Adaptive Edge Attention for Graph Matching with Outliers

Jingwei Qu[1*], **Haibin Ling**[2], **Chenrui Zhang**[1], **Xiaoqing Lyu**[1] and **Zhi Tang**[1]
[1]Wangxuan Institute of Computer Technology, Peking University, Beijing, China
[2]Department of Computer Science, Stony Brook University, Stony Brook, NY 11794 USA
qujingwei@pku.edu.cn, hling@cs.stonybrook.edu, {chenrui.zhang, lvxiaoqing, tangzhi}@pku.edu.cn

Abstract

Graph matching aims at establishing correspondence between node sets of given graphs while keeping the consistency between their edge sets. However, outliers in practical scenarios and equivalent learning of edge representations in deep learning methods are still challenging. To address these issues, we present an Edge Attention-adaptive Graph Matching (EAGM) network and a novel description of edge features. EAGM transforms the matching relation between two graphs into a node and edge classification problem over their assignment graph. To explore the potential of edges, EAGM learns edge attention on the assignment graph to 1) reveal the impact of each edge on graph matching, as well as 2) adjust the learning of edge representations adaptively. To alleviate issues caused by the outliers, we describe an edge by aggregating the semantic information over the space spanned by the edge. Such rich information provides clear distinctions between different edges (e.g., inlier-inlier edges vs. inlier-outlier edges), which further distinguishes outliers in the view of their associated edges. Extensive experiments demonstrate that EAGM achieves promising matching quality compared with state-of-the-arts, on cases both with and without outliers. Our source code along with the experiments is available at https://github.com/bestwei/EAGM.

1 Introduction

Aiming at establishing node correspondence between graphs while keeping the consistency between their edges, graph matching has been widely applied in computer vision, *e.g.*, shape matching, object recognition, and tracking, *etc*. From finding node correspondence in pure cases (*i.e.*, all nodes are inliers), to establishing matches between inliers while overcoming interference caused by outliers, there are many open issues to be solved in this field. Actually, the outliers are common in practical scenarios. In this paper, we focus on addressing the general graph matching problem with outliers.

*Contact Author

As a classical combinatorial optimization problem, graph matching is NP-hard and generally solved by acceptable suboptimal solutions. Early methods [Gold and Rangarajan, 1996; Leordeanu *et al.*, 2009; Cho *et al.*, 2010; Egozi *et al.*, 2012; Liu and Qiao, 2013] propose approximate algorithms with some relaxations. Then machine learning methods [Torresani *et al.*, 2008; Caetano *et al.*, 2009; Leordeanu *et al.*, 2011; Leordeanu *et al.*, 2012; Cho *et al.*, 2013] compute node and edge affinities by simple and shallow parametric models, while the accuracy promotion is limited. Recently, encouraged by the successes of deep neural networks in many research areas, deep learning methods [Zanfir and Sminchisescu, 2018; Wang *et al.*, 2019; Yu *et al.*, 2020a; Wang *et al.*, 2020] are introduced for learning the combinatorial solver in graph neural networks (GNN). Point positions and object appearance in visual matching tasks provide rich geometric and semantic features for these methods. However, two important challenges remain unsolved.

First, the outliers bring inevitable interference to the matching process in practical scenarios. Previous methods usually propose enhanced node features to reduce interference of outliers, but devote insufficient discrimination on the features of associated edges. Second, the initial graph structure in visual matching tasks is generally constructed in heuristic ways (*e.g.*, Delauney triangulation or k-nearest), and the obtained edges are all utilized during the learning process. Some edges thus generated are necessary, but others may be noisy. In fact, the impact of each edge on graph matching is different. Besides, different from links in social networks and bonds in molecular graphs, features of these edges are typically constructed by simple combinations (*e.g.*, concatenation or element-wise multiplication) over features from their associated nodes. Thus reasonable and informative descriptions of edges are worth exploring.

To address the two challenges mentioned above, we propose an Edge Attention-adaptive Graph Matching (EAGM) network (as illustrated in Figure 1), and a novel descriptive approach of edge features. Given a pair of graphs to be matched, EAGM transforms the matching between them into a node and edge classification problem over their assignment graph. The geometric features and semantic features of the assignment graph are fused to encode jointly position and appearance information. Then EAGM learns the edge attention on the assignment graph with fused features, which indicates

the impact of each edge on graph matching. The edge attention is imposed into the learning of node and edge classification on the assignment graph, which adjusts the learning of edge representations adaptively. The final predicted labels indicate whether the two original nodes or edges are matched in the graph matching problem.

To alleviate the interference caused by outliers, we design the edge semantic features to distinguish outliers from the aspect of their adjacent edges, rather than enhancing the node features only. We consider each edge as consisting of discrete points, *i.e.*, its passing pixel points in visual matching tasks. Then we describe its semantic features by involving the convolutional features of all passed pixel points. Compared with the previous approaches relying on features of the associated nodes, such edge features are more informative and independent. It can provide strong distinctions between inlier-inlier and inlier-outlier edges. Since the information of associated nodes is also included, the differences between the two kinds of edges further enhance the discrimination between inliers and outliers.

In summary, the main contributions of this paper include:

- We propose an Edge Attention-adaptive Graph Matching network, EAGM, which adjusts edge attention adaptively during learning graph matching. The impact of each edge on graph matching is described by the learned edge attention. The final predicted node and edge labels of the assignment graph indicate the matching relations between the two original graphs.
- We devise a novel edge feature by encoding semantic-aware information distributed spatially around edge pixels. Such feature not only provides reasonable and independent description for edges in visual matching tasks, but also distinguishes outliers by their adjacent edges.
- EAGM achieves promising performance on three popular benchmarks, on experiments both with and without outliers.

