subgraph patterns to infer the existence of links, and to automatically learn heuristic algorithms.

In addition, the graph neural network method based on a message passing architecture is also considered to be a spatial method. It uses different aggregation schemes to aggregate the characteristic messages of neighbors in a graph. The message passing neural network further integrates edge information in the aggregation process. The graph attention network proposed by Velikovic *et al.* can aggregate neighborhood information according to

the weights derived from a trainable attention mechanism [22]. However, all these methods focus on just learning node embeddings to capture the local network structure around a given node. Such models are at most as powerful as the Weisfeiler-Leman graph isomorphism test [23], but they cannot distinguish nodes in symmetric or isomorphic positions in the network.

In this paper, we develop a novel graph embedding method to extract high-level information combined with the relative position of nodes, which can greatly assist graph matching.

### *2.3. Combinatorial Optimization*

In combinatorial optimization problems, enumeration and exhaustive methods are not practical. The solution strategies are either discrete or are effectively reduced to a discrete set. Many traditional algorithms for solving combinatorial optimization problems involve the use of manually constructed heuristic algorithms. However, due to the difficulty of problem-solving, it is usually not the optimal solution.

Mathematically, the graph matching problem can be considered a combinatorial optimization problem, which is also an NP-Hard problem. Using mathematical methods to solve graph matching problems can be divided into relaxation optimization and matrix decomposition methods [24]. For the spectral relaxation optimization method, the computation speed is relatively fast, but the constraints on the obtained solution are often over-relaxed. Of course, the input discrete variables can also be converted into continuous variables to make the solution more compact. However, the following problems generally exist in these methods: First, they are not robust to outliers and noise. Second, they usually have high computational overheads. Besides, the optimal approximate solution obtained by traditional mathematical methods may be locally optimal solutions different from the ground truth.

With the rise of deep learning, the use of reinforcement learning and neural networks to solve combinatorial optimization problems has become a topical trend. The Sinkhorn network [25] can solve the linear assignment problem, by performing the doubly stochastic normalization operations of row normalization and column normalization on all non-negative matrices to minimize the cost. Furthermore, it is an approximate Hungarian algorithm [26] that can optimize polynomial complexity. Recently, Patrini *et al.* [27] proposed a Sinkhorn autoencoder to achieve the target of minimizing distance, and utilizing reinforcement learning to solve the combinatorial optimization problem. Furthermore, a novel and simple end-to-end training

framework was proposed by Rolínek *et al.* [10], which incorporates the most advanced graph matching combinatorial solver.

In this paper, we develop a novel subgraph detection and graph representation method to solve the combinatorial optimization problem. It can effectively reduce the dimensionality of the quadratic assignment problem for combinatorial optimization to a linear assignment problem.

## 3. Problem Definition and Notations

In this paper, we consider the graph $\mathcal{G} = (V, A, X)$ which is consist of a finite set of nodes $V$, an adjacency matrix $A$, and a set of attributes $X$ for the nodes, which originate from images using various CNN models [16, 28, 29]. The goal of graph matching is to establish a correspondence between two attributed graphs, which minimizes the sum of local and geometric costs of assignment between the vertices of the two graphs.

Let $\mathcal{G}_s = (V_s, A_s, X_s)$ be a source graph, with $|V_s| = n$, $A_s \in \{0,1\}^{n\times n}$, and feature matrix $X_s \in \mathbb{R}^{n\times F}$, where $F$ represents the $F$-dimensional feature vector of nodes in graph stacked to columns. The target graph to be matched is $\mathcal{G}_t = (V_t, A_t, X_t)$, with $|V_t| = m$, $A_t \in \{0,1\}^{m\times m}$, and feature matrix $X_t \in \mathbb{R}^{m\times F}$, whose w.l.o.g. $n \leq m$. We also construct a vector of length $nm$. The element $\mathbf{v} \in \{0,1\}^{nm\times 1}$ indicates the match of vertices in two graphs, where $\mathbf{v}_{i,j} = 1$ if vertex $i \in V_s$ is matched to vertex $j \in V_t$ and $\mathbf{v}_{i,j} = 0$ if otherwise. It is worth noting that all the vertex matches are subject to the one-to-one mapping constraints, i.e. $\sum_{j\in V_t} \mathbf{v}_{i,j} = 1\ \forall i \in V_s$ and $\sum_{i\in V_s} \mathbf{v}_{i,j} \leq 1\ \forall j \in V_t$.

Furthermore, we construct a square symmetric positive matrix $M \in \mathbb{R}^{nm\times nm}$ referred to as the affinity matrix, to encode the edge-to-edge affinity between two graphs by their off-diagonal elements. Specifically, $M_{ip;jq}$ measures how well edge $(i,j)$ in graph $\mathcal{G}_s$ matches with edge $(p,q)$ in graph $\mathcal{G}_t$ with $\{i! = p\} \cup \{j! = q\}$. The diagonal entries of the affinity matrix can also indicate the node-to-node affinity between two graphs. Therefore, the pairwise graph matching between $\mathcal{G}_s$ and $\mathcal{G}_t$ can be formulated as the edge-preserving, quadratic assignment programming (QAP) [30] problem:

$$\begin{aligned} &\underset{\mathbf{v}}{\text{argmax}}\ \mathbf{v}^\top M\mathbf{v} \\ &\text{s.t.}\ \sum_{j\in V_t} \mathbf{v}_{i,j} = 1\ \forall i \in V_s, \sum_{i\in V_s} \mathbf{v}_{i,j} \leq 1\ \forall j \in V_t, \end{aligned} \tag{1}$$

Table 1: Notations and Description

| Symbol | Definition |
|---|---|
| $\mathcal{G}$ | original graph |
| $V$ | nodes-set of graph |
| $\|V\|$ | node number of nodes-set |
| $r$ | maximum shortest path distance |
| $K$ | network layer |
| $P$ | anchor-set |
| $p$ | anchor number |
| $q_{v,u}$ | relative position coefficient between pair of nodes $v$ and $u$ |
| $d_{v,u}$ | shortest path distance between pair of nodes $v$ and $u$ |
| $I^{(k)}(v,u)$ | information aggregation function between node $v$ and $u$ in $k$-th layer |
| $W^{(k)}$ | trainable weight matrix in $k$-th layer |
| ${h_v}^{(k)}$ | hidden representation of node $v$ in $k$-th layer |
| $N^t(v)$ | $t$-hop neighbor of node $v$ |
| $P_v(v)$ | path from node $v$ to $u$ |
| $\delta$ | leaf node threshold |
| $\alpha$ | hyper-parameter to balance the position information |
| $\beta$ | hyper-parameter to balance the structure information |
| $T$ | transform matrix of the original graph to the subgraph relation graph |
| $H_s$ | final hidden representation of source graph |
| $H_t$ | final hidden representation of target graph |

where $\mathbf{v} \in \{0,1\}^{nm\times 1}$ and $M \in \mathbb{R}^{nm\times nm}$. In this work, we intend to resolve the graph matching problem based on the supervised matching of graphs, and we aim to learn an end-to-end model which can effectively extract graph information and match through given pair-wise ground-truth correspondences for a set of graphs, and generalize to unseen graph pairs. The notation used in this paper is summarized in Table 1.

## 4. Methods

In this section, we provide details of the proposed end-to-end position-aware and structure-based graph matching method, The overall pipeline is

shown in Figure 2. In the figure, the blue source graph $\mathcal{G}_s$ are extracted together with their node-wise high-level graph feature representations. This is done using position-aware node embedding and subgraph based structure embedding. The graph features are matched using a matrix product operation and the Sinkhorn activation function. The result is compared with the available ground truth. The final output of the model is the matching accuracy of the pair of nodes between the source graph and the target graph. Note that the deep feature extractor of the proposed method can be any CNN model, such as VGG-16 [16], which is adapted to convert images into graphs with features. These graphs are the source and target graphs, which are the paired input of our model.

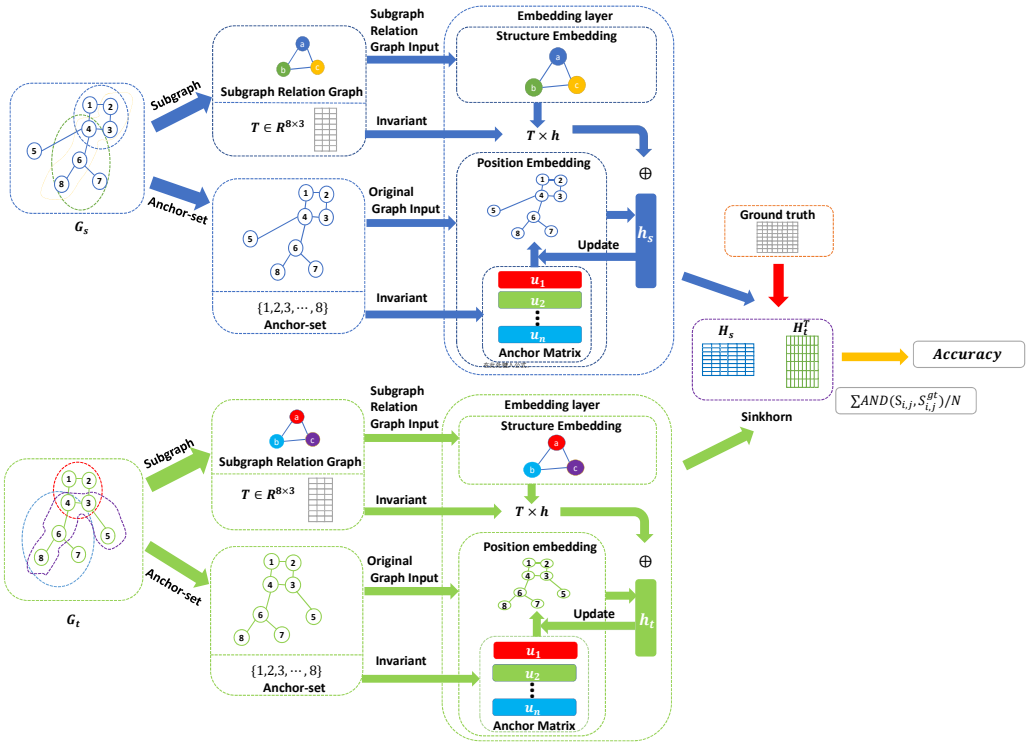


Figure 2: Overview of the end-to-end position and structure embedding networks for deep graph matching.

The model consists of three main components, namely a) a position-aware node embedding module, b) a subgraph-based structure embedding module,

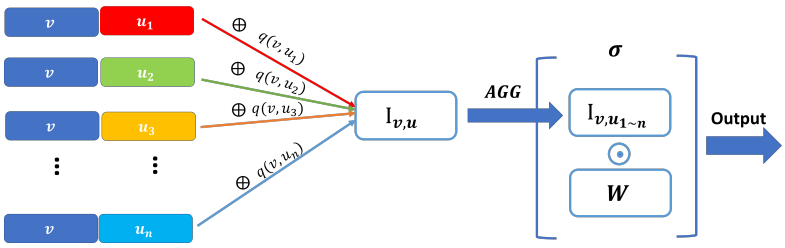


Figure 3: Procedure of Position Embedding.

and c) a graph feature matching module. The aim of the position-aware node embedding step, as described in Section 4.1, is to consider the information available from the relative structure. The second step is to iteratively combine the subgraph structure information described in Section 4.2, and the final part is the matching of two graphs with extracted graph representation described in Section 4.3.

### *4.1. Position-aware Node Embedding*

We establish our position-aware node embedding procedure with two key components: First, before the node embedding process, we propose a simple but effective strategy to construct an anchor-set for each graph which is used throughout the entire learning process. Here, we refer to the nodes used as reference position coordinates as *anchors*, and the set of these nodes as the *Anchor-set*. We aggregate the information of the node and each anchor in the anchor-set, in this way we replace the original graph embedding method that relies on message passing between local network neighborhoods. Second, we design an information aggregation mechanism for message aggregation between nodes and anchors in the anchor-set. With the addition of position-aware node embedding, we can effectively eliminate the ambiguity-prone mismatching of similar nodes in different positions.

#### *4.1.1. Anchor-set Construction for Small Graph*

The position anchor framework is motivated by PGNN [31] which emphasizes large graphs (i.e. those consisting of at least thousands of nodes). However, in this paper, we further study the anchor mechanism and focus on small graphs constructed from images. Specifically, the anchor-sets first proposed for large graphs are completely randomly chosen subsets of all nodes from the graph. They are intended to sample multiple anchor-sets $P = \{P_{i,j}\}$, where the number of anchor-sets $|P| = \log^2 |V|$, $i = 1, 2, ..., \log |V|$,

$j = 1, 2, ...c \log |V|$. Here, $|V|$ is the number of nodes in the graph and c is a hyper-parameter. Each node is sampled independently and randomly with a probability of $1/2^i$. The idea central to our method is that the anchor-sets can include each node or any of its one-hop neighbors. Large anchor sets have low position information with a large sampling probability. Small anchor sets have high position information with low sampling probability. The balance of these different sizes of anchor-sets may lead to efficient embeddings. However, this random sampling operation can result in the final anchor sets sampling almost all nodes in the graph. The proof is as follows:

*The sampling probability for each node in the anchor-set $P_{i,j}$ is $1/2^i$, and the number of anchors in each anchor-set $P_{i,j}$ is*

$$p_i = \frac{|V|}{2^i} \quad \forall i \in [1, \log |V|], \tag{2}$$

*where $|V|$ is the number of nodes in the graph.*

*Since the number of anchor points $p_i$ in the anchor-set $P_{i,j}$ is only related to element i, the construction of anchor-sets $P = \{P_{i,j}|(i \in [1, \log |V|], j \in [1, c \log |V|])\}$ is actually the repeated construction of anchor-sets $P = \{P_{i,j}|(i \in [1, \log |V|], j = 1)\}$ for $c \log |V|$ multiplied by the uniform sampling probability for each anchor in each anchor set construction step. Hence, we consider each anchor in a non-repetitive operation, i.e. the anchors in anchor-sets $P = \{P_i|(i \in [1, \log |V|])\}$. Therefore, the number of anchors p in the entire set of anchor-sets $P = \{P_i|(i \in [1, \log |V|])\}$ is given by:*

$$p = \sum_{i=1}^{log|V|} p_i = |V| - 1. \tag{3}$$

*The anchor sets obtained through fair non-repetitive sampling operation contain $|V| - 1$ anchors for a $|V|$ node graph and the sampling probability for each anchor is $(|V| - 1)/|V|$, which tends to unity for large graphs.*

Based on the derivation above, we propose an effective strategy to construct an anchor set $P$ consisting of all nodes in the graph, denoted by $P = V$, and serving as a stable reference for all nodes. According to our central idea in this paper, when all nodes in a graph become an anchor in an anchor-set, each node aggregates information with anchors based on a relative position relation. This is equivalent to each target node being subject to a relative position-based attention mechanism based on the global arrangement of nodes in the graph.

*4.1.2. Aggregation of Node Embedding*

With the anchor-set selected, we continue to compute the position-aware node embedding through the information aggregation between the nodes, in which each anchor cooperates with its relative position attributes. Here, we first compute the relative position coefficient $q_{v,u}$ between a pair of nodes $v$ and $u$ as:

$$q_{v,u} = \frac{e^{-d_{v,u}}}{\sum_{i \in V} e^{-d_{v,i}}}, \tag{4}$$

where $d_{v,u}$ is the shortest path distance between the two nodes. Considering the time complexity of calculating the shortest path, we stipulate that the maximum shortest path distance is not more than $r$, otherwise, the value is infinite. In this way, we can effectively eliminate the interference of anchors that are too far away from the node when the coefficient $q_{v,u}$ equals zero. We define the information aggregation function $I(v,u)^{(k)}$ for $k$-th layer between nodes $v$ and $u$ as:

$$I(v,u)^{(k)} = q_{v,u}\text{CONCAT}(h_v^{(k-1)}, h_u^{(k-1)}), \tag{5}$$

where $q_{v,u}$ is the relative position coefficient between node $u$ and $v$, $h_v^{(k-1)}$ and $h_u^{(k-1)}$ are hidden representations containing the position information in $(k-1)$-th layer. The message aggregation function CONCAT concatenates the hidden representations $h_v^{(k-1)}$ and $h_u^{(k-1)}$.

We further aggregate the information of each pair of nodes and the anchor with a non-linear transformation applied after the aggregation to achieve higher expressive power. The hidden representation passed to the next layer is:

$$h_v^{(k)} = \sigma(\text{AGG}(I(v,u)^{(k)} | \ \forall u \in V)W^{(k)}), \tag{6}$$

where AGG is a permutation-invariant function (e.g., the sum), and $W^{(k)}$ is a learnable weight vector for the $k$-th layer.

The general position-aware node embedding framework is summarized in Algorithm 1.

*4.2. Subgraph-based Structure Embedding*

In addition to the relative structural information, we further introduce subgraph structure embedding during the feature extraction and the information aggregation for each node. The subgraph-based structure embedding

**Algorithm 1:** Position-aware node embedding framework.

**Input:** input original graph $\mathcal{G} = (V, A, X)$, anchor-set:$S = V$, relative position coefficient $q$, trainable weight in $k$-th layer $W^{(k)}$, network layer $k \in [1, K]$

**Output:** Position-aware node embedding $h_p$

$h_v = X_v$
**for** $k \in [1, K]$ **do**
  **for** $v \in V$ **do**
    $I(v, u)^{(k)} = q_{v,u}\text{CONCAT}(h_v^{(k-1)}, h_u^{(k-1)})$
    $h_v^{(k)} = \sigma(\text{AGG}(I(v, u)^{(k)})W^{(k)}),\ \forall u \in V$
  **end**
**end**
**return** $h_v^{(K)}$

can be split into two steps: a) We first divide the original graph into different subgraphs according to the pre-defined structure. Each subgraph can be regarded as a new supernode, and the edges between new nodes are determined by the relationships between the nodes in each subgraph, thus we can construct a so-called **subgraph relation graph**. b) In the second step, we aggregate the structural arrangement information of the subgraphs in the subgraph relation graph and merge them with the relevant nodes in the original graph while concurrently aggregating the local information at the node level of the original graph.

