Deep Learning of Graph Matching

Andrei Zanfir[2] and Cristian Sminchisescu[1,2]
andrei.zanfir@imar.ro, cristian.sminchisescu@math.lth.se
[1]Department of Mathematics, Faculty of Engineering, Lund University
[2]Institute of Mathematics of the Romanian Academy

Abstract

The problem of graph matching under node and pair-wise constraints is fundamental in areas as diverse as combinatorial optimization, machine learning or computer vision, where representing both the relations between nodes and their neighborhood structure is essential. We present an end-to-end model that makes it possible to learn all parameters of the graph matching process, including the unary and pairwise node neighborhoods, represented as deep feature extraction hierarchies. The challenge is in the formulation of the different matrix computation layers of the model in a way that enables the consistent, efficient propagation of gradients in the complete pipeline from the loss function, through the combinatorial optimization layer solving the matching problem, and the feature extraction hierarchy. Our computer vision experiments and ablation studies on challenging datasets like PASCAL VOC keypoints, Sintel and CUB show that matching models refined end-to-end are superior to counterparts based on feature hierarchies trained for other problems.

1. Introduction and Related Work

The problem of graph matching – establishing correspondences between two graphs represented in terms of both local node structure and pair-wise relationships, be them visual, geometric or topological – is important in areas like combinatorial optimization, machine learning, image analysis or computer vision, and has applications in structure-from-motion, object tracking, 2d and 3d shape matching, image classification, social network analysis, autonomous driving, and more. Our emphasis in this paper is on matching graph-based image representations but the methodology applies broadly, to any graph matching problem where the unary and pairwise structures can be represented as deep feature hierarchies with trainable parameters.

Unlike other methods such as RANSAC [12] or iterative closest point [4], which are limited to rigid displacements, graph matching naturally encodes structural information that can be used to model complex relationships and more diverse transformations. Graph matching operates with affinity matrices that encode similarities between unary and pairwise sets of nodes (points) in the two graphs. Typically it is formulated mathematically as a quadratic integer program [25, 3], subject to one-to-one mapping constraints, i.e. each point in the first set must have an unique correspondence in the second set. This is known to be NP-hard so methods often solve it approximately by relaxing the constraints and finding local optima [19, 38].

Learning the parameters of the graph affinity matrix has been investigated by [7, 20] or, in the context of the more general hyper-graph matching model [10], by [21]. In those cases, the number of parameters is low, often controlling geometric affinities between pairs of points rather than the image structure in the neighborhood of those points. Recently there has been a growing interest in using deep features for both geometric and semantic visual matching tasks, either by training the network to directly optimize a matching objective [8, 27, 16, 36] or by using pre-trained, deep features [23, 14] within established matching architectures, all with considerable success.

Our objective in this paper is to marry the (shallow) graph matching to the deep learning formulations. We propose to build models where the graphs are defined over unary node neighborhoods and pair-wise structures computed based on learned feature hierarchies. We formulate a complete model to learn the feature hierarchies so that graph matching works best: the feature learning and the graph matching model are refined in a single deep architecture that is optimized jointly for consistent results. Methodologically, our contributions are associated to the construction of the different matrix layers of the computation graph, obtaining analytic derivatives all the way from the loss function down to the feature layers in the framework of matrix back-propagation, the emphasis on computational efficiency for backward passes, as well as a voting based loss function. The proposed model applies generally, not just for matching different images of a category, taken in different scenes (its primary design), but also to different images of the same scene, or from a video.

2. Problem Formulation

Input. We are given two input graphs $\mathcal{G}_1 = (V_1, E_1)$ and $\mathcal{G}_2 = (V_2, E_2)$, with $|V_1| = n$ and $|V_2| = m$. Our goal is to establish an assignment between the nodes of the two graphs, so that a criterion over the corresponding nodes and edges is optimized (see below).

Graph Matching. Let $\mathbf{v} \in \{0,1\}^{nm\times 1}$ be an indicator vector such that $\mathbf{v}_{ia} = 1$ if $i \in V_1$ is matched to $a \in V_2$ and 0 otherwise, while respecting one-to-one mapping constraints. We build a square symmetric positive matrix $\mathbf{M} \in \mathbb{R}^{nm\times nm}$ such that $\mathbf{M}_{ia;jb}$ measures how well every pair $(i,j) \in E_1$ matches with $(a,b) \in E_2$. For pairs that do not form edges, their corresponding entries in the matrix are set to 0. The diagonal entries contain node-to-node scores, whereas the off-diagonal entries contain edge-to-edge scores. The optimal assignment $\mathbf{v}^*$ can be formulated as

$$\mathbf{v}^* = \underset{\mathbf{v}}{\operatorname{argmax}}\, \mathbf{v}^\top \mathbf{M}\mathbf{v}, \text{ s.t. } \mathbf{C}\mathbf{v} = \mathbf{1}, \mathbf{v} \in \{0,1\}^{nm\times 1} \quad (1)$$

The binary matrix $\mathbf{C} \in \mathbb{R}^{nm\times nm}$ encodes one-to-one mapping constraints: $\forall a \sum_i \mathbf{v}_{ia} = 1$ and $\forall i \sum_a \mathbf{v}_{ia} = 1$. This is known to be NP-hard, so we relax the problem by dropping both the binary and the mapping constraints, and solve

$$\mathbf{v}^* = \underset{\mathbf{v}}{\operatorname{argmax}}\, \mathbf{v}^\top \mathbf{M}\mathbf{v}, \text{ s.t. } \|\mathbf{v}\|_2 = 1 \quad (2)$$

The optimal $\mathbf{v}^*$ is then given by the leading eigenvector of the matrix $\mathbf{M}$. Since $\mathbf{M}$ has non-negative elements, by using Perron-Frobenius arguments, the elements of $\mathbf{v}^*$ are in the interval $[0,1]$, and we interpret $\mathbf{v}^*_{ia}$ as the confidence that i matches a.

Learning. We estimate the matrix $\mathbf{M}$ parameterized in terms of unary and pair-wise point features computed over input images and represented as deep feature hierarchies. We learn the feature hierarchies end-to-end in a loss function that also integrates the matching layer. Specifically, given a training set of correspondences between pairs of images, we adapt the parameters so that the matching minimizes the error, measured as a sum of distances between predicted and ground truth correspondences. In our experiments, we work with graphs constructed over points that correspond to the 2d image projections of the 3d structure of the same physical object in motion (in the context of videos), or over point configurations that correspond to the same semantic category (matching instances of visual categories, e.g. different birds). The main challenge is the propagation of derivatives of the loss function through a factorization of the affinity matrix $\mathbf{M}$, followed by matching (in our formulation, this is an optimization problem, solved using eigen-decomposition) and finally the full feature extraction hierarchy used to compute the unary and pair-wise point representations.

2.1. Derivation Preliminaries

In practice, we build an end-to-end deep network that integrates a feature extracting component that outputs the required descriptors $\mathbf{F}$ for building the matrix $\mathbf{M}$. We solve the assignment problem (2) and compute a matching loss $L(\mathbf{v}^*)$ between the solution $\mathbf{v}^*$ and the ground-truth. The network must be able to pass gradients w.r.t the loss function from the last to the first layer. The key gradients to compute – which we cover in §3 – are $\partial \mathbf{L}/\partial \mathbf{M}$ and $\partial \mathbf{L}/\partial \mathbf{F}$. This computation could be difficult in the absence of an appropriate factorization, as the computational and memory costs become prohibitive. Moreover, as some of our layers implement complex matrix functions, a matrix generalization of back-propagation is necessary [15] for systematic derivations and computational efficiency. In the sequel we cover its main intuition and refer to [15] for details.

Matrix backpropagation. We denote $\mathbf{A} : \mathbf{B} = \text{Tr}(\mathbf{A}^\top \mathbf{B}) = \text{vec}(\mathbf{A})\text{vec}(\mathbf{B})^\top$. For matrix derivatives, if we denote by f a function that outputs $f(\mathbf{X}) = \mathbf{Y}$ and by L the network loss, the basis for the derivation starts from the Taylor expansion of the matrix functions [26] at the two layers. By deriving the functional $\mathcal{L}$ expresssing the total variation $d\mathbf{Y}$ in terms of $d\mathbf{X}$,

$$d\mathbf{Y} = \mathcal{L}(d\mathbf{X}) \quad (3)$$

and then using that

$$\frac{\partial L \circ f}{\partial \mathbf{X}} : d\mathbf{X} = \frac{\partial L}{\partial \mathbf{Y}} : \mathcal{L}(d\mathbf{X}) = \mathcal{L}^*(\frac{\partial L}{\partial \mathbf{Y}}) : d\mathbf{X} \quad (4)$$

we obtain the equality $\partial(L\circ f)/\partial \mathbf{X} = \mathcal{L}^*(\partial L/\partial \mathbf{Y})$, where $\mathcal{L}^*$ is the adjoint operator of $\mathcal{L}$. This recipe for finding the partial derivatives is used across all of our network layers. The derivations of $\mathcal{L}$ and $\mathcal{L}^*$ are layer-specific and are given in the following sections.

2.2. Affinity Matrix Factorization

Zhou and De la Torre [38] introduced a novel factorization of the matrix $\mathbf{M}$ that is generally applicable to all state-of-the-art graph matching methods. It explicitly exposes the graph structure of the set of points and the unary and pair-wise scores between nodes and edges, respectively,

$$\mathbf{M} = [\text{vec}(\mathbf{M}_p)] + (\mathbf{G}_2 \otimes \mathbf{G}_1)[\text{vec}(\mathbf{M}_e)](\mathbf{H}_2 \otimes \mathbf{H}_1)^\top \quad (5)$$

where $[\mathbf{x}]$ represents the diagonal matrix with $\mathbf{x}$ on the main diagonal, and $\otimes$ is the Kronecker product. The matrix

$\mathbf{M}_p \in \mathbb{R}^{n\times m}$ represents the unary term, measuring node-to-node similarities, whereas $\mathbf{M}_e \in \mathbb{R}^{p\times q}$ measures edge-to-edge similarity; p, q are the numbers of edges in each graph, respectively. The two matrices encode the first-order and second-order potentials. To describe the structure of each graph, we define, as in [38], the node-edge incidence matrices as $\mathbf{G}, \mathbf{H} \in \{0,1\}^{n\times p}$, where $g_{ic} = h_{jc} = 1$ if the c^{th} edge starts from the i^{th} node and ends at the j^{th} node. We have two pairs, $\{\mathbf{G}_1, \mathbf{H}_1\} \in \{0,1\}^{n\times p}$ and $\{\mathbf{G}_2, \mathbf{H}_2\} \in \{0,1\}^{m\times q}$, one for each image graph.

One simple way to build $\mathbf{M}_e$ and $\mathbf{M}_p$ is

$$\mathbf{M}_e = \mathbf{X}\mathbf{\Lambda}\mathbf{Y}^\top, \mathbf{M}_p = \mathbf{U}^1\mathbf{U}^{2\top} \tag{6}$$

where $\mathbf{X} \in \mathbb{R}^{p\times 2d}$ and $\mathbf{Y} \in \mathbb{R}^{q\times 2d}$ are the per-edge feature matrices, constructed such that for any c^{th} edge that starts from the i^{th} node and ends at the j^{th} node, we set the edge descriptor as the concatenation of the descriptors extracted at the two nodes

$$\mathbf{X}_c = [\mathbf{F}^1_i | \mathbf{F}^1_j], \mathbf{Y}_c = [\mathbf{F}^2_i | \mathbf{F}^2_j] \tag{7}$$

The matrices $\mathbf{F}^1, \mathbf{U}^1 \in \mathbb{R}^{n\times d}$ and $\mathbf{F}^2, \mathbf{U}^2 \in \mathbb{R}^{m\times d}$ contain per-node feature vectors of dimension d, extracted at possibly different levels in the network, and $\mathbf{\Lambda}$ is a $2d \times 2d$ *block-symmetric* parameter matrix. Superscripts $1, 2$ indicate over which input image (source or target) are the features computed.

3. Deep Network Optimization for Graph Matching

In this section we describe how to integrate and learn the graph matching model end-to-end, by implementing the required components in an efficient way. This allows us to back-propagate gradients all the way from the loss function down to the feature layers. The main components of our approach are shown in Fig. 1.

3.1. Affinity Matrix Layer

If we define the node-to-node adjacency matrices $\mathbf{A}_1 \in \{0,1\}^{n\times n}, \mathbf{A}_2 \in \{0,1\}^{m\times m}$, with $a_{ij} = 1$ if there is an edge from the i^{th} node to the j^{th} node, then

$$\mathbf{A}_1 = \mathbf{G}_1\mathbf{H}_1^\top, \mathbf{A}_2 = \mathbf{G}_2\mathbf{H}_2^\top \tag{8}$$

The *Affinity Matrix* layer receives as input the required hierarchy of features, and the adjacency matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ used to reconstruct the optimal $\mathbf{G}_1, \mathbf{H}_1, \mathbf{G}_2, \mathbf{H}_2$ matrices, which verify the equations (8). It is easier to describe the connectivity of the graphs by adjacency matrices than by node-edge incidence matrices, but we still need the latter for efficient backward passes at higher layers of the network. Next, we describe the forward and the backward passes of this layer, as parts of the trainable deep network.

Forward pass.

