Self-supervised Learning of Visual Graph Matching

Chang Liu*, Shaofeng Zhang*, Xiaokang Yang, and Junchi Yan†

MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University
{only-changer, sherrylone, xkyang, yanjunchi}@sjtu.edu.cn

Abstract. Despite the rapid progress made by existing graph matching methods, expensive or even unrealistic node-level correspondence labels are often required. Inspired by recent progress in self-supervised contrastive learning, we propose an end-to-end label-free self-supervised contrastive graph matching framework (SCGM). Unlike in vision tasks like classification and segmentation, where the backbone is often forced to extract object instance-level or pixel-level information, we design an extra objective function at node-level on graph data which also considers both the visual appearance and graph structure by node embedding. Further, we propose two-stage augmentation functions on both raw images and extracted graphs to increase the variance, which has been shown effective in self-supervised learning. We conduct experiments on standard graph matching benchmarks, where our method boosts previous state-of-the-arts under both label-free self-supervised and fine-tune settings. Without the ground truth labels for node matching nor the graph/image-level category information, our proposed framework SCGM outperforms several deep graph matching methods. By proper fine-tuning, SCGM can surpass the state-of-the-art supervised deep graph matching methods. Code is available at https://github.com/Thinklab-SJTU/ThinkMatch-SCGM.

Keywords: self-supervise, graph matching, contrastive learning.

1 Introduction

Graph matching (GM) has been a fundamental NP-hard problem, which has wide spectra of applications from computer vision, pattern recognition, to operational research. Traditional graph matching solvers either for two-graph matching [24,6,51] or multiple-graph matching [50,36,42] are mostly based on specific algorithms designed by human experts. Recently, machine learning-based approaches, especially deep network-based solvers are becoming more and more popular for their flexible data-driven nature and the easiness of making full use of the computing resource of the graphical processing unit (GPU). However, most existing graph matching networks [29,37,32,53,10] as well as the shallow learning models [5] follow a supervised learning framework and thus call for

* : Equal Contribution, † : Correspondence Author

ground truth node-level correspondences for training, which can be tedious or even impossible in practice.

In this paper, we resort to the **S**elf-supervised learning with **C**ontrastive learning and develop a learning network for visual **G**raph **M**atching without manually labeled node correspondence for supervision, called **SCGM**. In particular, we pursuit the most mild setting for label-free self-supervised learning of graph matching: even a single pair of matchable graphs is not required, let alone their node correspondences. In another word, we only assume there are multiple graphs (or images for graph extraction), which can be totally different and unmatchable to each other. In contrast, existing self-supervised GM model [38] requires access to the matchable pairs of graphs, and in fact often multiple such graphs to derive a global consistency or smooth loss. Moreover, existing graph matching methods need to assume that the matchable pairs of graphs always belong to the same category, which requires extra category information (normally only the same or very similar categories of objects can be matched) other than the node correspondences, while neither do we assume to require such category information. The essential difference of our approach is that we first perform data augmentation on the graph (with its visual appearance for images) to generate a set of its copies, such that **we can train neural networks to match any two of the copies from the same origin and the "ground truth" for this training can be easily calculated.** Besides, contrastive learning is performed by minimizing the distance of corresponding node features, that is, the node and its correspondences from the augmented graphs.

In particular, graph matching has been widely applied in computer vision ranging from image retrieval [40], object recognition [34], multi-object tracking [30] to action recognition [12], just to name a few. In this context as considered in this paper, it calls for effective learning of both visual appearances as well as graph structures. Based on this observation, we aim to incorporate both visual data augmentation as well as graph structure mixing to conduct self-supervised learning, and accordingly, both the convolutional neural networks and the graph neural networks are involved. Moreover, the node-level contrastive objective is proposed to help the encoder capture node-level information. In a nutshell, the main contributions are as follows:

1) To overcome the unrealistic labeling requirement for node correspondence bottleneck in existing supervised GM models, we propose a self-supervised based framework for label-free GM called SCGM. To the best of our knowledge, this is the first work for successfully learning a GM network without node correspondence labels (except for the finetuning stage), and even without the access to candidate matchable graphs[1]. In contrast, the (only) self-supervised method GANN [38] assumes the access to matchable graphs. In other words, GANN needs extra category information while SCGM does not.

[1] Typically matchable graphs are those falling into the same category, like the images of different cats, which is also the main setting of this paper. While it can be more general for graphs, e.g. there is partial matching between graphs.

2) We propose a novel two-stage augmentation strategy on both raw images and constructed graphs to increase the variance of the views (copies) for self-supervised learning. Specifically for the graph augmentation, we propose a novel mixing strategy targeted to the previous arts, which helps reduce vicinal risk. To the best of our knowledge, we are the first to apply mixup in graph matching.

3) To better capture node-level information, we propose node-level contrastive objectives to auxiliarily train the encoder. In contrast, previous deep graph matching methods directly use a pretrained encoder (typically from ImageNet), which can be more focused on image-level information.

4) We evaluate our method in both label-free self-supervised and finetune settings in the common graph matching dataset. Results show that our method achieves promising performance. Specifically, our method outperforms GANN [38] with 10% accuracy under the label-free setting and more than 1% accuracy compared to BBGM [31] after finetuning in the Pascal VOC dataset.

2 Related work

2.1 Graph matching

Graph matching has been a standing problem in vision and pattern recognition, which attracted studies in developing both learning-free methods and learning-based models. For the latter which is the focus of this paper, most existing works assume that the node level correspondence is given, and this setting can be unrealistic due to the tedious labeling efforts. Readers are referred to the survey of deep graph matching [52].

We further briefly discuss the related GM works as follows. **1) Supervised models:** Most learning-based GM models fall into the supervised paradigm whereby the label information refers to the node correspondence among two graphs, which in fact require tedious human efforts. For pure graph matching, the graph neural network is introduced in the seminal work [29] without considering the visual information. In the context of GM with vision information as considered in this paper, the seminal work [54] for the first time shows how to end-to-end train a CNN to extract tailored visual features for GM. The subsequent works [37,57,53,58,19] further improve the performance for visual GM by introducing different techniques as well as the use of GNN. In particular, one recent competitive model [31] proposes to integrate classic combinatorial solvers with deep networks together. While NGMv2 [39] directly deals with the general Lawler's QAP form [26], which solves the problem via applying vertex classification on the association graph. **2) Label-free self-supervised models:** There also merge a few label-free self-supervised deep GM networks. In NGMv2 [39], a model is also devised by treating the smoothed matching outputs from pairwise matchings as the pseudo ground truth for self-supervised learning and in GANN [38] the clustering and matching are jointly addressed with a side product of self-supervised GM network training, which achieves state-of-the-art performance and becomes the major competitor to our work.

