

DEEP GRAPH MATCHING VIA BLACKBOX DIFFERENTIATION OF COMBINATORIAL SOLVERS

Michal Rolínek¹, Paul Swoboda², Dominik Zietlow¹, Anselm Paulus¹, Vít Musil³, Georg Martius¹

¹ Max-Planck-Institute for Intelligent Systems, Tübingen, Germany

² Max-Planck-Institut für Informatik, Saarbrücken, Germany

³ Università degli Studi di Firenze, Italy

michal.rolinek@tuebingen.mpg.de

ABSTRACT

Building on recent progress at the intersection of combinatorial optimization and deep learning, we propose an end-to-end trainable architecture for deep graph matching that contains unmodified combinatorial solvers. Using the presence of heavily optimized combinatorial solvers together with some improvements in architecture design, we advance state-of-the-art on deep graph matching benchmarks for keypoint correspondence. In addition, we highlight the conceptual advantages of incorporating solvers into deep learning architectures, such as the possibility of post-processing with a strong multi-graph matching solver or the indifference to changes in the training setting. Finally, we propose two new challenging experimental setups.

1 Introduction

Matching discrete structures is a recurring theme in numerous branches of computer science. Aside from extensive analysis of its theoretical and algorithmic aspects [9, 26], there is also a wide range of applications. Computer vision, in particular, is abundant of tasks with a matching flavor; optical flow [4, 49, 50], person re-identification [25, 45], stereo matching [12, 36], pose estimation [11, 25], object tracking [39, 57], to name just a few. Matching problems are also relevant in a variety of scientific disciplines including biology [28], language processing [40], bioinformatics [19], correspondence problems in computer graphics [43] or social network analysis [35].

Particularly, in the domain of computer vision, the matching problem has two parts: **extraction of local features** from raw images and **resolving conflicting evidence** e.g. multiple long-term occlusions in a tracking context. Each of these parts can be addressed efficiently in separation, namely by deep networks on the one side and by specialized purely combinatorial algorithms on the other. The latter requires a clear abstract formulation of the combinatorial problem. Complications arise if concessions on *either* side harm performance. Deep networks on their own have a limited capability of *combinatorial generalization* [6] and purely combinatorial approaches typically rely on fixed features that are often suboptimal in practice. To address this, many *hybrid* approaches have been proposed.

In case of *deep graph matching* some approaches rely on finding suitable differentiable relaxations [60, 62], while others benefit from a tailored architecture design [24, 27, 59, 64]. What all these approaches have in common is that they compromise on the combinatorial side in the sense that the resulting “combinatorial block” would not be competitive in a purely combinatorial setup.

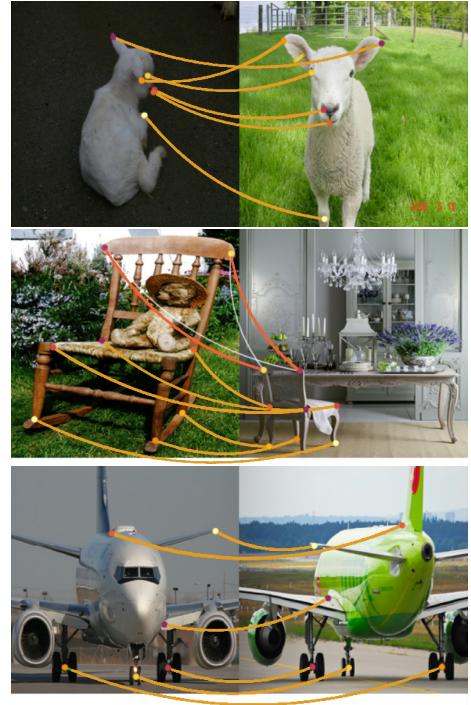


Figure 1: Example keypoint matchings of the proposed architecture on SPair-71k.

In this work, we present a novel type of end-to-end architecture for semantic keypoint matching that **does not make any concessions on the combinatorial side** while maintaining strong feature extraction. We build on recent progress at the intersection of combinatorial optimization and deep learning [56] that allows to seamlessly embed **blackbox implementations** of a wide range of combinatorial algorithms into deep networks in a **mathematically sound** fashion. As a result, we can leverage heavily optimized graph matching solvers [51, 53] based on dual block coordinate ascent for Lagrange decompositions.

Since the combinatorial aspect is handled by an expert algorithm, we can focus on the rest of the architecture design: building representative graph matching instances from visual and geometric information. In that regard, we leverage the recent findings [24] that large performance improvement can be obtained by correctly incorporating relative keypoint locations via SplineCNN [22].

Additionally, we observe that correct matching decisions are often simplified by leveraging global information such as viewpoint, rigidity of the object or scale (see also Fig. 1). With this in mind, we propose a natural **global feature attention mechanism** that allows to adjust the weighting of different node and edge features based on a global feature vector.

Finally, the proposed architecture allows a stronger post-processing step. In particular, we use a multi-graph matching solver [53] during evaluation to jointly resolve multiple graph matching instances in a consistent fashion.

On the experimental side, we achieve state-of-the-art results on standard keypoint matching datasets Pascal VOC (with Berkeley annotations [8, 20]) and Willow ObjectClass [14]. Motivated by lack of challenging standardised benchmarks, we additionally propose two new experimental setups. The first one is the evaluating on SPair-71k [38] a high-quality dataset that was recently released in the context of *dense image matching*. As the second one, we suggest to drop the common practice of keypoint pre-filtering and as a result force the future methods to address the presence of keypoints without a match.

The contributions presented in this paper can be summarized as follows.

1. We present a novel and conceptually simple architecture that seamlessly incorporates a combinatorial graph matching solver. In addition, improvements are attained on the feature extraction side by processing global image information.
2. We introduce two new experimental setups and suggest them as future benchmarks.
3. We perform an extensive evaluation on existing benchmarks as well as on the newly proposed ones. Our approach reaches higher matching accuracy than previous methods, particularly in more challenging scenarios.
4. We exhibit further advantages of incorporating a combinatorial solver:
 - (i) possible post-processing with a multi-graph matching solver,
 - (ii) an effortless transition to more challenging scenarios with unmatchable keypoints.

2 Related Work

2.1 Combinatorial Optimization Meets Deep Learning

The research on this intersection is driven by two main paradigms.

The first one attempts to improve combinatorial optimization algorithms with deep learning methods. Such examples include the use of reinforcement learning for increased performance of branch-and-bound decisions [5, 25, 30] as well as of heuristic greedy algorithms for NP-Hard graph problems [7, 17, 29, 32].