1. Given $\mathbf{A}_1$, $\mathbf{A}_2$, recover the matrices $\mathbf{G}_1, \mathbf{H}_1, \mathbf{G}_2, \mathbf{H}_2$, such that $\mathbf{A}_1 = \mathbf{G}_1\mathbf{H}_1^\top, \mathbf{A}_2 = \mathbf{G}_2\mathbf{H}_2^\top$
2. Given $\mathbf{F}^1, \mathbf{F}^2$, build $\mathbf{X}, \mathbf{Y}$ according to (7)
3. Build $\mathbf{M}_e = \mathbf{X}\mathbf{\Lambda}\mathbf{Y}^\top$
4. Given $\mathbf{U}^1, \mathbf{U}^2$, build $\mathbf{M}_p = \mathbf{U}^1\mathbf{U}^{2\top}$
5. Build $\mathbf{M}$ according to (5) and make $\mathbf{G}_1, \mathbf{H}_1, \mathbf{G}_2, \mathbf{H}_2$ available for the upper layers

Backward pass. Assuming the network provides $\partial L/\partial \mathbf{M}_e$ and $\partial L/\partial \mathbf{M}_p$, this layer must return $\partial L/\partial \mathbf{F}^1$, $\partial L/\partial \mathbf{F}^2$, $\partial L/\partial \mathbf{U}^1$ and $\partial L/\partial \mathbf{U}^2$; it must also compute $\partial L/\partial \mathbf{\Lambda}$ in order to update the matrix $\mathbf{\Lambda}$. We assume $\partial L/\partial \mathbf{M}_e$ and $\partial L/\partial \mathbf{M}_p$ as input, and not $\partial L/\partial \mathbf{M}$, because the subsequent layer can take advantage of this special factorization for efficient computation. We note that matrix $\mathbf{\Lambda}$ must have the following form in order for $\mathbf{M}$ to be symmetric with positive elements

$$\mathbf{\Lambda} = \begin{pmatrix} \mathbf{\Lambda}_1 & \mathbf{\Lambda}_2 \\ \mathbf{\Lambda}_2 & \mathbf{\Lambda}_1 \end{pmatrix}, \mathbf{\Lambda}_{ij} > 0, \forall i, j \tag{9}$$

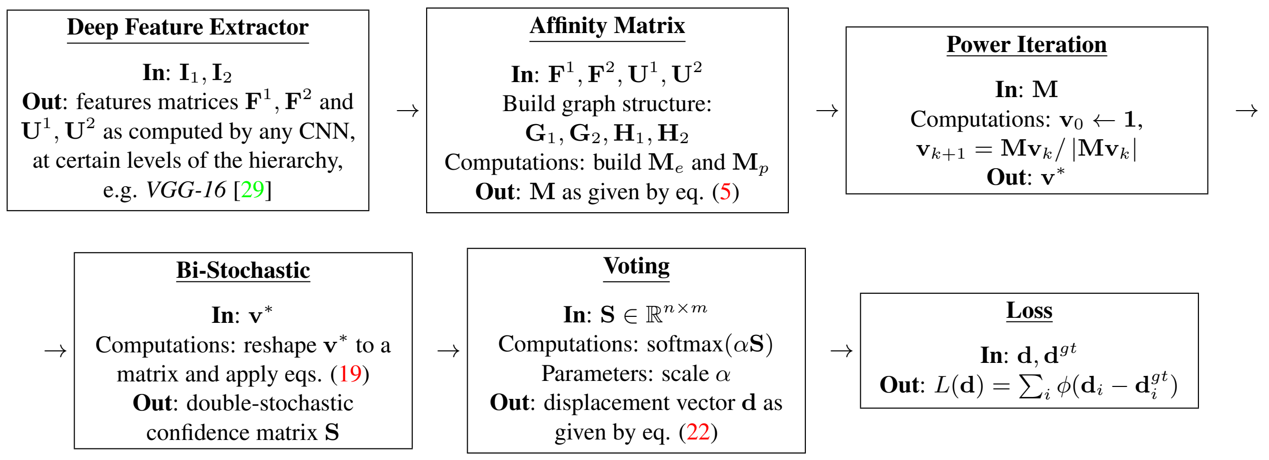
Writing the variation of the loss layer in terms of the variation of edge matrix and using the recipe (4),

$$\begin{aligned} dL &= \frac{\partial L}{\partial \mathbf{M}_e} : d\mathbf{M}_e = \frac{\partial L}{\partial \mathbf{M}_e} : d(\mathbf{X}\mathbf{\Lambda}\mathbf{Y}^\top) \\ &= \frac{\partial L}{\partial \mathbf{M}_e} : (d\mathbf{X}\mathbf{\Lambda}\mathbf{Y}^\top + \mathbf{X}d\mathbf{\Lambda}\mathbf{Y}^\top + \mathbf{X}\mathbf{\Lambda}d\mathbf{Y}^\top) \\ &= \frac{\partial L}{\partial \mathbf{M}_e}\mathbf{Y}\mathbf{\Lambda}^\top : d\mathbf{X} + \mathbf{X}^\top\frac{\partial L}{\partial \mathbf{M}_e}\mathbf{Y} : d\mathbf{\Lambda} + \\ &+ \frac{\partial L}{\partial \mathbf{M}_e}^\top \mathbf{X}\mathbf{\Lambda} : d\mathbf{Y} \end{aligned} \tag{10}$$

We identify the terms

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{X}} &= \frac{\partial L}{\partial \mathbf{M}_e}\mathbf{Y}\mathbf{\Lambda}^\top \\ \frac{\partial L}{\partial \mathbf{\Lambda}} &= \mathbf{X}^\top\frac{\partial L}{\partial \mathbf{M}_e}\mathbf{Y} \\ \frac{\partial L}{\partial \mathbf{Y}} &= \frac{\partial L}{\partial \mathbf{M}_e}\mathbf{X}\mathbf{\Lambda} \end{aligned} \tag{11}$$

To compute the partial derivatives $\partial L/\partial \mathbf{F}^1$ and $\partial L/\partial \mathbf{F}^2$, we identify and sum up the corresponding $1 \times d$ sub-blocks in the matrices $\partial L/\partial \mathbf{X}$ and $\partial L/\partial \mathbf{Y}$. The partial derivative $\partial L/\partial \mathbf{\Lambda}$ is used to compute the derivatives $\partial L/\partial \mathbf{\Lambda}_1$ and $\partial L/\partial \mathbf{\Lambda}_2$. Note that in implementing the positivity condition from (9), one can use a **ReLU** unit.


Deep Feature Extractor
In: $\mathbf{I}_1, \mathbf{I}_2$
Out: features matrices $\mathbf{F}^1, \mathbf{F}^2$ and $\mathbf{U}^1, \mathbf{U}^2$ as computed by any CNN, at certain levels of the hierarchy, e.g. *VGG-16* [29]

$\rightarrow$

Affinity Matrix
In: $\mathbf{F}^1, \mathbf{F}^2, \mathbf{U}^1, \mathbf{U}^2$
Build graph structure: $\mathbf{G}_1, \mathbf{G}_2, \mathbf{H}_1, \mathbf{H}_2$
Computations: build $\mathbf{M}_e$ and $\mathbf{M}_p$
Out: $\mathbf{M}$ as given by eq. (5)

$\rightarrow$

Power Iteration
In: $\mathbf{M}$
Computations: $\mathbf{v}_0 \leftarrow \mathbf{1}$, $\mathbf{v}_{k+1} = \mathbf{M}\mathbf{v}_k / |\mathbf{M}\mathbf{v}_k|$
Out: $\mathbf{v}^*$

$\rightarrow$

$\rightarrow$

Bi-Stochastic
In: $\mathbf{v}^*$
Computations: reshape $\mathbf{v}^*$ to a matrix and apply eqs. (19)
Out: double-stochastic confidence matrix $\mathbf{S}$

$\rightarrow$

Voting
In: $\mathbf{S} \in \mathbb{R}^{n \times m}$
Computations: softmax$(\alpha \mathbf{S})$
Parameters: scale α
Out: displacement vector $\mathbf{d}$ as given by eq. (22)

$\rightarrow$

Loss
In: $\mathbf{d}, \mathbf{d}^{gt}$
Out: $L(\mathbf{d}) = \sum_i \phi(\mathbf{d}_i - \mathbf{d}_i^{gt})$

Figure 1: Computational pipeline of our fully trainable graph matching model. In training, gradients w.r.t. the loss function are passed through a deep feature extraction hierarchy controlling the unary and pair-wise terms associated to the nodes and edges of the two graphs, the factorization of the resulting affinity matrix, the eigen-decomposition solution of the matching problem, and the voting-based assignment layer. For each module we show the inputs, outputs and the key variables involved. Detailed derivations for the associated computations are given in the corresponding paper sections.

3.2. Power Iteration Layer

Computing the leading eigenvector $\mathbf{v}^*$ of the affinity matrix $\mathbf{M}$ can be done using power iterations

$$\mathbf{v}_{k+1} = \frac{\mathbf{M}\mathbf{v}_k}{\|\mathbf{M}\mathbf{v}_k\|} \tag{12}$$

We initialize $\mathbf{v}_0 = \mathbf{1}$. We use $\|\cdot\|$ as the ℓ_2 norm, $\|\cdot\|_2$.

Forward pass. We run the assignment from (12) for N iterations, and output the vector $\mathbf{v}^* = \mathbf{v}_N$. Recall that $\mathbf{M} \in \mathbb{R}^{nm \times nm}$, where n, m are the number of nodes in each graph and p, q respectively the number of edges, the time complexity of the forward pass, per power iteration, is $O(n^2m^2)$, when the matrix $\mathbf{M}$ is dense. If we exploit the sparsity of $\mathbf{M}$ the cost drops to $O(\mathbf{nnz})$ where $\mathbf{nnz}$ represents the number of non-zero elements of the matrix $\mathbf{M}$, being equal to $pq + nm$.

Backward pass. To compute gradients, we express the variation of the loss and identify the required partial derivatives

$$\begin{aligned} dL &= \frac{\partial L}{\partial \mathbf{v}_{k+1}} : d\mathbf{v}_{k+1} = \frac{\partial L}{\partial \mathbf{v}_{k+1}} : d\frac{\mathbf{M}\mathbf{v}_k}{\|\mathbf{M}\mathbf{v}_k\|} \\ d\frac{\mathbf{M}\mathbf{v}_k}{\|\mathbf{M}\mathbf{v}_k\|} &= \frac{(\mathbf{I} - \mathbf{v}_{k+1}\mathbf{v}_{k+1}^\top)}{\|\mathbf{M}\mathbf{v}_k\|} d(\mathbf{M}\mathbf{v}_k) = \\ &= \frac{(\mathbf{I} - \mathbf{v}_{k+1}\mathbf{v}_{k+1}^\top)}{\|\mathbf{M}\mathbf{v}_k\|}(d\mathbf{M}\mathbf{v}_k + \mathbf{M}d\mathbf{v}_k) \end{aligned} \tag{13}$$

Knowing that $\mathbf{y} : \mathbf{A}\mathbf{x} = \mathbf{y}\mathbf{x}^\top : \mathbf{A} = \mathbf{A}^\top\mathbf{y} : \mathbf{x}$, and using the symmetry of $\mathbf{M}$, we derive

$$\begin{aligned} dL = &\frac{(\mathbf{I} - \mathbf{v}_{k+1}\mathbf{v}_{k+1}^\top)}{\|\mathbf{M}\mathbf{v}_k\|} \frac{\partial L}{\partial \mathbf{v}_{k+1}} \mathbf{v}_k^\top : d\mathbf{M} + \\ &+ \mathbf{M}\frac{(\mathbf{I} - \mathbf{v}_{k+1}\mathbf{v}_{k+1}^\top)}{\|\mathbf{M}\mathbf{v}_k\|} \frac{\partial L}{\partial \mathbf{v}_{k+1}} : d\mathbf{v}_k \end{aligned} \tag{14}$$

Noticing that $d\mathbf{v}_k$ is still a function of $d\mathbf{M}$, the gradients $\partial L / \partial \mathbf{M}$ and $\partial L / \partial \mathbf{v}_k$ are iteratively built:

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{M}} &= \sum_k \frac{(\mathbf{I} - \mathbf{v}_{k+1}\mathbf{v}_{k+1}^\top)}{\|\mathbf{M}\mathbf{v}_k\|} \frac{\partial L}{\partial \mathbf{v}_{k+1}} \mathbf{v}_k^\top \\ \frac{\partial L}{\partial \mathbf{v}_k} &= \mathbf{M}\frac{(\mathbf{I} - \mathbf{v}_{k+1}\mathbf{v}_{k+1}^\top)}{\|\mathbf{M}\mathbf{v}_k\|} \frac{\partial L}{\partial \mathbf{v}_{k+1}} \end{aligned} \tag{15}$$

where we receive $\partial L / \partial \mathbf{v}^*$ from the upper network layers. The computational cost of (15) is $O(m^2n^2)$ – regardless of the sparsity of $\mathbf{M}$ – and the memory complexity is $\Theta(m^2n^2)$. Such costs are prohibitive in practice. By employing techniques of **matrix back-propagation** [15], we can exploit the special factorization (5) of matrix $\mathbf{M}$, to make operations both memory and time efficient. In fact, set aside efficiency, it would not be obvious how a classic approach would be used in propagating derivatives through a complex factorization like (5). Exploiting (5) and the recipe from (4), and omitting for clarity the term $\mathbf{M}_p$, we obtain (16) as detailed in Fig. 2.