2.2 Visual correspondence learning

Beyond deep graph matching, unsupervised learning has been an attractive paradigm for general visual correspondence problems including optical flow estimation, object tracking, stereo matching, and etc. **1) Optical flow:** Differing from the supervised flow network [8] that calls for dense pixel correspondence labels, concurrent work [18] propose unsupervised flow network by utilizing the classic photometric consistency over consecutive frames, with smoothness regularization, as the main metric for loss design. Along this direction, further, development is made in subsequent works either by better handling the occlusion (Unflow [27], OccAwareFlow [45]), and depth estimation [59] etc. Dense contrastive learning is also introduced in [44] for visual pre-training with shown benefits to downstream detection and segmentation tasks. While the self-supervising technique is also applied in registration with an explicit mapping function [46]. **2) Object tracking:** Spatio-temporal relevance and appearance consistency are often explored in tracker training as the self-supervision signal, which has shown promising performance in recent trackers [21,48]. In contrast, we focus on sparse correspondence problems with particular care on learning the graph structure.

2.3 Contrastive learning

We generally divide the prior methods into two parts, i.e., task-irrelevant and task-relevant contrastive learning, where the former one aims to learn general information for better transferability. The latter one aims to learn backbone from the task-specific task, which can be used as a pre-trained model. **1) Task-irrelevant methods:** The classical objective of contrastive learning [13,3,16] is instance discrimination, which takes two views augmented from the same images as positives and others as negatives. Moco [16] proposes a memory bank module to store the negatives and they develop two tricks (stop-gradient and momentum update encoders) to prevent collapses [43]. SimCLR [3] proposes a simple, yet effective framework, which takes views of different images in one mini-batch as negatives. BYOL [11] directly discards the negative pairs. Correspondingly, a predictor module and EMA[22] update strategy are developed to prevent collapses. Recent work [4] is also devoted to exploring how to prevent collapses in SSL and hard negative mining by mixup [23,20]. These task-irrelevant methods [55,3] all show their strong transferability to other datasets or downstream tasks. **2) Task-relevant methods:** Task-relevant methods usually design pretext-tasks[7], which are similar to the target task. DenseCL [44] aims to improve the performance of detection and segmentation. Thus, they intuitively propose pixel-level contrastive learning, where for each query pixel in the feature map, they first find the corresponding closest pixel in the other view and maximize the agreement with them. Such a pretext task makes sense since, for segmentation and detection, pixel-wise information is required. PixPro [49] proposes to utilize the position information of each pixel before augmentation and maximize the agreement of the same pixel in the feature map of two views.

3 Preliminaries

InfoNCE based contrastive methods. In general, the objective for InfoNCE [13] attracts representations of views from the same data sample e.g. image, and pushes the negative pairs away, which can be formulated as:

$$\mathcal{L}_{info} = -\mathbb{E}\left[\log \frac{\exp\left(\frac{\mathbf{z}_i^A \cdot \mathbf{z}_j^B}{\tau}\right)}{\sum_j \exp\left(\frac{\mathbf{z}_{j\neq i}^A \cdot \mathbf{z}_j^A}{\tau}\right) + \sum_j \exp\left(\frac{\mathbf{z}_i^A \cdot \mathbf{z}_j^B}{\tau}\right)}\right] \tag{1}$$

where $\mathbf{z}_i^A$ and $\mathbf{z}_i^B$ are the embedding of view A and view B of the i-th sample, respectively. τ is the temperature parameter [3,16]. The above objective function can be divided into two parts, i.e., **alignment** and **uniformity**, which can be expressed as:

$$\begin{aligned}\mathcal{L}_{info} = & \underbrace{\mathbb{E}_{(x,x^+)\sim p_{pos}}\left[\frac{-f(x)^\top f(x^+)}{\tau}\right]}_{\text{alignment}} + \\ & \underbrace{\mathbb{E}_{\substack{(x,x^+)\sim p_{pos} \\ \{x_i^-\}_{i=1}^N \sim p_{data}}}\left[\log\left(e^{\frac{f(x)^\top f(x^+)}{\tau}} + \sum_i e^{\frac{f(x)^\top f(x_i^-)}{\tau}}\right)\right]}_{\text{uniformity}}\end{aligned} \tag{2}$$

where x^+ and x^- are positive samples and negative samples. Previous methods [43,4] have been devoted to exploring the effect of the two terms (alignment and uniformity), where the first term is the key to learning representations, while the second term can prevent collapses (all the data are mapped to a single point).

Mixup in supervised learning. Mixup [56] is an effective regularization with negligible computational overhead. It conducts a linear interpolation of two data instances in both input and label spaces and trains a model by minimizing the loss on the generated data and labels, which can be formulated as:

$$\mathcal{L}_{Mixup}((\mathbf{x}_i, \mathbf{y}_i), (\mathbf{x}_j, \mathbf{y}_j, \lambda)) = \mathcal{L}((1-\lambda)\cdot \mathbf{x}_i + \lambda \cdot \mathbf{x}_j, (1-\lambda)\cdot \mathbf{y}_i + \lambda \cdot \mathbf{y}_j) \tag{3}$$

where λ is the pre-defined mixing rate. $\mathbf{x}_i$ and $\mathbf{y}_i$ mean the i-th data and label.

4 The Proposed SCGM

For matching of graphs extracted from visual images, we perform two-stage augmentation on both raw image data and the extracted graphs. The objective is designated based on the resulting node information on augmented visual graphs. Note that previous self-supervised learning works either perform augmentation on image data [3,16], or on graph data alone [15,14]. While visual graph matching provides a pertinent test-bed for combing the two areas with new self-supervised learning approaches, which have not been studied in graph matching literature.

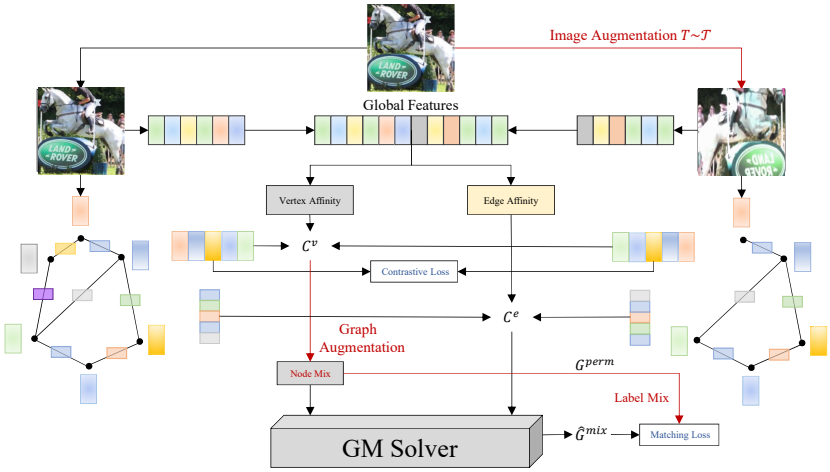


Fig. 1. Framework of the proposed SCGM (self-supervised contrastive graph matching framework). The proposed two-stage augmentation modules are highlighted in red.