*4.2.1. Subgraph Construction*

Here we retain the global structural arrangement information from the original graph to the greatest extent, and extract the key representative structural features for the subgraphs. Note that it is particularly important to select the nodes of the subgraph relation graph, i.e. pre-defined subgraphs in original graphs. In this paper, we define three different types of subgraph structures, namely a) paths, b) trees, and c) circuits. Combinations of these three basic subgraph types can be used to construct any required graph structure.

Since the search for all subgraphs in a graph is NP-hard, we propose a novel searching method to detect the $t$-hop neighbors of nodes to find subgraphs, and then save the search path in real-time to reduce the time

complexity. The corresponding detection Algorithm 2 is designed to handle how to find the three different types of subgraph structures.

For tree structure detection, we first initialize two sets: $N^t(v)$ to store the $t$-hop neighbor of node $v$, where $t \in [1, D]$ and $D$ is the maximum neighbor depth of the graph, and $P_v(u)$ stores the path from node $v$ to node $u$. We traverse the adjacency matrix to find all edges, and store the 1-hop neighbor set of each node. The tree-structural subgraph is selected for further processing if the number of leaf nodes exceeds the preset threshold $\delta$. For the path and circuit subgraphs, we continue the search for the $t$-hop neighbors of each node through the combination of several existing $t$-hop neighbors. If a path from node $v$ to node $u$ already exists in the set $P_v(u)$ and there is no repeating node in the path, we identify the path as a circuit subgraph. The searching progress for three different types of subgraph structure is shown in Fig. 4.

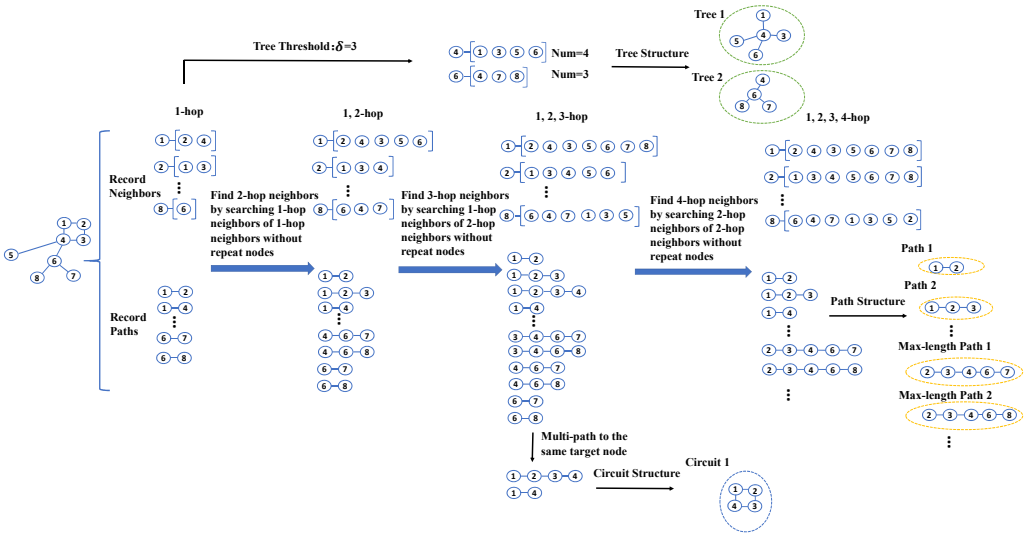


Figure 4: The searching progress for three different types of subgraph structure.

After obtaining these three different types of different subgraphs, we construct the so-called **subgraph relation graph** to capture the internal relationship between different subgraph types. This process is shown in Fig. 5:

**Definition 1:** ***Subgraph Relation Graph.*** Let an undirected graph $\hat{\mathcal{G}} = (\hat{V}, \hat{A}, T)$ constructed from the original graph $\mathcal{G}$, in which the node

**Algorithm 2:** The searching algorithm for the three different types of subgraph structure.

**Input:** input original graph $\mathcal{G} = (V, A, X)$, max neighbor depth $D$, min node leaf $\delta$

**Output:** output subgraph set $S$

initialize subgraph set $S = \{S_i\}$, neighbor set $\{N^t(v)\}$, path set $\{P_v(u)\}$, subgraph relation graph $\hat{\mathcal{G}}$ with $|\hat{V}|$ vertices
**for** $t \in [1, D]$ **do**
  **for** $v \in V$ **do**
    **if** $t == 1$ **then**
      **for** $u \in V$ **do**
        Add $u, v$ into $N^1(v), N^1(u)$
        Add $v, u$ into $P_v(u), P_u(v)$
        **if** $|N^1(v)| \geq \delta$ **then**
          add $(N^1(v) + [v])$ as Tree Subgraph into $S$
        **end**
      **end**
    **else**
      **for** $u \in N^i(v) | 2^i < t \leq 2^{i+1}$ **do**
        **for** $r \in N^{t-2^i}$ **do**
          **if** $P_v(r) \in \{P_v(u)\} and\ r \notin P_v(u)$ **then**
            Add $P_v(u)$ as Circuit Subgraph into $S$
          **else**
            Add $r, v$ into $N^t(v), P_v(r)$
            Add $\{P_v(u), P_u(r)\}$ to $P_v(r)$
            Add $P_v(r)$ to $P_r(v)$
          **end**
        **end**
      **end**
    **end**
  **end**
**end**
Add $P_v(u)$ as Path Subgraph into $S$
**return** $S$

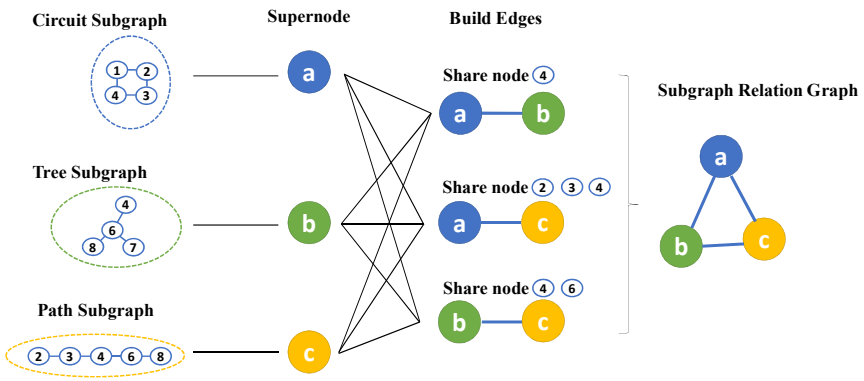


Figure 5: An example of the *Subgraph Relation Graph*.

set $\hat{V}$ contains $|\hat{V}|$ nodes, each denoting a different subgraph identified in the original graph. The adjacency matrix element of $\hat{A}$ equals one if two nodes share the same node in the original graph. The transformation matrix $T \in \mathbb{R}^{|V|*|\hat{V}|}$ represents the matrix that transforms the original graph into the subgraph relation graph.

Let node $v_i$ belong to node-set $V$ of the source graph, and node $\hat{v_j}$ belong to node-set $\hat{V}$ in the target graph corresponding to subgraph $S_j$ in the source graph. In this way, the transformation matrix $T$ has the element:

$$T_{ij} = \begin{cases} 1 & v_i \in S_j, \\ 0 & \text{else .} \end{cases} \tag{7}$$

Algorithm 3 gives an overview of the subgraph relation graph construction procedure.

*4.2.2. Merged Structure Embedding*

We augment the graph affinity embedding to better fuse the high-order information from the nodes by developing a merged structure message-passing scheme. We commence by aggregating the structural arrangement information of the subgraph relation graph at the subgraph level with the classical embedding procedure:

$$h_{\hat{v_i}}^{(k)} = \text{COMBINE}(h_{\hat{v_i}}^{(k-1)}, \text{AGG}\{h_{\hat{v_j}}^{(k-1)} | \hat{v_j} \in N(\hat{v_i})\}), \tag{8}$$

**Algorithm 3:** Construction of the subgraph relation graph.

**Input:** input subgraph set $S$

**Output:** output subgraph relation graph $\hat{\mathcal{G}} = (\hat{V}, \hat{A}, T)$

Create a new subgraph set $\hat{S}$ by combining one of the longest path subgraphs, one of the tree subgraphs, and the circuit subgraphs from subgraph set $S$.

**for** *all* $S_i$ *in subgraph set* $\hat{S}$ **do**
    Assign $S_i$ to each node $\hat{v_i}$
    **if** $\exists v \in S_i \cap S_j$ **then**
        Add edge $\hat{A_{ij}}$ between node $\hat{v_i}$ and $\hat{v_j}$
    **end**
    **if** $v_i \in S_j$ **then**
        $T_{ij} = 1$
    **else**
        $T_{ij} = 0$
    **end**
**end**

**return** $\hat{\mathcal{G}}$

where $N(\hat{v_i})$ denotes the set of neighbors of node $\hat{v_i}$ (i.e. which corresponds to subgraph $S_i$ in the original graph) in the subgraph relation graph. Here, the functions AGG and COMBINE are sometimes folded into a single aggregation update. Specifically, AGG is a function that aggregates all neighbor features of the node $S_i$, and COMBINE updates the representation of node $S_i$ according to the output of AGG. We perform information aggregation and combination at the structural level. We then further merge them with the position-aware node embedding extracted from the original graph using the following procedure:

$$h_v^{(k)} = \mathrm{COMBINE}_{\mathrm{M}}(h_v^{(k)}, \mathrm{AGG}_{\mathrm{M}}(h_{\hat{v_i}}^{(k)} | \forall v \in S_i)), \tag{9}$$

where $h_{\hat{v_i}}^{(k)}$ denotes the hidden representation of subgraph node $S_i$ extracted using Eq. 8, $h_v^{(k)}$ represents the hidden node embedding combined with structure and position in the $k$-th layer. The function is the sum over all subgraph embeddings that contain the node $v$ and is computed as $AGG_M(h_S) = T * h_S$. We define as $\mathrm{COMBINE}_{\mathrm{M}}(h_v, h_s) = \alpha h_s + \beta h_v$ where $\alpha$ and $\beta$ are two hyper-

parameters that can be used to balance the weight of position relationships and structural arrangement information.

The position-aware node embedding algorithm is summarized in Algorithm 4.

**Algorithm 4:** Merge structure embedding framework.

**Input:** input original graph $\mathcal{G} = (V, A, X)$, subgraph relation graph $\hat{\mathcal{G}} = (\hat{V}, \hat{A}, T)$, position-aware node embedding $h_p$, trainable weight $W$, network layer $k \in [1, K]$

**Output:** Subgraph-based Structure embedding $h_s$

$h_v = X_v$
**for** $k \in [1, K]$ **do**
  **for** $v \in V$ **do**
    $h_{\hat{v}_i}^{(k)} = \mathrm{COMBINE}(h_{\hat{v}_i}^{(k-1)}, \mathrm{AGG}\{h_{\hat{v}_j}^{(k-1)} | \hat{v}_j \in N(\hat{v}_i)\}))$
    $h_v^{(k)} = \mathrm{COMBINE}_{\mathrm{M}}(h_v^{(k)}, \mathrm{AGG}_{\mathrm{M}}(h_{\hat{v}_i}^{(k)} | \forall S_i \supset v))$
  **end**
**end**
**return** $h_v^{(K)}$

### *4.3. Graph Feature Matching*

With the proposed position-aware node embedding framework and subgraph-based structural embedding framework to hand, we encode each node with high-order graph structure information and position information and construct a high-level embedding space. Such an embedding scheme allows us to simplify the second-order affinity matrix to a linear one.

With the final hidden representation of the source graph $H_s \in \mathbb{R}^{n*F'}$ and the target graph $H_t \in \mathbb{R}^{m*F'}$ to hand, we can obtain a soft correspondence between these graphs using the corresponding affinity matrix $\mathrm{H_{affinity}} = \exp\left(\mathrm{H_s \Gamma H_t^T}\right)$. Each element in the affinity matrix $\mathrm{H_{affinity}} \in \mathbb{R}^{\mathrm{n*m}}$ represents the affinity between a pair of nodes in two graphs considering node features, position-aware information, and high-order subgraph information in the graph, with $\Gamma$ as a learnable weight matrix. We further apply the Sinkhorn normalization [32] to obtain a rectangular doubly-stochastic correspondence matrix to fulfill the constraints on injectivity from the original graph $\mathcal{G}_s$ to the target graph $\mathcal{G}_t$. Here, the Sinkhorn operation works

iteratively by iterative row-wise and column-wise normalization,

$$\begin{aligned} H^r_{\text{affinity}} &= H^c_{\text{affinity}} \oslash (H^c_{\text{affinity}} \mathbf{1_m} \mathbf{1_m}^T) \\ H^c_{\text{affinity}} &= H^r_{\text{affinity}} \oslash (\mathbf{1_n} \mathbf{1_n}^T H^r_{\text{affinity}}) \end{aligned} \tag{10}$$

where $\oslash$ indicates element-wise division, and $\mathbf{1_m} \in \mathbb{R}^{m*1}$, $\mathbf{1_n} \in \mathbb{R}^{n*1}$ are column vectors whose elements are all ones, $H^r_{\text{affinity}}$ and $H^c_{\text{affinity}}$ are the intermediate results of row normalization and column normalization respectively. We simplify these two iterative processes giving a continuous relaxation of the permutation matrix:

$$S = \text{Sinkhorn}(\text{H}_{\text{affinity}}) \tag{11}$$

We further employ the cross-entropy loss function as the permutation loss between the predicted permutation matrix and the ground truth:

$$L = -\sum_{i \in \mathcal{V}_s, j \in \mathcal{V}_t} \left( \mathbf{S}^{gt}_{i,j} \log \mathbf{S}_{i,j} + \left(1 - \mathbf{S}^{gt}_{i,j}\right) \log \left(1 - \mathbf{S}_{i,j}\right) \right), \tag{12}$$

where $\mathbf{S}^{gt}$ is the ground truth permutation matrix with each element as the ground truth node-to-node correspondence, and $S$ is the predicted permutation matrix given by Eq. 11. In this way, our cross-entropy loss can directly learn the permutation loss based on linear allocation cost in an end-to-end fashion, no matter how the number of nodes and edges in the graph change.

To illustrate in detail the complete framework adopted in our method, we provide an illustration of the processing steps involved in obtaining an embedding commencing from an image input through to the final output. Also shown is an illustration of the network structure used in each layer. In the example we take the batch size as 8, the number of key points as 12, and the number of motifs as 21. The complete data analysis workflow schematic is shown in Figure 6. The steps are as follows. Firstly, the resized image is input to a pre-trained VGG-16 model, and we compute the outputs of the $Relu4\_2$ and $Relu5\_1$ operations to extract node features and edge features. We align the keypoints with the guidance of the keypoint coordinates to obtain graph features. This also forms the input of the proposed position and structure embedding network. The operation of image feature extraction closely follows previous graph matching methods [5, 8, 10, 17] so as to guarantee equity in comparison. Then, for position-aware node embedding, the network concatenates the representation of each node with the anchor

representation multiplied by a weighting coefficient. A sum operator and one-layer non-linear transformation are performed to obtain the position embedding. For subgraph-based structure embedding, after constructing the motif graph, a single graph convolutional layer is applied to obtain the high dimensional structural embedding. The graph embedding can be computed using the combination of the position embedding and the corresponding structure embedding. We compute the affinity matrix using the paired graph embeddings combined with a learnable weight. The final accuracy of the method is computed by utilizing the available ground truth and a permutation matrix obtained from the affinity using the Sinkhorn activation function.

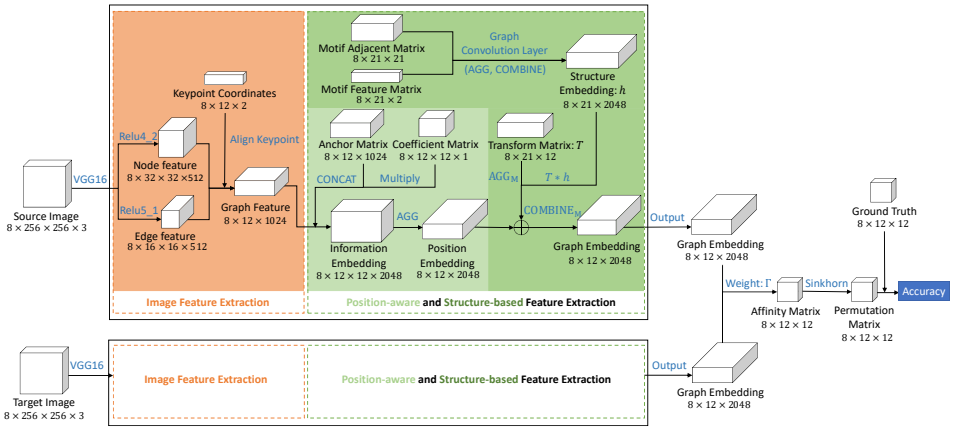


Figure 6: The entire data analysis workflow example with batch size as 8, the number of key points as 12, and the number of motifs as 21.

## 5. Experiments

In this section, we verify our method in three different tasks: We first perform our experiments in real-world tasks of supervised keypoint matching in four different kinds of natural images. Then, we further demonstrate the effectiveness of each procedure in our framework in an ablation study. Finally, we conduct the experiment on transfer learning across different categories, to evaluate the generalization capability of our methods.

### *5.1. Datasets*

**Willow ObjectClass.** The Willow Object Class dataset [33] contains five object categories including faces, cars, ducks, motorbikes, and wine bot-

tles. This dataset is a benchmark used to measure the ability of image classification and recognition. There are 9963 images in total, including 24640 labeled objects. In the experiment, we select a small subset for each category for training and use the remainder for testing. Each category has at least 40 images.

**Pascal VOC 2011.** The Pascal Visual Object Classes (VOC) 2011 dataset [11] contains 20 object sub-categories and four major categories in total, which are vehicles, indoors, animals, and people. Each image in the dataset has three annotations including category labels, attributes, locations, and bounding boxes. Note that most images in this dataset contain complicated component scenes.

**CUB-200-2011.** The Caltech-UCSD Birds-200-2011 (CUB-200-2011) dataset [34] contains 11788 images with 200 categories of birds. Each image has clear annotations containing 1 subcategory label, 15 part locations, 312 binary attributes, and 1 bounding box.