The other mindset aims at enhancing the expressivity of neural nets by turning combinatorial algorithms into differentiable building blocks. The work on differentiable quadratic programming [3] served as a catalyst and progress was achieved even in more discrete settings [21, 37, 58]. In a recent culmination of these efforts [56], a “differentiable wrapper” was proposed for *blackbox implementations* of algorithms minimizing a linear discrete objective, effectively allowing free flow of progress from combinatorial optimization to deep learning.

2.2 Combinatorial Graph Matching

This problem, also known as the quadratic assignment problem [33] in the combinatorial optimization literature, is famous for being one of the practically most difficult NP-complete problems. There exist instances with less than 100 nodes that can be extremely challenging to solve with existing approaches [10]. Nevertheless, in computer vision

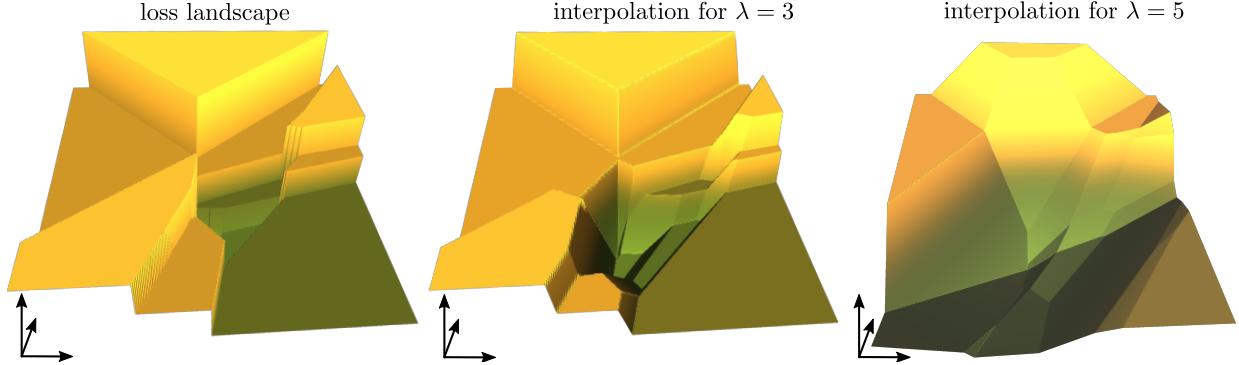


Figure 2: Differentiation of a piecewise constant loss resulting from incorporating a graph matching solver. A two-dimensional section of the loss landscape is shown (left) along with two differentiable interpolations of increasing strengths (middle and right).

efficient algorithmic approaches have been proposed that can routinely solve sparse instances with hundreds of nodes. Among those, solvers based on Lagrangian decomposition [51, 54, 65] have been shown to perform especially well, being able to quickly produce high quality solutions with small gaps to the optimum. Lagrange decomposition solvers split the graph matching problem into many small subproblems linked together via Lagrange multipliers. These multipliers are iteratively updated in order to reach agreement among the individual subproblems, typically with subgradient based techniques [48] or dual block coordinate ascent [52].

Graph matching solvers have a rich history of applications in computer vision. A non-exhaustive list includes uses for finding correspondences of landmarks between various objects in several semantic object classes [54, 55, 66], for estimating sparse correspondences in wide-displacement optical flow [2, 54], for establishing associations in multiple object tracking [13], for object categorization [18], and for matching cell nuclei in biological image analysis [28].

2.3 Peer Methods

Wider interest in deep graph matching was ignited by [62] where a fully differentiable graph matching solver based on spectral methods was introduced. While differentiable relaxation of quadratic graph matching has reappeared [60], most methods [27, 59, 61] rely on the Sinkhorn iterative normalization [1, 47] for the linear assignment problem or even on a single row normalization [24]. Another common feature is the use of various graph neural networks [6, 34, 44] sometimes also in a cross-graph fashion [59] for refining the node embeddings provided by the backbone architecture. There has also been a discussion regarding suitable loss functions [59, 61, 62]. Recently, nontrivial progress has been achieved by extracting more signal from the available geometric information [24, 64].

3 Methods

3.1 Differentiability of Combinatorial Solvers

When incorporating a combinatorial solver into a neural network, differentiability constitutes the principal difficulty. Such solvers take continuous inputs (vertex and edge costs in our case) and return a discrete output (an indicator vector of the optimal matching). This mapping is piecewise constant because a small change of the costs typically does not affect the optimal matching. Therefore, the gradient exists almost everywhere but is equal to zero. This prohibits any gradient-based optimization.

A recent method proposed in [56] offers a mathematically-backed solution to overcome these obstacles. It introduces an efficient “implicit interpolation” of the solver’s mapping while still treating the solver as a blackbox. In end effect, the intact solver is executed on the forward pass and as it turns out, only one other call to the solver is sufficient to provide meaningful gradient information during the backward pass.

Specifically, the method of [56] applies to *solvers* that solve an optimization problem of the form

$$w \in \mathbb{R}^N \mapsto y(w) \in Y \subset \mathbb{R}^N \quad \text{such that} \quad y(w) = \arg \min_{y \in Y} w \cdot y, \quad (1)$$

Algorithm 1 Forward and Backward Pass

```

function FORWARDPASS( $c^v, c^e$ )
    ( $\mathbf{v}, \mathbf{e}$ ) := GraphMatching( $c^v, c^e$ )
        // Run the solver
    save ( $\mathbf{v}, \mathbf{e}$ ) and ( $c^v, c^e$ )
        // Needed for backward pass
    return ( $\mathbf{v}, \mathbf{e}$ )
function BACKWARDPASS( $\nabla L(\mathbf{v}, \mathbf{e}), \lambda$ )
    load ( $\mathbf{v}, \mathbf{e}$ ) and ( $c^v, c^e$ )
    ( $c_\lambda^v, c_\lambda^e$ ) := ( $c^v, c^e$ ) +  $\lambda \nabla L(\mathbf{v}, \mathbf{e})$ 
        // Calculate modified costs
    ( $\mathbf{v}_\lambda, \mathbf{e}_\lambda$ ) := GraphMatching( $c_\lambda^v, c_\lambda^e$ )
        // One more call to the solver
    return  $\frac{1}{\lambda} (\mathbf{v}_\lambda - \mathbf{v}, \mathbf{e}_\lambda - \mathbf{e})$ 

```

where w is the continuous input and Y is *any* discrete set. This general formulation covers large classes of combinatorial algorithms that include the shortest path problem, the traveling salesman problem and many others. As will be shown in the subsequent sections, graph matching is also included in this definition.