4.1 Two-stage augmentation with label mixing

Self-supervised learning methods often benefit from the variance of augmented views, where they force encoders to learn invariance over two views [43]. The intuition behind their success is maximizing the mutual information between two views $\mathcal{I}(\mathbf{Z}^A, \mathbf{Z}^B)$, where $\mathbf{Z}$ is the global semantic features. We do not directly follow the augmentation paradigm of existing methods [3,4,11]. Instead, we design two-stage augmentation on both raw images and constructed graphs to further increase the variance.

Augmentation on raw images. For the first-stage, to make sure each view can cover most of the node pixel in $\mathcal{P}$, we increase the crop size, which potentially increases the mutual information of two views $\mathcal{I}(\mathbf{X}^A, \mathbf{X}^B)$. Hence, we also increase the probability of other augmentations (gray-scale, Gaussian blur) to expand the variance of two views. The details (way and probability) of augmentations will be discussed in the experiment section about hyperparameters.

Augmentations on graph. Augmentation on graph can be more subtle than on image. Many previous self-supervised methods in graph domain have tried different combinations of augmentations [15,14], e.g., edge/node dropping, add edge and re-weighting. However, node dropping and edge dropping are inappropriate for graph matching due to the lack of labeled nodes. Another approach is mixup [33], which not only helps classify data in decision boundary, but also reduces vicinal risk [2]. Given graph $\mathcal{G} = \{\mathbf{X}, \mathbf{A}\}$, where $\mathbf{X} \in \mathbb{R}^{N \times d}$ is the node sets. N is the number of nodes and d is the feature dimension. Then, for nodes in i-th graph, we generate new data by:

$$\mathbf{x}_i = (1 - \lambda) \cdot \mathbf{x}_i + \lambda \cdot \mathbf{x}_j \tag{4}$$

where $\mathbf{x}_i \in \mathbb{R}^{1 \times d}$ is the i-th node. $j \in [1, 2, \cdots, N]$ is a random value. Compared with the mixup in supervised learning, interpolating labels in graph matching calls for more non-trivial scheme design, which is detailed as follows.

Mixing labels. Given two graphs with N_1 and N_2 nodes, we refer them to the number of nodes in view A and view B respectively. We denote the ground truth of

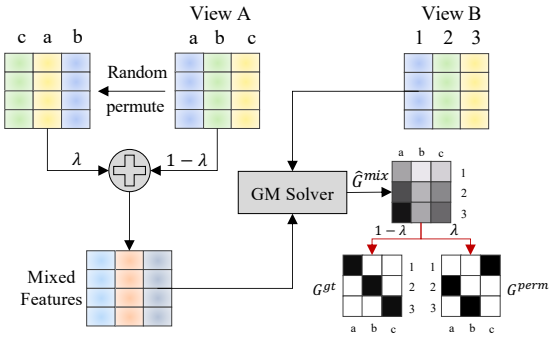


Fig. 2. Illustration of our mixing techniques for deep visual graph matching. Assume the pseudo ground truth of matching pairs are $[a \leftrightarrow 1]$, $[b \leftrightarrow 2]$, $[c \leftrightarrow 3]$, we first permute the node index of view A, then mix it with original feature. Correspondingly, the matching matching relation of $[a \leftrightarrow 1]$ becomes to $(1-\lambda)\cdot[a \leftrightarrow 1] + \lambda \cdot [c \leftrightarrow 1]$.

node matching matrix as $\mathbf{G}^{gt} \in \mathbb{R}^{N_1 \times N_2}$, which can be easily calculated **without supervision** since view A and view B are from the same origin figure. Note that we only mix data of view A. Suppose one generated node $\hat{\mathbf{x}}_i^A = (1-\lambda)\mathbf{x}_i^A + \lambda \mathbf{x}_j^A$, and $\mathbf{x}_i^A$, $\mathbf{x}_j^A$ should match with $\mathbf{x}_i^B$, $\mathbf{x}_j^B$, respectively. Then, the generated $\hat{\mathbf{x}}_i^A$ will match with both $\mathbf{x}_i^B$, $\mathbf{x}_j^B$ nodes with $(1-\lambda)$ and λ weights, respectively. Based on the random mixing index $(i \rightarrow j)$, we can construct a new matching matrix $\mathbf{G}^{perm}$. Finally, we formulate the weighted objective as: (see Fig. 2 as an illustration)

$$\mathcal{L}_{match} = (1-\lambda)\cdot\mathcal{L}(\hat{\mathbf{G}}^{mix}, \mathbf{G}^{gt}) + \lambda \cdot \mathcal{L}(\hat{\mathbf{G}}^{mix}, \mathbf{G}^{perm}) \tag{5}$$

where $\hat{\mathbf{G}}^{mix}$ is the predicted matching matrix after the GM solver and $\mathcal{L}$ is the loss function of graph matching which is required to be compatible and the same with that used by the adopted deep GM backbone in SCGM, such as the permutation loss (for NGM [39]), and the Hamming loss (for BBGM [31]). Note that our method can be applied in both stat-of-the-art deep GM models in an out-of-box manner.

4.2 Node level contrastive objective

A general paradigm of graph matching is to use the pre-trained encoder [17,35] on ImageNet as backbone to obtain feature map, followed by some traditional algorithms [6] to construct graphs. However, this supervised pre-trained encoder only captures global information [31]. To enhance the ability of extracting finer-grained information of encoders, we design a node-level contrastive loss. Given a batch images $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ and a set of node positions $\mathcal{P} = \{p_i\}_{i=1}^N$, we first generate two batch views $\mathbf{X}^A$ and $\mathbf{X}^B$ via a distribution of augmentation $\mathcal{T}$. Then, we feed the two batch views to encoder $f(\cdot, \theta)$ to get feature map $\mathbf{M}^A, \mathbf{M}^B \in \mathbb{R}^{C \times H \times W}$, respectively. In line with previous contrastive methods [3,49], we add a pixel-level projection head after the encoder:

$$\mathbf{C}^A = \mathbf{W}_b(\sigma(\mathbf{W}_a\mathbf{M}^A)),\ \mathbf{C}^B = \mathbf{W}_b(\sigma(\mathbf{W}_a\mathbf{M}^B)) \tag{6}$$

where σ is the activation function. Instead of directly contrasting all the pixels in feature map like PixPro [49], which has been shown harmful to learn encoders [47], we

utilize the prior of node positions. For each batch view, we select the node index by:

$$\mathcal{P}_i^{A(B)} = \{(i,h,w)|(h,w) \in p_i\} \tag{7}$$

where $\mathcal{P}_i^{A(B)}$ is the set of node index of the i-th image. Since view A and view B may cover different nodes (due to random cropping), we select the intersection set by:

$$\mathcal{S}_i^{A(B)} = \{(i,h,w)|(i,h,w) \in \mathcal{P}_i^A, (i,h,w) \in \mathcal{P}_i^B\} \tag{8}$$