**IMC-PT-SparseGM.** The IMC-PT-SparseGM dataset [35] contains 16 categories and about 25061 images. This dataset gathers data from 16 tourist attractions around the world. Compared with the datasets mentioned above, the number of images in this dataset is the largest.

### *5.2. Alternative for Comparison*

The state-of-art benchmark methods used for comparison experiments are:

**GMN.** (Graph Matching Network, GMN) Zanfir *et al.* [10] developed the deep graph matching method employing an end-to-end deep learning framework. To extract image features, this framework adopts VGG-16 [16] networks, with first-order and second-order features extracted from the shallower layer (relu4_2) and deeper layer (relu5_1) of VGG-16, respectively. It also uses Delaunay triangulation to build graphs. Factorized Graph Matching (FGM) has been used to construct the affinity matrix efficiently. The method is based on supervised learning using a loss function on the offset loss. Although this method is relatively simple, it pioneered the use of deep learning to solve graph-matching problems.

**PIA-GM/PCA-GM.** Permutation loss and intra-graph affinity-based graph matching (PIA-GM) and permutation loss and cross-graph affinity-based graph matching (PCA-GM) [17] replace the offset loss on GMN with the permutation loss. This can improve the learning process. The image preprocessing of PIA-GM is the same as GMN. It employs 3 GNN layers to

perform intra-graph convolution. The feature dimension of the GNN layer is 2048. The main difference between PIA-GM and the original GMN architecture is that it introduces graph neural networks to learn the node embedding of the graph. The difference between PCA-GM and PIA-GM is that the former changes the method of graph convolution, from intra-graph convolution to cross-graph convolution. The aim is to achieve better extraction of information for similar nodes between two input graphs and better generation of the node embedding.

**IPCA-GM.** (Iterative Permutation loss and cross-graph affinity-based graph matching, IPCA-GM [36]). To better combine combinatorial optimization with a deep learning solution, IPCA-GM considers the iterative interaction between the Sinkhorn layer and the GCN layer to learn a better feature embedding. This approach fully considers the embedding information for nodes, and reduces the original two-dimensional QAP problem to a one-dimensional one, thus reducing the complexity and improving the learning efficiency.

**CIE-H.** (Channel-Independent Embedding and Hungarian attention, CIE-H [37]). This follows the CNN-GNN-metric-Sinkhorn pipeline proposed by PCA-GM and improves PCA-GM in two ways: First, the edge attribute information is taken into account and combined with node embedding. This is used to produce a channel-independent embedding. Second, a Hungarian Attention module dynamically constructs a structured and sparsely connected layer. This considers the most significant matching pair during training.

**NGM/NHGM.** (Neural Graph Matching Network (NGM) and Neural Hyper Graph Matching Network, NHGM [8]). These are recognized as a learnable solution to Lawler's quadratic assignment problem. This first transforms Lawler's QAP into an association graph and founds a solution, which is equivalent to the vertex classification problem on the association graph. Finally, a graph neural network is used to solve the vertex classification problem. NGM considers the task of matching graph pairs, and can be regarded as Lawler's QAP solver using VGG-16. On the other hand, NHGM is a hypergraph matching solver with VGG-16. Both of these methods are based on supervised learning using permutation loss.

**GANN-GM/GANN-MGM.** (GANN-GM [5]) is self-supervised learning graduated assignment neural network for matching graph pairs. (GANN-MGM [5]) is a self-supervised learning graduated assignment neural network for multi-graph matching. Both were proposed by Wang *et al.* [5]. GANN-GM introduces a self-supervised learning framework by leveraging graph

matching solvers to provide pseudo labels to train the neural network module in a deep graph matching pipeline. A general graph matching solver is proposed for various graph matching settings based on the classic Graduated Assignment (GA) algorithm.

### *5.3. Experimental Settings*

The experiments are conducted using two GeForce GTX 1080 Ti GPUs. We employ a batch size of 8 in training and evaluate our model in 2000 epochs for each iteration. In terms of experimental settings, we employ the Adam [38] optimizer to train our models with a learning rate of $1 \times 10^{-4}$. To overcome the over-smoothing problem common to graph neural networks, which arises from the transmission of information between nodes caused by deepening the network layer. This results in the representations of different nodes becoming similar and causes ambiguities in matching. We adopt a two-layer graph embedding and restrict the degree of smoothing in our experiments. When we perform experimental comparisons and ablation studies, we train and test each dataset independently. For the cross-category generalization study on the Willow ObjectClass dataset, we train the model on the Pascal VOC dataset excluding the car and motorbike classes, as these images overlap with the Willow dataset, and then test on the Willow ObjectClass dataset afterward. To guarantee equity, the splits between the training and testing datasets are identical for all of the methods compared [5, 8, 10, 17]. The details of the hyperparameter settings are summarized in Table 2.

Table 2: Summary of parameter settings for datasets under study.

| Datasets | Average Node Number | K | r | $\delta$ | $\alpha$ | $\beta$ |
|---|---|---|---|---|---|---|
| Pascal VOC | 9.07 | 2 | 1 | 2 | $1 \times 10^{-4}$ | $1 \times 10^{-5}$ |
| Willow ObjectClass | 10.00 | 2 | 1 | 2 | $1 \times 10^{-4}$ | $1 \times 10^{-4}$ |
| CUB 2011 | 12.00 | 2 | 1 | 2 | $1 \times 10^{-4}$ | $1 \times 10^{-5}$ |
| IMC-PT-SparseGM | 21.36 | 2 | 1 | 3 | $1 \times 10^{-4}$ | $1 \times 10^{-4}$ |

### *5.4. Performance Comparisons on Four Datasets*

In this experiment, the source graph $\mathcal{G}_s$ and target graph $\mathcal{G}_t$ are composed of keypoints extracted from two different images. Both two graphs maintain node consistency during the training and testing stages. We perform the same operations on four datasets to convert the images into graphs. First, we filter the outliers from the dataset. Next, we crop the images around their bounding boxes, which provide the pixel location of the object boundings in

images. Their image sizes are further resized to $256 \times 256$. The preprocessed images are then fed to VGG-16 to extract the node features. The graphs are constructed by Delaunay triangulation [39] of the extracted feature points.

We evaluate the graph matching capacity of the proposed method using the matching accuracy, which is defined by $accuracy = \sum \mathrm{AND}(S_{i,j}, S^{gt}_{i,j})/N$ from [5, 17, 36]. Here, the AND operation is a logical operation. Note that $S_{i,j}$ is the element of the predicted permutation matrix representing the correspondence matching of node $i$ and node $j$ from two different graphs. Similarly, $S^{gt}_{i,j}$ is the correspondence of the ground truth between the two nodes, and $N$ is the number of matching node pairs. We use ground truth node correspondences (which are set to one if two nodes are matched, and zero otherwise) for the image pairs, and perform an averaging operation to compute the final matching accuracy.

Table 3: Matching accuracy on the Willow ObjectClass database(%).

| Method | Car | Duck | Face | Motorbike | Winebottle | Mean accuracy |
|---|---|---|---|---|---|---|
| GMN [10] | 67.90 | 76.70 | 99.80 | 69.20 | 83.10 | 79.34 |
| NGM [8] | 84.20 | 77.60 | 99.40 | 76.80 | 88.30 | 85.30 |
| NHGM [8] | 86.50 | 72.20 | 99.90 | 79.30 | 89.40 | 85.50 |
| CIE-H [37] | 82.20 | 81.20 | **100.00** | **90.00** | **97.60** | 90.20 |
| PIA-GM [17] | 88.60 | 87.00 | **100.00** | 70.30 | 87.80 | 86.74 |
| PCA-GM [17] | 87.60 | 83.60 | **100.00** | 77.60 | 88.40 | 87.44 |
| IPCA-GM [36] | 90.40 | 88.60 | **100.00** | 83.00 | 88.30 | 90.06 |
| PCA-GM+Position+Structure | **92.90** | **90.80** | **100.00** | 79.00 | 94.40 | **91.42** |
| IPCA-GM+Position+Structure | 92.70 | 88.70 | **100.00** | 80.20 | 94.90 | 91.30 |

The experimental results on the Willow ObjectClass, Pascal VOC, CUB and IMC-PT-SparseGM datasets are shown in Table 3, Table 4, Table 5, and Table 6, respectively. We also compare the proposed methods(i.e. PCA-GM+Position+Structure and IPCA-GM+Position+Structure) with the alternative methods introduced in Section 5.2. Specifically, our methods achieve the best performance in the classification accuracy of three different categories on the Willow ObjectClass dataset, while the CIE-H method outperforms the Motorbike and Wine bottle classes. Note that the CIE-H method also achieves good results on the VOC and CUB datasets. However, it performs poorly on the IMC-PT-SparseGM dataset. The VOC dataset contains 20 object sub-categories and four major categories. It is therefore much more complicated than the remaining datasets under study. As observed from Table 4, many methods perform better than others in one or several specific categories. Our method performs slightly lower than the highest average ac-

Table 4: Matching accuracy on the Pascal VOC database(%).

| Method | Aero | Bike | Bird | Boat | Bottle | Bus | Car | Cat | Chair | Cow | Table | Dog | Horse | Motorbike | Person | Plant | Sheep | Sofa | Train | TV | Mean accuracy |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GMN [10] | 31.90 | 47.20 | 51.90 | 40.80 | 68.70 | 72.20 | 53.60 | 52.80 | 34.60 | 48.60 | **72.30** | 47.70 | 54.80 | 51.00 | 38.60 | 75.10 | 49.50 | 45.00 | 83.00 | 86.30 | 55.30 |
| NGM [8] | 50.10 | 63.50 | 57.90 | 53.40 | 79.80 | 77.10 | 73.60 | 68.20 | 41.10 | 66.40 | 40.80 | 60.30 | 61.90 | 63.50 | 45.60 | 77.10 | 69.30 | 65.50 | 79.20 | 88.20 | 64.13 |
| NHGM [8] | 52.40 | 62.20 | 58.30 | 55.70 | 78.70 | 77.70 | **74.40** | 70.70 | 42.00 | 64.60 | 53.80 | 61.00 | 61.90 | 60.80 | 46.80 | 79.10 | 66.80 | 55.10 | 80.90 | 88.70 | 64.58 |
| CIE-H [37] | 49.94 | 63.13 | **70.65** | 52.98 | **82.43** | 75.36 | 67.66 | 72.30 | 42.35 | **66.88** | 69.90 | **69.52** | **70.74** | 61.96 | 46.67 | 85.04 | 70.00 | 61.75 | 80.23 | **91.78** | 67.56 |
| PIA-GM [17] | 41.50 | 55.80 | 60.90 | 51.90 | 75.00 | 75.80 | 59.60 | 65.20 | 33.30 | 65.90 | 62.80 | 62.70 | 67.70 | 62.10 | 42.90 | 80.20 | 64.30 | 59.50 | 82.70 | 90.10 | 63.00 |
| PCA-GM [17] | 40.90 | 55.00 | 65.80 | 47.90 | 76.90 | **77.90** | 63.50 | 67.40 | 33.70 | 65.50 | 63.60 | 61.30 | 68.90 | 62.80 | 44.90 | 77.50 | 67.40 | 57.50 | 86.70 | 90.90 | 63.80 |
| IPCA-GM [36] | **53.78** | 66.22 | 67.14 | **61.20** | 80.39 | 75.27 | 72.55 | **72.52** | **44.55** | 65.24 | 54.30 | 67.24 | 67.90 | 64.21 | **47.93** | 84.35 | **70.79** | 63.98 | 83.80 | 90.83 | **67.70** |
| PCA-GM+Position+Structure | 49.34 | **68.51** | 63.32 | 56.37 | 81.34 | 75.81 | 65.8 | 70.93 | 42.90 | 66.15 | 70.46 | 64.21 | 66.78 | **65.05** | 42.31 | **85.06** | 67.60 | **67.25** | **86.87** | 90.06 | 67.31 |
| IPCA-GM+Position+Structure | 48.72 | 65.52 | 56.08 | 52.86 | 78.76 | 73.66 | 64.4 | 66.18 | 39.73 | 65.18 | 63.15 | 61.90 | 66.06 | 62.68 | 42.21 | 82.89 | 64.46 | 60.8 | 86.40 | 89.45 | 64.56 |

Table 5: Matching accuracy on the CUB database(%).

| Method | Mean accuracy |
|---|---|
| GANN-GM [5] | 48.44 |
| GANN-MGM [5] | 78.32 |
| CIE-H [37] | 92.06 |
| PIA-GM [17] | 88.56 |
| PCA-GM [17] | 92.17 |
| IPCA-GM [36] | 92.18 |
| PCA-GM+Position+Structure | 91.46 |
| IPCA-GM+Position+Structure | **92.34** |

Table 6: Matching accuracy on the IMC-PT-SparseGM (%).

| Method | Reichstag | Sacre_coeur | St_peters_square | Mean accuracy |
|---|---|---|---|---|
| GANN-GM [5] | 76.02 | 44.15 | 50.49 | 56.89 |
| GANN-MGM [5] | 67.41 | 42.72 | 44.42 | 51.52 |
| CIE-H [37] | 42.24 | 28.47 | 30.78 | 33.83 |
| PIA-GM [17] | 71.46 | 41.31 | 42.64 | 51.80 |
| PCA-GM [17] | 69.38 | 39.86 | 42.10 | 50.40 |
| IPCA-GM [36] | 72.96 | 43.80 | 44.93 | 53.89 |
| PCA-GM+Position+Structure | **96.28** | **75.93** | 81.66 | **84.63** |
| IPCA-GM+Position+Structure | 95.71 | 74.12 | **82.06** | 83.96 |

curacy. For the CUB and IMC-PT-SparseGM datasets, our method achieves the highest mean accuracy performance compared with the alternatives. It can therefore be concluded that our model performs well on all four datasets under study, while the alternatives generally fail to perform well on at least one of the datasets. This also clearly demonstrates the advantage of our method in terms of generalization ability.

### *5.5. Ablation Study*

Here we conduct an ablation study to demonstrate that our novel method for embedding the positional and structural arrangement information can be effective in improving variants of the methods of the performance of

Table 7: Matching accuracy of ablation study on IMC-PT-SparseGM (%).

| Method | Reichstag | Sacre_coeur | St_peters_square | Mean accuracy |
|---|---|---|---|---|
| PCA-GM | 69.38 | 39.86 | 42.10 | 50.40 |
| PCA-GM+Position | 92.02 | 69.07 | 78.80 | 79.96 |
| PCA-GM+Structure | 93.83 | 70.44 | 78.78 | 81.02 |
| PCA-GM+Position+Structure | 96.49 | 72.01 | 83.00 | 83.83 |
| IPCA-GM | 72.96 | 43.80 | 44.93 | 53.89 |
| IPCA-GM+Position | 93.03 | 65.37 | 77.24 | 78.55 |
| IPCA-GM+Structure | 92.79 | 62.66 | 76.98 | 77.48 |
| IPCA-GM+Position+Structure | 95.71 | 74.12 | 82.06 | 83.96 |

Table 8: CNN backbone of ablation study on IMC-PT-SparseGM (%).

| Method | Reichstag | Sacre_coeur | St_peters_square | Mean accuracy | Time (min) |
|---|---|---|---|---|---|
| PCA-GM+Position+Structure (VGG-16) | 96.49 | 72.01 | 83.00 | 83.83 | 1173 |
| PCA-GM+Position+Structure (VGG-19) | 96.35 | 72.07 | 83.21 | 83.87 | 1358 |
| PCA-GM+Position+Structure (ResNet-50) | 96.45 | 71.84 | 83.08 | 83.79 | 979 |
| PCA-GM+Position+Structure (MobileNet-v3) | 96.31 | 71.45 | 83.24 | 83.67 | 815 |
| IPCA-GM+Position+Structure (VGG-16) | 95.71 | 74.12 | 82.06 | 83.96 | 1434 |
| IPCA-GM+Position+Structure (VGG-19) | 95.53 | 73.76 | 81.79 | 83.69 | 1706 |
| IPCA-GM+Position+Structure (ResNet-50) | 95.78 | 74.15 | 82.07 | 84.00 | 1179 |
| IPCA-GM+Position+Structure (MobileNet-v3) | 95.62 | 74.00 | 81.93 | 83.85 | 1068 |

graph matching. To this end, we construct three models based on the graph matching methods PCA-GM and IPCA-GM. The three variants are a) the original method combined with position-aware embedding, b) the original method combined with structure embedding, and c) the original method combined with complete position and structure embedding. Considering that our method has the alternative best performance on the IMC-PT-SparseGM data set compared with other methods, we conduct the ablation study on the IMC-PT-SparseGM dataset. The experimental results are shown in Table 7.

The proposed position-aware node embedding and subgraph-based structure embedding greatly improve the classification accuracy of the original method in each category. Note that the IMC-PT-SparseGM dataset consists of tourist attractions, of which many (especially the buildings) are symmetrical, indicating that the semantic information of the keypoints is usually ambiguous. Therefore, the incorporation of position information for the graph matching methods can effectively assist the classification task. The high-order subgraph-based structural arrangement information is also valuable, since the global structural features of the tourist attractions can provide more comprehensive information than just local features alone.

Furthermore, in this paper we make use of VGG-16 to extract image features. Note that the proposed method is not designed to be critically dependent on VGG-16, and the chosen feature extractor step can be replaced by any of the other available CNN models. Furthermore, we conduct an ablation study that compares the different CNN backbones applied in the proposed method, both in terms of performance and also computation time. Specifically, we include VGG-19, ResNet-50, and MobileNet-v3 in addition to VGG-16. Each of these backbone models is sourced from the TorchVision library [40]. The results are shown in Table 8, from which it can be concluded that using the choice of different backbone models has little effect on

the performance. Although there are minor differences in the initial feature extraction capability, the high-order feature extraction and matching algorithm do have a major influence on the final graph matching outcome. As far as time consumption is concerned, MobileNet-v3 shows clear advantages in terms of computation time due to its lightweight design. However, this is at the cost of sacrificing some accuracy. Here, we are at pains to ensure fairness of comparison with alternative methods. Based on these considerations we opt to use VGG-16 as the backbone architecture in image feature extraction.