2 Related Work

Graph matching has received significant attention in decades. Readers are referred to [Yan *et al.*, 2020] for a comprehensive acquaintance. Early methods [Gold and Rangarajan, 1996; Leordeanu *et al.*, 2009; Cho *et al.*, 2010; Egozi *et al.*, 2012; Zhou and De la Torre, 2012; Liu and Qiao, 2013; Wang *et al.*, 2018] usually search acceptable sub-optimal solutions via heuristics. [Leordeanu *et al.*, 2009] utilizes original discrete constraints to optimize graph matching with climbing and convergence properties. Recently, [Yu *et al.*, 2020b] proposes a determinant regularization technique on the correspondence matrix to perform gradient-efficient continuation optimization. However, being limited by handcrafted affinities, these methods still lack robust ability on real-work matching tasks.

Inspired by growing data and features, some methods [Torresani *et al.*, 2008; Caetano *et al.*, 2009; Leordeanu *et al.*, 2011; Leordeanu *et al.*, 2012; Cho *et al.*, 2013] devote attention to learn parameters of the corresponding affinity matrix, rather than the handcrafted affinities. [Cho *et al.*, 2013] utilizes structured SVM to parameterize a graph model, and apply its structural attributes to visual matching. Weight parameters can also be learned via semi-supervised [Leordeanu *et al.*, 2011] or unsupervised [Leordeanu *et al.*, 2012] models. Although the affinity weights are learnable, the node representations and structural information are still to be mined.

Recent deep learning methods [Zanfir and Sminchisescu, 2018; Wang *et al.*, 2019; Yu *et al.*, 2020a; Wang *et al.*, 2020] utilize features extracted by convolutional neural networks (CNN) or geometric information to initialize graph features, and learn a combinatorial solver by GNN. The pioneer work [Zanfir and Sminchisescu, 2018] computes unary and pairwise node affinities by CNN features, and adopts spectral matching [Leordeanu and Hebert, 2005] as a differentiable (but fixed) combinatorial solver. [Wang *et al.*, 2019] adopts GNN to aggregate graph structure information into node representations, and a Sinkhorn net is introduced as the combinatorial solver on node affinities. [Yu *et al.*, 2020a] devises a channel-independent embedding method, and utilizes Hungarian attention to smooth objective learning for graph matching. [Wang *et al.*, 2020] converts matching between two graphs into a node classification problem in their assignment graph, and learns the affinity functions and combinatorial solver simultaneously.

Despite the satisfactory progress achieved, the above methods are still sensitive to common outliers in practical scenarios, or lack impact adjustment of each edge on graph matching. In this paper, we overcome these problems by learning the edge attention-adaptive graph matching and by designing a novel edge semantic description.

3 Definition of Graph Matching

An undirected graph is represented by $\mathbb{G} = (\mathbb{V}, \mathbb{E}, \mathcal{V}, \mathcal{E})$,

- $\mathbb{V} = \{v_1, \dots, v_n\}$ denotes the node set, $|\mathbb{V}| = n$,
- $\mathbb{E} \subseteq \mathbb{V} \times \mathbb{V}$ denotes the edge set, $|\mathbb{E}| = m$,
- $\mathcal{V} = \{\mathbf{v}_i | \mathbf{v}_i \in \mathbb{R}^{d_\mathrm{v}}, i = 1, \dots, n\}$ denotes the node feature set,
- $\mathcal{E} = \{\mathbf{e}_i | \mathbf{e}_i \in \mathbb{R}^{d_\mathrm{e}}, i = 1, \dots, m\}$ denotes the edge feature set.

Given two graphs $\mathbb{G}_1 = (\mathbb{V}_1, \mathbb{E}_1, \mathcal{V}_1, \mathcal{E}_1)$ and $\mathbb{G}_2 = (\mathbb{V}_2, \mathbb{E}_2, \mathcal{V}_2, \mathcal{E}_2)$ with[1] $|\mathbb{V}_1| = |\mathbb{V}_2| = n$, the graph matching problem is to find a node correspondence $\mathbf{X} \in \{0,1\}^{n \times n}$ between $\mathbb{G}_1$ and $\mathbb{G}_2$. $\mathbf{X}_{ia} = 1$ iff $v_i \in \mathbb{V}_1$ corresponds to $v_a \in \mathbb{V}_2$. Graph matching is generally formulated by a quadratic assignment programming (QAP) problem [Loiola *et al.*, 2007].

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \mathbf{x}^\top \mathbf{K} \mathbf{x}$$
$$\mathbf{x} = \mathrm{vec}(\mathbf{X}) \in \{0,1\}^{n^2}, \ \mathbf{X}\mathbf{1}_n = \mathbf{1}_n, \text{ and } \ \mathbf{X}^\top \mathbf{1}_n = \mathbf{1}_n$$

The vector $\mathbf{x}^*$ denotes the desired node correspondence. The required one-to-one constraint is described by $\mathbf{X}\mathbf{1}_n = \mathbf{1}_n$ and $\mathbf{X}^\top \mathbf{1}_n = \mathbf{1}_n$ ($\mathbf{1}_n$ denotes a vector of n ones). $\mathbf{K} \in \mathbb{R}^{n^2 \times n^2}$ is the corresponding affinity matrix,

$$\mathbf{K}_{ia,jb} = \begin{cases} s^{\mathrm{v}}_{ia}, & \text{if } i = j \ \& \ a = b, \\ s^{\mathrm{e}}_{iajb}, & \text{else if } \mathbf{A}^1_{ij} \mathbf{A}^2_{ab} > 0, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

[1] We assume the two graphs have the same size n, and definitions can be extended to varied sizes, *e.g.*, by adding dummy nodes.