Then, the node-level contrastive loss of view A (similar for view B) is formulated as:

$$\mathcal{L}_{node}^A = -\mathbb{E}_{ihw} \log \frac{\exp(\mathbf{C}_{ihw}^A \cdot \mathbf{C}_{ihw}^B)}{\exp(\mathbf{C}_{ihw}^A \cdot \mathbf{C}_{ihw}^B) + \mathcal{N}} \tag{9}$$

where $(i,h,w) \in \mathcal{S}$ is the h-th row, w-th column pixel vector in feature map of i-th image. $\mathcal{N}$ is the negative part, which can be written as:

$$\mathcal{N} = \sum_{i'h'w'} \exp(\mathbf{C}_{ihw}^A \mathbf{C}_{i'h'w'}^B) + \exp(\mathbf{C}_{ihw}^A \mathbf{C}_{i'h'w'}^A) \tag{10}$$

where $(i',h',w') \in \mathcal{S}$ and $(i',h',w') \neq (i,h,w)$. The contrastive loss is the average of the two views:

$$\mathcal{L}_{node} = \mathcal{L}_{node}^A + \mathcal{L}_{node}^B \tag{11}$$

By the node-level contrastive loss, the encoder avoids contrasting each pixel, where some pixels are background (meaningless even harmful [47]). The overall objective function simply combines the contrastive loss and matching loss:

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{node} + \mathcal{L}_{match} \tag{12}$$

where α is a pre-defined hyper-parameter about the ratio of contrastive loss.

5 Experiments

We conduct experiments on a popular open-source graph matching framework called ThinkMatch[2], which provides the implementation of many deep graph matching methods with protocol to popular datasets, for the purpose of better **reproducibility** and more fair comparison. We will release the code of our method and the configurations of all compared methods used in our experiments after publishing the paper. Experiments run on Intel(R) Xeon(R) E5-2678 v3 CPUs (2.50GHz) and 8 GTX 2080 Ti GPUs.

5.1 Protocols

Hyperparameters for reproducibility The hyperparameters in our model can be basically divided into two parts: data augmentation and self-supervised learning. For the data augmentation part, the range of the cropping rate is $[0.3, 1.0]$, the range of the scale ratio is $[0.75, 1.33]$, the flip rate is 0.05 for vertical and 0.25 for horizontal, the color jitters parameters are $[0.4, 0.4, 0.4, 0.1]$ with the rate of 0.8, the rate of gray-scale is 0.2, the sigma of the Gaussian blur is $[0.1, 2.0]$ and the rate of Gaussian blur is 25. For the self-supervised learning part, the mix rate λ is set as 0.4 and the contrastive loss ratio α is set as 0.2. For the other hyperparameters of graph matching, we remain them as the same with the original backbone deep graph matching methods in ThinkMatch.

[2] https://github.com/Thinklab-SJTU/ThinkMatch

Datasets We mainly use two real-world image datasets in our experiments:

1) Pascal VOC dataset [1] consists of 20 classes with keypoint labels on the natural images. The instances of this dataset vary by scale, pose, and illumination. The number of keypoints in each image ranges from 6 to 23, which is relatively large in the common GM dataset. We follow the standard data preparation procedure of [37]. Each object is cropped to its bounding box and scaled to 256 * 256 px.

2) Willow Object dataset [5] contains 256 images from 5 categories, of which each category is represented with at least 40 real-world images. Please note that the instances in the same class share 10 distinctive keypoints, which means there are no outliers. We follow the standard train-test split protocol in existing works [31,39].

Compared methods We compare with various baselines, including learning-free, supervised learning, self-supervised learning graph matching methods. Please note SCGM is a pure label-free self-supervised learning framework, therefore, it is unfair to compare our method with existing supervised learning deep graph methods.

Learning-free methods: **IFPF** [24] iteratively improves the solution via integer projection, given a continuous or discrete solution. **RRWM** [6] proposes a random walk view of the graph matching problem, with a re-weighted jump on graph matching. **PSM** [9] improves the spectral graph matching algorithm through a probabilistic view, which presents a probabilistic interpretation of the spectral relaxation scheme. **GNCCP** [25] follows the convex-concave path-following algorithm without involving the relaxation explicitly. **BPF** [41] designs a branch switching technique to deal with the singular point issue in the path following algorithms.

Supervised methods: We compare our proposed method with existing popular deep graph matching methods: **GMN** [54], **PCA** [37], **NGM** [39], **IPCA** [37], **CIE** [53], **BBGM** [31], **NGMv2** [39], and the last two methods BBGM and NGMv2 are the state-of-the-art methods. Please note that the reported results of BBGM are slightly worse than the results in their paper since the protocol in ThinkMatch does not filter out keypoints out of the bounding box, as discussed in [39].

Self-supervised methods: **GANN** [38] is the only self-supervised method of deep graph matching. We compare SCGM with the two-graph matching version of GANN. Please note that GANN still uses more information than SCGM, since GANN requires category information (two graphs belong to the same category) while SCGM does not. In fact, as discussed in the method section, learning of SCGM is still feasible even the graphs are all from different categories.

Evaluation metrics For testing, given the pair-wise images sampled from the dataset, graph matching methods predict a permutation matrix $\mathbf{X}^{pred} \in \mathbb{R}^{n_1 \times n_2}$. Based on $\mathbf{X}^{pred}$ and ground truth $\mathbf{X}^{gt} \in \mathbb{R}^{n_1 \times n_2}$ (note that $\sum \mathbf{X}^{gt}$ equals the number of inliers, since the rows and columns of outliers are always zeros.). Two popular evaluation metrics are used: accuracy and F1 score in the experiments on the real image datasets.

$$\begin{aligned} \textbf{Precision} &= \sum \left(\mathbf{X}^{pred} * \mathbf{X}^{gt}\right) / \sum \mathbf{X}^{pred} \\ \textbf{Accuracy } (\text{ Recall }) &= \sum \left(\mathbf{X}^{pred} * \mathbf{X}^{gt}\right) / \sum \mathbf{X}^{gt} \\ \textbf{F1 score} &= (2 \cdot \text{Recall} \cdot \text{Precision}) / (\text{Recall} + \text{Precision}) \end{aligned} \tag{13}$$

where $*$ denotes element-wise multiplication. We use accuracy instead of F1 score in the non-outlier (intersection) settings since there is no need for calculating the F1 score.

Table 1. Average performance across all the objects w.r.t accuracy (%) on Pascal VOC with standard intersection filtering. The methods are grouped by supervised learning, learning-free, and self-supervised learning from top to bottom (see "Label").