Note that $(\cdot)_{n \times m}$ is the operator that reshapes an $nm \times 1$ vector into an $n \times m$ matrix. For derivations we use the property that, for any compatible matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{V}$, $(\mathbf{A} \otimes \mathbf{B})^\top \mathrm{vec}(\mathbf{V}) = \mathrm{vec}(\mathbf{B}^\top \mathbf{V}\mathbf{A})$. Consequently, by also considering the unary term $\mathbf{M}_p$, it follows that

$$
\begin{aligned}
\frac{(\mathbf{I}-\mathbf{v}_{k+1}\mathbf{v}_{k+1}^\top)}{\|\mathbf{M}\mathbf{v}_k\|}\frac{\partial L}{\partial \mathbf{v}_{k+1}}\mathbf{v}_k^\top : d\mathbf{M} &= \frac{(\mathbf{I}-\mathbf{v}_{k+1}\mathbf{v}_{k+1}^\top)}{\|\mathbf{M}\mathbf{v}_k\|}\frac{\partial L}{\partial \mathbf{v}_{k+1}}\mathbf{v}_k^\top : (\mathbf{G}_2\otimes\mathbf{G}_1)[\mathrm{vec}(d\mathbf{M}_e)](\mathbf{H}_2\otimes\mathbf{H}_1)^\top \\
&= \mathrm{diag}\left((\mathbf{G}_2\otimes\mathbf{G}_1)^\top \frac{(\mathbf{I}-\mathbf{v}_{k+1}\mathbf{v}_{k+1}^\top)}{\|\mathbf{M}\mathbf{v}_k\|}\frac{\partial L}{\partial \mathbf{v}_{k+1}}\mathbf{v}_k^\top(\mathbf{H}_2\otimes\mathbf{H}_1)\right) : d\mathbf{M}_e \\
&= (\mathbf{G}_2\otimes\mathbf{G}_1)^\top \frac{(\mathbf{I}-\mathbf{v}_{k+1}\mathbf{v}_{k+1}^\top)}{\|\mathbf{M}\mathbf{v}_k\|}\frac{\partial L}{\partial \mathbf{v}_{k+1}} \odot (\mathbf{H}_2\otimes\mathbf{H}_1)^\top\mathbf{v}_k : d\mathbf{M}_e \\
&= \mathbf{G}_1^\top\left(\frac{(\mathbf{I}-\mathbf{v}_{k+1}\mathbf{v}_{k+1}^\top)}{\|\mathbf{M}\mathbf{v}_k\|}\frac{\partial L}{\partial \mathbf{v}_{k+1}}\right)_{n\times m}\mathbf{G}_2 \odot \mathbf{H}_1^\top\Big(\mathbf{v}_k\Big)_{n\times m}\mathbf{H}_2 : d\mathbf{M}_e \qquad (16)
\end{aligned}
$$

Figure 2: Derivations expressing the variation of the loss function w.r.t the variation of the edge affinity matrix $\mathbf{M}_e$, given the variation of the loss function w.r.t the variation of the whole affinity matrix $\mathbf{M}$, from eq. (14)

$$
\frac{\partial L}{\partial \mathbf{M}_e} = \sum_k \mathbf{G}_1^\top\left(\frac{(\mathbf{I}-\mathbf{v}_{k+1}\mathbf{v}_{k+1}^\top)}{\|\mathbf{M}\mathbf{v}_k\|}\frac{\partial L}{\partial \mathbf{v}_{k+1}}\right)_{n\times m}\mathbf{G}_2 \odot \odot\, \mathbf{H}_1^\top(\mathbf{v}_k)_{n\times m}\mathbf{H}_2 \qquad (17)
$$

$$
\frac{\partial L}{\partial \mathbf{M}_p} = \sum_k \frac{(\mathbf{I}-\mathbf{v}_{k+1}\mathbf{v}_{k+1}^\top)}{\|\mathbf{M}\mathbf{v}_k\|}\frac{\partial L}{\partial \mathbf{v}_{k+1}} \odot \mathbf{v}_k \qquad (18)
$$

With careful ordering of operations, the complexities for the backward pass are now $O(\max(m^2q, n^2p))$ and $\Theta(pq)$. Taking into account the sparsity of the node-edge incidence matrices $\mathbf{G}, \mathbf{H}$, efficiency can be further improved.

3.3. Bi-Stochastic Layer

We make the result of the *Power Iteration* layer double-stochastic by mapping the eigenvector $\mathbf{v}^*$ on the L1 constraints of the matching problem (1): $\forall a, \sum_i \mathbf{v}_{ia} = 1$ and $\forall i, \sum_a \mathbf{v}_{ia} = 1$. This is suggested by multiple authors [13, 37, 19], as it was observed to significantly improve performance. We introduce a new *Bi-Stochastic* layer that takes as input any correspondence vector $\mathbf{v}^* \in \mathbb{R}_+^{nm\times 1}$, reshaped to a matrix of size $n\times m$, and outputs the double-stochastic variant, as described in [30]. Even though the original algorithm assumes only square matrices, the process can be generalized as shown in [9].

Forward pass. Given a starting matrix $\mathbf{S}_0 = \big(\mathbf{v}^*\big)_{n\times m}$, we run the following assignments for a number of iterations

$$
\mathbf{S}_{k+1} = \mathbf{S}_k[\mathbf{1}_n^\top\mathbf{S}_k]^{-1}, \mathbf{S}_{k+2} = [\mathbf{S}_{k+1}\mathbf{1}_m]^{-1}\mathbf{S}_{k+1} \qquad (19)
$$

Backward pass. Given a starting gradient $\partial L/\partial \mathbf{S}$, we iteratively compute

$$
\begin{aligned}
\frac{\partial L}{\partial \mathbf{S}_{k+1}} &= [\mathbf{S}_{k+2}\mathbf{1}_m]^{-1}\frac{\partial L}{\partial \mathbf{S}_{k+2}} - \\
&- \mathrm{diag}\left([\mathbf{S}_{k+2}\mathbf{1}_m]^{-2}\frac{\partial L}{\partial \mathbf{S}_{k+2}}\mathbf{S}_{k+2}^\top\right)\mathbf{1}_m^\top \qquad (20)\\
\frac{\partial L}{\partial \mathbf{S}_k} &= \frac{\partial L}{\partial \mathbf{S}_{k+1}}[\mathbf{1}_n^\top\mathbf{S}_{k+1}]^{-1} - \\
&- \mathbf{1}_n\mathrm{diag}\left([\mathbf{1}_n^\top\mathbf{S}_{k+1}]^{-2}\mathbf{S}_{k+1}^\top\frac{\partial L}{\partial \mathbf{S}_{k+1}}\right)^\top \qquad (21)
\end{aligned}
$$

3.4. Converting Confidence-maps to Displacements

We use a special layer, called *Voting* layer to convert from the confidence map $\mathbf{S} \in \mathbb{R}^{n\times m}$ – passed on by the *Bi-Stochastic* layer – to a displacement vector. The idea is to normalize, for each assigned point i, its corresponding candidate scores given by the ith row of $\mathbf{S}$, denoted $\mathbf{S}(i, 1...m)$. We then use it to weight the matrix of positions $\mathbf{P} \in \mathbb{R}^{m\times 2}$ and subtract the position of match i.

$$
\mathbf{d}_i = \frac{\exp[\alpha\mathbf{S}(i,1...m)]}{\sum_j \exp[\alpha\mathbf{S}(i,j)]}\mathbf{P} - \mathbf{P}_i \qquad (22)
$$

where [] is now just a bulkier bracket. In practice we set α to large values (e.g. 200). By softly voting over the fixed candidate locations, our solution can interpolate for more precise localization. We also set confidences to 0 for candidates that are too far away from assigned points, or those added through padding. Hence these do not contribute to the loss, given by

$$
L(\mathbf{d}) = \sum_i \phi(\mathbf{d}_i - \mathbf{d}_i^{gt}) \qquad (23)
$$

where $\phi(\mathbf{x}) = \sqrt{\mathbf{x}^\top\mathbf{x} + \epsilon}$ is a robust penalty term, and $\mathbf{d}^{gt}$ is the ground-truth displacement from the source point to the correct assignment. This layer is implemented in multiple, fully differentiable steps: a) first, scale the input by α, b) use a spatial map for discarding candidate locations that are further away than a certain threshold from the starting location and use it to modify the confidence maps, c) use a softmax layer to normalize the confidence maps, d) compute the displacement map. The discard map sets the confidences to 0 for points that are further away than a certain distance, or for points that were added through padding. Such points do not contribute in the final loss, given by (23).

4. Experiments

In this section we describe the models used as well as detailed experimental matching results, both quantitative (including ablation studies) and qualitative, on three challenging datasets: MPI Sintel, CUB, and PASCAL keypoints.

Deep feature extraction network. We rely on the VGG-16 architecture from [29], that is pretrained to perform classification in the ImageNet ILSVRC [28] but we can use any other deep network architecture. We implement our deep learning framework in MatConvNet [31]. As edge features $\mathbf{F}$ we use the output of layer `relu5_1` (and the entire hierarchy under it), and for the node features $\mathbf{U}$ we use the output of layer `relu4_2` (with the parameters of the associated hierarchy under it). Features are all normalized to 1 through normalization layers, right before they are used to compute the affinity matrix $\mathbf{M}$. We conduct experiments for geometric and semantic correspondences on MPI-Sintel [6], Caltech-UCSD Birds-200-2011 [32] and PASCAL VOC [11] with Berkeley annotations [5].

Matching networks. *GMNwVGG* is our proposed Graph Matching Network based on a VGG feature extractor. The suffix *-U* means that default (initial) weights are used; *-T* means trained end-to-end; the *GMNwVGG-T w/o V* variant does not use, at testing, the *Voting* layer in order to compute the displacements, but directly assigns indices of maximum value across the rows of the confidence map $\mathbf{S}$, as correspondences. *NNwVGG* gives nearest-neighbour matching based on deep node descriptors $\mathbf{U}$.

MPI-Sintel. Besides the main datasets CUB and PASCAL, typically employed in validating matching methods, we also use Sintel in order to demonstrate the generality and flexibility of the formulation. The Sintel input images are consecutive frames in a movie and exhibit large displacements, large appearance changes, occlusion, non-rigid movements and complex atmospheric effects (only included in the *final* set of images). The Sintel training set consists of 23 video sequences (organized as folders) and 1064 frames. In order to make sure that we are fairly training and evaluating, as images from the same video depict instances of the same objects, we split the data into 18 folders (i.e. 796 image pairs) for training and 5 folders (i.e. 245 image pairs) for testing. To be able to set a high number of correspondences under the constraints of memory usage and available computational resolutions of our descriptors, we partition the input images in 4×4 blocks padded for maximum displacement, which we match one by one. Note that for this particular experiment we do not use the *Bi-Stochastic* layer, as the one-to-one mapping constraints do not apply to the nature of this problem (the assignment can be many-to-one e.g. in scaling). Notice that our model is designed to establish correspondences between two images containing similar structures, generally from different scenes, not tailored explicitly for optical flow, where additional smoothness constraints can be exploited. A main point of our Sintel experiments is to show the scalability of our method which operates with affinity matrices of size $10^6 \times 10^6$. A complete forward and backward pass runs in roughly **2 seconds** on a 3.2 Ghz Intel Xeon machine, with Titan X Pascal GPU. [1] We show quantitative results in table 1. We use the Percentage of Correct Keypoints metric to test our matching performance, with a threshold of 10 pixels. We compare against other state-of-the-art matching methods, following the protocol in [8]. Notice that even untrained, our network remains competitive, as the graph structure acts as a regularizer. After training, the network has significantly increased accuracy. Visual examples are given in fig. 4.

Method	PCK@10 pixels
SIFT flow [22]	89.0
DaisyFF [34]	87.3
DSP [17]	85.3
DM [33]	89.2
UCN [8]	91.5
NNwVGG-U	85.9
NNwVGG-T	87.01
GMNwVGG-U	88.03
GMNwVGG-T	**92.6**

Table 1: Comparative matching results for Sintel.

CUB. This dataset contains 11,788 images of 200 bird categories, with bounding box object localization and 15 annotated parts. We use the test set built by [16] which consists of 5,000 image pairs that are within 3 nearest neighbors apart on the pose graph of [18], so we can expect that birds are in somewhat similar poses in each pair to be tested. But across all images, there is a significant variation in pose, articulation and appearance. To train our model, we sample

[1] One possible speed-up could be to not propagate gradients through expensive layers like the relaxation during each power iteration, but do it only once at convergence[1, 2]. In principle, this would not be the correct gradient and in our tests this approach indeed did not converge.

random pairs of images from the training set of CUB-200-2011, which are not present in the test set. The number of points in the two graphs are maximum $n = 15$ and $m = 256$ – as sampled from a 16×16 grid – with a Delaunay triangulation on the first graph, and fully connected on the second. In the *Voting* layer, we no longer discard candidate locations that are too far away from the source points. A complete forward and backward pass runs in 0.3 seconds.

Method	EPE (in pixels)	PCK@0.05
GMNwVGG-U	41.63	0.63
NNwVGG-U	59.05	0.46
GMNwVGG-T w/o V	18.22	0.83
GMNwVGG-T	**17.02**	**0.86**

Table 2: Our models (with ablations) on CUB.

We show quantitative results in table 2. The "PCK@α" metric [35] represents the percentage of predicted correspondences that are closer than $\alpha\sqrt{w^2 + h^2}$ from ground-truth locations, where w, h are image dimensions. Qualitative results are shown in fig. 5.

Our end-to-end fully trainable matching models significantly outperform nearest neighbor approaches (EPE error is halved) based on deep features or similar graph matching formulations based on deep features not refined jointly with the assignment model. Our model *GMNwVGG-T w/o V* which does not use, at testing, the *Voting* layer in order to compute the displacements is inferior to the soft-voting mechanism of our compete method *GMNwVGG-T*. We also achieve state-of-the-art results compared to UCN [8], WarpNet [16], SIFT [24] and DSP [17], *cf.* fig. 3.

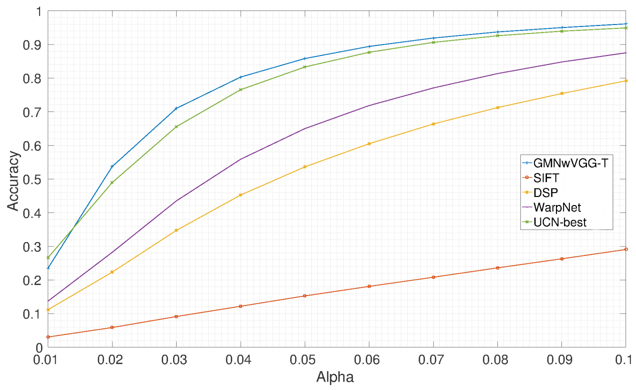


Figure 3: Our methods and other state-of-the-art techniques on CUB.