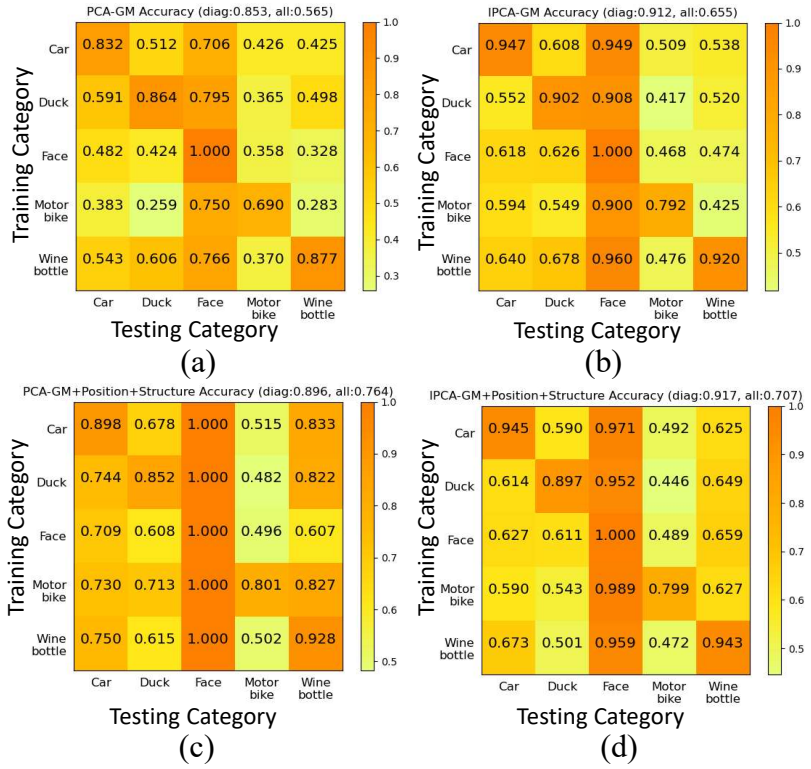


Figure 7: Cross-category generalization study represented by confusion matrix on Willow ObjectClass dataset. Each column of the matrix represents one category of images used for model training, while each row represents one class of testing samples. a) and b) show the matching accuracy of comparison methods in different category generalizations, respectively. c) and d) are obtained from the proposed methods.

### *5.6. Cross-category Generalization Study*

To verify the generalization capability of our methods on different types of categories, we construct the confusion matrix for the training and testing

instances from the different categories. The experimental results are presented in Fig. 7 and Fig. 8. We first train each of the five classes separately for the Willow dataset. We then use the distinct training classes to validate the generalization capabilities for all five object classes.

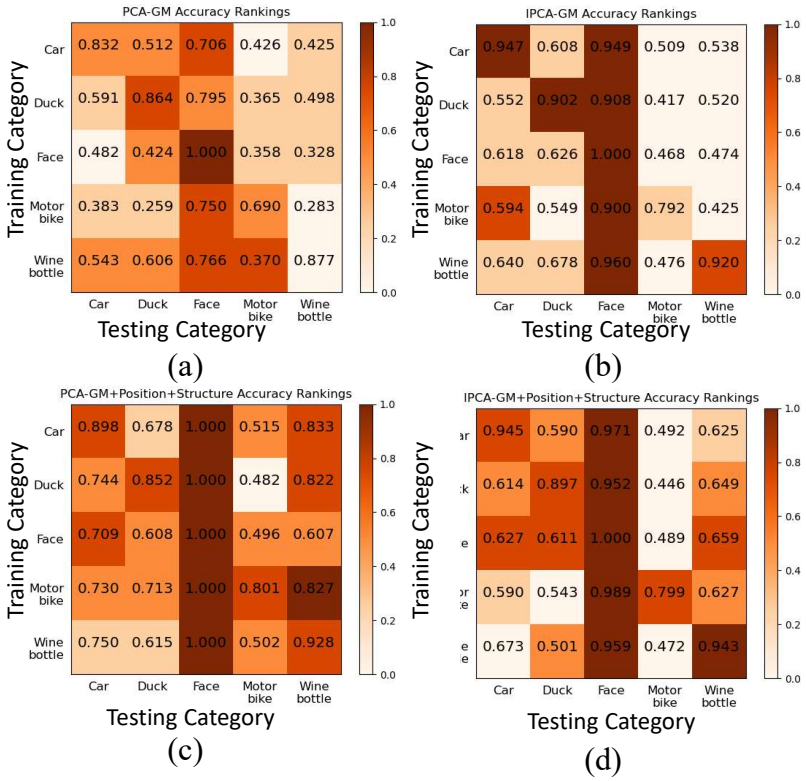


Figure 8: Cross-category generalization study method ranking on Willow ObjectClass dataset. The re-color figure from Fig. 7 by sorting all the matching results obtained by four different methods from large to small.

The confusion matrix for the dataset illustrating transfer learning among the five object categories is given in Fig. 7. The models are trained on the object categories in the rows and then tested on the object categories in the columns. The diagonal of the matrix shows the classification ability of the method on each object class. The darker the grid color in the figure, the higher the matching accuracy. We compare each element in Fig. 7 (a) with Fig. 7(c), and Fig. 7(b) with Fig. 7(d). The proposed graph matching method combined with position and structural arrangement information has improved the generalization ability for all classes compared with the original

methods. Furthermore, the accuracy of the model in the face category reaches 100% in Fig. 7(c), and is also high in both Fig. 7(b) and Fig. 7(d). It may be related to the simplicity and lower noise in the face samples.

In Fig. 8, we re-color Fig. 7 by sorting all the matching results obtained by the four different methods from large to small. The larger the value, the darker the color. This is also a reflection of the generalization ability of the model. Moreover, it gives the priority to the proposed position and structure framework based on higher accuracy scores in the confusion matrix. The generalization ability of the models provided in this study is substantially greater than the alternative methods due to the addition of position information and the correlation connection of the subgraph structure. Furthermore, IPCA-GM+Position+Structure has greater diagonal accuracy, indicating that it is more suitable for the training category.

## 6. Limitations

This work aims to address the issue of ambiguous matching that results from the absence of discriminability based on local structure and the semantic similarity of different nodes. We therefore focus on the feature extraction problem during the graph-matching task. However, we do not consider issues arising from the fact that the image itself may contain noise and other abnormalities. Our position-aware node embedding module and subgraph-based structural embedding module are adaptive plug-ins that can boost the performance of other methods too. However, this is at the expense of additional computing time consumption and memory usage.

Future work will aim to investigate the noisy image-matching task so that it can be made more practically applicable. In addition, and from a problem-solving perspective, we can consider how to deal with matching graphs that have more complicated structural relations. These include hypergraphs and directed graphs or using various relative position and subgraph relational strategies.

## 7. Conclusion

In this paper, we propose a novel graph matching method, which combines and incorporates a relative position-aware node embedding and together with a subgraph-based structure embedding into the node-wise embedding between graphs. The experiments include comparisons with alternatives, an

ablation study and the cross-category generalization study to demonstrate the effectiveness of the proposed method with its embedding modules. Specifically, the matching accuracy on several real-world datasets compared with peer methods demonstrate the state-of-the-art performance of our method.

## References


[1] J. Hou, M. Pelillo, H. Yuan, Hypergraph matching via game-theoretic hypergraph clustering, Pattern Recognition 125 (2022) 108526.

[2] Y. Cheng, X. Zhu, J. Qian, F. Wen, P. Liu, Cross-modal graph matching network for image-text retrieval, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 18 (4) (2022) 1–23.

[3] B. Jiang, H. Zhao, J. Tang, B. Luo, A sparse nonnegative matrix factorization technique for graph matching problems, Pattern Recognition 47 (2) (2014) 736–747.

[4] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, A. J. Smola, Learning graph matching, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (6) (2009) 1048–1058.

[5] R. Wang, J. Yan, X. Yang, Graduated assignment for joint multi-graph matching and clustering with application to unsupervised graph matching network learning, Advances in Neural Information Processing Systems 33 (2020) 19908–19919.

[6] H. Zhang, Z. Huang, X. Lin, Z. Lin, W. Zhang, Y. Zhang, Efficient and high-quality seeded graph matching: Employing higher-order structural information, ACM Transactions on Knowledge Discovery from Data (TKDD) 15 (3) (2021) 1–31.

[7] K. Henderson, B. Gallagher, L. Li, L. Akoglu, T. Eliassi-Rad, H. Tong, It's who you know: graph mining using recursive structural features, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2011, pp. 663–671.

[8] R. Wang, J. Yan, 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) 1–1.

[9] S. Berretti, A. Del Bimbo, E. Vicario, Efficient matching and indexing of graph models in content-based retrieval, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (10) (2001) 1089–1105.

[10] A. Zanfir, C. Sminchisescu, Deep learning of graph matching, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2684–2693.

[11] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge, International Journal of Computer Vision 88 (2) (2010) 303–338.

[12] W. Burger, M. J. Burge, Scale-invariant feature transform, in: Digital Image Processing, Springer, 2022, pp. 709–763.

[13] F. Cen, X. Zhao, W. Li, G. Wang, Deep feature augmentation for occluded image classification, Pattern Recognition 111 (2021) 107737.

[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.

[15] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[16] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: International Conference on Learning Representations, 2014.

[17] R. Wang, J. Yan, X. Yang, Learning combinatorial embedding networks for deep graph matching, in: IEEE International Conference on Computer Vision, 2019, pp. 3056–3065.

[18] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, Advances in Neural Information Processing Systems 29 (2016) 3844–3852.

[19] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations, 2017.

[20] Y. Yang, Y. Qi, Image super-resolution via channel attention and spatial graph convolutional network, Pattern Recognition 112 (2021) 107798.

[21] M. Zhang, Y. Chen, Link prediction based on graph neural networks, Advances in Neural Information Processing Systems 31 (2018) 5165–5175.

[22] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations, 2017.

[23] V. Arvind, F. Fuhlbrück, J. Köbler, O. Verbitsky, On weisfeiler-leman invariance: Subgraph counts and related graph properties, Journal of Computer and System Sciences 113 (2020) 42–59.

[24] F. Zhou, F. De la Torre, Factorized graph matching, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 127–134.

[25] P. A. Knight, The sinkhorn–knopp algorithm: convergence and applications, SIAM Journal on Matrix Analysis and Applications 30 (1) (2008) 261–275.

[26] V. Traneva, S. Tranev, V. Atanassova, An intuitionistic fuzzy approach to the hungarian algorithm, in: International Conference on Numerical Methods and Applications, Springer, 2018, pp. 167–175.

[27] G. Patrini, R. van den Berg, P. Forre, M. Carioni, S. Bhargav, M. Welling, T. Genewein, F. Nielsen, Sinkhorn autoencoders, in: Uncertainty in Artificial Intelligence, 2020, pp. 733–743.

[28] Y. Quan, Y. Chen, Y. Shao, H. Teng, Y. Xu, H. Ji, Image denoising using complex-valued deep cnn, Pattern Recognition 111 (2021) 107639.

[29] W. Fang, F. Zhang, V. S. Sheng, Y. Ding, A method for improving cnn-based image recognition using dcgan, Computers, Materials and Continua 57 (1) (2018) 167–178.

[30] I. Sergienko, V. Shylo, S. Chupov, P. Shylo, Solving the quadratic assignment problem, Cybernetics and Systems Analysis 56 (1) (2020) 53–57.

[31] J. You, R. Ying, J. Leskovec, Position-aware graph neural networks, in: International Conference on Machine Learning, 2019, pp. 7134–7143.

[32] I. Papakis, A. Sarkar, A. Karpatne, A graph convolutional neural network based approach for traffic monitoring using augmented detections with optical flow, in: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), IEEE, 2021, pp. 2980–2986.

[33] M. Cho, K. Alahari, J. Ponce, Learning graphs to match, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 25–32.

[34] C. Wah, S. Branson, P. Welinder, P. Perona, S. Belongie, The Caltech-UCSD Birds-200-2011 Dataset, Tech. Rep. CNS-TR-2011-001, California Institute of Technology (2011).

[35] Y. Jin, D. Mishkin, A. Mishchuk, J. Matas, P. Fua, K. M. Yi, E. Trulls, Image matching across wide baselines: From paper to practice, International Journal of Computer Vision (2021) 517–547.

[36] R. Wang, J. Yan, X. Yang, Combinatorial learning of robust deep graph matching: an embedding based approach, IEEE Transactions on Pattern Analysis and Machine Intelligence (2020) 1–1.

[37] T. Yu, R. Wang, J. Yan, B. Li, Learning deep graph matching with channel-independent embedding and hungarian attention, in: International Conference on Learning Representations, 2019.

[38] I. K. M. Jais, A. R. Ismail, S. Q. Nisa, Adam optimization algorithm for wide and deep neural network, Knowledge Engineering and Data Science 2 (1) (2019) 41–46.

[39] Y. Liu, G. Yin, The delaunay triangulation learner and its ensembles, Computational Statistics and Data Analysis 152 (2020) 107030.

[40] S. Marcel, Y. Rodriguez, Torchvision the machine-vision package of torch (2010) 1485–1488.

# Position-aware and Structure Embedding Networks for Deep Graph Matching

Dongdong Chen[a], Yuxing Dai[b], Lichi Zhang*[a], Zhihong Zhang[b], Edwin R. Hancock[c]

[a]*Shanghai Jiao Tong University, Shanghai, 20030, Shanghai, China*
[b]*Xiamen University, Xiamen, 361005, Fujian, China*
[c]*University of York, York, YO105DD, York, UK*

**Abstract**

Graph matching refers to the process of establishing node correspondences based on edge-to-edge constraints between graph nodes. This can be formulated as a combinatorial optimization problem under node permutation and pairwise consistency constraints. The main challenge of graph matching is to effectively find the correct match while reducing the ambiguities produced by similar nodes and edges. In this paper, we present a novel end-to-end neural framework that converts graph matching to a linear assignment problem in a high-dimensional space. This is combined with relative position information at the node level, and high-order structural arrangement information at the subgraph level. By capturing the relative position attributes of nodes between different graphs and the subgraph structural arrangement attributes, we can improve the performance of graph matching tasks, and establish reliable node-to-node correspondences. Our method can be generalized to any graph embedding setting, which can be used as components to deal with various graph matching problems answered with deep learning methods. We validate our method on several real-world tasks, by providing ablation studies to evaluate the generalization capability across different categories. We also compare state-of-the-art alternatives to demonstrate performance.

*Keywords:*
Graph Matching, Graph Embedding, Deep Neural Network

*Corresponding author: lichizhang@sjtu.edu.cn (Lichi Zhang)

## 1. Introduction

Graph matching aims to establish the correspondence between two or more graphs based on their structural information [1]. It is widely used in combinatorial optimization, machine learning and computer vision due to its natural representation and convenient coding of the relationship between abstract data. Specifically, many applications can be implemented using the graph matching technique, such as image registration in medical image analysis, linking user accounts in social network analysis, image extrapolation in computer vision, finding coherent motions and semantic regions in crowd scenes [2].

There are many methods and matching solvers developed for addressing the graph matching problem [3, 4]. These can be divided into different categories depending on the specific perspectives adopted. For example, based on the number of graphs included, there are graph-to-graph matching and multi-graph matching methods [5]. Since graph-to-graph matching is the basis of multi-graph matching, improving the matching ability between two graphs can also be extended to improve multi-graph matching. Therefore, this paper focuses on the matching of pairs of graphs. Based on matching content, they can be divided into a) structure information-based graph matching and b) semantic information-based graph matching [6]. Note that in graph matching, especially for image-generated graphs, semantic features based on structural information have been proved to be important for classifying nodes [7]. In this way, the graph matching problems studied in this paper are based on both node feature information and structural information from the graphs. On the other hand, graph matching methods can also be divided into a) learning-based methods and b) learning-free methods [8]. Notably, learning-free methods generally seek approximate solutions for a given fixed affinity model, which is usually cast in the form of simple parameters and is approximately solved by the joint similarity of individual entities together with their mutual relationships [9]. However, their time complexity is often too high to satisfy and solve large-scale real-world problems.

In the case of deep learning-based graph matching solvers, the common feature of most methods is that they compromise in their evidence-combining strategy in the sense that the resulting unclear combinatorial element would not be competitive in a purely combinatorial setup [10]. However, these methods can only calculate the similarity score of the entire graph, and heavily rely on inefficient global matching procedures. Additionally, they only con-

sider the embedding of local information for nodes in the graph, which tend to ambiguously match similar nodes in different regions of the graph in an inconsistent manner [10]. Fig. 1 illustrates an example of the main idea of our relative position-aware information embedding strategy. Here it is shown that it fails to match the cats' ear positions correctly, since the node embedding information usually relies only on node semantic information, and the local structure information lacks effective discrimination ability. Therefore it is difficult to unambiguously distinguish the left and right ears of a cat. In this example, the relative position information is critical to graph matching, especially when both the semantic and structural information are very similar in the two regions of the graphs.

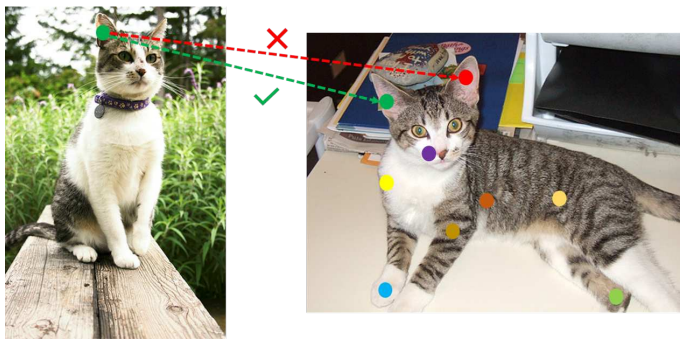

Figure 1: An exemplar failure case to establish the correct match of cat ears between two images, which are selected from the Pascal VOC dataset [11].

To address the above-mentioned challenges, here we propose a novel approach to solve the graph matching problem. We commence by utilizing node features extracted from the images to attribute the nodes, and design a position-aware node embedding algorithm to capture the relative position information of nodes in the graph. We further enhance the representation of local information of the nodes by merging a structural arrangement information representation of the subgraph in which the node is located. With the graph node-wise embedding to hand, we aim to obtain the required graph node permutations for node-to-node correspondence from raw pixel inputs. To our best knowledge, there are no methods that consider embedding both position and structural information for the graph matching problem.