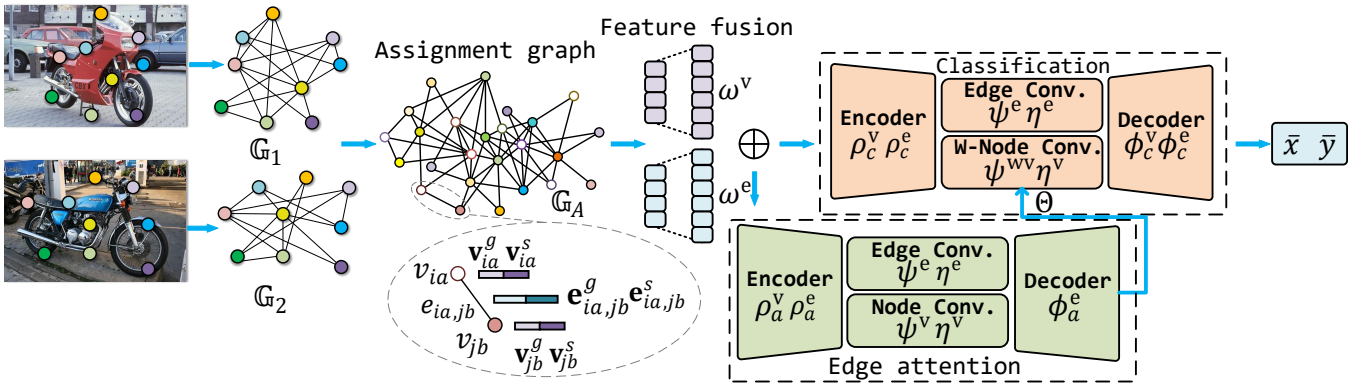


Figure 1: **Architecture of EAGM**. The assignment graph $\mathbb{G}_A$ is firstly generated for graphs $\mathbb{G}_1$ and $\mathbb{G}_2$. Its geometric features $\{\mathcal{V}_A^g, \mathcal{E}_A^g\}$ and semantic features $\{\mathcal{V}_A^s, \mathcal{E}_A^s\}$ are further fused, and the fused features $\{\mathcal{V}_A, \mathcal{E}_A\}$ are fed into the edge attention and classification modules. Next, the edge attention Θ is learned, and utilized to adjust the impact of each edge. Finally, the node and edge labels of $\mathbb{G}_A$ are predicted by the classification module to indicate the matching results between $\mathbb{G}_1$ and $\mathbb{G}_2$.

where $\mathbf{A} \in \mathbb{R}^{n\times n}$ is a symmetric adjacency matrix describing node connections, and $\mathbf{A}_{ij} = 1$ iff there is an edge $e_{i,j} = (v_i, v_j) \in \mathbb{E}$. s^{v}_{ia} measures the affinity between the node features $\mathbf{v}_i \in \mathcal{V}_1$ and $\mathbf{v}_a \in \mathcal{V}_2$. s^{e}_{iajb} measures the affinity between edge features $\mathbf{e}_{i,j} \in \mathcal{E}_1$ and $\mathbf{e}_{a,b} \in \mathcal{E}_2$.

4 Edge Attention-Adaptive Graph Matching

In the proposed EAGM (Figure 1), graph matching is learned by four modules:

- **Edge Semantic Description.** This module classifies node and edge features of the two given graphs $\mathbb{G}_1$ and $\mathbb{G}_2$ into the geometric features and the semantic features. The edge semantic features are designed emphatically.
- **Assignment Graph Generation.** This module generates the assignment graph $\mathbb{G}_A$ for $\mathbb{G}_1$ and $\mathbb{G}_2$, and fuses its geometric features and semantic features. The original graph matching between $\mathbb{G}_1$ and $\mathbb{G}_2$ is converted into the node and edge binary classification in $\mathbb{G}_A$.
- **Edge Attention.** The edge attention module is the core component of EAGM. It takes $\mathbb{G}_A$ with fused features as input, and learns the attention for each edge of $\mathbb{G}_A$ to adjust its impact on learning graph matching. This module adopts an encoder-decoder architecture, along with a convolution sub-module. The coupled encoder and decoder sub-modules transform fused features of $\mathbb{G}_A$ into latent representations, and predict the edge attention Θ based on the final graph state, respectively. The convolution sub-module consists of l_1 coupled edge convolution layers and node convolution layers, which learns node and edge representations for learning attention Θ.
- **Classification.** This module combines $\mathbb{G}_A$ with fused features and the edge attention Θ, and predicts final node and edge labels. It has similar autoencoder architecture with the former module. Its decoder outputs the predicted node and edge labels. The convolution sub-module consists of l_2 coupled edge convolution layers and weighted-node convolution layers, and utilizes the edge attention Θ to parameterize the aggregation in the weighted-node convolution layers. Besides, we propose a binary cross-entropy loss for guiding learning of edge representations and edge attention.

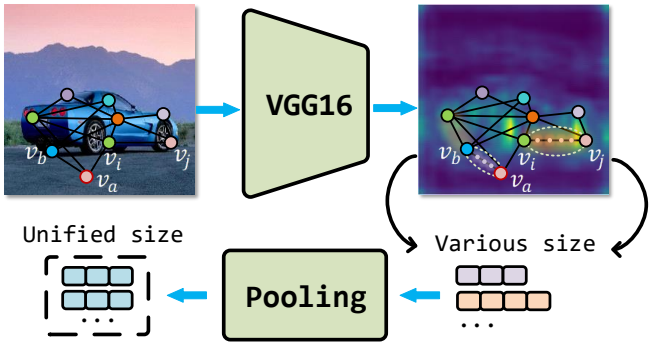


Figure 2: **Edge semantic feature**. The differences between edge semantic features $\mathbf{e}^s_{i,j}$ and $\mathbf{e}^s_{a,b}$ enhance the discrimination on the outlier v_a.

4.1 Edge Semantic Description

We categorize the node and edge features into two types, geometric feature and semantic feature,

- $\mathcal{V} = \{\mathcal{V}^g, \mathcal{V}^s\}$, $\mathcal{V}^g = \{\mathbf{v}^g_i | \mathbf{v}^g_i \in \mathbb{R}^{d^g_{\mathrm{v}}}, i = 1, \ldots, n\}$ and $\mathcal{V}^s = \{\mathbf{v}^s_i | \mathbf{v}^s_i \in \mathbb{R}^{d^s_{\mathrm{v}}}, i = 1, \ldots, n\}$ denote the node geometric and semantic feature set, respectively,
- $\mathcal{E} = \{\mathcal{E}^g, \mathcal{E}^s\}$, $\mathcal{E}^g = \{\mathbf{e}^g_i | \mathbf{e}^g_i \in \mathbb{R}^{d^g_{\mathrm{e}}}, i = 1, \ldots, m\}$ and $\mathcal{E}^s = \{\mathbf{e}^s_i | \mathbf{e}^s_i \in \mathbb{R}^{d^s_{\mathrm{e}}}, i = 1, \ldots, m\}$ denote the edge geometric and semantic feature set, respectively.