Method	Label	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv	Avg
GMN [54]	✓	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 [37]	✓	49.8	61.9	65.3	57.2	78.8	75.6	64.7	69.7	41.6	63.4	50.7	67.1	66.7	61.6	44.5	81.2	67.8	59.2	78.5	90.4	64.8
NGM [39]	✓	50.1	63.5	57.9	53.4	79.8	77.1	73.6	68.2	41.1	66.4	40.8	60.3	61.9	63.5	45.6	77.1	69.3	65.5	79.2	88.2	64.1
IPCA [37]	✓	53.8	66.2	67.1	61.2	80.4	75.3	72.6	72.5	44.6	65.2	54.3	67.2	67.9	64.2	47.9	84.4	70.8	64.0	83.8	90.8	67.7
CIE [53]	✓	52.5	68.6	70.2	57.1	82.1	77.0	70.7	73.1	43.8	69.9	62.4	70.2	70.3	66.4	47.6	85.3	71.7	64.0	83.8	91.7	68.9
BBGM [31]	✓	**61.9**	71.1	**79.7**	**79.0**	**87.4**	**94.0**	**89.5**	**80.2**	**56.8**	**79.1**	64.6	**78.9**	76.2	75.1	**65.2**	**98.2**	77.3	**77.0**	94.9	**93.9**	79.0
NGMv2 [39]	✓	61.8	**71.2**	77.6	78.8	87.3	93.6	87.7	79.8	55.4	77.8	**89.5**	78.8	**80.1**	**79.2**	62.6	97.7	**77.7**	75.7	**96.7**	93.2	**80.1**
IPFP [24]	✗	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	36.7
RRWM [6]	✗	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	48.4	70.5	41.3	43.0
PSM [9]	✗	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	43.0
GNCCP [25]	✗	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
BPF [41]	✗	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
GANN [38] (two-graph)	✗	19.2	20.5	24.1	27.9	30.8	50.9	36.4	22.3	24.4	23.2	39.8	21.7	20.5	23.9	15.8	42.2	29.8	17.1	61.8	78.0	31.5
SCGM + BBGM [31]	✗	**37.6**	**49.9**	**54.8**	54.5	**65.6**	**56.4**	60.6	**52.3**	36.8	**51.4**	**50.4**	**47.2**	**59.4**	51.2	38.3	**91.3**	**59.3**	52.7	**83.1**	**88.4**	**57.1**
SCGM + NGMv2 [39]	✗	34.3	48.2	51.0	52.2	63.3	56.0	**62.0**	50.1	38.5	49.9	39.9	46.2	54.8	**52.1**	37.4	82.3	56.8	51.4	80.2	78.8	54.3

Table 2. Average performance across all the objects w.r.t F1 score (%) (the higher the better) on Pascal VOC without filtering. It denotes that all graphs can contain outliers. The methods are grouped by supervised learning and self-supervised learning from top to bottom (see "Label").

Method	Label	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv	Avg
GMN [54]	✓	28.0	55.0	33.1	27.0	79.0	52.0	26.0	40.2	28.4	36.0	29.8	33.7	39.4	43.0	22.1	71.8	30.8	25.9	58.8	78.0	41.9
PCA [37]	✓	27.5	56.5	36.6	27.7	77.8	49.2	23.9	42.3	27.4	38.2	**38.7**	36.5	39.3	42.8	25.6	74.3	32.6	24.7	51.5	74.3	42.4
BBGM [31]	✓	37.0	65.0	**50.1**	34.8	86.7	**67.1**	25.4	**56.1**	**41.6**	**58.0**	38.3	**52.9**	**55.0**	**66.6**	30.7	**96.5**	**49.5**	36.4	**76.4**	**83.1**	**55.4**
NGMv2 [39]	✓	**39.4**	**66.1**	49.6	**41.0**	**87.9**	59.6	**46.3**	52.9	39.5	53.1	31.0	49.7	51.0	60.3	**42.2**	91.5	41.3	**37.1**	65.7	74.8	54.0
GANN [38] (two-graph)	✗	12.6	19.5	16.6	18.5	41.1	32.4	19.3	12.3	**24.3**	17.2	**38.0**	12.2	15.9	18.2	19.4	35.5	14.8	15.4	41.5	60.8	24.3
SCGM + BBGM [31]	✗	18.9	**43.5**	**32.3**	**29.5**	**64.4**	**36.1**	20.3	**28.8**	23.9	**28.8**	23.7	**23.3**	**31.4**	33.4	21.1	**83.2**	**25.5**	**27.0**	**49.4**	**72.9**	**35.9**
SCGM + NGMv2 [39]	✗	**19.7**	42.2	29.5	23.9	62.3	35.2	**21.2**	27.3	23.5	25.9	22.8	22.7	29.7	**35.7**	**21.3**	67.5	24.6	21.6	44.4	65.6	33.3

5.2 Experiments on the Pascal VOC dataset

We first test our proposed self-supervised learning framework SCGM on the Pascal VOC dataset, which is a popular and relatively challenging dataset in the graph matching area. There are two common experiment settings in this dataset for pair-wise graph matching: 1) intersection ($\therefore \cap \therefore$) denotes the keypoints of each graph are filtered that only the keypoints present in both source and target image are preserved for the matching. 2) all keypoints ($\therefore \cup \therefore$) denote the keypoints are not filtered, which means there exist outliers in both the pair-wise graphs and the number of keypoints in two graphs may be different. In general, all-keypoints settings are harder than the intersection settings due to the outliers. We conduct the experiments on both intersection and all keypoints settings. The results are shown in Table 1 and Table 2. We adapt SCGM on BBGM and NGMv2, which are the two state-of-the-art deep graph matching methods.

In the intersection experiments, we compare our SCGM with supervised learning graph matching methods, traditional learning-free solvers, and self-supervised GANN.

Table 3. Mean accuracy (%) across all the objects in the Willow Object dataset. The methods are grouped by supervised learning and self-supervised learning. Please note that our SCGM is trained in the Pascal VOC dataset and finetuned in Willow Object.

Method	Label	car	duck	face	motorbike	winebottle	Average
GMN [54]	✓	67.9	76.7	99.8	69.2	83.1	79.3
PCA [37]	✓	87.6	83.6	**100.0**	77.6	88.4	87.4
NGM [39]	✓	84.2	77.6	99.4	76.8	88.3	85.3
IPCA [37]	✓	90.4	88.6	**100.0**	83.0	88.3	90.1
CIE [53]	✓	85.8	82.1	99.9	88.4	88.7	89.0
BBGM [31]	✓	96.8	89.9	**100.0**	**99.8**	**99.4**	97.2
NGMv2 [39]	✓	**97.4**	**93.4**	**100.0**	98.6	98.3	**97.5**
GANN [38] (two-graph)	✗	85.4	**89.8**	**100.0**	88.6	96.4	**92.0**
SCGM + BBGM [31]	✗	**91.3**	73.0	**100.0**	95.6	**96.6**	91.3
SCGM + NGMv2 [39]	✗	91.2	74.4	99.7	**96.8**	92.7	91.0

From the perspective of information required, SCGM uses the same information as the learning-free solvers since we do not require any ground truth permutations. Please note that even GANN requires more information than SCGM since GANN needs to know two graphs belong to the same category while SCGM does not.