PASCAL VOC keypoints. This dataset is an extension of PASCAL VOC 2011 and contains annotations of body parts for 20 semantic classes. There are 7,000 annotated examples for training, and 1,700 for testing, but we can form pairs between any two examples of the same class. While the numbers of examples for each class are heavily imbalanced, at training we draw random examples from each class, according to its corresponding probability. At train and test time we crop each example around its bounding box, re-scale it to 256×256 and pass it to the network. We train one matching network for all classes, hence our model is agnostic to the semantic category. This dataset is considerably more challenging than CUB: bounding-boxes can range from very large to extremely small (e.g. 30×30), the training and testing pairs cover combinations of two images from each class (meaning that we cannot expect them to contain objects in the same pose) and the appearance variation is more extreme. We rely on the same protocol for evaluation and setting meta-parameters as in the CUB experiment. Results are shown in Fig. 6, and comparisons in table 3.

Method	PCK@0.1 (Class average)
conv4 flow [23]	24.9
SIFT flow [22]	24.7
UCN [8]	38.9
NNwVGG-U	25.4
GMNwVGG-U	29.8
GMNwVGG-T	**40.6**

Table 3: Our models as well as other state-of-the art approaches on PASCAL VOC.

We test following the same protocol as for **CUB-200-2011** and obtain improvements over state-of-the-art. Notice that results of UCN [8] differ from the paper, as we use the straight average for the 20 semantic classes. Their paper reports the weighted average based on class frequency – under that metric UCN obtains PCK@0.1 = 44 and we obtain 45.3, so the improvement is preserved compared to the direct averaging case.

5. Conclusions

We have presented an end-to-end learning framework for graph matching with general applicability to models containing deep feature extraction hierarchies and combinatorial optimization layers. We formulate the problem as a quadratic assignment under unary and pair-wise node relations represented using deep parametric feature hierarchies. All model parameters are trainable and the graph matching optimization is included within the learning formulation. As such, the main challenges are the calculation of back-propagated derivatives through complex matrix layers and the implementation of the entire framework (factorization of the affinity matrix, bi-stochastic layers) in a computationally efficient manner. Our experiments and ablation studies on diverse datasets like PASCAL VOC keypoints, Sintel and CUB show that fully learned graph matching models

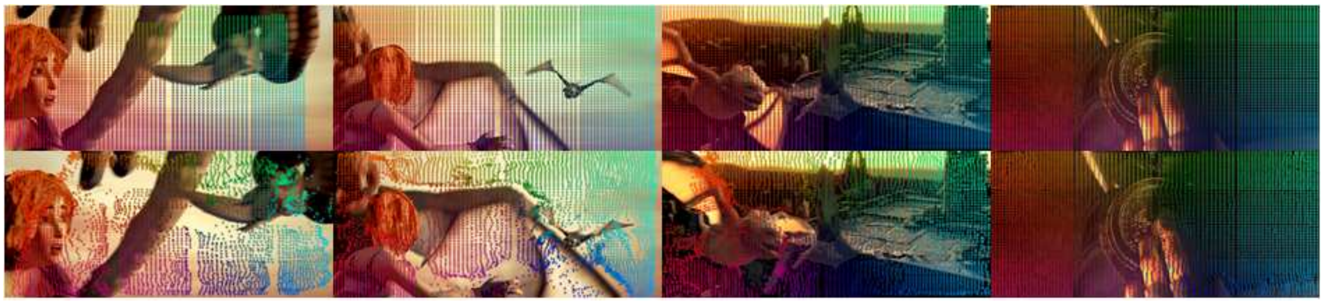


Figure 4: Four visual examples on the MPI-Sintel test partition, which exhibit large motions and occlusion areas. From **top** to **bottom**: the source images with the initial grid of points overlaid and the target images with the corresponding matches as found by our fully trained network. Colors are unique and they encode correspondences. Even for fast moving objects, points tend to track the surface correctly, without sliding – see e.g. the dragon's wing, claw, and the flying monster.

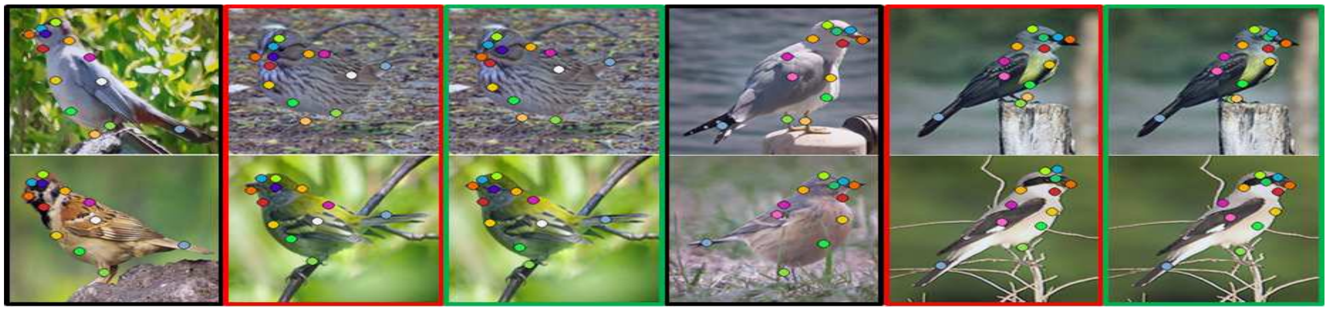


Figure 5: Four qualitative examples of our best performing network *GMNwVGG-T*, on the CUB-200-2011 test-set. Images with a black contour represent the source, whereas images with a red contour represent targets. Color-coded correspondences are found by our method. The green framed images show ground-truth correspondences. The colors of the drawn circular markers uniquely identify 15 semantic keypoints.

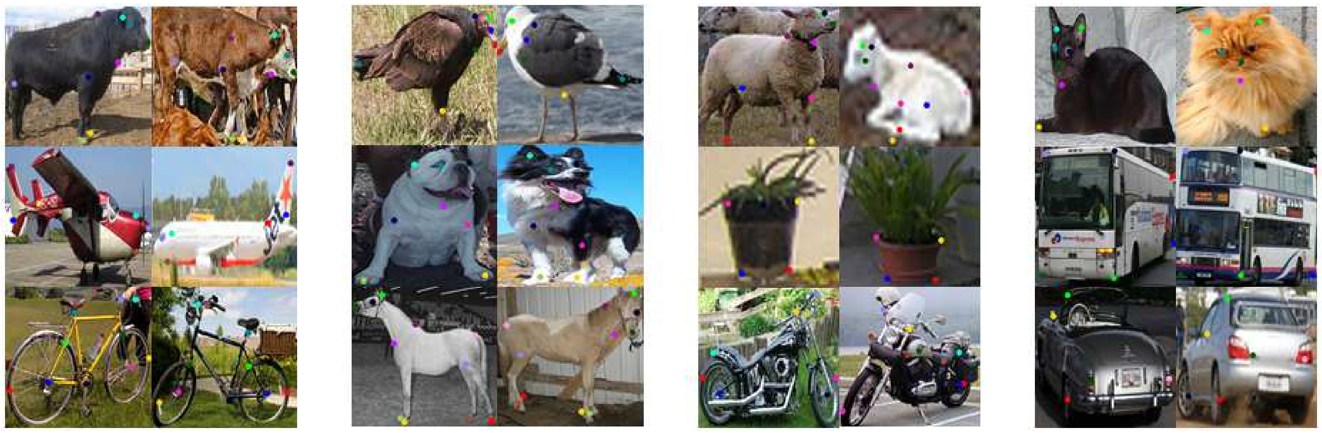


Figure 6: Twelve qualitative examples of our best performing network *GMNwVGG-T* on the PASCAL VOC test-set. For every pair of examples, the left shows the source image and the right the target. Colors identify the computed assignments between points. The method finds matches even under extreme appearance and pose changes.

surpass nearest neighbor counterparts, or approaches that use deep feature hierarchies that were not refined jointly with (and constrained by) the quadratic assignment problem.

Acknowledgments: This work was supported in part by the European Research Council Consolidator grant SEED, CNCS-UEFISCDI PN-III-P4-ID-PCE-2016-0535, EU Horizon 2020 Grant No. 688835 DE-ENIGMA, and SSF.

References

[1] B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. *CoRR*, abs/1703.00443, 2017.

[2] J. T. Barron and B. Poole. The fast bilateral solver. *CoRR*, abs/1511.03296, 2015.

[3] A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 26–33. IEEE, 2005.

[4] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.

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

[6] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision*, pages 611–625. Springer, 2012.

[7] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, and A. J. Smola. Learning graph matching. *IEEE transactions on pattern analysis and machine intelligence*, 31(6):1048–1058, 2009.

[8] C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker. Universal correspondence network. In *Advances in Neural Information Processing Systems*, pages 2406–2414, 2016.

[9] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. In *NIPS*, volume 2, page 6, 2006.

[10] O. Duchenne, F. Bach, I.-S. Kweon, and J. Ponce. A tensor-based algorithm for high-order graph matching. *IEEE transactions on pattern analysis and machine intelligence*, 33(12):2383–2395, 2011.

[11] M. Everingham and J. Winn. The pascal visual object classes challenge 2011 (voc2011) development kit. *Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep*, 2011.

[12] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[13] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Transactions on pattern analysis and machine intelligence*, 18(4):377–388, 1996.

[14] B. Ham, M. Cho, C. Schmid, and J. Ponce. Proposal flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3475–3484, 2016.

[15] C. Ionescu, O. Vantzos, and C. Sminchisescu. Training deep networks with structured layers by matrix backpropagation. *arXiv preprint arXiv:1509.07838*, 2015.

[16] A. Kanazawa, D. W. Jacobs, and M. Chandraker. Warpnet: Weakly supervised matching for single-view reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3253–3261, 2016.

[17] J. Kim, C. Liu, F. Sha, and K. Grauman. Deformable spatial pyramid matching for fast dense correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2307–2314, 2013.

[18] J. Krause, H. Jin, J. Yang, and L. Fei-Fei. Fine-grained recognition without part annotations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5546–5555, 2015.

[19] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1482–1489. IEEE, 2005.

[20] M. Leordeanu, R. Sukthankar, and M. Hebert. Unsupervised learning for graph matching. *International journal of computer vision*, 96(1):28–45, 2012.

[21] M. Leordeanu, A. Zanfir, and C. Sminchisescu. Semi-supervised learning and optimization for hypergraph matching. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2274–2281. IEEE, 2011.

[22] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):978–994, 2011.

[23] J. L. Long, N. Zhang, and T. Darrell. Do convnets learn correspondence? In *Advances in Neural Information Processing Systems*, pages 1601–1609, 2014.

[24] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[25] J. Maciel and J. P. Costeira. A global solution to sparse correspondence problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):187–199, 2003.

[26] J. R. Magnus, H. Neudecker, et al. Matrix differential calculus with applications in statistics and econometrics. 1995.

[27] I. Rocco, R. Arandjelović, and J. Sivic. Convolutional neural network architecture for geometric matching. *arXiv preprint arXiv:1703.05593*, 2017.

[28] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

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

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

[31] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. In *Proceeding of the ACM Int. Conf. on Multimedia*, 2015.

[32] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[33] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep match-

ing. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1385–1392. IEEE, 2013.

[34] H. Yang, W. Lin, and J. Lu. Daisy filter flow: A generalized approach to discrete dense correspondences. CVPR, 2014.

[35] Y. Yang and D. Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2878–2890, 2013.

[36] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. Lift: Learned invariant feature transform. In *European Conference on Computer Vision*, 2016.

[37] R. Zass and A. Shashua. Probabilistic graph and hypergraph matching. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[38] F. Zhou and F. De la Torre. Factorized graph matching. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 127–134. IEEE, 2012.