Specifically, we utilize the 2D Cartesian coordinates of each node v_i as its geometric feature $\mathbf{v}^g_i = [x_i, y_i]$, and concatenate the geometric features of head node v_{h_i} and tail node v_{t_i} associated with each edge e_i to form its geometric feature $\mathbf{e}^g_i = [\mathbf{v}^g_{h_i}; \mathbf{v}^g_{t_i}]$.

For the node semantic feature, we extracted CNN feature maps from the input images, and interpolate them as the size

of images. The spatially corresponding feature vector located at (x_i, y_i) for each node is selected as its semantic feature $\mathbf{v}_i^s \in \mathbb{R}^{d_{\mathrm{v}}^s}$. The node semantic features are L2 normalized.

To clearly explain the design of edge semantic feature, we take an edge $e_{i,j} = (v_i, v_j)$ in Figure 2 as an example. We firstly adopt Bresenham's line algorithm [Bresenham, 1965] to compute the pixel points passed by the edge $e_{i,j}$, and collect them as a point set $\mathbb{P} = \{v_i, p_1, \ldots, p_k, v_j\}$. Then the semantic features of these points are constructed by the above operations of node semantic features, and form a feature set $\mathcal{V}_{\mathbb{P}}^s = \{\mathbf{v}_i^s | \mathbf{v}_i^s \in \mathbb{R}^{d_{\mathrm{v}}^s}, i = 1, \ldots, k+2\}$. Next we stack these features, and obtain the initial edge semantic feature $\mathbf{e}_{i,j}^s \in \mathbb{R}^{k \times d_{\mathrm{v}}^s}$. Notably, different edges pass different amounts of pixel points (*e.g.*, $e_{i,j}$ and $e_{a,b}$), but the size of all edge features should be unified in a GNN. Thus we finally adopt the max pooling operation on the initial edge semantic features to obtain the unified size d_{e}^s, *e.g.*, $\mathbf{e}_{i,j}^s \in \mathbb{R}^{d_{\mathrm{e}}^s}$. The edge semantic features are also L2 normalized.

4.2 Assignment Graph Generation

Inspired by the successful assignment graph in [Leordeanu and Hebert, 2005; Cho *et al.*, 2010; Wang *et al.*, 2020], we transform the graph matching problem between the two graphs $\mathbb{G}_1$ and $\mathbb{G}_2$ into a node and edge binary classification problem in the assignment graph $\mathbb{G}_A = (\mathbb{V}_A, \mathbb{E}_A, \mathcal{V}_A, \mathcal{E}_A)$. $\mathbb{G}_A$ describes all candidate node and edge correspondences between $\mathbb{G}_1$ and $\mathbb{G}_2$. For structure generation of $\mathbb{G}_A$, each candidate node correspondence $v_i \in \mathbb{V}_1$ and $v_a \in \mathbb{V}_2$ generates a node $v_{ia} \in \mathbb{V}_A$. An edge $e_{ia,jb} = (v_{ia}, v_{jb}) \in \mathbb{E}_A$ is constructed iff there are two edges $e_{i,j} \in \mathbb{E}_1$ and $e_{a,b} \in \mathbb{E}_2$.

For feature initialization of $\mathbb{G}_A$, we concatenate the initial geometric and semantic features of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively.

$$\begin{aligned} \mathbf{v}_{ia}^g &= [\mathbf{v}_i^g; \mathbf{v}_a^g] \in \mathbb{R}^{2d_{\mathrm{v}}^g}, & \mathbf{v}_{ia}^s &= [\mathbf{v}_i^s; \mathbf{v}_a^s] \in \mathbb{R}^{2d_{\mathrm{v}}^s} \\ \mathbf{e}_{ia,jb}^g &= [\mathbf{e}_{i,j}^g; \mathbf{e}_{a,b}^g] \in \mathbb{R}^{2d_{\mathrm{e}}^g}, & \mathbf{e}_{ia,jb}^s &= [\mathbf{e}_{i,j}^s; \mathbf{e}_{a,b}^s] \in \mathbb{R}^{2d_{\mathrm{e}}^s} \end{aligned}$$

To enable the following modules to learn about both position and appearance jointly, we further fuse the geometric and semantic features of $\mathbb{G}_A$ by two multilayer perceptrons (MLPs) ω^{v} and ω^{e}. They embed a node geometric feature and an edge geometric feature into their semantic features respectively:

$$\mathcal{V}_A = \mathcal{V}_A^s + \omega^{\mathrm{v}}(\mathcal{V}_A^g),\ \mathcal{E}_A = \mathcal{E}_A^s + \omega^{\mathrm{e}}(\mathcal{E}_A^g) \tag{2}$$

where $\omega^{\mathrm{v}}(\mathcal{V}_A^g)$ and $\omega^{\mathrm{e}}(\mathcal{E}_A^g)$ denote the embedded node geometric features and edge geometric features, by applying ω^{v} and ω^{e} to each node and each edge respectively. Then the updated graph $\mathbb{G}_A$ with fused features is passed to the following edge attention module and classification module as input.

Therefore, the original graph matching between $\mathbb{G}_1$ and $\mathbb{G}_2$ is converted into the binary classification problem in $\mathbb{G}_A$, *i.e.*, positive and negatives nodes or edges are equivalent to the two original nodes or edges being matched or not.