As shown in Table 1, our SCGM reaches the highest accuracy 57.1% among both the learning-free and self-supervised methods, which is even higher than the supervised learning GMN (55.3%). It is promising since SCGM does not require any ground truth matchings as labels, while GMN relies on such information for supervision.

In all keypoints experiments, we remove the traditional solvers since this setting is too hard for them and compare our SCGM with other deep graph matching baselines. As Table 2 shows, our SCGM surpasses GANN for more than 10%, with less information used than GANN. For illustration, we visualize some images from the data to show the matching results of SCGM in Fig. 5 (later in this section).

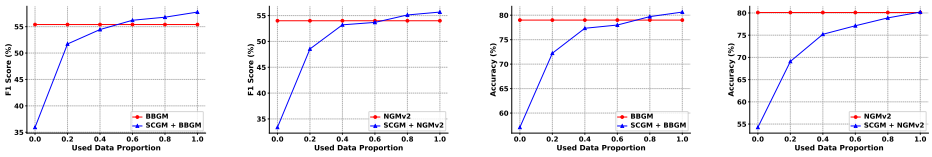
The experiments on Pascal VOC have verified SCGM. We notice that "SCGM + BBGM" is always better than "SCGM + NGMv2", but the performance of the original NGMv2 is better than BBGM. We think it is because NGMv2 uses a GNN as its back-end solver but BBGM uses a traditional one, making BBGM's training easier.

5.3 Experiments on the Willow Object dataset

Then, we test SCGM on the Willow Object dataset, which is a traditional graph matching dataset. However, we have to face the issue that there are only 256 images in the Willow Object dataset, which is clearly not enough for our self-supervised training[3]. Therefore, we use the model trained by SCGM on the Pascal VOC dataset as a prior and utilize few data from the Willow Object dataset to finetune the pre-trained model.

The results are shown in Table 3. We can see that SCGM outperforms many supervised learning methods including GMN, PCA, NGM, IPCA, and CIE. When compared with GANN, SCGM can reach better accuracy in the car, face, motorbike, and winebottle, except for the duck category. It is reasonable since **there are no duck images in the Pascal VOC dataset**, and it turns out SCGM is not trained well for the duck

[3] Unlike ours, GANN [38] utilizes another direction of self-supervised learning, which does not require the amount of data for pre-training.



(a) BBGM; all keypoints (b) NGMv2; all keypoints (c) BBGM; intersection (d) NGMv2; intersection

Fig. 3. Results after finetuning with part of the data in the Pascal VOC dataset for our proposed self-supervised learning framework SCGM combined with BBGM and NGMv2. The settings of both intersection and all keypoints are reported.

Table 4. Ablation test about the sensitivity of hyper-parameters, as conducted on the Pascal VOC dataset with all keypoints. The backbone of SCGM is BBGM.

	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
α	34.0%	35.6%	**35.9%**	35.1%	33.8%	33.0%	33.7%	33.9%	34.3%	34.2%	33.2%
	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
λ	33.7%	35.0%	35.7%	35.3%	**35.9%**	35.4%	25.8%	24.0%	21.8%	22.8%	21.6%

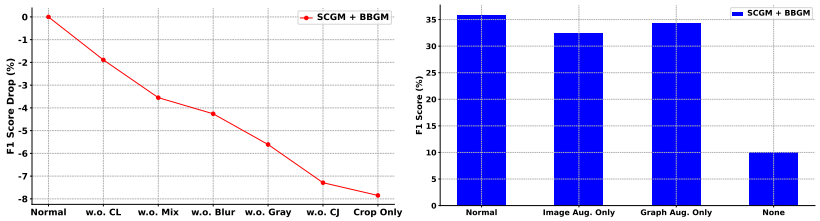


Fig. 4. F1 score drop by removing components from vanilla SCGM with BBGM as the backbone solver. **(left)**: Removing the components one by one; See x-axis from left to right by incrementally removing: contrastive loss, label mixing, Gaussian blur, gray-scale, color jitters, and finally, only crop remains. **(right)**: Removing the components categorized by graph augmentation and image augmentation. The x-axis denotes the normal SCGM, SCGM with only image augmentation, SCGM with only graph augmentation, and SCGM with no augmentation.

category. If we compare the other four categories, the average accuracy of "SCGM + BBGM" is 95.9%, which is 3% higher than GANN (92.6%).

To sum up, in the Willow Object dataset, our SCGM framework also reaches a competitive performance. It also shows that SCGM has some transferability and can generalize to other datasets to some extent.

5.4 Performance after finetuning with initialization

Except for pure self-supervised learning, our framework SCGM can also provide a better parameter initialization for existing graph matching methods. The self-supervised

Table 5. Results after finetuning with part of the data from VOC. SCGM is pretrained on VOC and SPair-71k respectively, with BBGM as the backbone solver.

Method	Pretrained Dataset	Used Data Portion for Finetuning				
		20%	40%	60%	80%	100%
BBGM	-	-	-	-	-	55.40%
SCGM + BBGM	Pascal VOC	**51.72%**	54.45%	56.21%	56.79%	57.76%
SCGM + BBGM	SPair-71k	50.42%	**55.89%**	**56.57%**	**57.91%**	**59.52%**

training procedure in SCGM can be regarded as finding a better-pretrained initialization. Therefore, we can finetune the model pretrained by SCGM in Table 1 and Table 2. To test initialization, we gradually add the amount of labeled data from 0% to 100%.

The results of finetuning in the intersection and all keypoints setting are shown in Fig. 3. We plot the results of normal training with all labeled data as a horizontal red line. The accuracy of finetuning with our initialization increases fast as more data is used. In all keypoints settings, finetuning with initialization can overhead the performance of the normal training with about 60% labeled data usage. Similar to previous experiments, SCGM is better combined with BBGM than with NGMv2. When finetune with all labeled data, our proposed framework can always reach the better accuracy: (a) 57.8% *vs.* 55.3%; (b) 55.9% *vs.* 54.0%; (c) 80.6% *vs.* 79.0%; (d) 80.2% *vs.* 80.1%; In the commonly used intersection settings, SCGM improves BBGM's accuracy by 1%.

We believe these results are promising since what we have done is only changing the initialization. It shows that our self-supervised framework SCGM can provide a better initialization for the graph matching problem.