The main contributions of our works are as follows:

1. Relative position information has not been considered in the existing

process of graph matching, which hampers their applications in terms of reducing the ambiguity of matching accuracy. We propose to construct an anchor set for small graphs extracted from images, and fully consider the relative position information related to the matching of keypoints, which is also proved to be effective in experiments.

2. We propose a novel subgraph detection and graph representation method, and extract subgraph structural arrangement information for improved node embedding. Previous graph matching studies only match using node level and edge level information, but ignore the higher-order neighborhood structure information. Here we introduce subgraph arrangement structure information into the graph matching task, by capturing it using the subgraph structure. This arrangement procedure combines node and edge level information with relative position information. We can therefore implement a hierarchical and comprehensive extraction of image features, which further improves the matching accuracy.

3. We demonstrate in our experiments that our method outperforms alternative work on several widely-studied real-world datasets. In the ablation study, we also show the effectiveness of the introduced components of our new method of graph matching on these datasets. We show that our position-aware node embedding module and subgraph-based structural embedding module can be added to existing deep learning graph matching methods, to further boost their classification performance.

The remainder of the paper is organized as follows: Section 2 provides a brief review of the related work on image feature extraction, graph embedding, and combinatorial optimization. Section 3 defines the specific graph matching problem that we intend to resolve in this work. Section 4 provides the details of our method, and the corresponding experiments to validate our new method are presented in Section 5. Finally, we conclude the paper and provide an extended discussion of potential future work in Section 7.

## 2. Related Work

In this section, we review existing deep learning solvers in dealing with graph matching problems.

### *2.1. Image Feature Extraction*

Feature extraction refers to the computation of higher-level features from the original pixels in an image. It can capture the differences between pixels

of various object classes. Conventional feature extraction usually proceeds in an unsupervised manner, and no image category labels are used when extracting information from the pixels. Commonly used traditional features include GIST (Generalized Search Trees), SIFT (Scale-Invariant Feature Transform), and LBP (Local Binary Patterns) [12].

However, these hand-crafted feature extraction methods can not be optimized according to the training from images and their labels. However, they fail to provide more comprehensive feature information to better represent important image information. The method based on deep learning overcomes this shortcoming using a soft-coded feature extractor [13]. Recently, the CNN model has been developed and enhanced in detail. For example, Szegedy *et al.* [14] significantly increase the depth of the CNN in their GoogleNet with three different types of convolution operation. He *et al.* [15] have proposed the residual neural network (ResNet) which includes a cross-layer connection that passes information from the input across deeper layers. It also adds to the result of convolution by introducing shortcut connections to solve the problem of vanishing loss gradient.

In this paper, we use VGG-16 [16] as the backbone architecture for image feature extraction, and which also closely follows the previous graph-matching works described in [5, 8, 10, 17]. This guarantees equity in the comparisons performed in our experiments. It can adapt to different sizes of the convolutional kernel to capture more discriminative decision functions with fewer parameters. It has been proved effective in deep visual feature extraction.

### *2.2. Graph Embedding*

At present, most studies related to graph embedding focus on static graphs, which can be divided into (a) spectral-based and (b) spatially-based methods.

#### *2.2.1. Spectral Methods for Graph Embedding*

The earliest graph embedding method provides local convolutions of graph structure data in the spectral frequency domain. The Fourier transform of the graph is used to transform the time domain data into a frequency domain signal. Then the frequency domain signal is convolved with local features. Finally, the frequency domain signal is transformed back into the time domain.

Although spectral frequency-based methods can effectively avoid the cross-influence of data in the time domain, it needs to calculate the Laplace matrix eigendecomposition and a representation of the graph structure corresponding to the sample of the graph structure. This makes it difficult to apply in real scenes due to the large computational overheads. To solve the problem of high computational complexity, Defferrard *et al.* [18] proposed to use Chebyshev polynomials instead of the exact calculation of graph eigenvalues and eigenvectors. In addition, Kifp and Welling [19] simplified the algorithm with first-order polynomials and used it to solve the task of semi-supervised node classification.

In general, spectral methods have suffered from the following three defects: a) The generalization ability to new unseen and variable graph structures is poor. If any node or edge in the graph changes, the spectral domain of the graph will also change rapidly, so that the filter learned from the original graph can not be well matched to the new graph. b) The alignment of Fourier patterns is difficult. It is challenging to align graphs with different typologies in the spectral domain, especially when the difference between graphs is large. Such differences in the spectral domain will be magnified, making it harder to align the Fourier modes. c) Spectral methods can only solve the problem for undirected graphs. This means it is limited to satisfying the constraint of a symmetric transformation matrix.

*2.2.2. Spatial Methods for Graph Embedding*

Spatial approaches have an advantage over spectral approaches in that they can operate on problems where the graph structure varies significantly in the dataset being studied [20]. However, they generally require sophisticated data transformations to enable effective learning. For example, given the strong hypothesis that the existence of node-connected edges is based on sometimes ad-hoc and contrived or artificial settings, Zhang *et al.* [21] extracted local subgraphs around each target link. They use this to learn the functional mapping of subgraph patterns to infer the existence of links, and to automatically learn heuristic algorithms.

In addition, the graph neural network method based on a message passing architecture is also considered to be a spatial method. It uses different aggregation schemes to aggregate the characteristic messages of neighbors in a graph. The message passing neural network further integrates edge information in the aggregation process. The graph attention network proposed by Velikovic *et al.* can aggregate neighborhood information according to

the weights derived from a trainable attention mechanism [22]. However, all these methods focus on just learning node embeddings to capture the local network structure around a given node. Such models are at most as powerful as the Weisfeiler-Leman graph isomorphism test [23], but they cannot distinguish nodes in symmetric or isomorphic positions in the network.

In this paper, we develop a novel graph embedding method to extract high-level information combined with the relative position of nodes, which can greatly assist graph matching.

### *2.3. Combinatorial Optimization*

In combinatorial optimization problems, enumeration and exhaustive methods are not practical. The solution strategies are either discrete or are effectively reduced to a discrete set. Many traditional algorithms for solving combinatorial optimization problems involve the use of manually constructed heuristic algorithms. However, due to the difficulty of problem-solving, it is usually not the optimal solution.

Mathematically, the graph matching problem can be considered a combinatorial optimization problem, which is also an NP-Hard problem. Using mathematical methods to solve graph matching problems can be divided into relaxation optimization and matrix decomposition methods [24]. For the spectral relaxation optimization method, the computation speed is relatively fast, but the constraints on the obtained solution are often over-relaxed. Of course, the input discrete variables can also be converted into continuous variables to make the solution more compact. However, the following problems generally exist in these methods: First, they are not robust to outliers and noise. Second, they usually have high computational overheads. Besides, the optimal approximate solution obtained by traditional mathematical methods may be locally optimal solutions different from the ground truth.

With the rise of deep learning, the use of reinforcement learning and neural networks to solve combinatorial optimization problems has become a topical trend. The Sinkhorn network [25] can solve the linear assignment problem, by performing the doubly stochastic normalization operations of row normalization and column normalization on all non-negative matrices to minimize the cost. Furthermore, it is an approximate Hungarian algorithm [26] that can optimize polynomial complexity. Recently, Patrini *et al.* [27] proposed a Sinkhorn autoencoder to achieve the target of minimizing distance, and utilizing reinforcement learning to solve the combinatorial optimization problem. Furthermore, a novel and simple end-to-end training

framework was proposed by Rolínek *et al.* [10], which incorporates the most advanced graph matching combinatorial solver.

In this paper, we develop a novel subgraph detection and graph representation method to solve the combinatorial optimization problem. It can effectively reduce the dimensionality of the quadratic assignment problem for combinatorial optimization to a linear assignment problem.

## 3. Problem Definition and Notations

In this paper, we consider the graph $\mathcal{G} = (V, A, X)$ which is consist of a finite set of nodes $V$, an adjacency matrix $A$, and a set of attributes $X$ for the nodes, which originate from images using various CNN models [16, 28, 29]. The goal of graph matching is to establish a correspondence between two attributed graphs, which minimizes the sum of local and geometric costs of assignment between the vertices of the two graphs.

Let $\mathcal{G}_s = (V_s, A_s, X_s)$ be a source graph, with $|V_s| = n$, $A_s \in \{0,1\}^{n\times n}$, and feature matrix $X_s \in \mathbb{R}^{n\times F}$, where $F$ represents the $F$-dimensional feature vector of nodes in graph stacked to columns. The target graph to be matched is $\mathcal{G}_t = (V_t, A_t, X_t)$, with $|V_t| = m$, $A_t \in \{0,1\}^{m\times m}$, and feature matrix $X_t \in \mathbb{R}^{m\times F}$, whose w.l.o.g. $n \leq m$. We also construct a vector of length $nm$. The element $\mathbf{v} \in \{0,1\}^{nm\times 1}$ indicates the match of vertices in two graphs, where $\mathbf{v}_{i,j} = 1$ if vertex $i \in V_s$ is matched to vertex $j \in V_t$ and $\mathbf{v}_{i,j} = 0$ if otherwise. It is worth noting that all the vertex matches are subject to the one-to-one mapping constraints, i.e. $\sum_{j\in V_t} \mathbf{v}_{i,j} = 1\ \forall i \in V_s$ and $\sum_{i\in V_s} \mathbf{v}_{i,j} \leq 1\ \forall j \in V_t$.

Furthermore, we construct a square symmetric positive matrix $M \in \mathbb{R}^{nm\times nm}$ referred to as the affinity matrix, to encode the edge-to-edge affinity between two graphs by their off-diagonal elements. Specifically, $M_{ip;jq}$ measures how well edge $(i,j)$ in graph $\mathcal{G}_s$ matches with edge $(p,q)$ in graph $\mathcal{G}_t$ with $\{i! = p\} \cup \{j! = q\}$. The diagonal entries of the affinity matrix can also indicate the node-to-node affinity between two graphs. Therefore, the pairwise graph matching between $\mathcal{G}_s$ and $\mathcal{G}_t$ can be formulated as the edge-preserving, quadratic assignment programming (QAP) [30] problem:

$$\begin{aligned} &\underset{\mathbf{v}}{\text{argmax}}\ \mathbf{v}^\top M\mathbf{v} \\ &\text{s.t.}\ \sum_{j\in V_t} \mathbf{v}_{i,j} = 1\ \forall i \in V_s, \sum_{i\in V_s} \mathbf{v}_{i,j} \leq 1\ \forall j \in V_t, \end{aligned} \tag{1}$$

Table 1: Notations and Description

| Symbol | Definition |
|---|---|
| $\mathcal{G}$ | original graph |
| $V$ | nodes-set of graph |
| $\|V\|$ | node number of nodes-set |
| $r$ | maximum shortest path distance |
| $K$ | network layer |
| $P$ | anchor-set |
| $p$ | anchor number |
| $q_{v,u}$ | relative position coefficient between pair of nodes $v$ and $u$ |
| $d_{v,u}$ | shortest path distance between pair of nodes $v$ and $u$ |
| $I^{(k)}(v,u)$ | information aggregation function between node $v$ and $u$ in $k$-th layer |
| $W^{(k)}$ | trainable weight matrix in $k$-th layer |
| ${h_v}^{(k)}$ | hidden representation of node $v$ in $k$-th layer |
| $N^t(v)$ | $t$-hop neighbor of node $v$ |
| $P_v(v)$ | path from node $v$ to $u$ |
| $\delta$ | leaf node threshold |
| $\alpha$ | hyper-parameter to balance the position information |
| $\beta$ | hyper-parameter to balance the structure information |
| $T$ | transform matrix of the original graph to the subgraph relation graph |
| $H_s$ | final hidden representation of source graph |
| $H_t$ | final hidden representation of target graph |

where $\mathbf{v} \in \{0,1\}^{nm\times 1}$ and $M \in \mathbb{R}^{nm\times nm}$. In this work, we intend to resolve the graph matching problem based on the supervised matching of graphs, and we aim to learn an end-to-end model which can effectively extract graph information and match through given pair-wise ground-truth correspondences for a set of graphs, and generalize to unseen graph pairs. The notation used in this paper is summarized in Table 1.

## 4. Methods

In this section, we provide details of the proposed end-to-end position-aware and structure-based graph matching method, The overall pipeline is

shown in Figure 2. In the figure, the blue source graph $\mathcal{G}_s$ are extracted together with their node-wise high-level graph feature representations. This is done using position-aware node embedding and subgraph based structure embedding. The graph features are matched using a matrix product operation and the Sinkhorn activation function. The result is compared with the available ground truth. The final output of the model is the matching accuracy of the pair of nodes between the source graph and the target graph. Note that the deep feature extractor of the proposed method can be any CNN model, such as VGG-16 [16], which is adapted to convert images into graphs with features. These graphs are the source and target graphs, which are the paired input of our model.

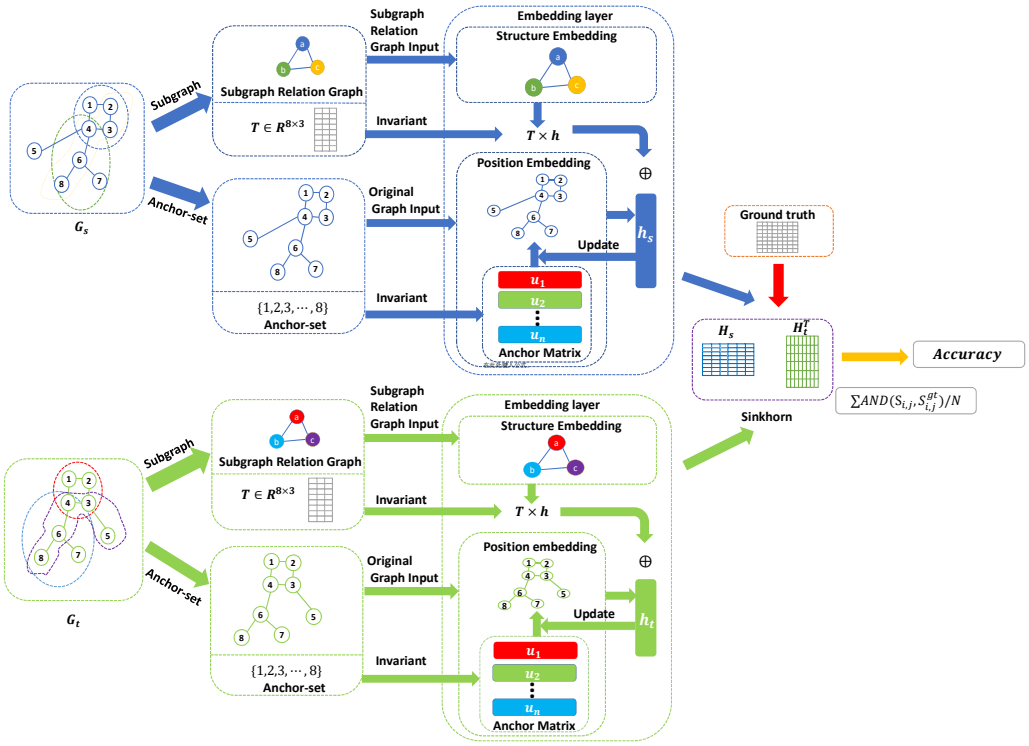


Figure 2: Overview of the end-to-end position and structure embedding networks for deep graph matching.

The model consists of three main components, namely a) a position-aware node embedding module, b) a subgraph-based structure embedding module,

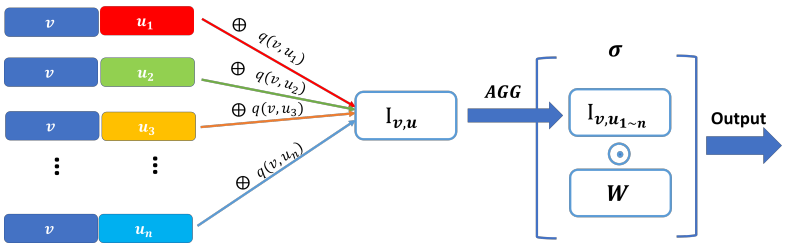


Figure 3: Procedure of Position Embedding.

and c) a graph feature matching module. The aim of the position-aware node embedding step, as described in Section 4.1, is to consider the information available from the relative structure. The second step is to iteratively combine the subgraph structure information described in Section 4.2, and the final part is the matching of two graphs with extracted graph representation described in Section 4.3.

### *4.1. Position-aware Node Embedding*

We establish our position-aware node embedding procedure with two key components: First, before the node embedding process, we propose a simple but effective strategy to construct an anchor-set for each graph which is used throughout the entire learning process. Here, we refer to the nodes used as reference position coordinates as *anchors*, and the set of these nodes as the *Anchor-set*. We aggregate the information of the node and each anchor in the anchor-set, in this way we replace the original graph embedding method that relies on message passing between local network neighborhoods. Second, we design an information aggregation mechanism for message aggregation between nodes and anchors in the anchor-set. With the addition of position-aware node embedding, we can effectively eliminate the ambiguity-prone mismatching of similar nodes in different positions.