CVPR2018 Paper Translation

姓名：	张皓飞
学号：	2015301876
班号：	08031502

图匹配的深度学习

Andrei Zanfir[2] and Cristian Sminchisescu[1,2]

andrei.zanfir@imar.ro, cristian.sminchisescu@math.lth.se

[1]Department of Mathematics, Faculty of Engineering, Lund University

[2]Institute of Mathematics of the Romanian Academy

摘要

在节点和配对约束下的图匹配问题是组合优化、机器学习或计算机视觉等许多领域中的基本问题，其中表示节点之间的关系及其邻域结构是至关重要的。我们提出一个端到端的模型，使得能够学习图形匹配过程的所有参数，包括表示为深度特征提取层次的一元节点邻域和两元节点邻域。挑战在于通过求解匹配问题的组合优化层以及有限元，以一种能够从损失函数在完整流水线中实现梯度的一致、有效传播的方式建立模型的不同矩阵计算层并提取层次。我们在 PASCAL VOC 关键点、Sintel 和 CUB 等具有挑战性的数据集上的计算机视觉实验和烧蚀研究表明，基于针对其他问题训练的特征层次结构的匹配模型优于端到端的匹配模型。

1. 介绍和相关工作

图匹配问题——建立两个用局部节点结构和成对关系表示的图之间的对应关系，不管它们是视觉的、几何的或拓扑的——在诸如组合优化、机器学习、图像分析或协作等领域中都很重要。计算机视觉，在结构从运动、目标跟踪、二维和三维形状匹配、图像分类、社会网络分析、自主性驾驶等方面都有应用。本文的重点是匹配基于图的图像表示，但是这种方法广泛地应用于任何图匹配问题，其中一元结构和成对结构可以用具有可训练参数的深特征层次来表示。

不像 RANSAC[12]或迭代最近点[4]等仅限于刚性位移的其他方法，图形匹配自然地编码可用于建模复杂关系和更多样化变换的结构信息。图匹配使用关联矩阵操作，关联矩阵编码两个图中的一元节点集和二元节点集（点）之间的相似性。典型地，它被数学地表示为二次整数程序[25，3]，受一对一映射约束，即，第一组中的每个点必须在第二组中具有唯一的对应关系。这是已知的 nPHARD，所以方法通常通过放宽约束和找到局部最优解来近似求解[19, 38]。

学习图关联矩阵的参数已经由[7，20]或，在更一般的超图匹配模型[10]的背景下，由[21]研究。在这些情况下，参数的数量较低，通常控制点对之间的几何相似性，而不是控制这些点附近的图像结构。近来，通过训练网络以直接优化匹配目标[8、27、16、36]或通过在已建立的匹配体系结构中使用预先训练的深层特征[23、14]，对于将深层特征用于几何和语义视觉匹配任务越来越感兴趣并都取得了相当大的成功。

本文的目的是将（浅）图匹配与深度学习公式相结合。我们建议建立模型，其中图定义在一元节点邻域上，并且基于学习的特征层次计算成对结构。我们建立了一个完整的模型来学习特征层次，使得图匹配工作得最好：特征学习和图匹配模型在一个单一的深层结构中被细化，该深层结构被联合优化以获得一致的结果。在方法上，我们的贡献与计算图的不同矩阵层的构造有关，在矩阵反向传播框架中从损失函数一直到特征层都获得解析导数，强调计算效率。反向传递的公信力，以及基于投票的损失函数。所提出的模型一般不仅适用于匹配不同场景（其主要设计）中拍摄的类别的不同图像，还适用于同一场景的不同图像，或来自视频的不同图像。

2. 问题公式化

输入： 我们定义两个输入图 $G_1=(V_1,E_1)$ 和 $G_2=(V_2,E_2)$ 并且 $|V_1|=n$，$|V_2|=m$。我们的目标是在两个图的节点之间建立分配，从而优化了对应节点和边缘的准则（参见下文）。

图匹配： 令 $\mathbf{v}\in\{0,1\}^{nm\times 1}$ 为一个指示向量，且 $\mathbf{v}_{ia}=1$ 如果 $i\in V_1$ 匹配 $a\in V_2$，否则为 0 作为满射的限制。我们构建一个对称正矩阵 $\mathbf{M}\in\mathbb{R}^{nm\times nm}$，$\mathbf{M}_{ia;jb}$ 代表 $(i,j)\in E_1$ 匹配 $(a,b)\in E_2$ 的结果。对于不形成边缘的对，它们在矩阵中的对应条目被设置为 0。对角线条目包含节点到节点的分数，而非对角线条目包含边缘到边缘的分数。最优 $\mathbf{v}^*$ 的取值可以按如下构建：

$$\mathbf{v}^*=\arg\max_{v}\mathbf{v}^{\mathrm{T}}\mathbf{M}\mathbf{v},\ s.t.\ \mathbf{C}\mathbf{v}=1,\ \mathbf{v}\in\{0,1\}^{nm\times 1}\qquad(1)$$

二进制矩阵 $\mathbf{C}\in\mathbb{R}^{nm\times nm}$ 编码一一映射限制条件：

$$\forall a\sum_i\mathbf{v}_{ia}=1,\forall i\sum_a\mathbf{v}_{ia}=1$$

这是人们所熟知的 NP 完全问题，因此，我们通过删除二进制和映射约束来放松问题，并解决：

$$\mathbf{v}^*=\arg\max_{\mathbf{v}}\mathbf{v}^{\mathrm{T}}\mathbf{M}\mathbf{v},\ s.t.\ \|\mathbf{v}\|_2=1\qquad(2)$$

然后由矩阵 $\mathbf{M}$ 的前导特征向量给出最优的 $\mathbf{v}^*$ 由于 $\mathbf{M}$ 具有非负元素，因此通过使用 Perron-Frobenius 参数，$\mathbf{v}^*$ 的元素在区间 $[0,1]$，我们将 $\mathbf{v}^*_{ia}$ 解释为 i 匹配 a 的置信度。

学习： 我们根据输入图像上计算的一元和二元点特征估计参数化的矩阵 $\mathbf{M}$，并将其表示为深特征层次结构。我们学习的特征层次结构的端到端的损失函数，也集成了匹配层。具体地说，给定一对图像之间的对应训练集，我们调整参数，使得匹配误差最小化，所述误差是以预测与地面真实对应之间的距离之和测量的。在我们的实验中，我们使用在对应于运动中的相同物理对象的 3d 结构的 2d 图像投影的点上构建的图（在视频上下文中），或者使用对应于相同语义类别的点配置（视觉的匹配实例）。类别，例如不同的鸟）。主要的挑战是通过关联矩阵 $\mathbf{M}$ 的因式分解传播损失函数的导数，然后进行匹配（在我们的公式中，这是一个优化问题，使用特征分解来解决），最后是计算所用的全特征提取层次。一元和对偶点表示。

2.1 初等推导

在实践中，我们建立了一个端到端的深层网络，该网络集成了一个特征提取组件，该组件输出构建矩阵 $\mathbf{M}$ 所需的描述符 $\mathbf{F}$。我们解决了指派问题（2），并且计算了解 $\mathbf{v}^*$ 与真是值之间的匹配损失 $L(\mathbf{v}^*)$。网络必须能够将梯度函数从最后一层传递到第一层。我们需要计算的关键梯度是 $\partial\mathbf{L}/\partial\mathbf{M}$ 和 $\partial\mathbf{L}/\partial\mathbf{F}$。由于计算和存储成本变得令人望而却步，在缺乏适当的因式分解的情况下，这种计算会很困难。此外，由于我们的一些层实现复杂的矩阵函数，为了系统的推导和计算效率，反向传播的矩阵推广是必要的[15]。在接下来中，我们涵盖了它的主要直觉，并参考[15]的细节。

矩阵反向传播：

我们使用符号 $\mathbf{A}:\mathbf{B}=tr(\mathbf{A}^{\mathrm{T}}\mathbf{B})=vec(\mathbf{A})vec(\mathbf{B})^{\mathrm{T}}$。对于矩阵的导数，我们用函数记号 f 表示输出为 $f(\mathbf{X})=\mathbf{Y}$ 并且 L 为网络的损失。两层的导数的基础来自于矩阵函数的泰勒展开[26]。通过对 L 函数的求导根据 $d\mathbf{X}$ 求出 $d\mathbf{Y}$，

$$d\mathbf{Y}=L(d\mathbf{X})\qquad(3)$$

并且使用

$$\frac{\partial L\circ f}{\partial\mathbf{X}}:d\mathbf{X}=\frac{\partial L}{\partial\mathbf{Y}}:L(d\mathbf{X})=L^*(\frac{\partial L}{\partial\mathbf{Y}}):d\mathbf{X}\qquad(4)$$

我们取得等式 $\partial(L\circ f)/\partial\mathbf{X}=L^*(\partial L/\partial\mathbf{Y})$，这里的 L^* 是 L 的伴随算子。这个求偏导数的方法用于我们所有的网络层。L 和 L^* 的推导是层特异性的，并在以下各节中给出。

2.2 关联矩阵分解

Zhou 和 De la Torre [38]引入了矩阵 $\mathbf{M}$ 的一种新的因式分解，它通常适用于所有最先进的图匹配方法。它显式地公开了点集的图结构，以及节点和边缘之间的一元和二元得分，

$$\mathbf{M}=[vec(\mathbf{M}_p)]+(\mathbf{G}_2\otimes\mathbf{G}_1)[vec(\mathbf{M}_e)](\mathbf{H}_2\otimes\mathbf{H}_1)^{\mathrm{T}}\ (5)$$

这里的 $[\mathbf{x}]$ 代表以 $\mathbf{x}$ 作为主对角线的对角矩阵，$\otimes$ 是 Kronecker 乘法。矩阵 $\mathbf{M}_p\in\mathbb{R}^{n\times m}$ 代表一元项，

表示点对点的相似度，类似地，$\mathbf{M}_e \in \mathbb{R}^{p\times q}$ 表示边对边的相似度；p，q 是每个图的边的数量。这两个矩阵编码一阶和二阶势能。为了描述每个图的结构，我们定义点对边的关联度矩阵[38] $\mathbf{G},\mathbf{H} \in \{0,1\}^{n\times p}$，这里 $g_{ic} = h_{jc} = 1$ 如果第 c 个边以地 i 个点为起点且以第 j 个点为终点。我们有两对 $\{\mathbf{G}_1,\mathbf{H}_1\} \in \{0,1\}^{n\times p}$ 和 $\{\mathbf{G}_2,\mathbf{H}_2\} \in \{0,1\}^{m\times q}$ 每一个对应着一个图。

一个简单的方式构建 $\mathbf{M}_e$ 和 $\mathbf{M}_p$ 为

$$\mathbf{M}_e = \mathbf{X\Lambda Y}^{\mathrm{T}}, \mathbf{M}_p = \mathbf{U}^1\mathbf{U}^{2\mathrm{T}} \tag{6}$$

这里 $\mathbf{X} \in \mathbb{R}^{p\times 2d}$ 和 $\mathbf{Y} \in \mathbb{R}^{q\times 2d}$ 是每个边缘特征矩阵，构造成使得对于从第 i 个节点开始并结束于第 j 个节点的任何第 c 个边，我们将边描述符设置为在两个节点处提取的描述符的连接

$$\mathbf{X}_c = [\mathbf{F}_i^1 \mid \mathbf{F}_j^1], \mathbf{Y}_c = [\mathbf{F}_i^2 \mid \mathbf{F}_j^2] \tag{7}$$

矩阵 $\mathbf{F}^1,\mathbf{U}^1 \in \mathbb{R}^{n\times d}$ 和 $\mathbf{F}^2,\mathbf{U}^2 \in \mathbb{R}^{m\times d}$ 包含维数 d 的每个节点特征向量，在网络中可能以不同级别提取，并且 $\mathbf{\Lambda}$ 是 $2d\times 2d$ 的块对称参数矩阵。上标 1, 2 指示哪个输入图像（源或目标）是计算的特征。

3. 图匹配的深度网络的优化

在本节中，我们将描述如何通过有效地实现所需的组件来集成和学习端到端的图匹配模型。这使得我们可以将梯度从损失函数向下传播到特征层。我们的方法的主要组成部分如图 1 所示。

3.1 相似矩阵层

如果我们定义点对点的邻接矩阵 $\mathbf{A}_1 \in \{0,1\}^{n\times n}$，$\mathbf{A}_2 \in \{0,1\}^{m\times m}$，且如果有一条连接第 i 和第 j 个节点的边，则 $a_{ij} = 1$ 且

$$\mathbf{A}_1 = \mathbf{G}_1\mathbf{H}_1^{\mathrm{T}}, \mathbf{A}_2 = \mathbf{G}_2\mathbf{H}_2^{\mathrm{T}} \tag{8}$$

相似矩阵层接收所需的特征层次作为输入，以及用于重构最优 $\mathbf{G}_1,\mathbf{H}_1,\mathbf{G}_2,\mathbf{H}_2$ 矩阵的邻接矩阵 $\mathbf{A_1}$ 和 $\mathbf{A_2}$，从而验证公式(8)。用邻接矩阵描述图的连通性比用节点-边缘关联矩阵更容易，但是我们仍然需要后者，以便在网络的较高层进行有效的向后传递。接下来，我们描述了这一层的向前和向后通过，作为可训练的深度网络的一部分。

前向传递：

1. 给出 $\mathbf{A}_1,\mathbf{A}_2$，恢复矩阵 $\mathbf{G}_1,\mathbf{H}_1,\mathbf{G}_2,\mathbf{H}_2$ 使得 $\mathbf{A}_1 = \mathbf{G}_1\mathbf{H}_1^{\mathrm{T}}, \mathbf{A}_2 = \mathbf{G}_2\mathbf{H}_2^{\mathrm{T}}$
2. 给出 $\mathbf{F}^1,\mathbf{F}^2$，构建 $\mathbf{X},\mathbf{Y}$ 根据式（7）
3. 构建 $\mathbf{M}_e = \mathbf{X\Lambda Y}^{\mathrm{T}}$
4. 给出 $\mathbf{U}^1,\mathbf{U}^2$，构建 $\mathbf{M}_p = \mathbf{U}^1\mathbf{U}^{2\mathrm{T}}$
5. 构建 $\mathbf{M}$ 根据式（5）并且使得 $\mathbf{G}_1,\mathbf{H}_1,\mathbf{G}_2,\mathbf{H}_2$ 对于上一层可用。

反向传递：假设网络提供 $\partial L/\partial\mathbf{M}_e$ 和 $\partial L/\partial\mathbf{M}_p$，这层必须返回 $\partial L/\partial\mathbf{F}^1$，$\partial L/\partial\mathbf{F}^2$，$\partial L/\partial\mathbf{U}^1$ 和 $\partial L/\partial\mathbf{U}^2$；它也必须计算 $\partial L/\partial\mathbf{\Lambda}$ 以便更新矩阵 $\mathbf{\Lambda}$。我们假设 $\partial L/\partial\mathbf{M}_e$ 和 $\partial L/\partial\mathbf{M}_p$ 作为输入，而不是 $\partial L/\partial\mathbf{M}$ 因为后续层可以利用这种特殊的分解来进行有效的计算。我们注意到矩阵 $\mathbf{\Lambda}$ 必须具有以下形式，以便 $\mathbf{M}$ 与正元素对称

$$\mathbf{\Lambda} = \begin{pmatrix} \mathbf{\Lambda}_1 & \mathbf{\Lambda}_2 \\ \mathbf{\Lambda}_2 & \mathbf{\Lambda}_1 \end{pmatrix}, \mathbf{\Lambda}_{ij} > 0, \forall i,j \tag{9}$$