5.5 Ablation studies

The study focuses on two points: sensitivity of hyper-parameters used in SCGM, and impacts of different components of SCGM. First, we conduct the hyper-parameters experiments to show the impact of λ (the mix rate) and α (the contrastive loss rate). The results are shown in Table 4. We keep one hyper-parameter unchanged and vary the value of the other one to see its sensitivity. In the normal setting of SCGM, we use $\alpha = 0.2$ and $\lambda = 0.4$ as the bold value in the table. We can see our proposed SCGM is not sensitive to α, but sensitive to λ when $\lambda > 0.5$. We conjecture it is because $\lambda > 0.5$ denotes that the original features are no longer the main component in the mixed features, which may cause the model to learn the wrong features. Then, we conduct experiments to see the effect of removing some components in SCGM. As Fig. 4 shows, we remove some of the key components of SCGM. In the left figure, we remove the components of SCGM one by one, from contrastive loss to label mixing, to Gaussian blur, to gray-scale, to color jitters, and at last only the crop of images remains. We can see that the removal of contrastive loss and label mixing lead to a relatively larger drop of performance. Therefore, the importance of contrastive loss and label mixing in SCGM is more significant to some extent, and the effectiveness of such design is proved. In the right figure, we divide the augmentations into graph augmentation and image augmentation, corresponding to our two-stage augmentation mentioned in Sec. 4. We can see that the performance of only using graph augmentation is better than the performance of only using image augmentation, and both are better than that with no augmentation. It turns out that the design of such two-stage augmentation in SCGM is necessary in better self-supervised learning in the graph matching area.

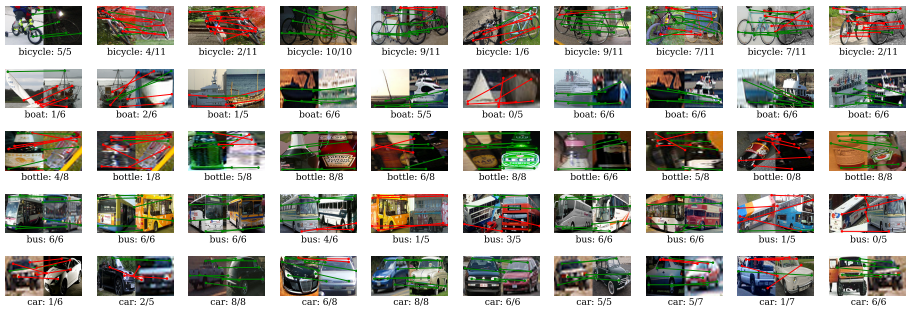


Fig. 5. Illustration of the matching results by SCGM combined with BBGM on the Pascal VOC. Green and red lines represent correct and incorrect node matchings respectively. The subtitle of each figure shows the correct matched/total keypoints count.

5.6 Ability of transfer learning

In the aforementioned experiments, we evaluate our method by the semi-supervised settings, which are also commonly used in BYOL [11]. In this subsection, we consider a new experiment scenario: pretraining SCGM on a larger dataset and then testing (transferring) on the Pascal VOC dataset. We think it is more useful to train the neural networks from a large-scale dataset and transfer the neural networks to downstream tasks, as a typical setting of self-supervised learning. Therefore, we conduct the experiments of transferring pretrained SCGM on Spair-71k [28] to the Pascal VOC dataset. Spair-71k is a large dataset in the graph matching area, which contains 70,958 images. We use the same training protocol as Section 4.4 and the results are shown in Table 5. The first two rows correspond to the results from Fig. 3(a), and the last row is the new experiments of pretraining from SPair-71k and testing (transferring) to VOC. It shows that pretrained on the larger dataset does improve the performance of our proposed SCGM, since SCGM performs much better when pretrained on the SPair-71k dataset.

6 Conclusion

In this paper, we have presented a self-supervised constrictive learning approach for visual graph matching, whereby neither node level correspondence label nor graph level class label is needed. The model involves contrastive learning with both convolution networks and graph neural networks. Moreover, we further generalize our model to the case that part of the graphs from the dataset are labeled by ground truth node correspondence. Comprehensive experimental results on natural images show the efficacy of our method, which outperforms the existing self-supervised graph matching methods. In the future, we aim to adapt SCGM to more complex settings, e.g. partial matching.

Acknowledgements This work was supported in part by by National Key Research and Development Program of China (2020AAA0107600), National Science of Foundation China (61972250, 72061127003), and Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102).

References

1. Bourdev, L., Malik, J.: Poselets: Body part detectors trained using 3d human pose annotations. In: 2009 IEEE 12th International Conference on Computer Vision. pp. 1365–1372. IEEE (2009) 9
2. Chapelle, O., Weston, J., Bottou, L., Vapnik, V.: Vicinal risk minimization. Advances in neural information processing systems pp. 416–422 (2001) 6
3. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International conference on machine learning. pp. 1597–1607. PMLR (2020) 4, 5, 6, 7
4. Chen, X., He, K.: Exploring simple siamese representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15750–15758 (2021) 4, 5, 6
5. Cho, M., Alahari, K., Ponce, J.: Learning graphs to match. In: ICCV. pp. 25–32 (2013) 1, 9
6. Cho, M., Lee, J., Lee, K.M.: Reweighted random walks for graph matching. In: European conference on Computer vision. pp. 492–505. Springer (2010) 1, 7, 9, 10
7. Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: ICCV. pp. 1422–1430 (2015) 4
8. Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. In: ICCV. pp. 2758–2766 (2015) 4
9. Egozi, A., Keller, Y., Guterman, H.: A probabilistic approach to spectral graph matching. TPAMI **35**, 18–27 (2013) 9, 10
10. Fey, M., Lenssen, J.E., Morris, C., Masci, J., Kriege, N.M.: Deep graph matching consensus. In: ICLR (2020) 1
11. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P.H., Buchatskaya, E., Doersch, C., Pires, B.A., Guo, Z.D., Azar, M.G., et al.: Bootstrap your own latent: A new approach to self-supervised learning. arXiv preprint arXiv:2006.07733 (2020) 4, 6, 14
12. Guo, M., Chou, E., Huang, D.A., Song, S., Yeung, S., Fei-Fei, L.: Neural graph matching networks for fewshot 3d action recognition. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 653–669 (2018) 2
13. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). vol. 2, pp. 1735–1742. IEEE (2006) 4, 5
14. Hafidi, H., Ghogho, M., Ciblat, P., Swami, A.: Graphcl: Contrastive self-supervised learning of graph representations. arXiv preprint arXiv:2007.08025 (2020) 5, 6
15. Hassani, K., Khasahmadi, A.H.: Contrastive multi-view representation learning on graphs. In: International Conference on Machine Learning. pp. 4116–4126. PMLR (2020) 5, 6
16. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9729–9738 (2020) 4, 5
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016) 7
18. Jason, J.Y., Harley, A.W., Derpanis, K.G.: Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In: ECCV. pp. 3–10. Springer (2016) 4