4.3 Edge Attention Learning

The edge attention module takes the updated graph $\mathbb{G}_A$ with fused features as input, and learns the attention θ_i of each edge $e_i \in \mathbb{E}_A$ for graph matching. It consists of three sub-modules, encoder, convolution, and decoder sub-module.

The encoder sub-module embeds the fused features $\mathcal{V}_A$ and $\mathcal{E}_A$ into latent representations by two MLPs ρ_a^{v} and ρ_a^{e}, respectively. The updated graph $\mathbb{G}_A = (\mathbb{V}_A, \mathbb{E}_A, \rho_a^{\mathrm{v}}(\mathcal{V}_A), \rho_a^{\mathrm{e}}(\mathcal{E}_A))$ is fed into the following convolution sub-module.

The convolution sub-module consists of l_1 coupled edge convolution layers and node convolution layers, which updates latent representations of $\mathbb{G}_A$. Overall, the edge convolution layer aggregates the representations of the two associated nodes of each edge to update the representation of this edge. The following node convolution layer aggregates the representations of all the edges adjacent to each node to update the node representation.

Edge convolution. The edge convolution layer updates the edge representations of $\mathbb{G}_A$ by firstly aggregating the representations $\mathbf{v}_{h_i}$ and $\mathbf{v}_{t_i}$ of head and tail nodes of each edge $e_i \in \mathbb{E}_A$ to produce an intermediate edge representation $\hat{\mathbf{e}}_i$.

$$\hat{\mathbf{e}}_i = \psi^{\mathrm{e}}(\mathbf{v}_{h_i} \circ \mathbf{v}_{t_i}) \tag{3}$$

where the operation $\circ$ denotes the Hadamard product between two vectors. Then the edge representation is updated by transforming the concatenation of the aggregated representation $\hat{\mathbf{e}}_i$ and the original edge representation $\mathbf{e}_i$.

$$\mathbf{e}_i' = \eta^{\mathrm{e}}([\hat{\mathbf{e}}_i; \mathbf{e}_i]) \tag{4}$$

Node convolution. The node convolution layer updates the node representations of $\mathbb{G}_A$ by firstly aggregating the representations of associated edges $\mathcal{E}_A^i$ for each node $v_i \in \mathbb{V}_A$ to generate an intermediate node representation $\hat{\mathbf{v}}_i$.

$$\hat{\mathbf{v}}_i = \psi^{\mathrm{v}}(\mathcal{E}_A^i) = \sum\nolimits_{\mathbf{e}_k \in \mathcal{E}_A^i} \mathbf{e}_k \tag{5}$$

Then the node representation is updated by a similar way in the edge convolution layer.

$$\mathbf{v}_i' = \eta^{\mathrm{v}}([\hat{\mathbf{v}}_i; \mathbf{v}_i]) \tag{6}$$

The three functions ψ^{e}, η^{e}, and η^{v} are implemented by MLPs.

The decoder sub-module predicts the edge attention Θ by the final graph state to indicate the impact of each edge on graph matching. Since we focus on learning attention for edges, this sub-module is implemented by only one MLP ϕ_a^{e}.

$$\Theta = \phi_a^{\mathrm{e}}(\mathcal{E}_A),\ \Theta \in \mathbb{R}^{m_1 m_2} \tag{7}$$

4.4 Classification

The classification module accepts $\mathbb{G}_A$ with fused features and the learned edge attention Θ as input, and predicts node and edge labels of $\mathbb{G}_A$ to reason whether the two original nodes and edges are matched respectively. The edge attention Θ is imposed into the learning of edge representations. This module has three similar sub-modules with the edge attention module. The encoder also embeds fused features of $\mathbb{G}_A$ into latent representations by two MLPs ρ_c^{v} and ρ_c^{e} respectively.

The convolution sub-module consists of l_2 coupled edge convolution layers and weighted-node convolution layers.

Weighted-Node convolution. Different from the node convolution layer in the edge attention module, the aggregation in the weighted-node convolution layer is parameterized by the attention of associated edges Θ^i for each node $v_i \in \mathbb{V}_A$.

$$\hat{\mathbf{v}}_i = \psi^{\mathrm{wv}}(\mathcal{E}_A^i, \Theta^i) = \sum\nolimits_{\mathbf{e}_k \in \mathcal{E}_A^i, \theta_k \in \Theta^i} \exp(-\theta_k)\mathbf{e}_k \tag{8}$$

Method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	d.table	dog	horse	m.bike	person	plant	sheep	sofa	train	tv	Avg.
IPFP	25.1	26.4	41.4	50.3	43.0	32.9	37.3	32.5	33.6	28.2	26.9	26.1	29.9	32.0	28.8	62.9	28.2	45.0	69.3	33.8	36.6
RRWM	30.9	40.0	46.4	54.1	52.3	35.6	47.4	37.3	36.3	34.1	28.8	35.0	39.1	36.2	39.5	67.8	38.6	49.4	70.5	41.3	43.0
PSM	32.6	37.5	49.9	53.2	47.8	34.6	50.1	35.5	37.2	36.3	23.1	32.7	42.4	37.1	38.5	62.3	41.7	54.3	72.6	40.8	43.1
GNCCP	28.9	37.1	46.2	53.1	48.0	36.3	45.5	34.7	36.3	34.2	25.2	35.3	39.8	39.6	40.7	61.9	37.4	50.5	67.0	34.8	41.6
ABPF	30.9	40.4	47.3	54.5	50.8	35.1	46.7	36.3	40.9	38.9	16.3	34.8	39.8	39.6	39.3	63.2	37.9	50.2	70.5	41.3	42.7
GMN	31.9	47.2	51.9	40.8	68.7	72.2	53.6	52.8	34.6	48.6	72.3	47.7	54.8	51.0	38.6	75.1	49.5	45.0	83.0	86.3	55.3
PCA	40.9	55.0	65.8	47.9	76.9	77.9	63.5	67.4	33.7	65.5	63.6	61.3	68.9	62.8	44.9	77.5	67.4	57.5	86.7	90.9	63.8
CIE	**51.2**	**69.2**	**70.1**	55.0	82.8	72.8	69.0	**74.2**	39.6	**68.8**	71.8	**70.0**	**71.8**	**66.8**	44.8	85.2	**69.9**	65.4	85.2	92.4	68.9
LCSGM	46.9	58.0	63.6	69.9	87.8	79.8	**71.8**	60.3	44.8	64.3	79.4	57.5	64.4	57.6	52.4	96.1	62.9	**65.8**	94.4	92.0	68.5
EAGM	49.4	62.1	64.6	**75.3**	**90.9**	**80.9**	71.1	61.3	**48.7**	65.9	**87.5**	58.4	66.3	60.1	**56.3**	**97.1**	64.7	60.6	**96.0**	**93.0**	**70.5**