#### *4.1.1. Anchor-set Construction for Small Graph*

The position anchor framework is motivated by PGNN [31] which emphasizes large graphs (i.e. those consisting of at least thousands of nodes). However, in this paper, we further study the anchor mechanism and focus on small graphs constructed from images. Specifically, the anchor-sets first proposed for large graphs are completely randomly chosen subsets of all nodes from the graph. They are intended to sample multiple anchor-sets $P = \{P_{i,j}\}$, where the number of anchor-sets $|P| = \log^2 |V|$, $i = 1, 2, ..., \log |V|$,

$j = 1, 2, ...c \log |V|$. Here, $|V|$ is the number of nodes in the graph and c is a hyper-parameter. Each node is sampled independently and randomly with a probability of $1/2^i$. The idea central to our method is that the anchor-sets can include each node or any of its one-hop neighbors. Large anchor sets have low position information with a large sampling probability. Small anchor sets have high position information with low sampling probability. The balance of these different sizes of anchor-sets may lead to efficient embeddings. However, this random sampling operation can result in the final anchor sets sampling almost all nodes in the graph. The proof is as follows:

*The sampling probability for each node in the anchor-set $P_{i,j}$ is $1/2^i$, and the number of anchors in each anchor-set $P_{i,j}$ is*

$$p_i = \frac{|V|}{2^i} \quad \forall i \in [1, \log |V|], \tag{2}$$

*where $|V|$ is the number of nodes in the graph.*

*Since the number of anchor points $p_i$ in the anchor-set $P_{i,j}$ is only related to element i, the construction of anchor-sets $P = \{P_{i,j}|(i \in [1, \log |V|], j \in [1, c \log |V|])\}$ is actually the repeated construction of anchor-sets $P = \{P_{i,j}|(i \in [1, \log |V|], j = 1)\}$ for $c \log |V|$ multiplied by the uniform sampling probability for each anchor in each anchor set construction step. Hence, we consider each anchor in a non-repetitive operation, i.e. the anchors in anchor-sets $P = \{P_i|(i \in [1, \log |V|])\}$. Therefore, the number of anchors p in the entire set of anchor-sets $P = \{P_i|(i \in [1, \log |V|])\}$ is given by:*

$$p = \sum_{i=1}^{log|V|} p_i = |V| - 1. \tag{3}$$

*The anchor sets obtained through fair non-repetitive sampling operation contain $|V| - 1$ anchors for a $|V|$ node graph and the sampling probability for each anchor is $(|V| - 1)/|V|$, which tends to unity for large graphs.*

Based on the derivation above, we propose an effective strategy to construct an anchor set $P$ consisting of all nodes in the graph, denoted by $P = V$, and serving as a stable reference for all nodes. According to our central idea in this paper, when all nodes in a graph become an anchor in an anchor-set, each node aggregates information with anchors based on a relative position relation. This is equivalent to each target node being subject to a relative position-based attention mechanism based on the global arrangement of nodes in the graph.

*4.1.2. Aggregation of Node Embedding*

With the anchor-set selected, we continue to compute the position-aware node embedding through the information aggregation between the nodes, in which each anchor cooperates with its relative position attributes. Here, we first compute the relative position coefficient $q_{v,u}$ between a pair of nodes $v$ and $u$ as:

$$q_{v,u} = \frac{e^{-d_{v,u}}}{\sum_{i \in V} e^{-d_{v,i}}}, \tag{4}$$

where $d_{v,u}$ is the shortest path distance between the two nodes. Considering the time complexity of calculating the shortest path, we stipulate that the maximum shortest path distance is not more than $r$, otherwise, the value is infinite. In this way, we can effectively eliminate the interference of anchors that are too far away from the node when the coefficient $q_{v,u}$ equals zero. We define the information aggregation function $I(v,u)^{(k)}$ for $k$-th layer between nodes $v$ and $u$ as:

$$I(v,u)^{(k)} = q_{v,u}\text{CONCAT}(h_v^{(k-1)}, h_u^{(k-1)}), \tag{5}$$

where $q_{v,u}$ is the relative position coefficient between node $u$ and $v$, $h_v^{(k-1)}$ and $h_u^{(k-1)}$ are hidden representations containing the position information in $(k-1)$-th layer. The message aggregation function CONCAT concatenates the hidden representations $h_v^{(k-1)}$ and $h_u^{(k-1)}$.

We further aggregate the information of each pair of nodes and the anchor with a non-linear transformation applied after the aggregation to achieve higher expressive power. The hidden representation passed to the next layer is:

$$h_v^{(k)} = \sigma(\text{AGG}(I(v,u)^{(k)} | \ \forall u \in V)W^{(k)}), \tag{6}$$

where AGG is a permutation-invariant function (e.g., the sum), and $W^{(k)}$ is a learnable weight vector for the $k$-th layer.

The general position-aware node embedding framework is summarized in Algorithm 1.

*4.2. Subgraph-based Structure Embedding*

In addition to the relative structural information, we further introduce subgraph structure embedding during the feature extraction and the information aggregation for each node. The subgraph-based structure embedding

**Algorithm 1:** Position-aware node embedding framework.

**Input:** input original graph $\mathcal{G} = (V, A, X)$, anchor-set:$S = V$, relative position coefficient $q$, trainable weight in $k$-th layer $W^{(k)}$, network layer $k \in [1, K]$

**Output:** Position-aware node embedding $h_p$

$h_v = X_v$
**for** $k \in [1, K]$ **do**
  **for** $v \in V$ **do**
    $I(v, u)^{(k)} = q_{v,u}\text{CONCAT}(h_v^{(k-1)}, h_u^{(k-1)})$
    $h_v^{(k)} = \sigma(\text{AGG}(I(v, u)^{(k)})W^{(k)}),\ \forall u \in V$
  **end**
**end**
**return** $h_v^{(K)}$

can be split into two steps: a) We first divide the original graph into different subgraphs according to the pre-defined structure. Each subgraph can be regarded as a new supernode, and the edges between new nodes are determined by the relationships between the nodes in each subgraph, thus we can construct a so-called **subgraph relation graph**. b) In the second step, we aggregate the structural arrangement information of the subgraphs in the subgraph relation graph and merge them with the relevant nodes in the original graph while concurrently aggregating the local information at the node level of the original graph.

*4.2.1. Subgraph Construction*

Here we retain the global structural arrangement information from the original graph to the greatest extent, and extract the key representative structural features for the subgraphs. Note that it is particularly important to select the nodes of the subgraph relation graph, i.e. pre-defined subgraphs in original graphs. In this paper, we define three different types of subgraph structures, namely a) paths, b) trees, and c) circuits. Combinations of these three basic subgraph types can be used to construct any required graph structure.

Since the search for all subgraphs in a graph is NP-hard, we propose a novel searching method to detect the $t$-hop neighbors of nodes to find subgraphs, and then save the search path in real-time to reduce the time

complexity. The corresponding detection Algorithm 2 is designed to handle how to find the three different types of subgraph structures.

For tree structure detection, we first initialize two sets: $N^t(v)$ to store the $t$-hop neighbor of node $v$, where $t \in [1, D]$ and $D$ is the maximum neighbor depth of the graph, and $P_v(u)$ stores the path from node $v$ to node $u$. We traverse the adjacency matrix to find all edges, and store the 1-hop neighbor set of each node. The tree-structural subgraph is selected for further processing if the number of leaf nodes exceeds the preset threshold $\delta$. For the path and circuit subgraphs, we continue the search for the $t$-hop neighbors of each node through the combination of several existing $t$-hop neighbors. If a path from node $v$ to node $u$ already exists in the set $P_v(u)$ and there is no repeating node in the path, we identify the path as a circuit subgraph. The searching progress for three different types of subgraph structure is shown in Fig. 4.

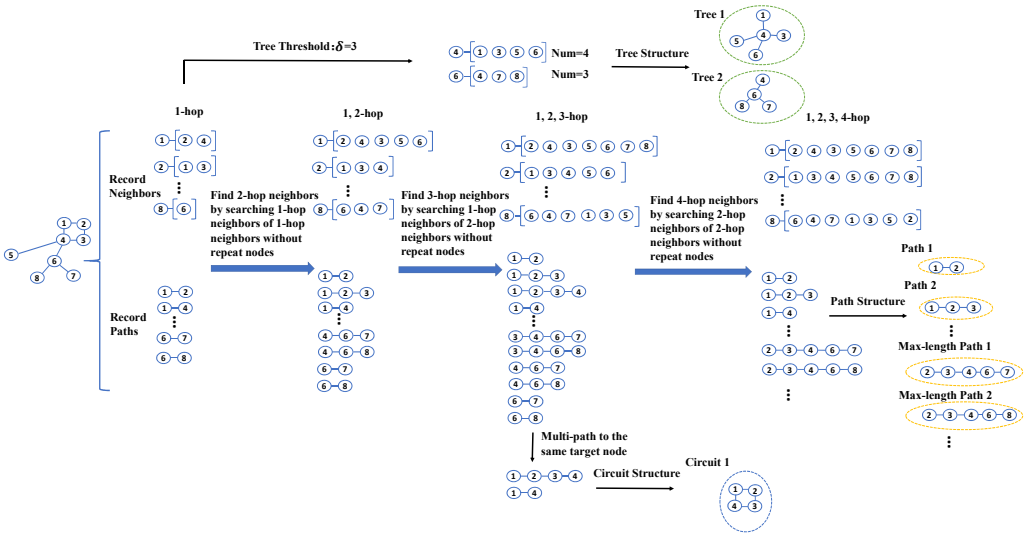


Figure 4: The searching progress for three different types of subgraph structure.

After obtaining these three different types of different subgraphs, we construct the so-called **subgraph relation graph** to capture the internal relationship between different subgraph types. This process is shown in Fig. 5:

**Definition 1:** ***Subgraph Relation Graph.*** Let an undirected graph $\hat{\mathcal{G}} = (\hat{V}, \hat{A}, T)$ constructed from the original graph $\mathcal{G}$, in which the node

**Algorithm 2:** The searching algorithm for the three different types of subgraph structure.

**Input:** input original graph $\mathcal{G} = (V, A, X)$, max neighbor depth $D$, min node leaf $\delta$

**Output:** output subgraph set $S$

initialize subgraph set $S = \{S_i\}$, neighbor set $\{N^t(v)\}$, path set $\{P_v(u)\}$, subgraph relation graph $\hat{\mathcal{G}}$ with $|\hat{V}|$ vertices
**for** $t \in [1, D]$ **do**
  **for** $v \in V$ **do**
    **if** $t == 1$ **then**
      **for** $u \in V$ **do**
        Add $u, v$ into $N^1(v), N^1(u)$
        Add $v, u$ into $P_v(u), P_u(v)$
        **if** $|N^1(v)| \geq \delta$ **then**
          add $(N^1(v) + [v])$ as Tree Subgraph into $S$
        **end**
      **end**
    **else**
      **for** $u \in N^i(v) | 2^i < t \leq 2^{i+1}$ **do**
        **for** $r \in N^{t-2^i}$ **do**
          **if** $P_v(r) \in \{P_v(u)\} and\ r \notin P_v(u)$ **then**
            Add $P_v(u)$ as Circuit Subgraph into $S$
          **else**
            Add $r, v$ into $N^t(v), P_v(r)$
            Add $\{P_v(u), P_u(r)\}$ to $P_v(r)$
            Add $P_v(r)$ to $P_r(v)$
          **end**
        **end**
      **end**
    **end**
  **end**
**end**
Add $P_v(u)$ as Path Subgraph into $S$
**return** $S$

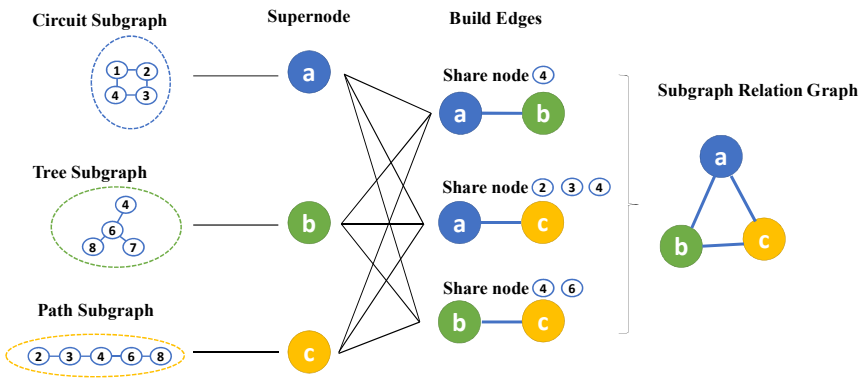


Figure 5: An example of the *Subgraph Relation Graph*.

set $\hat{V}$ contains $|\hat{V}|$ nodes, each denoting a different subgraph identified in the original graph. The adjacency matrix element of $\hat{A}$ equals one if two nodes share the same node in the original graph. The transformation matrix $T \in \mathbb{R}^{|V|*|\hat{V}|}$ represents the matrix that transforms the original graph into the subgraph relation graph.

Let node $v_i$ belong to node-set $V$ of the source graph, and node $\hat{v_j}$ belong to node-set $\hat{V}$ in the target graph corresponding to subgraph $S_j$ in the source graph. In this way, the transformation matrix $T$ has the element:

$$T_{ij} = \begin{cases} 1 & v_i \in S_j, \\ 0 & \text{else .} \end{cases} \tag{7}$$

Algorithm 3 gives an overview of the subgraph relation graph construction procedure.

*4.2.2. Merged Structure Embedding*

We augment the graph affinity embedding to better fuse the high-order information from the nodes by developing a merged structure message-passing scheme. We commence by aggregating the structural arrangement information of the subgraph relation graph at the subgraph level with the classical embedding procedure:

$$h_{\hat{v_i}}^{(k)} = \text{COMBINE}(h_{\hat{v_i}}^{(k-1)}, \text{AGG}\{h_{\hat{v_j}}^{(k-1)} | \hat{v_j} \in N(\hat{v_i})\}), \tag{8}$$

**Algorithm 3:** Construction of the subgraph relation graph.

**Input:** input subgraph set $S$

**Output:** output subgraph relation graph $\hat{\mathcal{G}} = (\hat{V}, \hat{A}, T)$

Create a new subgraph set $\hat{S}$ by combining one of the longest path subgraphs, one of the tree subgraphs, and the circuit subgraphs from subgraph set $S$.

**for** *all* $S_i$ *in subgraph set* $\hat{S}$ **do**
    Assign $S_i$ to each node $\hat{v_i}$
    **if** $\exists v \in S_i \cap S_j$ **then**
        Add edge $\hat{A_{ij}}$ between node $\hat{v_i}$ and $\hat{v_j}$
    **end**
    **if** $v_i \in S_j$ **then**
        $T_{ij} = 1$
    **else**
        $T_{ij} = 0$
    **end**
**end**

**return** $\hat{\mathcal{G}}$

where $N(\hat{v_i})$ denotes the set of neighbors of node $\hat{v_i}$ (i.e. which corresponds to subgraph $S_i$ in the original graph) in the subgraph relation graph. Here, the functions AGG and COMBINE are sometimes folded into a single aggregation update. Specifically, AGG is a function that aggregates all neighbor features of the node $S_i$, and COMBINE updates the representation of node $S_i$ according to the output of AGG. We perform information aggregation and combination at the structural level. We then further merge them with the position-aware node embedding extracted from the original graph using the following procedure:

$$h_v^{(k)} = \mathrm{COMBINE}_{\mathrm{M}}(h_v^{(k)}, \mathrm{AGG}_{\mathrm{M}}(h_{\hat{v_i}}^{(k)} | \forall v \in S_i)), \tag{9}$$

where $h_{\hat{v_i}}^{(k)}$ denotes the hidden representation of subgraph node $S_i$ extracted using Eq. 8, $h_v^{(k)}$ represents the hidden node embedding combined with structure and position in the $k$-th layer. The function is the sum over all subgraph embeddings that contain the node $v$ and is computed as $AGG_M(h_S) = T * h_S$. We define as $\mathrm{COMBINE}_{\mathrm{M}}(h_v, h_s) = \alpha h_s + \beta h_v$ where $\alpha$ and $\beta$ are two hyper-

parameters that can be used to balance the weight of position relationships and structural arrangement information.

The position-aware node embedding algorithm is summarized in Algorithm 4.

**Algorithm 4:** Merge structure embedding framework.

**Input:** input original graph $\mathcal{G} = (V, A, X)$, subgraph relation graph $\hat{\mathcal{G}} = (\hat{V}, \hat{A}, T)$, position-aware node embedding $h_p$, trainable weight $W$, network layer $k \in [1, K]$

**Output:** Subgraph-based Structure embedding $h_s$

$h_v = X_v$
**for** $k \in [1, K]$ **do**
  **for** $v \in V$ **do**
    $h_{\hat{v}_i}^{(k)} = \mathrm{COMBINE}(h_{\hat{v}_i}^{(k-1)}, \mathrm{AGG}\{h_{\hat{v}_j}^{(k-1)} | \hat{v}_j \in N(\hat{v}_i)\}))$
    $h_v^{(k)} = \mathrm{COMBINE}_{\mathrm{M}}(h_v^{(k)}, \mathrm{AGG}_{\mathrm{M}}(h_{\hat{v}_i}^{(k)} | \forall S_i \supset v))$
  **end**
**end**
**return** $h_v^{(K)}$

### *4.3. Graph Feature Matching*

With the proposed position-aware node embedding framework and subgraph-based structural embedding framework to hand, we encode each node with high-order graph structure information and position information and construct a high-level embedding space. Such an embedding scheme allows us to simplify the second-order affinity matrix to a linear one.

With the final hidden representation of the source graph $H_s \in \mathbb{R}^{n*F'}$ and the target graph $H_t \in \mathbb{R}^{m*F'}$ to hand, we can obtain a soft correspondence between these graphs using the corresponding affinity matrix $\mathrm{H_{affinity}} = \exp\left(\mathrm{H_s \Gamma H_t^T}\right)$. Each element in the affinity matrix $\mathrm{H_{affinity}} \in \mathbb{R}^{\mathrm{n*m}}$ represents the affinity between a pair of nodes in two graphs considering node features, position-aware information, and high-order subgraph information in the graph, with $\Gamma$ as a learnable weight matrix. We further apply the Sinkhorn normalization [32] to obtain a rectangular doubly-stochastic correspondence matrix to fulfill the constraints on injectivity from the original graph $\mathcal{G}_s$ to the target graph $\mathcal{G}_t$. Here, the Sinkhorn operation works

iteratively by iterative row-wise and column-wise normalization,

$$\begin{aligned} H^r_{\text{affinity}} &= H^c_{\text{affinity}} \oslash (H^c_{\text{affinity}} \mathbf{1_m} \mathbf{1_m}^T) \\ H^c_{\text{affinity}} &= H^r_{\text{affinity}} \oslash (\mathbf{1_n} \mathbf{1_n}^T H^r_{\text{affinity}}) \end{aligned} \tag{10}$$

where $\oslash$ indicates element-wise division, and $\mathbf{1_m} \in \mathbb{R}^{m*1}$, $\mathbf{1_n} \in \mathbb{R}^{n*1}$ are column vectors whose elements are all ones, $H^r_{\text{affinity}}$ and $H^c_{\text{affinity}}$ are the intermediate results of row normalization and column normalization respectively. We simplify these two iterative processes giving a continuous relaxation of the permutation matrix:

$$S = \text{Sinkhorn}(\text{H}_{\text{affinity}}) \tag{11}$$

We further employ the cross-entropy loss function as the permutation loss between the predicted permutation matrix and the ground truth:

$$L = -\sum_{i \in \mathcal{V}_s, j \in \mathcal{V}_t} \left( \mathbf{S}^{gt}_{i,j} \log \mathbf{S}_{i,j} + \left(1 - \mathbf{S}^{gt}_{i,j}\right) \log \left(1 - \mathbf{S}_{i,j}\right) \right), \tag{12}$$

where $\mathbf{S}^{gt}$ is the ground truth permutation matrix with each element as the ground truth node-to-node correspondence, and $S$ is the predicted permutation matrix given by Eq. 11. In this way, our cross-entropy loss can directly learn the permutation loss based on linear allocation cost in an end-to-end fashion, no matter how the number of nodes and edges in the graph change.