If L denotes the final loss of the network, the suggested gradient of the piecewise constant mapping $w \mapsto L(y(w))$ takes the form

$$\frac{dL(y(w))}{dy} := \frac{y(w_\lambda) - y(w)}{\lambda}, \quad (2)$$

in which w_λ is a certain modification of the input w depending on the gradient of L at $y(w)$. This is in fact the *exact gradient* of a piecewise linear interpolation of $L(y(w))$ in which a hyperparameter $\lambda > 0$ controls the interpolation range as Fig. 2 suggests.

It is worth pointing out that the framework does not require any *explicit* description of the set Y (such as via linear constraints). For further details and mathematical guarantees, see [56].

3.2 Graph Matching

The aim of graph matching is to find an assignment between vertices of two graphs that minimizes the sum of local and geometric costs.

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two directed graphs. We denote by $\mathbf{v} \in \{0, 1\}^{|V_1||V_2|}$ the indicator vector of matched vertices, that is $\mathbf{v}_{i,j} = 1$ if a vertex $i \in V_1$ is matched with $j \in V_2$ and $\mathbf{v}_{i,j} = 0$ otherwise. Analogously, we set $\mathbf{e} \in \{0, 1\}^{|E_1||E_2|}$ as the indicator vector of matched edges. Obviously, the vector \mathbf{e} is fully determined by the vector \mathbf{v} . Further, we denote by $\text{Adm}(G_1, G_2)$ the set of all pairs (\mathbf{v}, \mathbf{e}) that encode a valid matching between G_1 and G_2 .

Given two cost vectors $c^v \in \mathbb{R}^{|V_1||V_2|}$ and $c^e \in \mathbb{R}^{|E_1||E_2|}$, we formulate the graph matching optimization problem as

$$\text{GM}(c^v, c^e) = \arg \min_{(\mathbf{v}, \mathbf{e}) \in \text{Adm}(G_1, G_2)} \{c^v \cdot \mathbf{v} + c^e \cdot \mathbf{e}\}. \quad (3)$$

It is immediate that GM fits the definition of the solver given in (1). If $L = L(\mathbf{v}, \mathbf{e})$ is the loss function, the mapping

$$(c^v, c^e) \mapsto L(\text{GM}(c^v, c^e)) \quad (4)$$

is the piecewise constant function for which the scheme of [56] suggests

$$\nabla(L(\text{GM}(c^v, c^e))) := \frac{1}{\lambda} [\text{GM}(c_\lambda^v, c_\lambda^e) - \text{GM}(c^v, c^e)], \quad (5)$$

where the vectors c_λ^v and c_λ^e stand for

$$c_\lambda^v = c^v + \lambda \frac{\partial L}{\partial \mathbf{v}}(\text{GM}(c^v, c^e)) \quad \text{and} \quad c_\lambda^e = c^e + \lambda \frac{\partial L}{\partial \mathbf{e}}(\text{GM}(c^v, c^e)). \quad (6)$$

The implementation is listed in Alg. 1.

In our experiments, we use the Hamming distance between the proposed matching and the ground truth matching of vertices as a loss. In this case, L does not depend on \mathbf{e} and, consequently, $c_\lambda^e = c^e$.

A more sophisticated variant of graph matching involves more than two graphs. The aim of multi-graph matching is to find a matching for every pair of graphs such that these matchings are consistent in a global fashion (i.e. satisfy so-called cycle consistency, see Fig. 3) and minimize the global cost. Although the framework of [56] is also applicable to multi-graph matching, we will only use it for post-processing.

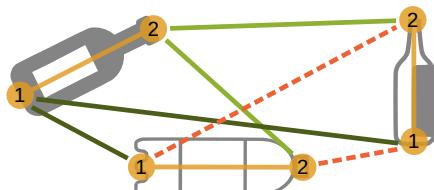


Figure 3: Cycle consistency in multi-graph matching. The partial matching induced by light and dark green edges prohibits including the dashed edges.

3.3 Cost Margin

One disadvantage of using Hamming distance as a loss function is that it reaches its minimum value zero even if the ground truth matching has only fractionally lower cost than competing matchings. This increases sensitivity to distribution shifts and potentially harms generalization. The issue was already observed in [42], where the method [56] was also applied. We adopt the solution proposed in [42], namely the *cost margin*. In particular, during training we increase the unary costs that correspond to the ground truth matching by $\alpha > 0$, i.e.

$$\overleftarrow{c}^v_{i,j} = \begin{cases} c^v_{i,j} + \alpha & \text{if } \mathbf{v}_{i,j}^* = 1 \\ c^v_{i,j} & \text{if } \mathbf{v}_{i,j}^* = 0 \end{cases} \quad \text{for } i \in V_1 \text{ and } j \in V_2, \quad (7)$$

where \mathbf{v}^* denotes the ground truth matching indicator vector. In all experiments, we use $\alpha = 1.0$.

3.4 Solvers

Graph matching. We employ a dual block coordinate ascent solver [51] based on a Lagrange decomposition of the original problem. In every iteration, a dual lower bound is monotonically increased and the resulting dual costs are used to round primal solutions using a minimum cost flow solver.

Multi-graph matching. We employ the solver from [53] that builds upon [51] and extends it to include additional constraints arising from cycle consistency. Primal solutions are rounded using a special form of permutation synchronization [41] allowing for partial matchings.

3.5 Architecture Design

Our end-to-end trainable architecture for keypoint matching consists of three stages. We call it BlackBox differentiation of Graph Matching solvers (BB-GM).

1. *Extraction of visual features* A standard CNN architecture extracts a feature vector for each of the keypoints in the image. Additionally, a global feature vector is extracted.
2. *Geometry-aware feature refinement* Keypoints are converted to a graph structure with spatial information. Then a graph neural network architecture is applied.
3. *Construction of combinatorial instance* Vertex and edge similarities are computed using the graph features and the global features. This determines a graph matching instance that is passed to the solver.

The resulting matching \mathbf{v} is compared to the ground truth matching \mathbf{v}^* and their Hamming distance $L(\mathbf{v}) = \mathbf{v} \cdot (1 - \mathbf{v}^*) + \mathbf{v}^* \cdot (1 - \mathbf{v})$ is the loss function to optimize.

While the first and the second stage (Fig. 4) are rather standard design blocks, the third one (Fig. 5) constitutes the principal novelty. More detailed descriptions follow.

3.5.1 Visual Feature Extraction

We closely follow previous work [24, 59, 62] and also compute the outputs of the `relu4_2` and `relu5_1` operations of the VGG16 [46] network pre-trained on ImageNet [16]. The spatially corresponding feature vector for each keypoint is recovered via bi-linear interpolation.