根据边缘矩阵的变化和使用公式（4）写出损失层的变化，

$$\begin{aligned} dL &= \frac{\partial L}{\partial\mathbf{M}_e} : d\mathbf{M}_e = \frac{\partial L}{\partial\mathbf{M}_e} : d(\mathbf{X\Lambda Y}^{\mathrm{T}}) \\ &= \frac{\partial L}{\partial\mathbf{M}_e} : (d\mathbf{X\Lambda Y}^{\mathrm{T}} + \mathbf{X}d\mathbf{\Lambda Y}^{\mathrm{T}} + \mathbf{X\Lambda}d\mathbf{Y}^{\mathrm{T}}) \\ &= \frac{\partial L}{\partial\mathbf{M}_e}\mathbf{Y\Lambda}^{\mathrm{T}} : d\mathbf{X} + \mathbf{X}^{\mathrm{T}}\frac{\partial L}{\partial\mathbf{M}_e}\mathbf{Y} : d\mathbf{\Lambda} + \\ &\quad + \frac{\partial L}{\partial\mathbf{M}_e}^{\mathrm{T}}\mathbf{X\Lambda} : d\mathbf{Y} \end{aligned} \tag{10}$$

我们确定项

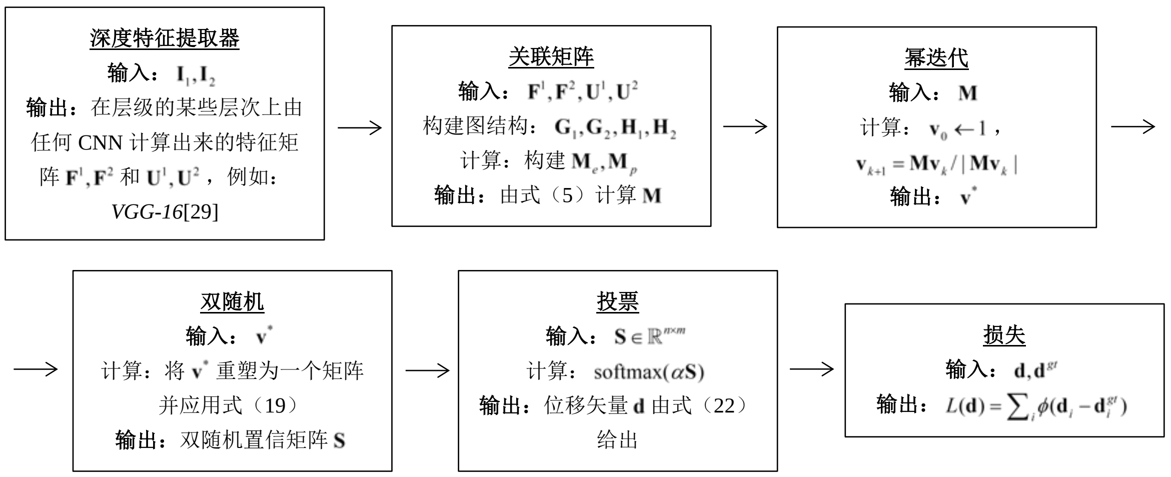

深度特征提取器
输入：$\mathbf{I}_1,\mathbf{I}_2$
输出：在层级的某些层次上由任何 CNN 计算出来的特征矩阵 $\mathbf{F}^1,\mathbf{F}^2$ 和 $\mathbf{U}^1,\mathbf{U}^2$，例如：*VGG-16*[29]
关联矩阵
输入：$\mathbf{F}^1,\mathbf{F}^2,\mathbf{U}^1,\mathbf{U}^2$
构建图结构：$\mathbf{G}_1,\mathbf{G}_2,\mathbf{H}_1,\mathbf{H}_2$
计算：构建 $\mathbf{M}_e,\mathbf{M}_p$
输出：由式（5）计算 $\mathbf{M}$
幂迭代
输入：$\mathbf{M}$
计算：$\mathbf{v}_0 \leftarrow 1$，$\mathbf{v}_{k+1}=\mathbf{M}\mathbf{v}_k/|\mathbf{M}\mathbf{v}_k|$
输出：$\mathbf{v}^*$
双随机
输入：$\mathbf{v}^*$
计算：将 $\mathbf{v}^*$ 重塑为一个矩阵并应用式（19）
输出：双随机置信矩阵 $\mathbf{S}$
投票
输入：$\mathbf{S}\in\mathbb{R}^{n\times m}$
计算：$\mathrm{softmax}(\alpha\mathbf{S})$
输出：位移矢量 $\mathbf{d}$ 由式（22）给出
损失
输入：$\mathbf{d},\mathbf{d}^{gt}$
输出：$L(\mathbf{d})=\sum_i\phi(\mathbf{d}_i-\mathbf{d}_i^{gt})$

图 1：我们完全可训练的图形匹配模型的计算流水线。在训练中，梯度 w.r.t.损失函数通过深度特征提取层次结构，控制与两个图的节点和边缘相关联的一元项和二元项、得到的亲和矩阵的因式分解、匹配的特征分解解。问题和基于投票的分配层。对于每个模块，我们展示了输入、输出和涉及的关键变量。相关的计算的详细推导在相应的论文章节中给出。

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{X}}&=\frac{\partial L}{\partial \mathbf{M}_e}\mathbf{T}\boldsymbol{\Lambda}^{\mathrm{T}}\\ \frac{\partial L}{\partial \boldsymbol{\Lambda}}&=\mathbf{X}^{\mathrm{T}}\frac{\partial L}{\partial \mathbf{M}_e}\mathbf{Y}\\ \frac{\partial L}{\partial \mathbf{Y}}&=\frac{\partial L}{\partial \mathbf{M}_e}\mathbf{X}\boldsymbol{\Lambda}\end{aligned}\tag{11}$$

为了计算偏导数 $\partial L/\partial\mathbf{F}^1$ 和 $\partial L/\partial\mathbf{F}^2$，我们定义和做和了矩阵 $\partial L/\partial\mathbf{X}$ 和 $\partial L/\partial\mathbf{Y}$ 中相应的 $1\times d$ 子块。注意到在从（9）中实现正性条件时，可以使用 **ReLU** 单元。

3.2 幂迭代层

使用幂迭代可以计算关联矩阵 $\mathbf{M}$ 的主导特征向量 $\mathbf{v}^*$

$$\mathbf{v}_{k+1}=\frac{\mathbf{M}\mathbf{v}_k}{\|\mathbf{M}\mathbf{v}_k\|}\tag{12}$$

我们初始化 $\mathbf{v}_0=1$，使用 $\|\bullet\|$ 代表 ℓ_2 范数 $\|\bullet\|_2$。

前向计算：我们运行式（12）中的赋值 N 次，则输出向量 $\mathbf{v}^*=\mathbf{v}_N$。回忆 $\mathbf{M}\in\mathbb{R}^{nm\times nm}$，这里的 n,m 是每张图的节点数，p,q 是每张图的边数，因此前向计算的时间复杂度为 $O(n^2m^2)$ 如果矩阵 $\mathbf{M}$ 是稠密的。如果我们利用 $\mathbf{M}$ 的稀疏性，成本下降到 $O(\mathbf{nnz})$，其中 $\mathbf{nnz}$ 表示矩阵 $\mathbf{M}$ 的非零元素的数量，等于 $pq+nm$。

反向计算：为了计算梯度，我们表达损失的变化和识别所需的偏导数

$$\begin{aligned}dL&=\frac{\partial L}{\partial \mathbf{v}_{k+1}}:d\mathbf{v}_{k+1}=\frac{\partial L}{\partial \mathbf{v}_{k+1}}:d\frac{\mathbf{M}\mathbf{v}_k}{\|\mathbf{M}\mathbf{v}_k\|}\\ d\frac{\mathbf{M}\mathbf{v}_k}{\|\mathbf{M}\mathbf{v}_k\|}&=\frac{\left(\mathbf{I}-\mathbf{v}_{k+1}\mathbf{v}_{k+1}{}^{\mathrm{T}}\right)}{\|\mathbf{M}\mathbf{v}_k\|}d\left(\mathbf{M}\mathbf{v}_k\right)\\ &=\frac{\left(\mathbf{I}-\mathbf{v}_{k+1}\mathbf{v}_{k+1}{}^{\mathrm{T}}\right)}{\|\mathbf{M}\mathbf{v}_k\|}\left(d\mathbf{M}\mathbf{v}_k+\mathbf{M}d\mathbf{v}_k\right)\end{aligned}\tag{13}$$

由于 $\mathbf{y}:\mathbf{A}\mathbf{x}=\mathbf{y}\mathbf{x}^{\mathrm{T}}:\mathbf{A}=\mathbf{A}^{\mathrm{T}}\mathbf{y}:\mathbf{x}$，并且使用 $\mathbf{M}$ 的对称性，我们得到

$$\begin{aligned}dL=&\frac{\left(\mathbf{I}-\mathbf{v}_{k+1}\mathbf{v}_{k+1}{}^{\mathrm{T}}\right)}{\|\mathbf{M}\mathbf{v}_k\|}\frac{\partial L}{\partial \mathbf{v}_{k+1}}\mathbf{v}_k^{\mathrm{T}}:\ d\mathbf{M}+\\ &+\mathbf{M}\frac{\left(\mathbf{I}-\mathbf{v}_{k+1}\mathbf{v}_{k+1}{}^{\mathrm{T}}\right)}{\|\mathbf{M}\mathbf{v}_k\|}\frac{\partial L}{\partial \mathbf{v}_{k+1}}:\ d\mathbf{v}_k\end{aligned}\tag{14}$$

注意到 $d\mathbf{v}_k$ 依旧是 $d\mathbf{M}$ 的函数，梯度 $\partial L/\partial\mathbf{M}$ 和 $\partial L/\partial\mathbf{v}_k$ 构建如下：

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{M}}&=\sum_k\frac{\left(\mathbf{I}-\mathbf{v}_{k+1}\mathbf{v}_{k+1}{}^{\mathrm{T}}\right)}{\|\mathbf{M}\mathbf{v}_k\|}\frac{\partial L}{\partial \mathbf{v}_{k+1}}\mathbf{v}_k^{\mathrm{T}}\\ \frac{\partial L}{\partial \mathbf{v}_k}&=\mathbf{M}\frac{\left(\mathbf{I}-\mathbf{v}_{k+1}\mathbf{v}_{k+1}{}^{\mathrm{T}}\right)}{\|\mathbf{M}\mathbf{v}_k\|}\frac{\partial L}{\partial \mathbf{v}_{k+1}}\end{aligned}\tag{15}$$

这里我们从上层网络层接收 $\partial L/\partial\mathbf{v}^*$。式（15）的计算代价是 $O(m^2n^2)$ —无论 $\mathbf{M}$ 的稀疏性如何，并且

$$\begin{aligned}\frac{(\mathbf{I}-\mathbf{v}_{k+1}\mathbf{v}_{k+1}^{\mathrm{T}})}{\|\mathbf{M}\mathbf{v}_k\|}\frac{\partial L}{\partial \mathbf{v}_{k+1}}\mathbf{v}_k^{\mathrm{T}}\mathbf{:}\ d\mathbf{M} &= \frac{(\mathbf{I}-\mathbf{v}_{k+1}\mathbf{v}_{k+1}^{\mathrm{T}})}{\|\mathbf{M}\mathbf{v}_k\|}\mathbf{v}_k^{\mathrm{T}} : (\mathbf{G}_2\otimes\mathbf{G}_1)[\mathrm{vec}(d\mathbf{M}_e)](\mathbf{H}_2\otimes\mathbf{H}_1)^{\mathrm{T}} \\ &= \mathrm{diag}\left((\mathbf{G}_2\otimes\mathbf{G}_1)^{\mathrm{T}}\frac{(\mathbf{I}-\mathbf{v}_{k+1}\mathbf{v}_{k+1}^{\mathrm{T}})}{\|\mathbf{M}\mathbf{v}_k\|}\frac{\partial L}{\partial \mathbf{v}_{k+1}}\mathbf{v}_k^{\mathrm{T}}(\mathbf{H}_2\otimes\mathbf{H}_1)\right) : d\mathbf{M}_e \\ &= (\mathbf{G}_2\otimes\mathbf{G}_1)^{\mathrm{T}}\frac{(\mathbf{I}-\mathbf{v}_{k+1}\mathbf{v}_{k+1}^{\mathrm{T}})}{\|\mathbf{M}\mathbf{v}_k\|}\frac{\partial L}{\partial \mathbf{v}_{k+1}}\odot(\mathbf{H}_2\otimes\mathbf{H}_1)^{\mathrm{T}}\mathbf{v}_k^{\mathrm{T}} : d\mathbf{M}_e \\ &= \mathbf{G}_1^{\mathrm{T}}\left(\frac{(\mathbf{I}-\mathbf{v}_{k+1}\mathbf{v}_{k+1}^{\mathrm{T}})}{\|\mathbf{M}\mathbf{v}_k\|}\frac{\partial L}{\partial \mathbf{v}_{k+1}}\right)_{n\times m}\mathbf{G}_2\odot\mathbf{H}_1^{\mathrm{T}}(\mathbf{v}_k)_{n\times m}\mathbf{H}_2 : d\mathbf{M}_e\end{aligned} \tag{16}$$

图 2：从方程式（14）出发，给出损失函数 w.r.t 的变化表示边缘亲和矩阵 $\mathbf{M}_e$ 的变化，给出损失函数 w.r.t 的变化表示整个亲和矩阵 $\mathbf{M}$ 的变化。