To illustrate in detail the complete framework adopted in our method, we provide an illustration of the processing steps involved in obtaining an embedding commencing from an image input through to the final output. Also shown is an illustration of the network structure used in each layer. In the example we take the batch size as 8, the number of key points as 12, and the number of motifs as 21. The complete data analysis workflow schematic is shown in Figure 6. The steps are as follows. Firstly, the resized image is input to a pre-trained VGG-16 model, and we compute the outputs of the $Relu4\_2$ and $Relu5\_1$ operations to extract node features and edge features. We align the keypoints with the guidance of the keypoint coordinates to obtain graph features. This also forms the input of the proposed position and structure embedding network. The operation of image feature extraction closely follows previous graph matching methods [5, 8, 10, 17] so as to guarantee equity in comparison. Then, for position-aware node embedding, the network concatenates the representation of each node with the anchor

representation multiplied by a weighting coefficient. A sum operator and one-layer non-linear transformation are performed to obtain the position embedding. For subgraph-based structure embedding, after constructing the motif graph, a single graph convolutional layer is applied to obtain the high dimensional structural embedding. The graph embedding can be computed using the combination of the position embedding and the corresponding structure embedding. We compute the affinity matrix using the paired graph embeddings combined with a learnable weight. The final accuracy of the method is computed by utilizing the available ground truth and a permutation matrix obtained from the affinity using the Sinkhorn activation function.

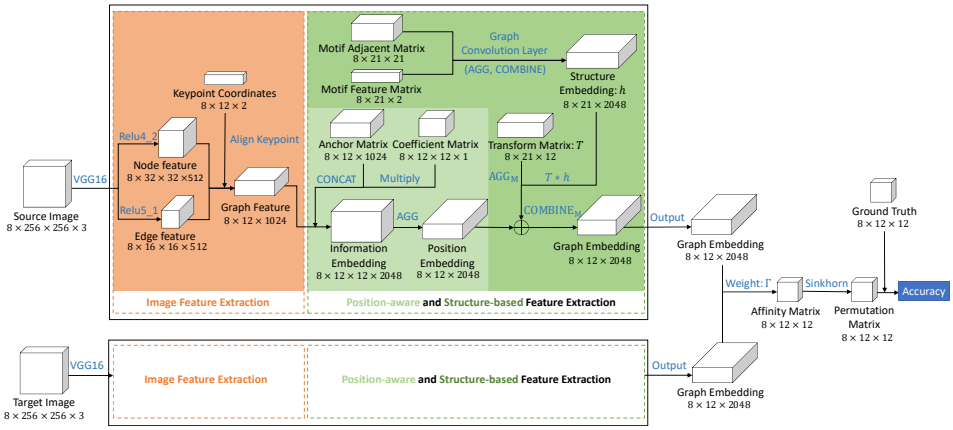


Figure 6: The entire data analysis workflow example with batch size as 8, the number of key points as 12, and the number of motifs as 21.

## 5. Experiments

In this section, we verify our method in three different tasks: We first perform our experiments in real-world tasks of supervised keypoint matching in four different kinds of natural images. Then, we further demonstrate the effectiveness of each procedure in our framework in an ablation study. Finally, we conduct the experiment on transfer learning across different categories, to evaluate the generalization capability of our methods.

### *5.1. Datasets*

**Willow ObjectClass.** The Willow Object Class dataset [33] contains five object categories including faces, cars, ducks, motorbikes, and wine bot-

tles. This dataset is a benchmark used to measure the ability of image classification and recognition. There are 9963 images in total, including 24640 labeled objects. In the experiment, we select a small subset for each category for training and use the remainder for testing. Each category has at least 40 images.

**Pascal VOC 2011.** The Pascal Visual Object Classes (VOC) 2011 dataset [11] contains 20 object sub-categories and four major categories in total, which are vehicles, indoors, animals, and people. Each image in the dataset has three annotations including category labels, attributes, locations, and bounding boxes. Note that most images in this dataset contain complicated component scenes.

**CUB-200-2011.** The Caltech-UCSD Birds-200-2011 (CUB-200-2011) dataset [34] contains 11788 images with 200 categories of birds. Each image has clear annotations containing 1 subcategory label, 15 part locations, 312 binary attributes, and 1 bounding box.

**IMC-PT-SparseGM.** The IMC-PT-SparseGM dataset [35] contains 16 categories and about 25061 images. This dataset gathers data from 16 tourist attractions around the world. Compared with the datasets mentioned above, the number of images in this dataset is the largest.

### *5.2. Alternative for Comparison*

The state-of-art benchmark methods used for comparison experiments are:

**GMN.** (Graph Matching Network, GMN) Zanfir *et al.* [10] developed the deep graph matching method employing an end-to-end deep learning framework. To extract image features, this framework adopts VGG-16 [16] networks, with first-order and second-order features extracted from the shallower layer (relu4_2) and deeper layer (relu5_1) of VGG-16, respectively. It also uses Delaunay triangulation to build graphs. Factorized Graph Matching (FGM) has been used to construct the affinity matrix efficiently. The method is based on supervised learning using a loss function on the offset loss. Although this method is relatively simple, it pioneered the use of deep learning to solve graph-matching problems.

**PIA-GM/PCA-GM.** Permutation loss and intra-graph affinity-based graph matching (PIA-GM) and permutation loss and cross-graph affinity-based graph matching (PCA-GM) [17] replace the offset loss on GMN with the permutation loss. This can improve the learning process. The image preprocessing of PIA-GM is the same as GMN. It employs 3 GNN layers to

perform intra-graph convolution. The feature dimension of the GNN layer is 2048. The main difference between PIA-GM and the original GMN architecture is that it introduces graph neural networks to learn the node embedding of the graph. The difference between PCA-GM and PIA-GM is that the former changes the method of graph convolution, from intra-graph convolution to cross-graph convolution. The aim is to achieve better extraction of information for similar nodes between two input graphs and better generation of the node embedding.

**IPCA-GM.** (Iterative Permutation loss and cross-graph affinity-based graph matching, IPCA-GM [36]). To better combine combinatorial optimization with a deep learning solution, IPCA-GM considers the iterative interaction between the Sinkhorn layer and the GCN layer to learn a better feature embedding. This approach fully considers the embedding information for nodes, and reduces the original two-dimensional QAP problem to a one-dimensional one, thus reducing the complexity and improving the learning efficiency.

**CIE-H.** (Channel-Independent Embedding and Hungarian attention, CIE-H [37]). This follows the CNN-GNN-metric-Sinkhorn pipeline proposed by PCA-GM and improves PCA-GM in two ways: First, the edge attribute information is taken into account and combined with node embedding. This is used to produce a channel-independent embedding. Second, a Hungarian Attention module dynamically constructs a structured and sparsely connected layer. This considers the most significant matching pair during training.

**NGM/NHGM.** (Neural Graph Matching Network (NGM) and Neural Hyper Graph Matching Network, NHGM [8]). These are recognized as a learnable solution to Lawler's quadratic assignment problem. This first transforms Lawler's QAP into an association graph and founds a solution, which is equivalent to the vertex classification problem on the association graph. Finally, a graph neural network is used to solve the vertex classification problem. NGM considers the task of matching graph pairs, and can be regarded as Lawler's QAP solver using VGG-16. On the other hand, NHGM is a hypergraph matching solver with VGG-16. Both of these methods are based on supervised learning using permutation loss.

**GANN-GM/GANN-MGM.** (GANN-GM [5]) is self-supervised learning graduated assignment neural network for matching graph pairs. (GANN-MGM [5]) is a self-supervised learning graduated assignment neural network for multi-graph matching. Both were proposed by Wang *et al.* [5]. GANN-GM introduces a self-supervised learning framework by leveraging graph

matching solvers to provide pseudo labels to train the neural network module in a deep graph matching pipeline. A general graph matching solver is proposed for various graph matching settings based on the classic Graduated Assignment (GA) algorithm.

### *5.3. Experimental Settings*

The experiments are conducted using two GeForce GTX 1080 Ti GPUs. We employ a batch size of 8 in training and evaluate our model in 2000 epochs for each iteration. In terms of experimental settings, we employ the Adam [38] optimizer to train our models with a learning rate of $1 \times 10^{-4}$. To overcome the over-smoothing problem common to graph neural networks, which arises from the transmission of information between nodes caused by deepening the network layer. This results in the representations of different nodes becoming similar and causes ambiguities in matching. We adopt a two-layer graph embedding and restrict the degree of smoothing in our experiments. When we perform experimental comparisons and ablation studies, we train and test each dataset independently. For the cross-category generalization study on the Willow ObjectClass dataset, we train the model on the Pascal VOC dataset excluding the car and motorbike classes, as these images overlap with the Willow dataset, and then test on the Willow ObjectClass dataset afterward. To guarantee equity, the splits between the training and testing datasets are identical for all of the methods compared [5, 8, 10, 17]. The details of the hyperparameter settings are summarized in Table 2.

Table 2: Summary of parameter settings for datasets under study.

| Datasets | Average Node Number | K | r | $\delta$ | $\alpha$ | $\beta$ |
|---|---|---|---|---|---|---|
| Pascal VOC | 9.07 | 2 | 1 | 2 | $1 \times 10^{-4}$ | $1 \times 10^{-5}$ |
| Willow ObjectClass | 10.00 | 2 | 1 | 2 | $1 \times 10^{-4}$ | $1 \times 10^{-4}$ |
| CUB 2011 | 12.00 | 2 | 1 | 2 | $1 \times 10^{-4}$ | $1 \times 10^{-5}$ |
| IMC-PT-SparseGM | 21.36 | 2 | 1 | 3 | $1 \times 10^{-4}$ | $1 \times 10^{-4}$ |

### *5.4. Performance Comparisons on Four Datasets*

In this experiment, the source graph $\mathcal{G}_s$ and target graph $\mathcal{G}_t$ are composed of keypoints extracted from two different images. Both two graphs maintain node consistency during the training and testing stages. We perform the same operations on four datasets to convert the images into graphs. First, we filter the outliers from the dataset. Next, we crop the images around their bounding boxes, which provide the pixel location of the object boundings in

images. Their image sizes are further resized to $256 \times 256$. The preprocessed images are then fed to VGG-16 to extract the node features. The graphs are constructed by Delaunay triangulation [39] of the extracted feature points.

We evaluate the graph matching capacity of the proposed method using the matching accuracy, which is defined by $accuracy = \sum \mathrm{AND}(S_{i,j}, S^{gt}_{i,j})/N$ from [5, 17, 36]. Here, the AND operation is a logical operation. Note that $S_{i,j}$ is the element of the predicted permutation matrix representing the correspondence matching of node $i$ and node $j$ from two different graphs. Similarly, $S^{gt}_{i,j}$ is the correspondence of the ground truth between the two nodes, and $N$ is the number of matching node pairs. We use ground truth node correspondences (which are set to one if two nodes are matched, and zero otherwise) for the image pairs, and perform an averaging operation to compute the final matching accuracy.

Table 3: Matching accuracy on the Willow ObjectClass database(%).

| Method | Car | Duck | Face | Motorbike | Winebottle | Mean accuracy |
|---|---|---|---|---|---|---|
| GMN [10] | 67.90 | 76.70 | 99.80 | 69.20 | 83.10 | 79.34 |
| NGM [8] | 84.20 | 77.60 | 99.40 | 76.80 | 88.30 | 85.30 |
| NHGM [8] | 86.50 | 72.20 | 99.90 | 79.30 | 89.40 | 85.50 |
| CIE-H [37] | 82.20 | 81.20 | **100.00** | **90.00** | **97.60** | 90.20 |
| PIA-GM [17] | 88.60 | 87.00 | **100.00** | 70.30 | 87.80 | 86.74 |
| PCA-GM [17] | 87.60 | 83.60 | **100.00** | 77.60 | 88.40 | 87.44 |
| IPCA-GM [36] | 90.40 | 88.60 | **100.00** | 83.00 | 88.30 | 90.06 |
| PCA-GM+Position+Structure | **92.90** | **90.80** | **100.00** | 79.00 | 94.40 | **91.42** |
| IPCA-GM+Position+Structure | 92.70 | 88.70 | **100.00** | 80.20 | 94.90 | 91.30 |

The experimental results on the Willow ObjectClass, Pascal VOC, CUB and IMC-PT-SparseGM datasets are shown in Table 3, Table 4, Table 5, and Table 6, respectively. We also compare the proposed methods(i.e. PCA-GM+Position+Structure and IPCA-GM+Position+Structure) with the alternative methods introduced in Section 5.2. Specifically, our methods achieve the best performance in the classification accuracy of three different categories on the Willow ObjectClass dataset, while the CIE-H method outperforms the Motorbike and Wine bottle classes. Note that the CIE-H method also achieves good results on the VOC and CUB datasets. However, it performs poorly on the IMC-PT-SparseGM dataset. The VOC dataset contains 20 object sub-categories and four major categories. It is therefore much more complicated than the remaining datasets under study. As observed from Table 4, many methods perform better than others in one or several specific categories. Our method performs slightly lower than the highest average ac-

Table 4: Matching accuracy on the Pascal VOC database(%).

| Method | Aero | Bike | Bird | Boat | Bottle | Bus | Car | Cat | Chair | Cow | Table | Dog | Horse | Motorbike | Person | Plant | Sheep | Sofa | Train | TV | Mean accuracy |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GMN [10] | 31.90 | 47.20 | 51.90 | 40.80 | 68.70 | 72.20 | 53.60 | 52.80 | 34.60 | 48.60 | **72.30** | 47.70 | 54.80 | 51.00 | 38.60 | 75.10 | 49.50 | 45.00 | 83.00 | 86.30 | 55.30 |
| NGM [8] | 50.10 | 63.50 | 57.90 | 53.40 | 79.80 | 77.10 | 73.60 | 68.20 | 41.10 | 66.40 | 40.80 | 60.30 | 61.90 | 63.50 | 45.60 | 77.10 | 69.30 | 65.50 | 79.20 | 88.20 | 64.13 |
| NHGM [8] | 52.40 | 62.20 | 58.30 | 55.70 | 78.70 | 77.70 | **74.40** | 70.70 | 42.00 | 64.60 | 53.80 | 61.00 | 61.90 | 60.80 | 46.80 | 79.10 | 66.80 | 55.10 | 80.90 | 88.70 | 64.58 |
| CIE-H [37] | 49.94 | 63.13 | **70.65** | 52.98 | **82.43** | 75.36 | 67.66 | 72.30 | 42.35 | **66.88** | 69.90 | **69.52** | **70.74** | 61.96 | 46.67 | 85.04 | 70.00 | 61.75 | 80.23 | **91.78** | 67.56 |
| PIA-GM [17] | 41.50 | 55.80 | 60.90 | 51.90 | 75.00 | 75.80 | 59.60 | 65.20 | 33.30 | 65.90 | 62.80 | 62.70 | 67.70 | 62.10 | 42.90 | 80.20 | 64.30 | 59.50 | 82.70 | 90.10 | 63.00 |
| PCA-GM [17] | 40.90 | 55.00 | 65.80 | 47.90 | 76.90 | **77.90** | 63.50 | 67.40 | 33.70 | 65.50 | 63.60 | 61.30 | 68.90 | 62.80 | 44.90 | 77.50 | 67.40 | 57.50 | 86.70 | 90.90 | 63.80 |
| IPCA-GM [36] | **53.78** | 66.22 | 67.14 | **61.20** | 80.39 | 75.27 | 72.55 | **72.52** | **44.55** | 65.24 | 54.30 | 67.24 | 67.90 | 64.21 | **47.93** | 84.35 | **70.79** | 63.98 | 83.80 | 90.83 | **67.70** |
| PCA-GM+Position+Structure | 49.34 | **68.51** | 63.32 | 56.37 | 81.34 | 75.81 | 65.8 | 70.93 | 42.90 | 66.15 | 70.46 | 64.21 | 66.78 | **65.05** | 42.31 | **85.06** | 67.60 | **67.25** | **86.87** | 90.06 | 67.31 |
| IPCA-GM+Position+Structure | 48.72 | 65.52 | 56.08 | 52.86 | 78.76 | 73.66 | 64.4 | 66.18 | 39.73 | 65.18 | 63.15 | 61.90 | 66.06 | 62.68 | 42.21 | 82.89 | 64.46 | 60.8 | 86.40 | 89.45 | 64.56 |

Table 5: Matching accuracy on the CUB database(%).

| Method | Mean accuracy |
|---|---|
| GANN-GM [5] | 48.44 |
| GANN-MGM [5] | 78.32 |
| CIE-H [37] | 92.06 |
| PIA-GM [17] | 88.56 |
| PCA-GM [17] | 92.17 |
| IPCA-GM [36] | 92.18 |
| PCA-GM+Position+Structure | 91.46 |
| IPCA-GM+Position+Structure | **92.34** |

Table 6: Matching accuracy on the IMC-PT-SparseGM (%).

| Method | Reichstag | Sacre_coeur | St_peters_square | Mean accuracy |
|---|---|---|---|---|
| GANN-GM [5] | 76.02 | 44.15 | 50.49 | 56.89 |
| GANN-MGM [5] | 67.41 | 42.72 | 44.42 | 51.52 |
| CIE-H [37] | 42.24 | 28.47 | 30.78 | 33.83 |
| PIA-GM [17] | 71.46 | 41.31 | 42.64 | 51.80 |
| PCA-GM [17] | 69.38 | 39.86 | 42.10 | 50.40 |
| IPCA-GM [36] | 72.96 | 43.80 | 44.93 | 53.89 |
| PCA-GM+Position+Structure | **96.28** | **75.93** | 81.66 | **84.63** |
| IPCA-GM+Position+Structure | 95.71 | 74.12 | **82.06** | 83.96 |

curacy. For the CUB and IMC-PT-SparseGM datasets, our method achieves the highest mean accuracy performance compared with the alternatives. It can therefore be concluded that our model performs well on all four datasets under study, while the alternatives generally fail to perform well on at least one of the datasets. This also clearly demonstrates the advantage of our method in terms of generalization ability.