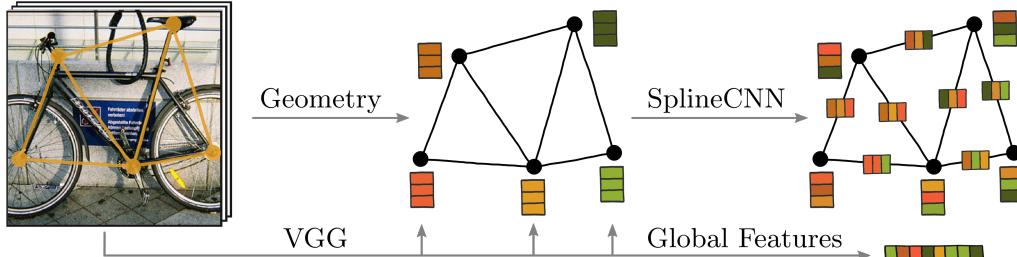


Figure 4: Extraction of features for a single image. Keypoint locations and VGG features are processed by a SplineCNN and a global feature vector is produced.

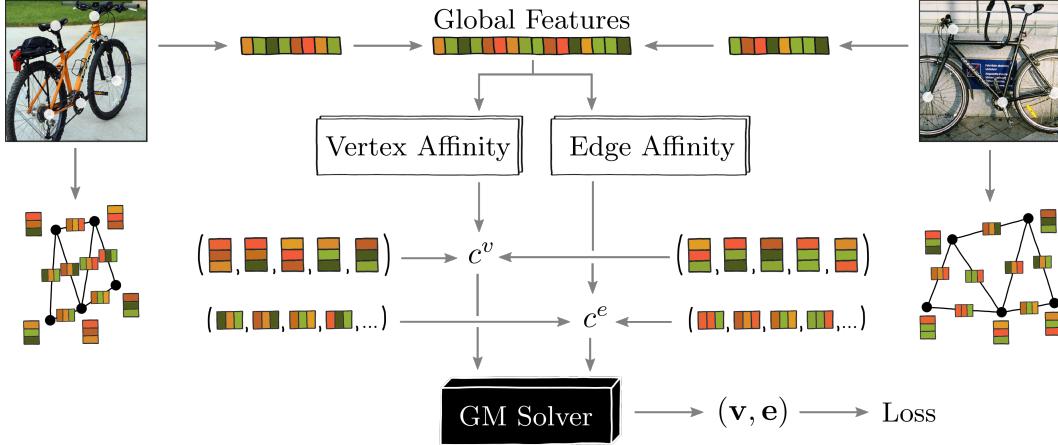


Figure 5: Construction of combinatorial instance for keypoint matching.

An image-wide global feature vector is extracted by max-pooling the output of the final VGG16 layer, see Fig. 4. Both the keypoint feature vectors and the global feature vectors are normalized with respect to the L^2 norm.

3.5.2 Geometric Feature Refinement

The graph is created as a Delaunay triangulation [15] of the keypoint locations. We deploy SplineCNN [22], an architecture that proved successful in point-cloud processing. Its inputs are the VGG vertex features and spatial edge attributes defined as normalized relative coordinates of the associated vertices (called anisotropic in [23, 24]). We use two layers of SplineCNN with MAX aggregations. The outputs are additively composed with the original VGG node features to produce the refined node features. For subsequent computation, we set the edge features as the differences of the refined node features. For illustration, see Fig. 4.

3.5.3 Matching Instance Construction

Both source and target image are passed through the two described procedures. Their global features are concatenated to one global feature vector g . A standard way to prepare a matching instance (the unary costs c^v) is to compute the inner product similarity (or affinity) of the vertex features $c_{i,j}^v = f_s^v(i) \cdot f_t^v(j)$, where $f_s^v(i)$ is the feature vector of the vertex i in the source graph and $f_t^v(j)$ is the feature vector of the vertex j in the target graph, possibly with a learnable vector or a matrix of coefficient as in [59].

In our case, the vector of “similarity coefficients” is produced as a linear transformation of g followed by a nonlinearity. In particular,

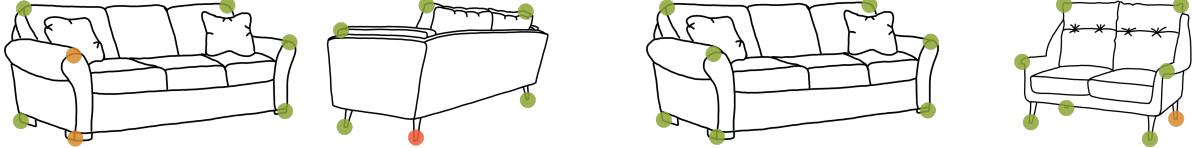
$$c_{i,j}^v = f_s^v(i)^T \text{diag}(\varphi(Ag)) f_t^v(j) \quad (8)$$

where A is a learnable matrix and φ is a \tanh nonlinearity. This allows for a gating-like behavior; the individual coordinates of the feature vectors may play a different role depending on the global feature vector g . It is intended to enable integrating various global semantic aspects such as rigidity of the object or the viewpoint perspective. Higher order cost terms c^e are calculated in the same vein using edge features instead of vertex features with an analogous learnable affinity layer. For an overview, see Fig. 5.

4 Experiments

We evaluate our method on the standard datasets for keypoint matching Pascal VOC with Berkeley annotations [8, 20] and Willow ObjectClass [14]. Additionally, we propose a harder setup for Pascal VOC that avoids keypoint filtering as a preprocessing step. Finally, we report our performance on a recently published dataset SPair-71k [38]. Even though this dataset was designed for a slightly different community, its high quality makes it very suitable also in this context. The two new experimental setups aim to address the lack of difficult benchmarks in this line of work. The code used for our experiments is available at <https://github.com/martius-lab/blackbox-deep-graph-matching>.

In some cases, we report our own evaluation of DGMC [24], the strongest competing method, which we denote by DGMC*. We used the publicly available implementation [23].



(a) Intersection filtering ($\cdot : \cap : \cdot$). Only the keypoints visible in both images are used (green), others are ignored (yellow, red). (b) Inclusion filtering ($\cdot : \subset : \cdot$). For any source image (left), only the targets (right) containing all the source keypoints are used.

Figure 6: Keypoint filtering strategies. The image pair in (a) would not occur under inclusion filtering (b) because the different perspectives lead to incomparable sets of keypoints. Intersection filtering is unaffected by viewpoints.