19. Jiang, B., Sun, P., Luo, B.: Glmnet: Graph learning-matching convolutional networks for feature matching. Pattern Recognition **121**, 108167 (2022) 3
20. Kalantidis, Y., Sariyildiz, M.B., Pion, N., Weinzaepfel, P., Larlus, D.: Hard negative mixing for contrastive learning. arXiv preprint arXiv:2010.01028 (2020) 4
21. Lai, Z., Lu, E., Xie, W.: Mast: A memory-augmented self-supervised tracker. In: CVPR (2020) 4
22. Lawrance, A., Lewis, P.: An exponential moving-average sequence and point process (ema1). Journal of Applied Probability **14**(1), 98–113 (1977) 4
23. Lee, K., Zhu, Y., Sohn, K., Li, C.L., Shin, J., Lee, H.: I-mix: A domain-agnostic strategy for contrastive representation learning. arXiv preprint arXiv:2010.08887 (2020) 4
24. Leordeanu, M., Hebert, M., Sukthankar, R.: An integer projected fixed point method for graph matching and map inference. In: NIPS (2009) 1, 9, 10
25. Liu, Z.Y., Qiao, H., Xu, L.: An extended path following algorithm for graph-matching problem. TPAMI **34**(7), 1451–1456 (2012) 9, 10
26. Loiola, E.M., de Abreu, N.M.M., Boaventura-Netto, P.O., Hahn, P., Querido, T.: A survey for the quadratic assignment problem. EJOR pp. 657–90 (2007) 3
27. Meister, S., Hur, J., Roth, S.: Unflow: Unsupervised learning of optical flow with a bidirectional census loss. arXiv preprint arXiv:1711.07837 (2017) 4
28. Min, J., Lee, J., Ponce, J., Cho, M.: Spair-71k: A large-scale benchmark for semantic correspondence. arXiv preprint arXiv:1908.10543 (2019) 14
29. Nowak, A., Villar, S., Bandeira, A., Bruna, J.: Revised note on learning quadratic assignment with graph neural networks. In: DSW (2018) 1, 3
30. Pei, W.Y., Yang, C., Meng, L., Hou, J.B., Tian, S., Yin, X.C.: Scene video text tracking with graph matching. IEEE Access **6**, 19419–19426 (2018) 2
31. Rolínek, M., Swoboda, P., Zietlow, D., Paulus, A., Musil, V., Martius, G.: Deep graph matching via blackbox differentiation of combinatorial solvers. In: European Conference on Computer Vision. pp. 407–424. Springer (2020) 3, 7, 9, 10, 11
32. Sarlin, P.E., DeTone, D., Malisiewicz, T., Rabinovich, A.: Superglue: Learning feature matching with graph neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4938–4947 (2020) 1
33. Shim, D., Kim, H.J.: Learning a domain-agnostic visual representation for autonomous driving via contrastive loss. arXiv preprint arXiv:2103.05902 (2021) 6
34. Shokoufandeh, A., Keselman, Y., Demirci, M.F., Macrini, D., Dickinson, S.: Many-to-many feature matching in object recognition: a review of three approaches. Computer Vision, IET **6**(6), 500–513 (2012) 2
35. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014) 7
36. SOLÉ-RIBALTA, A., SERRATOSA, F.: Graduated assignment algorithm for multiple graph matching based on a common labeling. International Journal of Pattern Recognition and Artificial Intelligence **27**(01), 1350001 (2013) 1
37. Wang, R., Yan, J., Yang, X.: Learning combinatorial embedding networks for deep graph matching. In: ICCV. pp. 3056–3065 (2019) 1, 3, 9, 10, 11
38. Wang, R., Yan, J., Yang, X.: Graduated assignment for joint multi-graph matching and clustering with application to unsupervised graph matching network learning. In: NeurIPS (2020) 2, 3, 9, 10, 11
39. Wang, R., Yan, J., Yang, X.: Neural graph matching network: Learning lawler's quadratic assignment problem with extension to hypergraph and multiple-graph matching. TPAMI (2021) 3, 7, 9, 10, 11

40. Wang, S., Wang, R., Yao, Z., Shan, S., Chen, X.: Cross-modal scene graph matching for relationship-aware image-text retrieval. In: The IEEE Winter Conference on Applications of Computer Vision. pp. 1508–1517 (2020) 2
41. Wang, T., Ling, H., Lang, C., Feng, S.: Graph matching with adaptive and branching path following. IEEE TPAMI (2017) 9, 10
42. Wang, T., Jiang, Z., Yan, J.: Clustering-aware multiple graph matching via decayed pairwise matching composition. AAAI (2020) 1
43. Wang, T., Isola, P.: Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In: International Conference on Machine Learning. pp. 9929–9939. PMLR (2020) 4, 5, 6
44. Wang, X., Zhang, R., Shen, C., Kong, T., Li, L.: Dense contrastive learning for self-supervised visual pre-training. In: ICCV (2021) 4
45. Wang, Y., Yang, Y., Yang, Z., Zhao, L., Xu, W.: Occlusion aware unsupervised learning of optical flow. In: CVPR. pp. 4884–4893 (2018) 4
46. Wang, Y., Solomon, J.M.: Prnet: Self-supervised learning for partial-to-partial registration. arXiv preprint arXiv:1910.12240 (2019) 4
47. Wang, Z., Li, Q., Zhang, G., Wan, P., Zheng, W., Wang, N., Gong, M., Liu, T.: Exploring set similarity for dense self-supervised representation learning. arXiv preprint arXiv:2107.08712 (2021) 7, 8
48. Wu, Q., Wan, J., Chan, A.B.: Progressive unsupervised learning for visual object tracking. In: CVPR (2021) 4
49. Xie, Z., Lin, Y., Zhang, Z., Cao, Y., Lin, S., Hu, H.: Propagate yourself: Exploring pixel-level consistency for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16684–16693 (2021) 4, 7
50. Yan, J., Tian, Y., Zha, H., Yang, X., Zhang, Y., Chu, S.: Joint optimization for consistent multiple graph matching. In: ICCV (2013) 1
51. Yan, J., Zhang, C., Zha, H., Liu, W., Yang, X., Chu, S.: Discrete hyper-graph matching. In: CVPR (2015) 1
52. Yan, J., Yang, S., Hancock, E.: Learning graph matching and related combinatorial optimization problems. In: IJCAI (2020) 3
53. Yu, T., Wang, R., Yan, J., Li, B.: Learning deep graph matching with channel-independent embedding and hungarian attention. In: ICLR (2019) 1, 3, 9, 10, 11
54. Zanfir, A., Sminchisescu, C.: Deep learning of graph matching. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2684–2693 (2018) 3, 9, 10, 11
55. Zbontar, J., Jing, L., Misra, I., LeCun, Y., Deny, S.: Barlow twins: Self-supervised learning via redundancy reduction. arXiv preprint arXiv:2103.03230 (2021) 4
56. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412 (2017) 5
57. Zhang, Z., Lee, W.S.: Deep graphical feature learning for the feature matching problem. In: ICCV. pp. 5087–5096 (2019) 3
58. Zhao, K., Tu, S., Xu, L.: Ia-gm: A deep bidirectional learning method for graph matching. In: AAAI (2021) 3
59. Zou, Y., Luo, Z., Huang, J.B.: Df-net: Unsupervised joint learning of depth and flow using cross-task consistency. In: ECCV. pp. 38–55. Springer (2018) 4