Table 1: Comparison of matching accuracy (%) on Pascal VOC. All results, except ours, are taken from [Yu *et al.*, 2020a; Wang *et al.*, 2020]. Numbers in red indicate the best performance.

To learn the more effective intermediate node representation $\hat{\mathbf{v}}_i$, the edge attention Θ^i adjusts the learning of edge representations $\mathcal{E}_A^i$ adaptively.

The decoder predicts the node labels $\bar{\mathbf{x}}$ and edge labels $\bar{\mathbf{y}}$ by the final graph state, and is implemented by two MLPs ϕ_c^{v} and ϕ_c^{e}.

$$\begin{aligned} \bar{\mathbf{x}} &= \phi_c^{\mathrm{v}}(\mathcal{V}_A),\ \bar{\mathbf{x}} \in \mathbb{R}^{n^2} \\ \bar{\mathbf{y}} &= \phi_c^{\mathrm{e}}(\mathcal{E}_A),\ \bar{\mathbf{y}} \in \mathbb{R}^{m_1 m_2} \end{aligned} \tag{9}$$

4.5 Loss

The ground-truth node-to-node and edge-to-edge correspondences are utilized to guide the training of our EAGM. We firstly define that the two edges $e_i \in \mathbb{E}_1$ and $e_a \in \mathbb{E}_2$ are matched iff their associated node pairs (v_{h_i}, v_{h_a}) and (v_{t_i}, v_{t_a}) are matched respectively. Then we build the ground-truth edge-to-edge correspondence matrix $\mathbf{Y} \in \{0,1\}^{m_1 \times m_2}$ between $\mathbb{G}_1$ and $\mathbb{G}_2$, where each element $\mathbf{Y}_{ia} = 1$ iff $e_i \in \mathbb{E}_1$ corresponds to $e_a \in \mathbb{E}_2$.

To guide the learning of edge representations and edge attention, we measure a binary cross-entropy loss $\mathcal{L}_{\mathrm{e}}$ between the predicted edge labels $\bar{\mathbf{y}}$ and the ground-truth edge binary labels $\mathbf{y} = \mathrm{vec}(\mathbf{Y}) \in \{0,1\}^{m_1 m_2}$.

$$\mathcal{L}_{\mathrm{e}} = -\sum_i^{m_1 m_2} (\mathbf{y}_i \log \bar{\mathbf{y}}_i + (1-\mathbf{y}_i)\log(1-\bar{\mathbf{y}}_i)) \tag{10}$$

Besides, we compute the binary cross-entropy loss $\mathcal{L}_{\mathrm{v}}$ between the predicted node labels $\bar{\mathbf{x}}$ and the ground-truth node labels $\mathbf{x}$, and the one-to-one matching constraint loss $\mathcal{L}_{\mathrm{c}}$,

$$\begin{aligned} \mathcal{L}_{\mathrm{v}} &= -\textstyle\sum_i^{n^2} (\mathbf{x}_i \log \bar{\mathbf{x}}_i + (1-\mathbf{x}_i)\log(1-\bar{\mathbf{x}}_i)) \\ \mathcal{L}_{\mathrm{c}} &= \|\mathbf{B}(\bar{\mathbf{z}} - \mathbf{x})\|_2 \end{aligned} \tag{11}$$

where $\mathbf{B} \in \{0,1\}^{2n \times n^2}$ is an auxiliary matrix, and $\bar{\mathbf{z}} \in \{0,1\}^{n^2}$ is an index vector (please refer to [Wang *et al.*, 2020] for the details of $\mathcal{L}_{\mathrm{v}}$ and $\mathcal{L}_{\mathrm{c}}$).

Finally, we combine the three losses to jointly guide the training of our EAGM,

$$\mathcal{L} = \mathcal{L}_{\mathrm{v}} + \lambda_{\mathrm{e}}\mathcal{L}_{\mathrm{e}} + \lambda_{\mathrm{c}}\mathcal{L}_{\mathrm{c}} \tag{12}$$

where $\lambda_{\mathrm{e}}, \lambda_{\mathrm{c}} \geq 0$ control the relative importance of $\mathcal{L}_{\mathrm{e}}$ and $\mathcal{L}_{\mathrm{c}}$ respectively.

5 Experiments

5.1 Implementation & Evaluation Benchmarks

In our implementation, we build up our EAGM based on the state-of-the-art method [Wang *et al.*, 2020], which learns affinities and the combinatorial solver simultaneously. For all experiments, optimization is achieved via ADAM optimizer [Kingma and Ba, 2015] with initial learning rate 1×10^{-3}, and exponential decaying 2% per 2000 iterations. The CNN features in Sec. 4.1 are the concatenated feature maps of `relu4_2` and `relu5_1` of a VGG16 [Simonyan and Zisserman, 2014] network pre-trained on ImageNet [Deng *et al.*, 2009]. We empirically set the number of convolutional layers $l_1 = 3$ and $l_2 = 10$ in the edge attention module and classification module respectively. The weights in Eq. 12 are set as $\lambda_{\mathrm{e}} = \lambda_{\mathrm{c}} = 0.1$ during training. All experiments are run on a single GTX-1080Ti GPU, and around 25 image pairs are processed per second.