Table 1: Impact of filtering strategies on test accuracy (%) for DGM [24] on Pascal VOC. Classes with drastic differences are highlighted.

Filter	airplane	bicycle	bird	boat	car	chair	motorcycle	sofa	train	tv	cow	horse	bottle	motorcycle	person	sheep	sofa	train	tv	Mean	
$\cdot : \cap : \cdot$	50.4	67.6	70.7	70.5	87.2	85.2	82.5	74.3	46.2	69.4	69.9	73.9	73.8	65.4	51.6	98.0	73.2	69.6	94.3	89.6	73.2 ± 0.5
$\cdot : \subset : \cdot$	45.5	66.6	54.5	67.8	87.2	86.4	85.6	73.2	38.5	67.3	86.9	64.9	78.9	60.3	61.5	96.8	68.7	93.5	93.6	85.0	73.1 ± 0.4

4.0.1 Runtime

All experiments were run on a single Tesla-V100 GPU. Due to the efficient C++ implementation of the solver [52], the computational bottleneck of the entire architecture is evaluating the VGG backbone. Around 30 image pairs were processed every second.

4.0.2 Hyperparameters

In all experiments, we use the exact same set of hyperparameters. Only the number of training steps is dataset-dependent. The optimizer in use is Adam [31] with an initial learning rate of 2×10^{-3} which is halved four times in regular intervals. Learning rate for finetuning the VGG weights is multiplied with 10^{-2} . We process batches of 8 image pairs and the hyperparameter λ from (2) is consistently set to 80.0. For remaining implementation details, the full code base will be made available.

4.0.3 Image Pair Sampling and Keypoint Filtering

The standard benchmark datasets provide images with annotated keypoints but do not define pairings of images or which keypoints should be kept for the matching instance. While it is the designer’s choice how this is handled during training it is imperative that only one pair-sampling and keypoint filtering procedure is used at test time. Otherwise, the change in the distribution of test pairs and the corresponding instances may have unintended effects on the evaluation metric (as we demonstrate below), and therefore hinder fair comparisons.

We briefly describe two previously used methods for creating evaluation data, discuss their impact, and propose a third one.

Keypoint intersection ($\cdot : \cap : \cdot$) Only the keypoints present in both source and target image are preserved for the matching task. Clearly, any pair of images can be processed this way, see Fig. 6a.

Keypoint inclusion ($\cdot : \subset : \cdot$) Target image keypoints have to include all the source image keypoints. The target keypoints that are not present in the source image are then disregarded. Not every pair of images obeys this property, see Fig. 6b.

When keypoint inclusion filtering is used on evaluation, some image pairs are discarded, which introduces some biases. In particular, pairs of images seen from different viewpoints become underrepresented, as such pairs often have incomparable sets of visible keypoints, see Fig. 6. Another effect is a bias towards a higher number of keypoints in a matching instance which makes the matching task more difficult. While the effect on mean accuracy is not strong, Tab. 1 shows large differences in individual classes.

Another unsatisfactory aspect of both methods is that label information is required at evaluation time, rendering the setting quite unrealistic. For this reason, we **propose to evaluate without any keypoint removal**.

Table 2: Keypoint matching accuracy (%) on Pascal VOC using standard intersection filtering ($\cdot : \cap : \cdot$). For GMN [62] we report the improved results from [59] denoted as GMN-PL. DGMC* is [24] reproduced using $\cdot : \cap : \cdot$. For DGMC* and BB-GM we report the mean over 5 restarts.

Method	airplane	bicycle	bird	boat	car	motorcycle	sofa	train	tv	motor	person	sheep	cow	bus	cup	Mean					
GMN-PL	31.1	46.2	58.2	45.9	70.6	76.5	61.2	61.7	35.5	53.7	58.9	57.5	56.9	49.3	34.1	57.9					
PCA-GM [59]	40.9	55.0	65.8	47.9	76.9	77.9	63.5	67.4	33.7	66.5	63.6	61.3	58.9	62.8	44.9	63.8					
NGM+ [60]	50.8	64.5	59.5	57.6	79.4	76.9	74.4	69.9	41.5	62.3	68.5	62.2	62.4	64.7	47.8	66.1					
GLMNet [27]	52.0	67.3	63.2	57.4	80.3	74.6	70.0	72.6	38.9	66.3	77.3	65.7	67.9	64.2	44.8	67.5					
CIE1-H [61]	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	68.9					
DGMC* [24]	50.4	67.6	70.7	70.5	87.2	85.2	82.5	74.3	46.2	69.4	69.9	73.9	73.8	65.4	51.6	73.2 ± 0.5					
BB-GM	61.5	75.0	78.1	80.0	87.4	93.0	89.1	80.2	58.1	77.6	76.5	79.3	78.6	78.8	66.7	97.4	76.4	77.5	97.7	94.4	80.1 ± 0.6

Table 3: F1 score (%) for Pascal VOC keypoint matching without filtering ($\cdot : \cup : \cdot$). As a reference we report an ablation of our method where the solver is forced to match all source keypoints, denoted as BB-GM-Max. BB-GM-Multi refers to using the multi-graph matching solver with cycle consistency [53] with sets of 5 images at evaluation. The reported statistics are over 10 restarts. The last row displays the percentage of unmatched keypoints in the test-set pairs.

Method	airplane	bicycle	bird	boat	car	motorcycle	sofa	train	tv	motor	person	sheep	cow	bus	cup	Mean					
BB-GM-Max	35.5	68.6	46.7	36.1	85.4	58.1	25.6	51.7	27.3	51.0	46.0	46.7	48.9	58.9	29.6	93.6	42.6	35.3	70.7	79.5	51.9 ± 1.0
BB-GM	42.7	70.9	57.5	46.6	85.8	64.1	51.0	63.8	42.4	63.7	47.9	61.5	63.4	69.0	46.1	94.2	57.4	39.0	78.0	82.7	61.4 ± 0.5
BB-GM-Multi	43.4	70.5	61.9	46.8	84.9	65.3	54.2	66.9	44.9	67.5	50.8	66.8	63.3	71.0	46.1	96.1	56.5	41.3	73.4	83.4	62.8 ± 0.5
Unmatched (%)	22.7	4.9	30.6	29.1	2.7	23.8	40.8	26.4	17.3	25.1	21.2	27.4	26.8	16.6	22.1	6.7	36.7	27.5	31.7	14.0	22.7

Unfiltered keypoints ($\cdot : \cup : \cdot$) For a given pair of images, the keypoints are used without any filtering. Matching instances may contain a different number of source and target vertices, and vertices without a corresponding match.