存储器复杂度是 $\Theta(m^2n^2)$。这样的成本在实践中是令人望而却步的。通过采用矩阵反向传播技术[15]，我们可以利用矩阵 $\mathbf{M}$ 的特殊因式分解（5），使运算既节省内存又节省时间。事实上，撇开效率不谈，经典方法如何通过复杂的因式分解来传播导数并不明显，比如（5）。利用（5）和来自（4）的配方，并省略了术语 $\mathbf{M}_p$，我们得到（16）如图 2 所示。

注意 $(\bullet)_{n\times m}$ 是将 $nm\times 1$ 向量整形成 $n\times m$ 矩阵的算子。对于导数，我们使用的性质是，对于任何相容矩阵 $\mathbf{A}$， $\mathbf{B}$ 和 $\mathbf{V}$，

$$(\mathbf{A}\otimes\mathbf{B})^{\mathrm{T}} vec(\mathbf{V}) = vec(\mathbf{B}^{\mathrm{T}}\mathbf{V}\mathbf{A})$$

因此，也考虑到一元项 $\mathbf{M}_p$，如下

$$\frac{\partial L}{\partial \mathbf{M}_e} = \sum_k \mathbf{G}_1^{\mathrm{T}}\left(\frac{\left(\mathbf{I}-\mathbf{v}_{k+1}\mathbf{v}_{k+1}{}^{\mathrm{T}}\right)}{\|\mathbf{M}\mathbf{v}_k\|}\frac{\partial L}{\partial \mathbf{v}_{k+1}}\right)_{n\times m}\mathbf{G}_2\odot \\ \odot\mathbf{H}_1^{\mathrm{T}}(\mathbf{v}_k)_{n\times m}\mathbf{H}_2 \tag{17}$$

$$\frac{\partial L}{\partial \mathbf{M}_p} = \sum_k \frac{\left(\mathbf{I}-\mathbf{v}_{k+1}\mathbf{v}_{k+1}{}^{\mathrm{T}}\right)}{\|\mathbf{M}\mathbf{v}_k\|}\frac{\partial L}{\partial \mathbf{v}_{k+1}}\odot\mathbf{v}_k \tag{18}$$

随着操作的仔细排序，后向传递的复杂性现在为 $O(\max(m^2q, n^2p))$ 和 $\Theta(pq)$。考虑到节点边缘关联矩阵 $\mathbf{G},\mathbf{H}$ 的稀疏性，可以进一步提高效率。

3.3 双随机层

通过将特征向量 $\mathbf{v}^*$ 映射到匹配问题(1)的 L1 约束上，使得幂迭代层的结果具有双重随机性：$\forall a, \sum_i v_{ia} = 1$ 和 $\forall i, \sum_a v_{ia} = 1$。这是由多个作者[13, 37, 19]提出的，因为它被观察到显著地改善了性能。我们引入了一个新的双随机层，它以任意的对应向量 $\mathbf{v}^* \in \mathbb{R}_+^{nm\times 1}$ 作为输入，重塑为大小为 $n\times m$ 的矩阵，并输出双随机变量，如[30]中所述。即使原来的算法只假定正方形矩阵，该过程可以被推广，如在[9]中所示。

前向计算：给出一个初始矩阵 $\mathbf{S}_0 = (\mathbf{v}^*)_{n\times m}$，我们运行前向赋值

$$\mathbf{S}_{k+1} = \mathbf{S}_k[\mathbf{1}_n^{\mathrm{T}}\mathbf{S}_k]^{-1}, \mathbf{S}_{k+2} = [\mathbf{S}_{k+1}\mathbf{1}_m]^{-1}\mathbf{S}_{k+1} \tag{19}$$

反向计算：给出一个初始梯度 $\partial L/\partial\mathbf{S}$，我们计算

$$\frac{\partial L}{\partial \mathbf{S}_{k+1}} = [\mathbf{S}_{k+2}\mathbf{1}_m]^{-1}\frac{\partial L}{\partial \mathbf{S}_{k+2}} - \\ -\mathrm{diag}\left([\mathbf{S}_{k+2}\mathbf{1}_m]^{-2}\frac{\partial L}{\partial \mathbf{S}_{k+2}}\mathbf{S}_{k+2}^{\mathrm{T}}\right)\mathbf{1}_m^{\mathrm{T}} \tag{20}$$

$$\frac{\partial L}{\partial \mathbf{S}_k} = \frac{\partial L}{\partial \mathbf{S}_{k+1}}[\mathbf{1}_n^{\mathrm{T}}\mathbf{S}_{k+1}]^{-1} - \\ -\mathbf{1}_n\,\mathrm{diag}\left([\mathbf{1}_n^{\mathrm{T}}\mathbf{S}_{k+1}]^{-2}\mathbf{S}_{k+1}^{\mathrm{T}}\frac{\partial L}{\partial \mathbf{S}_{k+1}}\right)^{\mathrm{T}} \tag{21}$$

3.4 将置信图转换为位移

我们使用一个特殊的层，称为投票层，将由双随机层传递的置信映射 $\mathbf{S}\in\mathbb{R}^{n\times m}$ 转换为位移矢量。其思想是对每个分配的点 i 进行归一化，其对应的候选得分由 $\mathbf{S}$ 的第 i 行给出，表示为 $\mathbf{S}(i,1\dots m)$。然后，我们使用它来加权位置 $\mathbf{P}\in\mathbb{R}^{m\times 2}$ 的矩阵，并减去匹配 i 的位置。

$$\mathbf{d}_i = \frac{\exp[\alpha\mathbf{S}(i,1\dots m)]}{\sum_j \exp[\alpha\mathbf{S}(i,j)]}\mathbf{P} - \mathbf{P}_i \tag{22}$$

这里[]现在只是一个笨重的支架。在实践中，我们将α设置为大值（例如 200）。通过软投票固定的候选位置，我们的解决方案可以插值更精确的定位。我们也为 0 的候选人设定了信心，这些人离分配点太远，或者通过填充增加。因此，这些不会造成损失，由如下给出。

$$L(\mathbf{d}) = \sum \phi(\mathbf{d}_i - \mathbf{d}_i^{gt}) \tag{23}$$

其中$\phi(\mathbf{x}) = \sqrt{\mathbf{x}^{\mathrm{T}}\mathbf{x} + \varepsilon}$是一个鲁棒惩罚项，$\mathbf{d}^{gt}$是从源点到正确赋值的地真位移。该层在多个完全可微的步骤中实现：a）首先，将输入缩放α，b）使用空间映射丢弃离起始位置更远的候选位置，并使用它来修改置信度映射，c）使用软映射层到 n 计算置信图，D）计算位移图。丢弃映射将距离超过一定距离的点或通过填充添加的点的置信度设置为 0。这样的点在最终损失中没有贡献，由（23）给出。

4. 实验

在这一节中，我们描述了使用的模型以及详细的实验匹配结果，包括定量（包括消融研究）和定性，在三个挑战性数据集：MPI SnTEL，CUB，和 PASCAL 关键点。

深度特征提取网络：我们依赖于[29]中的 VGG-16 体系结构，该体系结构在 ImageNet ILSVRC[28]中经过了执行分类的预训练，但是我们可以使用任何其他深层网络体系结构。我们在 MaTrimNET[31]中实现了我们的深度学习框架。作为边缘特征**F**，我们使用层 relu5_1 的输出（以及它下面的整个层次结构），而对于节点特征**U**，我们使用层 relu4_2 的输出（具有与之相关的层次结构的参数）。在将它们用于计算关联矩阵**M**之前，所有特征都通过归一化层被归一化到 1。我们在 MPISintel [6]、Caltech-UCSD Birds-200-2011[32]和 PASCAL VOC[11]上用 Berkeley 注释进行了几何和语义对应实验[5]。

匹配网络：*GMNwVGG* 是我们提出的基于 VGG 特征提取器的图形匹配网络。后缀*-U* 表示使用默认（初始）权重；*-T* 表示端到端的培训；*GMNwVGG-T w/o V* 变体在测试中不使用表决层来计算位移，而是直接在置信图**S**的行之间分配最大值的索引，如相应帷幕。*NNwVGG* 给出了基于深度节点描述符**U**的最近邻匹配。

MPI-Sintel：除了用于验证匹配方法的主要数据集 CUB 和 PASCAL 之外，我们还使用 Sintel 来演示公式的通用性和灵活性。Sintel 输入图像是电影中的连续帧，并且显示出大位移、大外观变化、遮挡、非刚性运动和复杂的大气效应(仅包括在最终的图像集中)。Sintel 培训集包括 23 个视频序列（组织为文件夹）和 1064 帧。为了确保我们正在进行公平的训练和评估，因为来自相同视频的图像描述了相同对象的实例，我们将数据分成 18 个文件夹（即 796 个图像对）用于训练，5 个文件夹（即 245 个图像对）用于测试。为了能够在内存使用和描述符的可用计算分辨率的限制下设置大量的对应关系，我们将输入图像分割成 4×4 块，以填充最大位移，逐个匹配。注意，对于这个特定的实验，我们不使用双随机层，因为一对一的映射约束不适用于这个问题的本质（分配可以是多对一的，例如在缩放中）。注意，我们的模型被设计为在包含相似结构的两个图像之间建立对应关系，这些图像通常来自不同的场景，没有明确地针对光流进行裁剪，其中可以利用额外的平滑性约束。我们的 Sintel 实验的一个重点在于展示我们方法的可伸缩性，该方法使用大小为$10^6 \times 10^6$的关联矩阵。在 3.2 GHz 的英特尔 Xeon 机上，一个完整的前向和后退通过大约 **2 秒**的时间运行，Titan X Pascal GPU[1]。我们在

方法	PCK@10 像素
SIFT flow [22]	89.0
DaisyFF [34]	87.3
DSP [17]	85.3
DM [33]	89.2
UCN [8]	91.5
NNwVGG-U	85.9
NNwVGG-T	87.01
GMNwVGG-U	88.03
GMNwVGG-T	**92.6**

表 1：Sintel 的比较匹配结果。

[1] 一种可能的加速方法是，不像每次幂迭代期间的松弛那样通过昂贵的层传播梯度，而是在收敛时只传播一次[1，2]。原则上，这不是正确的梯度，在我们的测试中，这种方法确实没有收敛。

表 1 中显示定量结果。我们使用正确的关键点度量的百分比来测试我们的匹配性能，阈值为 10 像素。我们与其他国家的最先进的匹配方法比较，按照协议[8]。请注意，即使未经训练，我们的网络仍然具有竞争性，因为图结构充当正则化器。经过训练，网络精度明显提高。在图 4 中给出了视觉示例。

CUB: 该数据集包含 200 个鸟类类别的 11788 个图像，具有包围盒对象定位和 15 个注释部分。我们使用由[16]建立的测试集，该测试集由 5000 个图像对组成，在[18]的姿态图上，这些图像对相距 3 个最近的邻居，所以我们可以预期，在待测试的每对图像中，鸟儿的姿态有些相似。但是在所有的图像中，姿势、关节和外观都有显著的变化。为了训练我们的模型，我们从 CUB-200-2011 训练集中随机采样图像对，这些图像在测试集中不存在。两个图中的点数是最大 $n=15$ 和 $m=256$，取自 16×16 网格，在第一个图上用 Delaunay 三角剖分，在第二个图上完全连接。在投票层中，我们不再丢弃离源点太远的候选位置。一个完整的前向后传球在 0.3 秒内完成。

方法	EPE(像素)	PCK@0.05
GMNwVGG-U	41.63	0.63
NNwVGG-U	59.05	0.46
GMNwVGG-T w/o V	18.22	0.83
GMNwVGG-T	**17.02**	**0.86**

表 2：我们的模型在 CUB 上。

我们在表 2 中显示了定量结果。“PCK@ α ”度量[35]表示来自地面真实位置的比 $\alpha\sqrt{w^2+h^2}$ 更接近的预测对应关系的百分比，其中 w,h 是图像维度。定性结果如图 5 所示。

我们的端到端完全可训练匹配模型显著优于基于深度特征的最近邻方法（EPE 误差减半）或基于深度特征的相似图匹配公式，这些公式没有与分配模型联合细化。我们的模型 *GMNwVGG-T W/O V*，它不使用，在测试、投票层，以计算位移不如我们的竞争方法的软投票机制 *GMNwVGG-T*。我们还实现了最先进的结果相比，UCN[8]，WARPNET[16]，SIFT[24]和 DSP[17]，参见图 3。

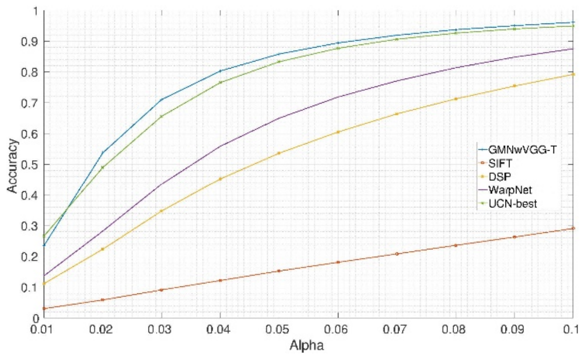


图 3：我们的方法和其他最先进的技术在 CUB

PASCAL VOC 关键点：该数据集是 PASCAL VOC 2011 的扩展，包含 20 个语义类的主体部分的注释。有 7000 个带注释的示例用于培训，1700 个用于测试，但是我们可以在同一个类的任意两个示例之间形成对。虽然每个类的实例数目严重失衡，但在训练中，我们根据其对应的概率，从每个类中

方法	EPE(像素)
conv4 flow [23]	24.9
SIFT flow [22]	24.7
UCN [8]	38.9
NNwVGG-U	25.4
GMNwVGG-U	29.8
GMNwVGG-T	**40.6**