Our EAGM is compared with the state-of-the-arts, including IPFP [Leordeanu *et al.*, 2009], RRWM [Cho *et al.*, 2010], PSM [Egozi *et al.*, 2012], GNCCP [Liu and Qiao, 2013], ABPF [Wang *et al.*, 2018], HARG [Cho *et al.*, 2013], DetGM [Yu *et al.*, 2020b], GMN [Zanfir and Sminchisescu, 2018], PCA [Wang *et al.*, 2019], CIE [Yu *et al.*, 2020a], and LCSGM [Wang *et al.*, 2020] (the last four are deep learning methods). The experiments are performed on two cases with and without outliers, including three benchmarks for keypoint matching: Pascal VOC [Everingham *et al.*, 2010] with Berkeley annotations [Bourdev and Malik, 2009], Willow Object [Cho *et al.*, 2013], and CMU House Sequence [Caetano *et al.*, 2006]. Graph edges are built between keypoints by Delaunay triangulation [Cignoni *et al.*, 1998]. Matching accuracy is evaluated by the number of correctly matched keypoint pairs averaged by the total number of keypoint pairs.

5.2 Evaluation Results without Outliers

Pascal VOC. This dataset contains images with labeled object bounding boxes and keypoints of 20 classes. We follow the standard protocol [Wang *et al.*, 2019]: (1) each object is cropped by its corresponding bounding box and scaled to 256×256, the number of inliers ranges from 6 to 23; (2) 7,020 images for training and 1,682 for testing. We generate 100,000 training samples for each class by randomly choosing two images. As shown in Table 1, EAGM outperforms the state-of-the-arts with the highest matching accuracy 70.5%. This dataset is difficult due to variations in object scale, appearance, pose, and background clutter. Compared with deep learning methods GMN, PCA, CIE, and LCSGM with equivalent edges, our EAGM learns edge attention to adjust the impact of different edges, and achieves more promising per-

Method	car	duck	face	m.bike	w.bottle	Avg.
IPFP	74.8	60.6	98.9	84.0	79.0	79.5
RRWM	86.3	75.5	**100**	94.9	94.3	90.2
PSM	88.0	76.8	**100**	96.4	97.0	91.6
GNCCP	86.4	77.4	**100**	95.6	95.7	91.0
ABPF	88.4	80.1	**100**	96.2	96.7	92.3
HARG	71.9	72.2	93.9	71.4	86.1	79.1
GMN	74.3	82.8	99.3	71.4	76.7	80.9
PCA	84.0	**93.5**	**100**	76.7	96.9	90.2
CIE	82.2	81.2	**100**	90	97.6	90.2
LCSGM	91.2	86.2	**100**	**99.4**	97.9	94.9
EAGM	**94.4**	89.7	**100**	99.3	**99.2**	**96.5**

Table 2: Comparison of matching accuracy (%) on Willow.

formance. The informative and reasonable edge semantic features also enhance the robustness of EAGM.

Willow Object. This dataset includes images of 5 classes, and each contains at least 40 images. Each image is labeled with 10 distinctive keypoints on its target object. Following [Cho *et al.*, 2013; Wang *et al.*, 2019], (1) each image is scaled to 256×256; (2) 20 images from each class are randomly selected for training, and 1000 pairs from the rest of each class are randomly chosen for testing. This dataset is easier than Pascal VOC with keypoints, due to aligned pose of images from the same class and lack of object scale variations. As shown in Table 2, both non-learning methods and learning-based methods achieve acceptable performance. The distinguishing edge features and learned edge attention of EAGM support it to achieve the best performance with accuracy 96.5%.

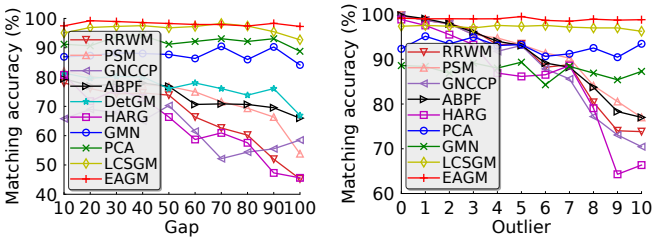
5.3 Evaluation Results with Outliers

Pascal VOC with outliers. To further demonstrate the effectiveness of our EAGM, we perform the comparison experiments on Pascal VOC with outliers. We adopt the pre-trained model in the experiments on Pascal VOC without outliers, but evaluate on the testing samples with outliers. Specifically, we randomly generate outliers in the area 256×256, and add them to the second image of each testing pair. The number of outliers ranges from 1 to 5. As illustrated in Table 3, although both of LCSGM and our EAGM achieve decreasing accuracy with the increasing outliers, our EAGM is more robust as keeping superiority.

CMU House Sequence. This dataset consists of 111 frames of image sequences, which contains the same house object with transformation cross sequence gaps. 30 keypoints are manually labeled and tracked across all frames. Each training sample is formed by two randomly selected frames, and 300,000 random training samples are generated. n_1 ($10 \leq n_1 \leq 30$) inliers are randomly chosen from the 30 keypoints for one frame of a training pair, and the other frame contains $n_2 = 30$ nodes, *i.e.*, there are $30 - n_1$ outliers. We compare the matching accuracy under two cases: (1) following the standard protocol [Cho *et al.*, 2010], *i.e.*, fixing $n_1 = 20, n_2 = 30$ and testing under varying sequence gap

Outlier	0	1	2	3	4	5
LCSGM	68.5	59.3	51.9	45.8	41.4	36.6
EAGM	**70.5**	**60.6**	**53.3**	**47.6**	**42.6**	**38.4**

Table 3: Matching accuracy (%) on Pascal VOC with outliers.



(a) Sampling gap (b) Outlier

Figure 3: Comparison of matching accuracy (%) on CMU House.