4.1 Pascal VOC

The Pascal VOC [20] dataset with Berkeley annotations [8] contains images with bounding boxes surrounding objects of 20 classes. We follow the standard data preparation procedure of [59]. Each object is cropped to its bounding box and scaled to 256×256 px. The resulting images contain up to 23 annotated keypoints, depending on the object category.

The results under the most common experimental conditions ($\cdot : \cap : \cdot$) are reported in Tab. 2 and we can see that BB-GM outperforms competing approaches.

4.1.1 All keypoints

We propose, see Sec. 4.0.3, to preserve all keypoints ($\cdot : \cup : \cdot$). Matching accuracy is no longer a good evaluation metric as it ignores false positives. Instead, we report F1-Score, the harmonic mean of precision and recall.

Since the underlying solver used by our method also works for partial matchings, our architecture is applicable out of the box. Competing architectures rely on either the Sinkhorn normalization or a softmax and as such, they are hard-wired to produce maximal matchings and do not offer a simple adjustment to the unfiltered setup. To simulate the negative impact of maximal matchings we provide an ablation of BB-GM where we modify the solver to output maximal matchings. This is denoted by BB-GM-Max.

In addition, we report the scores obtained by running the multi-graph matching solver [53] as post-processing. Instead of sampling pairs of images, we sample sets of 5 images and recover from the architecture the costs of the $\binom{5}{2} = 10$ matching instances. The multi-graph matching solver then searches for globally optimal set of consistent matchings. The results are provided in Tab. 3.

Note that sampling sets of 5 images instead of image pairs does not interfere with the statistics of the test set. The results are therefore comparable.

4.2 Willow ObjectClass

The Willow ObjectClass dataset contains a total of 256 images from 5 categories. Each category is represented by at least 40 images, all of them with consistent orientation. Each image is annotated with the same 10 distinctive category-specific keypoints, which means there is no difference between the described keypoint filtering methods. Following standard procedure, we crop the images to the bounding boxes of the objects and rescale to 256×256 px.

Table 4: Keypoint matching accuracy (%) on Willow ObjectClass. The columns Pt and Wt indicate training on Pascal VOC and Willow, respectively. Comparisons should be made only within the same training setting. For HARG-SSVM [14] we report the comparable figures from [59]. Twenty restarts were carried out.

Method	Pt	Wt	face	motorbike	car	duck	bottle
HARG-SSVM [59]	x	✓	91.2	44.4	58.4	55.2	66.6
GMN-PL [59, 62]	✓	x	98.1	65.0	72.9	74.3	70.5
	✓	✓	99.3	71.4	74.3	82.8	76.7
PCA-GM [59]	✓	x	100.0	69.8	78.6	82.4	95.1
	✓	✓	100.0	76.7	84.0	93.5	96.9
CIE [61]	✓	x	99.9	71.5	75.4	73.2	97.6
	✓	✓	100.0	90.0	82.2	81.2	97.6
NGM [60]	x	✓	99.2	82.1	84.1	77.4	93.5
GLMNet [27]	✓	✓	100.0	89.7	93.6	85.4	93.4
DGMC* [24]	✓	x	98.6 ± 1.1	69.8 ± 5.0	84.6 ± 5.2	76.8 ± 4.3	90.7 ± 2.4
	x	✓	100.0 ± 0.0	98.5 ± 1.5	98.3 ± 1.2	90.2 ± 3.6	98.1 ± 0.9
	✓	✓	100.0 ± 0.0	98.8 ± 1.6	96.5 ± 1.6	93.2 ± 3.8	99.9 ± 0.3
BB-GM	✓	x	100.0 ± 0.0	95.8 ± 1.4	89.1 ± 1.7	89.8 ± 1.7	97.9 ± 0.7
	x	✓	100.0 ± 0.0	99.2 ± 0.4	96.9 ± 0.6	89.0 ± 1.0	98.8 ± 0.6
	✓	✓	100.0 ± 0.0	98.9 ± 0.5	95.7 ± 1.5	93.1 ± 1.5	99.1 ± 0.4

Multiple training strategies have been used in prior work. Some authors decide to train only on the relatively small Willow dataset, or pretrain on Pascal VOC and fine-tune on Willow afterward [59]. Another approach is to pretrain on Pascal VOC and evaluate on Willow without fine-tuning, to test the transferability [60]. We report results for all different variants, following the standard procedure of using 20 images per class when training on Willow and excluding the classes *car* and *motorbike* from Pascal VOC when pre-training, as these images overlap with the Willow dataset. We also evaluated the strongest competing approach DGMC [24] under all settings.

The results are shown in Tab. 4. While our method achieves good performance, we are reluctant to claim superiority over prior work. The small dataset size, the multitude of training setups, and high standard deviations all prevent statistically significant comparisons.

4.3 SPair-71k

We also report performance on SPair-71k [38], a dataset recently published in the context of dense image matching. It contains 70,958 image pairs prepared from Pascal VOC 2012 and Pascal 3D+. It has several advantages over the Pascal VOC dataset, namely higher image quality, richer keypoint annotations, difficulty annotation of image-pairs, as well as the removal of the ambiguous and poorly annotated *sofas* and *dining tables*.

Again, we evaluated DGMC [24] as the strongest competitor of our method. The results are reported in Tab. 5 and Tab. 6. We consistently improve upon the baseline, particularly on pairs of images seen from very different viewpoints. This highlights the ability of our method to resolve instances with conflicting evidence. Some example matchings are presented in Fig. 1 and Fig. 7.

Table 5: Keypoint matching accuracy (%) on SPair-71k grouped by levels of difficulty in the viewpoint of the matching-pair. Statistics is over 5 restarts.

Method	Viewpoint difficulty			All
	easy	medium	hard	
DGMC*	79.4 ± 0.2	65.2 ± 0.2	61.3 ± 0.5	72.2 ± 0.2
BB-GM	84.8 ± 0.1	73.1 ± 0.2	70.6 ± 0.9	78.9 ± 0.4

Table 6: Keypoint matching accuracy (%) on SPair-71k for all classes.

Method	airplane	bicycle	bird	boat	motorcycle	bus	car	cat	chair	cow	dog	motorcycle	person	sheep	sofa	tv	Mean		
DGMC*	54.8	44.8	80.3	70.9	65.5	90.1	78.5	66.7	66.4	73.2	66.2	66.5	65.7	59.1	98.7	68.5	84.9	98.0	72.2 ± 0.2
BB-GM	66.9	57.7	85.8	78.5	66.9	95.4	86.1	74.6	68.3	78.9	73.0	67.5	79.3	73.0	99.1	74.8	95.0	98.6	78.9 ± 0.4

4.4 Ablation Studies

To isolate the impact of single components of our architecture, we conduct various ablation studies. The results on Pascal VOC are summarized in Tab. 7 where large performance differences are highlighted.