### *5.5. Ablation Study*

Here we conduct an ablation study to demonstrate that our novel method for embedding the positional and structural arrangement information can be effective in improving variants of the methods of the performance of

Table 7: Matching accuracy of ablation study on IMC-PT-SparseGM (%).

| Method | Reichstag | Sacre_coeur | St_peters_square | Mean accuracy |
|---|---|---|---|---|
| PCA-GM | 69.38 | 39.86 | 42.10 | 50.40 |
| PCA-GM+Position | 92.02 | 69.07 | 78.80 | 79.96 |
| PCA-GM+Structure | 93.83 | 70.44 | 78.78 | 81.02 |
| PCA-GM+Position+Structure | 96.49 | 72.01 | 83.00 | 83.83 |
| IPCA-GM | 72.96 | 43.80 | 44.93 | 53.89 |
| IPCA-GM+Position | 93.03 | 65.37 | 77.24 | 78.55 |
| IPCA-GM+Structure | 92.79 | 62.66 | 76.98 | 77.48 |
| IPCA-GM+Position+Structure | 95.71 | 74.12 | 82.06 | 83.96 |

Table 8: CNN backbone of ablation study on IMC-PT-SparseGM (%).

| Method | Reichstag | Sacre_coeur | St_peters_square | Mean accuracy | Time (min) |
|---|---|---|---|---|---|
| PCA-GM+Position+Structure (VGG-16) | 96.49 | 72.01 | 83.00 | 83.83 | 1173 |
| PCA-GM+Position+Structure (VGG-19) | 96.35 | 72.07 | 83.21 | 83.87 | 1358 |
| PCA-GM+Position+Structure (ResNet-50) | 96.45 | 71.84 | 83.08 | 83.79 | 979 |
| PCA-GM+Position+Structure (MobileNet-v3) | 96.31 | 71.45 | 83.24 | 83.67 | 815 |
| IPCA-GM+Position+Structure (VGG-16) | 95.71 | 74.12 | 82.06 | 83.96 | 1434 |
| IPCA-GM+Position+Structure (VGG-19) | 95.53 | 73.76 | 81.79 | 83.69 | 1706 |
| IPCA-GM+Position+Structure (ResNet-50) | 95.78 | 74.15 | 82.07 | 84.00 | 1179 |
| IPCA-GM+Position+Structure (MobileNet-v3) | 95.62 | 74.00 | 81.93 | 83.85 | 1068 |

graph matching. To this end, we construct three models based on the graph matching methods PCA-GM and IPCA-GM. The three variants are a) the original method combined with position-aware embedding, b) the original method combined with structure embedding, and c) the original method combined with complete position and structure embedding. Considering that our method has the alternative best performance on the IMC-PT-SparseGM data set compared with other methods, we conduct the ablation study on the IMC-PT-SparseGM dataset. The experimental results are shown in Table 7.

The proposed position-aware node embedding and subgraph-based structure embedding greatly improve the classification accuracy of the original method in each category. Note that the IMC-PT-SparseGM dataset consists of tourist attractions, of which many (especially the buildings) are symmetrical, indicating that the semantic information of the keypoints is usually ambiguous. Therefore, the incorporation of position information for the graph matching methods can effectively assist the classification task. The high-order subgraph-based structural arrangement information is also valuable, since the global structural features of the tourist attractions can provide more comprehensive information than just local features alone.

Furthermore, in this paper we make use of VGG-16 to extract image features. Note that the proposed method is not designed to be critically dependent on VGG-16, and the chosen feature extractor step can be replaced by any of the other available CNN models. Furthermore, we conduct an ablation study that compares the different CNN backbones applied in the proposed method, both in terms of performance and also computation time. Specifically, we include VGG-19, ResNet-50, and MobileNet-v3 in addition to VGG-16. Each of these backbone models is sourced from the TorchVision library [40]. The results are shown in Table 8, from which it can be concluded that using the choice of different backbone models has little effect on

the performance. Although there are minor differences in the initial feature extraction capability, the high-order feature extraction and matching algorithm do have a major influence on the final graph matching outcome. As far as time consumption is concerned, MobileNet-v3 shows clear advantages in terms of computation time due to its lightweight design. However, this is at the cost of sacrificing some accuracy. Here, we are at pains to ensure fairness of comparison with alternative methods. Based on these considerations we opt to use VGG-16 as the backbone architecture in image feature extraction.

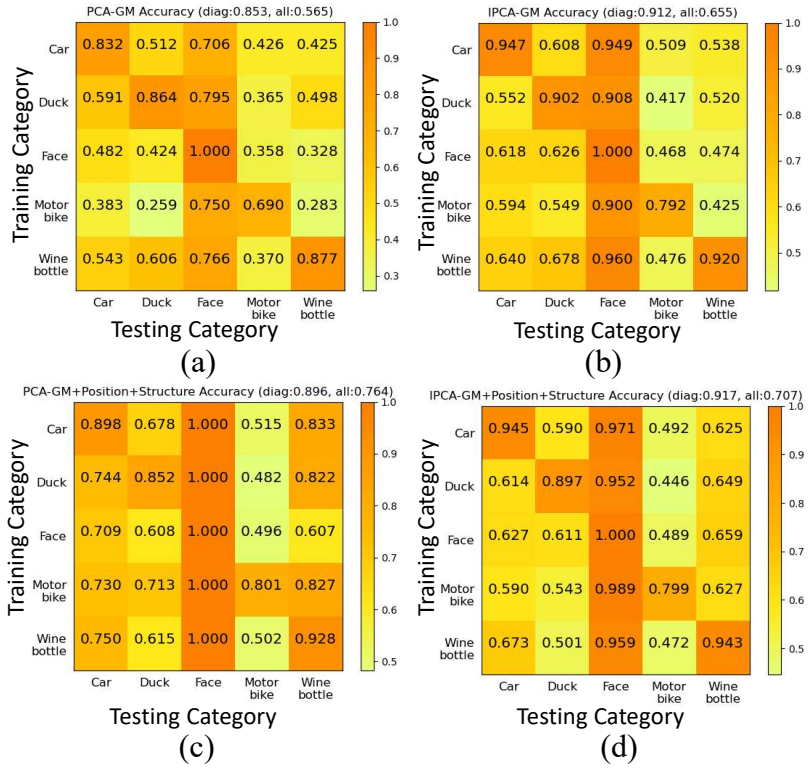


Figure 7: Cross-category generalization study represented by confusion matrix on Willow ObjectClass dataset. Each column of the matrix represents one category of images used for model training, while each row represents one class of testing samples. a) and b) show the matching accuracy of comparison methods in different category generalizations, respectively. c) and d) are obtained from the proposed methods.

### *5.6. Cross-category Generalization Study*

To verify the generalization capability of our methods on different types of categories, we construct the confusion matrix for the training and testing

instances from the different categories. The experimental results are presented in Fig. 7 and Fig. 8. We first train each of the five classes separately for the Willow dataset. We then use the distinct training classes to validate the generalization capabilities for all five object classes.

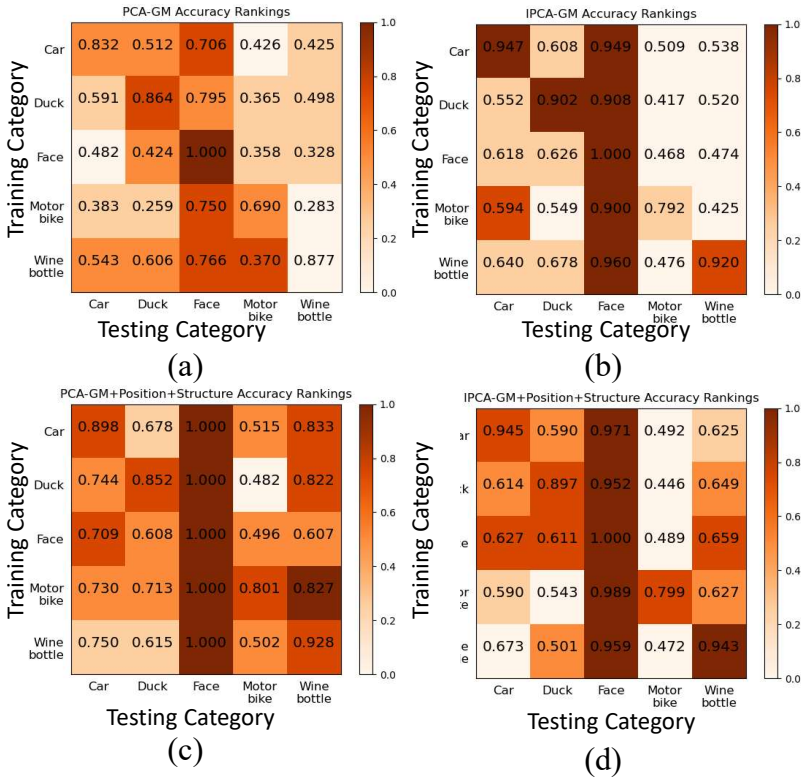


Figure 8: Cross-category generalization study method ranking on Willow ObjectClass dataset. The re-color figure from Fig. 7 by sorting all the matching results obtained by four different methods from large to small.

The confusion matrix for the dataset illustrating transfer learning among the five object categories is given in Fig. 7. The models are trained on the object categories in the rows and then tested on the object categories in the columns. The diagonal of the matrix shows the classification ability of the method on each object class. The darker the grid color in the figure, the higher the matching accuracy. We compare each element in Fig. 7 (a) with Fig. 7(c), and Fig. 7(b) with Fig. 7(d). The proposed graph matching method combined with position and structural arrangement information has improved the generalization ability for all classes compared with the original

methods. Furthermore, the accuracy of the model in the face category reaches 100% in Fig. 7(c), and is also high in both Fig. 7(b) and Fig. 7(d). It may be related to the simplicity and lower noise in the face samples.

In Fig. 8, we re-color Fig. 7 by sorting all the matching results obtained by the four different methods from large to small. The larger the value, the darker the color. This is also a reflection of the generalization ability of the model. Moreover, it gives the priority to the proposed position and structure framework based on higher accuracy scores in the confusion matrix. The generalization ability of the models provided in this study is substantially greater than the alternative methods due to the addition of position information and the correlation connection of the subgraph structure. Furthermore, IPCA-GM+Position+Structure has greater diagonal accuracy, indicating that it is more suitable for the training category.

## 6. Limitations

This work aims to address the issue of ambiguous matching that results from the absence of discriminability based on local structure and the semantic similarity of different nodes. We therefore focus on the feature extraction problem during the graph-matching task. However, we do not consider issues arising from the fact that the image itself may contain noise and other abnormalities. Our position-aware node embedding module and subgraph-based structural embedding module are adaptive plug-ins that can boost the performance of other methods too. However, this is at the expense of additional computing time consumption and memory usage.

Future work will aim to investigate the noisy image-matching task so that it can be made more practically applicable. In addition, and from a problem-solving perspective, we can consider how to deal with matching graphs that have more complicated structural relations. These include hypergraphs and directed graphs or using various relative position and subgraph relational strategies.

## 7. Conclusion

In this paper, we propose a novel graph matching method, which combines and incorporates a relative position-aware node embedding and together with a subgraph-based structure embedding into the node-wise embedding between graphs. The experiments include comparisons with alternatives, an

ablation study and the cross-category generalization study to demonstrate the effectiveness of the proposed method with its embedding modules. Specifically, the matching accuracy on several real-world datasets compared with peer methods demonstrate the state-of-the-art performance of our method.

## References


[1] J. Hou, M. Pelillo, H. Yuan, Hypergraph matching via game-theoretic hypergraph clustering, Pattern Recognition 125 (2022) 108526.

[2] Y. Cheng, X. Zhu, J. Qian, F. Wen, P. Liu, Cross-modal graph matching network for image-text retrieval, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 18 (4) (2022) 1–23.

[3] B. Jiang, H. Zhao, J. Tang, B. Luo, A sparse nonnegative matrix factorization technique for graph matching problems, Pattern Recognition 47 (2) (2014) 736–747.

[4] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, A. J. Smola, Learning graph matching, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (6) (2009) 1048–1058.

[5] R. Wang, J. Yan, X. Yang, Graduated assignment for joint multi-graph matching and clustering with application to unsupervised graph matching network learning, Advances in Neural Information Processing Systems 33 (2020) 19908–19919.

[6] H. Zhang, Z. Huang, X. Lin, Z. Lin, W. Zhang, Y. Zhang, Efficient and high-quality seeded graph matching: Employing higher-order structural information, ACM Transactions on Knowledge Discovery from Data (TKDD) 15 (3) (2021) 1–31.

[7] K. Henderson, B. Gallagher, L. Li, L. Akoglu, T. Eliassi-Rad, H. Tong, It's who you know: graph mining using recursive structural features, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2011, pp. 663–671.

[8] R. Wang, J. Yan, 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) 1–1.

[9] S. Berretti, A. Del Bimbo, E. Vicario, Efficient matching and indexing of graph models in content-based retrieval, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (10) (2001) 1089–1105.

[10] A. Zanfir, C. Sminchisescu, Deep learning of graph matching, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2684–2693.

[11] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge, International Journal of Computer Vision 88 (2) (2010) 303–338.

[12] W. Burger, M. J. Burge, Scale-invariant feature transform, in: Digital Image Processing, Springer, 2022, pp. 709–763.

[13] F. Cen, X. Zhao, W. Li, G. Wang, Deep feature augmentation for occluded image classification, Pattern Recognition 111 (2021) 107737.

[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.

[15] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[16] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: International Conference on Learning Representations, 2014.

[17] R. Wang, J. Yan, X. Yang, Learning combinatorial embedding networks for deep graph matching, in: IEEE International Conference on Computer Vision, 2019, pp. 3056–3065.

[18] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, Advances in Neural Information Processing Systems 29 (2016) 3844–3852.

[19] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations, 2017.

[20] Y. Yang, Y. Qi, Image super-resolution via channel attention and spatial graph convolutional network, Pattern Recognition 112 (2021) 107798.

[21] M. Zhang, Y. Chen, Link prediction based on graph neural networks, Advances in Neural Information Processing Systems 31 (2018) 5165–5175.

[22] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations, 2017.

[23] V. Arvind, F. Fuhlbrück, J. Köbler, O. Verbitsky, On weisfeiler-leman invariance: Subgraph counts and related graph properties, Journal of Computer and System Sciences 113 (2020) 42–59.

[24] F. Zhou, F. De la Torre, Factorized graph matching, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 127–134.

[25] P. A. Knight, The sinkhorn–knopp algorithm: convergence and applications, SIAM Journal on Matrix Analysis and Applications 30 (1) (2008) 261–275.

[26] V. Traneva, S. Tranev, V. Atanassova, An intuitionistic fuzzy approach to the hungarian algorithm, in: International Conference on Numerical Methods and Applications, Springer, 2018, pp. 167–175.

[27] G. Patrini, R. van den Berg, P. Forre, M. Carioni, S. Bhargav, M. Welling, T. Genewein, F. Nielsen, Sinkhorn autoencoders, in: Uncertainty in Artificial Intelligence, 2020, pp. 733–743.

[28] Y. Quan, Y. Chen, Y. Shao, H. Teng, Y. Xu, H. Ji, Image denoising using complex-valued deep cnn, Pattern Recognition 111 (2021) 107639.

[29] W. Fang, F. Zhang, V. S. Sheng, Y. Ding, A method for improving cnn-based image recognition using dcgan, Computers, Materials and Continua 57 (1) (2018) 167–178.

[30] I. Sergienko, V. Shylo, S. Chupov, P. Shylo, Solving the quadratic assignment problem, Cybernetics and Systems Analysis 56 (1) (2020) 53–57.

[31] J. You, R. Ying, J. Leskovec, Position-aware graph neural networks, in: International Conference on Machine Learning, 2019, pp. 7134–7143.

[32] I. Papakis, A. Sarkar, A. Karpatne, A graph convolutional neural network based approach for traffic monitoring using augmented detections with optical flow, in: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), IEEE, 2021, pp. 2980–2986.

[33] M. Cho, K. Alahari, J. Ponce, Learning graphs to match, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 25–32.

[34] C. Wah, S. Branson, P. Welinder, P. Perona, S. Belongie, The Caltech-UCSD Birds-200-2011 Dataset, Tech. Rep. CNS-TR-2011-001, California Institute of Technology (2011).

[35] Y. Jin, D. Mishkin, A. Mishchuk, J. Matas, P. Fua, K. M. Yi, E. Trulls, Image matching across wide baselines: From paper to practice, International Journal of Computer Vision (2021) 517–547.

[36] R. Wang, J. Yan, X. Yang, Combinatorial learning of robust deep graph matching: an embedding based approach, IEEE Transactions on Pattern Analysis and Machine Intelligence (2020) 1–1.

[37] T. Yu, R. Wang, J. Yan, B. Li, Learning deep graph matching with channel-independent embedding and hungarian attention, in: International Conference on Learning Representations, 2019.

[38] I. K. M. Jais, A. R. Ismail, S. Q. Nisa, Adam optimization algorithm for wide and deep neural network, Knowledge Engineering and Data Science 2 (1) (2019) 41–46.

[39] Y. Liu, G. Yin, The delaunay triangulation learner and its ensembles, Computational Statistics and Data Analysis 152 (2020) 107030.

[40] S. Marcel, Y. Rodriguez, Torchvision the machine-vision package of torch (2010) 1485–1488.