$\{10, 20, \ldots, 100\}$ with 560 image pairs (Figure 3(a)), where increasing gaps indicates deeper deformation; (2) fixing the gap as 50 and evaluating under varying $n_1 = 30, 29, \ldots, 20$ (Figure 3(b)), *i.e.*, increasing outliers from 0 to 10. It is obvious that the performance of non-learning methods degrades as the increasing deformation and outliers, but the deep learning methods show relatively stable accuracy. However, only our EAGM is still robust against 100 gaps. On the whole, EAGM outperforms other methods with at least 97.3% accuracy.

These experiments demonstrate that our EAGM with the informative edge semantic features is effective on graph matching with outliers. The fused features also make EAGM more robust, which involves geometric and semantic information jointly.

6 Conclusion

In this paper, we design an edge attention-adaptive graph matching network for graph matching with outliers, named EAGM, to effectively capture the novel edge semantic information that is robust to outliers. EAGM adjusts the impact of each edge on graph matching adaptively, and predicts node and edge binary labels of the associated assignment graph to indicate the matching relations between the two original graphs. Moreover, our novel edge semantic feature not only provides reasonable and independent description for edges, but also distinguishes outliers in the view of their associated edges. The experiments with and without outliers demonstrate that our EAGM achieves promising performance.

Acknowledgments

This work was supported by National Key Research and Development Program of China (No. 2019YFB1406303), National Natural Science Foundation of China (No. 61876003), and Beijing Natural Science Foundation (No. L192024). It is also a research achievement of Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology).

References

[Bourdev and Malik, 2009] Lubomir Bourdev and Jitendra Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *ICCV*, pages 1365–1372, 2009.

[Bresenham, 1965] Jack E Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

[Caetano *et al.*, 2006] Tiberio S Caetano, Terry Caelli, Dale Schuurmans, and Dante Augusto Couto Barone. Graphical models and point pattern matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(10):1646–1663, 2006.

[Caetano *et al.*, 2009] Tibério S Caetano, Julian J McAuley, Li Cheng, Quoc V Le, and Alex J Smola. Learning graph matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(6):1048–1058, 2009.

[Cho *et al.*, 2010] Minsu Cho, Jungmin Lee, and Kyoung Mu Lee. Reweighted random walks for graph matching. In *ECCV*, pages 492–505, 2010.

[Cho *et al.*, 2013] Minsu Cho, Karteek Alahari, and Jean Ponce. Learning graphs to match. In *ICCV*, pages 25–32, 2013.

[Cignoni *et al.*, 1998] Paolo Cignoni, Claudio Montani, and Roberto Scopigno. Dewall: A fast divide and conquer delaunay triangulation algorithm in ed. *Computer-Aided Design*, 30(5):333–341, 1998.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[Egozi *et al.*, 2012] Amir Egozi, Yosi Keller, and Hugo Guterman. A probabilistic approach to spectral graph matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(1):18–27, 2012.

[Everingham *et al.*, 2010] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[Gold and Rangarajan, 1996] Steven Gold and Anand Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(4):377–388, 1996.

[Kingma and Ba, 2015] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[Leordeanu and Hebert, 2005] Marius Leordeanu and Martial Hebert. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, pages 1482–1489, 2005.

[Leordeanu *et al.*, 2009] Marius Leordeanu, Martial Hebert, and Rahul Sukthankar. An integer projected fixed point method for graph matching and map inference. In *NIPS*, pages 1114–1122, 2009.

[Leordeanu *et al.*, 2011] Marius Leordeanu, Andrei Zanfir, and Cristian Sminchisescu. Semi-supervised learning and optimization for hypergraph matching. In *ICCV*, pages 2274–2281, 2011.

[Leordeanu *et al.*, 2012] Marius Leordeanu, Rahul Sukthankar, and Martial Hebert. Unsupervised learning for graph matching. *International Journal of Computer Vision*, 96(1):28–45, 2012.

[Liu and Qiao, 2013] Zhi-Yong Liu and Hong Qiao. Gnccp—graduated nonconvexityand concavity procedure. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(6):1258–1267, 2013.

[Loiola *et al.*, 2007] Eliane Maria Loiola, Nair Maria Maia de Abreu, Paulo Oswaldo Boaventura-Netto, Peter Hahn, and Tania Querido. A survey for the quadratic assignment problem. *European Journal of Operational Research*, 176(2):657–690, 2007.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[Torresani *et al.*, 2008] Lorenzo Torresani, Vladimir Kolmogorov, and Carsten Rother. Feature correspondence via graph matching: Models and global optimization. In *ECCV*, pages 596–609, 2008.

[Wang *et al.*, 2018] Tao Wang, Haibin Ling, Congyan Lang, and Songhe Feng. Graph matching with adaptive and branching path following. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(12):2853–2867, 2018.

[Wang *et al.*, 2019] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Learning combinatorial embedding networks for deep graph matching. In *ICCV*, pages 3056–3065, 2019.

[Wang *et al.*, 2020] Tao Wang, He Liu, Yidong Li, Yi Jin, Xiaohui Hou, and Haibin Ling. Learning combinatorial solver for graph matching. In *CVPR*, pages 7568–7577, 2020.

[Yan *et al.*, 2020] Junchi Yan, Shuang Yang, and Edwin R Hancock. Learning for graph matching and related combinatorial optimization problems. In *IJCAI*, 2020.

[Yu *et al.*, 2020a] Tianshu Yu, Runzhong Wang, Junchi Yan, and Baoxin Li. Learning deep graph matching with channel-independent embedding and hungarian attention. In *ICLR*, 2020.

[Yu *et al.*, 2020b] Tianshu Yu, Junchi Yan, and Baoxin Li. Determinant regularization for gradient-efficient graph matching. In *CVPR*, pages 7123–7132, 2020.

[Zanfir and Sminchisescu, 2018] Andrei Zanfir and Cristian Sminchisescu. Deep learning of graph matching. In *CVPR*, pages 2684–2693, 2018.

[Zhou and De la Torre, 2012] Feng Zhou and Fernando De la Torre. Factorized graph matching. In *CVPR*, pages 127–134, 2012.