Global Features / Fixed Affinity Influence of the global feature vector is removed. In (8), instead of A_g , we use a single learnable vector a .

VGG Fine-Tuning / Frozen VGG VGG pretrained on ImageNet [16] is used without fine-tuning. This ablation is in particular important for a fair comparison with DGMG [24] where no fine-tuning is applied as well.

Quadratic Costs / Unary Costs The matching instances consist only of unary costs c^v . The quadratic costs c^e are set to zero.

Comparison to Sinkhorn We use Sinkhorn normalization [1, 47] instead of the solver. We only obtained good performance after increasing an internal constant ε (intended to prevent division by zero) (from 10^{-8} to 10^{-4}). We believe this is connected to issues with “vanishing gradient” that were also reported in [24, 63].

Table 7: Ablations of BB-GM on Pascal VOC. Large performance differences are highlighted. Statistics is over 5 restarts.

Ablation	airplane	bicycle	bird	boat	car	chair	couch	diningtable	motorcycle	sofa	wheelchair	Mean
Unmodified	62.0	77.0	76.4	75.9	89.1	93.7	88.6	80.0	56.6	78.2	81.3	79.0 77.1 77.6 64.3 97.0 78.3 78.4 97.9 94.6 80.1 ± 0.3
Fixed Affinity	59.1	73.9	75.6	73.7	87.2	88.4	86.4	77.7	55.2	76.2	74.7	77.4 78.1 77.1 62.7 96.3 75.2 73.1 96.3 94.4 77.9 ± 0.7
Frozen VGG	57.8	73.5	75.2	77.5	87.2	92.4	87.1	79.0	59.1	77.8	77.4	77.9 77.0 77.4 63.0 96.9 75.8 72.7 97.5 94.6 78.9 ± 0.5
Unary Costs	60.7	74.4	77.2	78.0	87.0	92.3	89.6	80.4	55.5	77.3	65.2	79.0 79.2 77.2 63.0 97.0 75.8 73.3 96.0 93.6 78.6 ± 0.4
Sinkhorn	62.5	73.9	74.3	75.3	88.3	94.3	88.9	76.9	51.6	75.8	68.8	77.0 75.3 78.3 60.2 97.7 75.8 71.3 97.8 94.6 77.9 ± 0.5

5 Conclusion

We have demonstrated that deep learning architectures that integrate combinatorial graph matching solvers perform well on deep graph matching benchmarks.

Opportunities for future work now fall into multiple categories. For one, it should be tested whether such architectures can be useful outside the designated playground for deep graph matching methods. If more progress is needed, two major directions lend themselves: (i) improving the neural network architecture even further so that input costs to the matching problem become more discriminative and (ii) employing better solvers that improve in terms of obtained solution quality and ability to handle a more complicated and expressive cost structure (e.g. hypergraph matching solvers).

Finally, the potential of building architectures around solvers for other computer vision related combinatorial problems such as MULTICUT or MAX-CUT can be explored.



Figure 7: Example matchings from the SPair-71k dataset.

References

- [1] Ryan Prescott Adams and Richard S. Zemel. Ranking via sinkhorn propagation, 2011.
- [2] Hassan Abu Alhaija, Anita Sellent, Daniel Kondermann, and Carsten Rother. Graphflow–6d large displacement scene flow via graph matching. In *German Conference on Pattern Recognition*, pages 285–296. Springer, 2015.
- [3] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, ICML’17, pages 136–145, 2017.
- [4] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.
- [5] Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. Learning to branch. In *International Conference on Machine Learning*, ICML’18, pages 353–362, 2018.
- [6] Peter Battaglia, Jessica Blake Chandler Hamrick, Victor Bapst, Alvaro Sanchez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andy Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Jayne Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.
- [7] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. In *International Conference on Learning Representations, Workshop Track*, ICLR’17, 2017.
- [8] Lubomir Bourdev and Jitendra Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *IEEE International Conference on Computer Vision*, ICCV’09, pages 1365–1372, 2009.
- [9] Rainer Burkard, Mauro Dell’Amico, and Silvano Martello. *Assignment Problems*. Society for Industrial and Applied Mathematics, USA, 2009. ISBN 0898716632.
- [10] Rainer E Burkard, Stefan E Karisch, and Franz Rendl. QAPLIB—a quadratic assignment problem library. *Journal of Global optimization*, 10(4):391–403, 1997.
- [11] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR’17, 2017.
- [12] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR’18, pages 5410–5418, 2018.
- [13] Hwann-Tzong Chen, Horng-Horng Lin, and Tyng-Luh Liu. Multi-object tracking using dynamical graph matching. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 2, pages II–II. IEEE, 2001.
- [14] Minsu Cho, Kartek Alahari, and Jean Ponce. Learning graphs to match. In *IEEE International Conference on Computer Vision*, ICCV’13, 2013.
- [15] B. Delaunay. Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7: 793–800, 1934.
- [16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR’09, pages 248–255, 2009.
- [17] Michel Deudon, Pierre Courtnut, Alexandre Lacoste, Yossiri Adulyasak, and Louis-Martin Rousseau. Learning heuristics for the tsp by policy gradient. In *Intl. Conf. on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 170–181. Springer, 2018.
- [18] Olivier Duchenne, Armand Joulin, and Jean Ponce. A graph-matching kernel for object categorization. In *2011 International Conference on Computer Vision*, pages 1792–1799. IEEE, 2011.
- [19] Ahed Elmsallati, Connor Clark, and Jugal Kalita. Global alignment of protein-protein interaction networks: A survey. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 13(4):689–705, 2016. ISSN 1545-5963.
- [20] Mark Everingham, Luc Van Gool, Chris Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.
- [21] Aaron Ferber, Bryan Wilder, Bistra Dilkina, and Milind Tambe. Mipaal: Mixed integer program as a layer. *arXiv preprint arXiv:1907.05912*, 2019.
- [22] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR’18, pages 869–877, 2018.