表 3：我们的模型以及其他国家的方法接近 PASCAL VOC。

抽取随机实例。在火车和测试时间，我们将每一个例子都围绕它的边界框，将其重新缩放到 256×256，并将其传递给网络。我们训练一个匹配网络的所有类（意味着是我们不能期望它们以相同的姿势包含对象），因此，我们的模型是不可知的语义范畴。这个数据集比 CUB 更具挑战性：边界框的范围从非常大到非常小（例如，30×30），训练和测试对覆盖每个类的两个图像的组合，并且外观变化更极端。我们依赖于相同的协议进行评估和设置元参数，如在 CUB 实验中。结果在图 6 中示出，并且在表 3 中进行比较。

我们测试遵循相同的协议为 **CUB-200-2011**，并获得改进的最先进的。注意，UCN[8]的结果与论文不同，因为我们使用 20 个语义类的直平均值。他们的论文报道了基于类频率的加权平均——在该度量下，UCN 得到 PCK@0.1 = 44，我们获得了 45.3，因此与直接平均的情况相比，改进被保留。

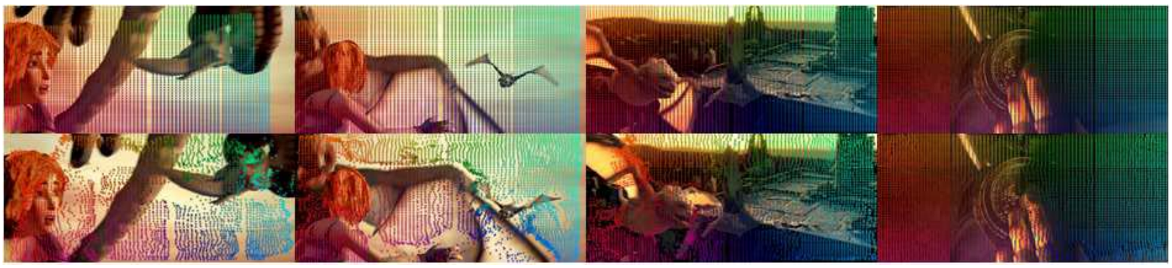


图 4：MPI SnTEL 测试分区上的四个可视化示例，其表现出大的运动和遮挡区域。从上到下：源图像与点重叠的初始网格和目标图像与对应的匹配，由我们完全训练的网络找到。颜色是独特的，它们编码对应。即使对于快速移动的物体，点趋向于正确地跟踪表面，而不会滑动——例如龙的翅膀、爪子和飞行的怪物。

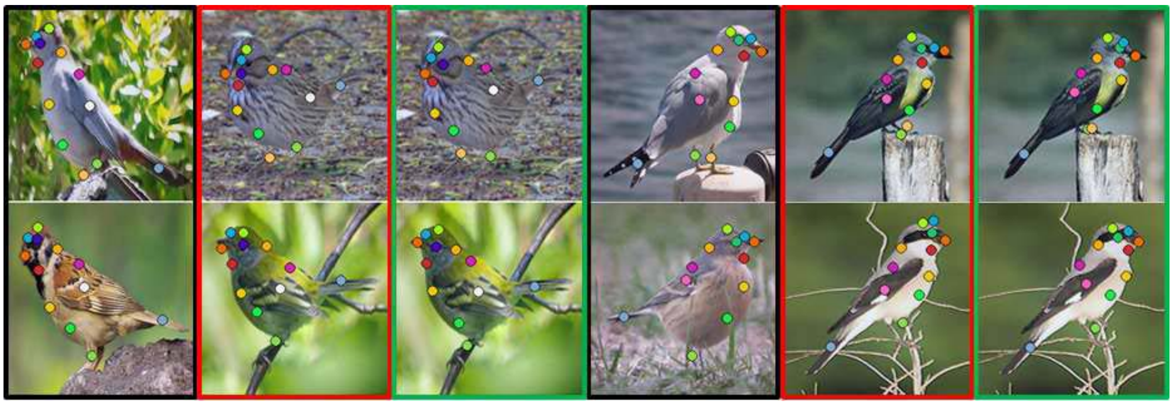


图 5：MPI SnTEL 测试分区上的四个可视化示例，其表现出大的运动和遮挡区域。从上到下：源图像与点重叠的初始网格和目标图像与对应的匹配，由我们完全训练的网络找到。颜色是独特的，它们编码对应。即使对于快速移动的物体，点趋向于正确地跟踪表面，而不会滑动——例如龙的翅膀、爪子和飞行的怪物。

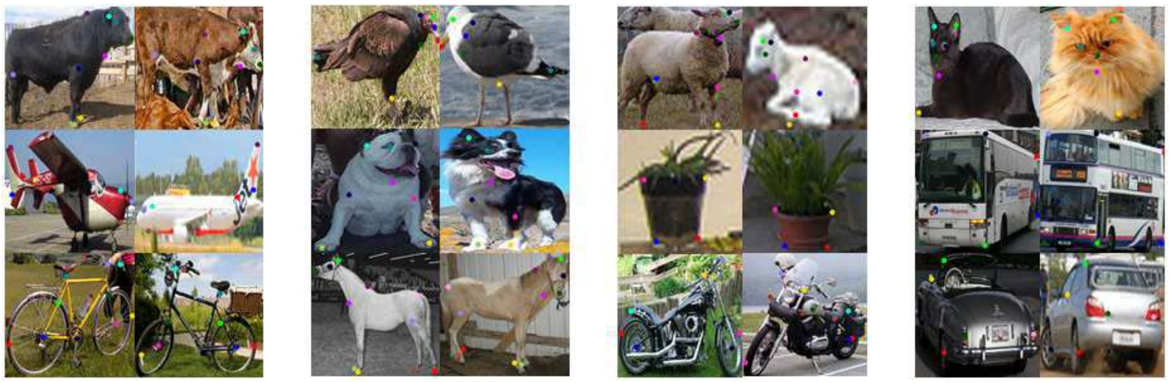


图 6：十二个性能最好的网络 GMNWVGG-T 在 PASCAL VOC 测试集上的例子。对于每一对示例，左边显示源图像，右边显示目标。颜色标识计算点之间的分配。即使在极端的外观和姿态变化下，该方法也能找到匹配。

5. 结论

我们提出了用于图匹配的端到端学习框架，该框架一般适用于包含深度特征提取层次和组合优化层的模型。我们将此问题描述为使用深参数特征层次表示的一元节点关系和成对节点关系下的二次指派。所有的模型参数都是可训练的，并且图形匹配优化被包括在学习公式中。因此，主要的挑战

是通过复杂矩阵层计算反向传播导数，并以计算有效的方式实现整个框架(亲和矩阵的因式分解，双随机层)。我们在 PASCAL VOC 关键点、Sintel 和 CUB 等不同数据集上的实验和烧蚀研究表明，完全学习的图匹配模型优于最近邻匹配模型，或使用深度特征层次结构的方法，而深度特征层次结构没有与（受约束）二次分配问题一起细化。

致谢：这项工作部分由欧洲研究理事会合并者，SEED, CNCSUEFISCDI PN-III-P4-ID-PCE-2016-0535, EU Horizon 2020 Grant No. 688835 DE-ENIGMA, 和 SSF.支持。

参考文献

[1] B. Amos and J. Z. Kolter. Optnet: Differentiable optimizationas a layer in neural networks. *CoRR*, abs/1703.00443,2017.
[2] J. T. Barron and B. Poole. The fast bilateral solver. *CoRR*,abs/1511.03296, 2015.
[3] A. C. Berg, T. L. Berg, and J. Malik. Shape matchingand object recognition using low distortion correspondences.In *Computer Vision and Pattern Recognition, 2005. CVPR2005. IEEE Computer Society Conference on*, volume 1,pages 26–33. IEEE, 2005.
[4] P. J. Besl and N. D. McKay. Method for registration of 3-dshapes. In *Robotics-DL tentative*, pages 586–606. InternationalSociety for Optics and Photonics, 1992.
[5] L. Bourdev and J. Malik. Poselets: Body part detectorstrained using 3d human pose annotations. In *Computer Vi-sion, 2009 IEEE 12th International Conference on*, pages1365–1372. IEEE, 2009.
[6] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. Anaturalistic open source movie for optical flow evaluation. In*European Conference on Computer Vision*, pages 611–625.Springer, 2012.
[7] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, and A. J.Smola. Learning graph matching. *IEEE transactions onpattern analysis and machine intelligence*, 31(6):1048–1058,2009.
[8] C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker. Universalcorrespondence network. In *Advances in Neural In-formation Processing Systems*, pages 2406–2414, 2016.
[9] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching.In *NIPS*, volume 2, page 6, 2006.
[10] O. Duchenne, F. Bach, I.-S. Kweon, and J. Ponce. Atensor-based algorithm for high-order graph matching. *IEEEtransactions on pattern analysis and machine intelligence*,33(12):2383–2395, 2011.
[11] M. Everingham and J.Winn. The pascal visual object classeschallenge 2011 (voc2011) development kit. *Pattern Analy-sis, Statistical Modelling and Computational Learning, Tech.Rep*, 2011.
[12] M. A. Fischler and R. C. Bolles. Random sample consensus:a paradigm for model fitting with applications to imageanalysis and automated cartography. *Communications of theACM*, 24(6):381–395, 1981.
[13] S. Gold and A. Rangarajan. A graduated assignment algorithmfor graph matching. *IEEE Transactions on patternanalysis and machine intelligence*, 18(4):377–388, 1996.
[14] B. Ham, M. Cho, C. Schmid, and J. Ponce. Proposal flow.In *Proceedings of the IEEE Conference on Computer Visionand Pattern Recognition*, pages 3475–3484, 2016.
[15] C. Ionescu, O. Vantzos, and C. Sminchisescu. Training deepnetworks with structured layers by matrix backpropagation.*arXiv preprint arXiv:1509.07838*, 2015.
[16] A. Kanazawa, D. W. Jacobs, and M. Chandraker. Warpnet:Weakly supervised matching for single-view reconstruction.In *Proceedings of the IEEE Conference on Computer Visionand Pattern Recognition*, pages 3253–3261, 2016.
[17] J. Kim, C. Liu, F. Sha, and K. Grauman. Deformable spatialpyramid matching for fast dense correspondences. In *Pro-ceedings of the IEEE Conference on Computer Vision andPattern Recognition*, pages 2307–2314, 2013.
[18] J. Krause, H. Jin, J. Yang, and L. Fei-Fei. Fine-grainedrecognition without part annotations. In *Proceedings of theIEEE Conference on Computer Vision and Pattern Recogni-tion*, pages 5546–5555, 2015.
[19] M. Leordeanu and M. Hebert. A spectral technique for correspondenceproblems using pairwise constraints. In *ComputerVision, 2005. ICCV 2005. Tenth IEEE International Confer-ence on*, volume 2, pages 1482–1489. IEEE, 2005.
[20] M. Leordeanu, R. Sukthankar, and M. Hebert. Unsupervisedlearning for graph matching. *International*

journal of com-puter vision, 96(1):28–45, 2012.

[21] M. Leordeanu, A. Zanfir, and C. Sminchisescu. Semisupervisedlearning and optimization for hypergraph matching.In *Computer Vision (ICCV), 2011 IEEE InternationalConference on*, pages 2274–2281. IEEE, 2011.

[22] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondenceacross scenes and its applications. *IEEE Transactionson Pattern Analysis and Machine Intelligence*, 33(5):978–994, 2011.

[23] J. L. Long, N. Zhang, and T. Darrell. Do convnets learn correspondence?In *Advances in Neural Information ProcessingSystems*, pages 1601–1609, 2014.

[24] D. G. Lowe. Distinctive image features from scaleinvariantkeypoints. *International journal of computer vi-sion*, 60(2):91–110, 2004.

[25] J. Maciel and J. P. Costeira. A global solution to sparse correspondenceproblems. *IEEE Transactions on Pattern Analysisand Machine Intelligence*, 25(2):187–199, 2003.

[26] J. R. Magnus, H. Neudecker, et al. Matrix differential calculuswith applications in statistics and econometrics. 1995.

[27] I. Rocco, R. Arandjelovi´c, and J. Sivic. Convolutional neuralnetwork architecture for geometric matching. *arXiv preprintarXiv:1703.05593*, 2017.

[28] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh,S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein,et al. Imagenet large scale visual recognition challenge.*International Journal of Computer Vision*, 115(3):211–252,2015.

[29] K. Simonyan and A. Zisserman. Very deep convolutionalnetworks for large-scale image recognition. *arXiv preprintarXiv:1409.1556*, 2014.

[30] R. Sinkhorn and P. Knopp. Concerning nonnegative matricesand doubly stochastic matrices. *Pacific Journal ofMathemat-ics*, 21(2):343–348, 1967.

[31] A. Vedaldi and K. Lenc. Matconvnet – convolutional neuralnetworks for matlab. In *Proceeding of the ACM Int. Conf. onMultimedia*, 2015.

[32] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie.The Caltech-UCSD Birds-200-2011 Dataset. Technical ReportCNS-TR-2011-001, California Institute of Technology,2011.

[33] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid.Deepflow: Large displacement optical flow with deep matching. In *Computer Vision (ICCV), 2013 IEEE InternationalConference on*, pages 1385–1392. IEEE, 2013.

[34] H. Yang, W. Lin, and J. Lu. Daisy filter flow: A generalizedapproach to discrete dense correspondences. CVPR, 2014.

[35] Y. Yang and D. Ramanan. Articulated human detectionwith flexible mixtures of parts. *IEEE Transactions on Pat-tern Analysis and Machine Intelligence*, 35(12):2878–2890,2013.

[36] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. Lift: Learned invariantfeature transform. In *European Conference on Com-puter Vision*, 2016.

[37] R. Zass and A. Shashua. Probabilistic graph and hypergraphmatching. In *Computer Vision and Pattern Recog-nition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8.IEEE, 2008.

[38] F. Zhou and F. De la Torre. Factorized graph matching.In *Computer Vision and Pattern Recognition (CVPR), 2012IEEE Conference on*, pages 127–134. IEEE, 2012.