- [23] Matthias Fey, Jan E Lenssen, Christopher Morris, Jonathan Masci, and Nils M Kriege. Deep graph matching consensus. <https://github.com/rusty1s/deep-graph-matching-consensus>, 2020. Commit: be1c4c.
- [24] Matthias Fey, Jan E Lenssen, Christopher Morris, Jonathan Masci, and Nils M Kriege. Deep graph matching consensus. In *International Conference on Learning Representations*, ICLR’20, 2020.
- [25] Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, NIPS’19, pages 15554–15566, 2019.
- [26] Martin Grohe, Gaurav Rattan, and Gerhard J. Woeginger. Graph Similarity and Approximate Isomorphism. In *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*, volume 117 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:16, 2018.
- [27] Bo Jiang, Pengfei Sun, Jin Tang, and Bin Luo. Glmnet: Graph learning-matching networks for feature matching. *arXiv preprint arXiv:1911.07681*, 2019.
- [28] Dagmar Kainmueller, Florian Jug, Carsten Rother, and Gene Myers. Active graph matching for automatic joint segmentation and annotation of *c. elegans*. In *Medical Image Computing and Computer-Assisted Intervention*, MICCAI’14, pages 81–88, 2014.
- [29] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, NIPS’17, pages 6348–6358, 2017.
- [30] Elias B. Khalil, Pierre Le Bodic, Le Song, George Nemhauser, and Bistra Dilkina. Learning to branch in mixed integer programming. In *AAAI Conference on Artificial Intelligence*, AAAI’16, pages 724–731, 2016.
- [31] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, ICLR’14, 2014.
- [32] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, ICLR’19, 2019.
- [33] Eugene L Lawler. The quadratic assignment problem. *Management science*, 9(4):586–599, 1963.
- [34] Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. Gated graph sequence neural networks. In *International Conference on Learning Representations*, ICLR’16, 2016.
- [35] Li Liu, William K. Cheung, Xin Li, and Lejian Liao. Aligning users across social networks using network embedding. In *International Joint Conference on Artificial Intelligence*, IJCAI’16, pages 1774–1780, 2016.
- [36] Wenjie Luo, Alexander G. Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR’16, pages 5695–5703, 2016.
- [37] Jaynta Mandi, Emir Demirovic, Peter J. Stuckey, and Tias Guns. Smart predict-and-optimize for hard combinatorial optimization problems. *arXiv preprint arXiv:1911.10092*, 2019.
- [38] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. SPair-71k: A Large-scale Benchmark for Semantic Correspondance. *arXiv preprint arXiv:1908.10543*, 2019.
- [39] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. *arXiv preprint arXiv:1510.07945*, 2015.
- [40] Vlad Niculae, Andre Martins, Mathieu Blondel, and Claire Cardie. SparseMAP: Differentiable sparse structured inference. In *International Conference on Machine Learning*, ICML’18, pages 3799–3808, 2018.
- [41] Deepti Pachauri, Risi Kondor, and Vikas Singh. Solving the multi-way matching problem by permutation synchronization. In *Advances in Neural Information Processing Systems*, NIPS’13, pages 1860–1868, 2013.
- [42] M. Rolínek, V. Musil, A. Paulus, M. Vlastelica, C. Michaelis, and G. Martius. Optimizing ranking-based metrics with blackbox differentiation. In *Conference on Computer Vision and Pattern Recognition*, CVPR’20, 2020.
- [43] Yusuf Sahillioglu. Recent advances in shape correspondence. *The Visual Computer*, pages 1–17, 2019.
- [44] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *Trans. Neur. Netw.*, 20(1):61–80, 2009. ISSN 1045-9227.
- [45] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR’15, pages 815–823, 2015.
- [46] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

- [47] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21, 05 1967.
- [48] Geir Storvik and Geir Dahl. Lagrangian-based methods for finding map solutions for mrf models. *IEEE Transactions on Image Processing*, 9(3):469–479, 2000.
- [49] Deqing Sun, Stefan Roth, and Michael J. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *International Journal of Computer Vision*, 106(2):115–137, 2014. ISSN 0920-5691.
- [50] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR’18, June 2018.
- [51] Paul Swoboda, Carsten Rother, Hassan Abu Alhaija, Dagmar Kainmüller, and Bogdan Savchynskyy. A study of lagrangean decompositions and dual ascent solvers for graph matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR’16, pages 7062–7071, 2016.
- [52] Paul Swoboda, Jan Kuske, and Bogdan Savchynskyy. A dual ascent framework for lagrangean decomposition of combinatorial problems. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR’17, pages 1596–1606, 2017.
- [53] Paul Swoboda, Ashkan Mokarian, Christian Theobalt, Florian Bernard, et al. A convex relaxation for multi-graph matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR’19, pages 11156–11165, 2019.
- [54] L. Torresani, V. Kolmogorov, and C. Rother. A dual decomposition approach to feature correspondence. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(2):259–271, 2013.
- [55] Nikolai Ufer and Bjorn Ommer. Deep semantic feature matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR’17, pages 6914–6923, 2017.
- [56] M. Vlastelica, A. Paulus, V. Musil, G. Martius, and M. Rolínek. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations*, ICLR’20, 2020.
- [57] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Visual tracking with fully convolutional networks. In *IEEE International Conference on Computer Vision*, ICCV’15, pages 3119–3127, 2015.
- [58] Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning*, pages 6545–6554, 2019.
- [59] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Learning combinatorial embedding networks for deep graph matching. In *IEEE International Conference on Computer Vision*, ICCV’19, pages 3056–3065, 2019.
- [60] Runzhong Wang, Junchi Yan, and Xiaokang Yang. Neural graph matching network: Learning lawler’s quadratic assignment problem with extension to hypergraph and multiple-graph matching. *arXiv preprint arXiv:1911.11308*, 2019.
- [61] Tianshu Yu, Runzhong Wang, Junchi Yan, and Baoxin Li. Learning deep graph matching with channel-independent embedding and hungarian attention. In *International Conference on Learning Representations*, ICLR’20, 2020.
- [62] Andrei Zanfir and Cristian Sminchisescu. Deep learning of graph matching. In *Conference on Computer Vision and Pattern Recognition*, CVPR’18, pages 2684–2693, 2018.
- [63] Yan Zhang, Jonathon Hare, and Adam Prügel-Bennett. Learning representations of sets through optimized permutations. *arXiv preprint arXiv:1812.03928*, 2018.
- [64] Zhen Zhang and Wee Sun Lee. Deep graphical feature learning for the feature matching problem. In *IEEE International Conference on Computer Vision*, ICCV’19, 2019.
- [65] Zhen Zhang, Qinfeng Shi, Julian McAuley, Wei Wei, Yanning Zhang, and Anton van den Hengel. Pairwise matching through max-weight bipartite belief propagation. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR’16, 2016.
- [66] Feng Zhou and Fernando De la Torre. Factorized graph matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR’12, pages 127–134, 2012.