Deep Learning of Graph Matching

Zanfir and Sminchisescu
CVPR 2018

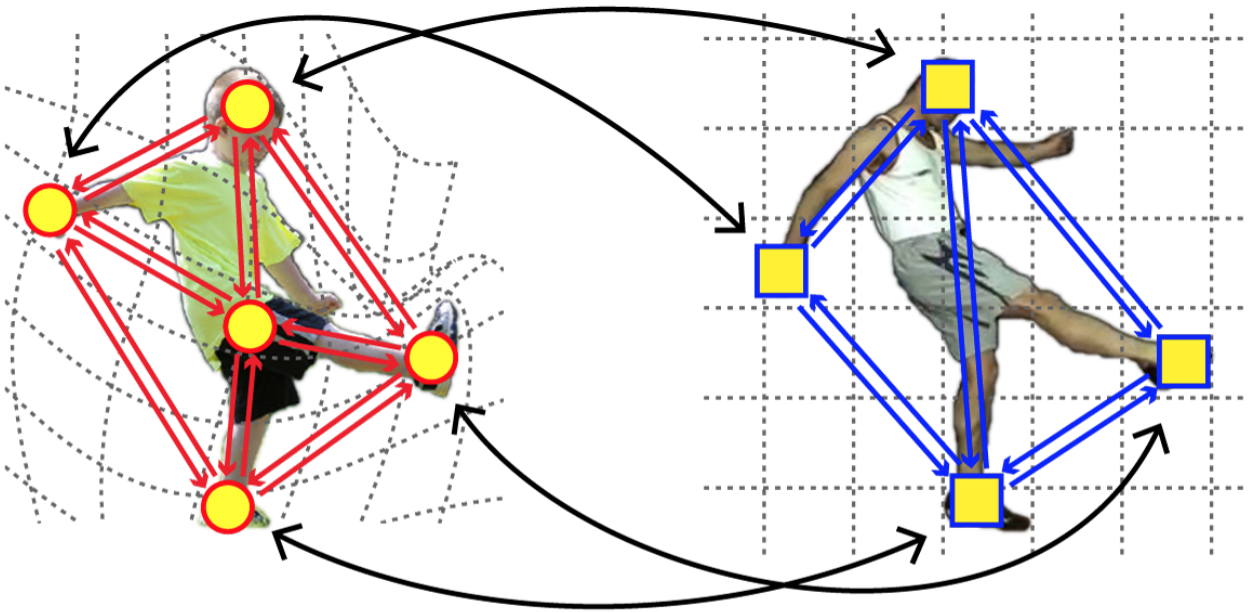
Presenter: Jack Lanchantin
https://qdata.github.io/deep2Read

Outline

Outline

Overview

- **Graph matching**: establishing correspondences between two graphs represented in terms of both local node structure and pair-wise relationships



- **This paper**: graph matching where the unary and pairwise structures are deep feature representation with trainable parameters

Problem Formulation

- **Given**: two input graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, with $|V_1| = n$ and $|V_2| = m$
- **Graph Matching Goal**: minimize some loss between the corresponding nodes and edges of the two graphs

Graph Matching

- Let $v \in \{0,1\}^{nm}$ be an indicator vector s.t. $v_{ia} = 1$ if $i \in V_1$ is matched to $a \in V_2$ and 0 otherwise

Graph Matching

- Let $v \in \{0,1\}^{nm}$ be an indicator vector s.t. $v_{ia} = 1$ if $i \in V_1$ is matched to $a \in V_2$ and 0 otherwise
- Let $M \in \mathbb{R}^{nm \times nm}$ be an affinity matrix that encodes similarities between unary and pairwise sets of nodes (points) in the two graphs
  - $M_{ia;jb}$ measures how well every pair $(i,j) \in E_1$ matches with $(a,b) \in E_2$
  - The diagonal entries contain node-to-node scores, whereas the off-diagonal entries contain edge-to-edge scores.

Graph Matching

- Let $v \in \{0,1\}^{nm}$ be an indicator vector s.t. $v_{ia} = 1$ if $i \in V_1$ is matched to $a \in V_2$ and 0 otherwise
- Let $M \in \mathbb{R}^{nm \times nm}$ be an affinity matrix that encodes similarities between unary and pairwise sets of nodes (points) in the two graphs
  - $M_{ia;jb}$ measures how well every pair $(i,j) \in E_1$ matches with $(a,b) \in E_2$
  - 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

$$\mathbf{v}^* = \arg\max_{\mathbf{v}} \mathbf{v}^\mathsf{T} \mathbf{M} \mathbf{v}$$
$$\text{s.t. } \mathbf{C}\mathbf{v} = 1, \mathbf{v} \in \{0,1\}^{nm}$$

where binary matrix $\mathbf{C}\mathbf{v} \in \mathbb{R}^{nm \times nm}$ encodes one-to-one mapping constraints: $\forall i \sum_a \mathbf{a}_{ia} = 1$ and $\forall a \sum_i \mathbf{v}_{ia} = 1$.

Graph Matching

- This is known to be NP-hard, so we relax the problem by dropping both the binary and the mapping constraints, and solve

$$\mathbf{v}^* = \arg\max_{\mathbf{v}} \mathbf{v}^\mathsf{T}\mathbf{M}\mathbf{v}$$
$$\text{s.t. } ||\mathbf{v}||_2 = 1$$

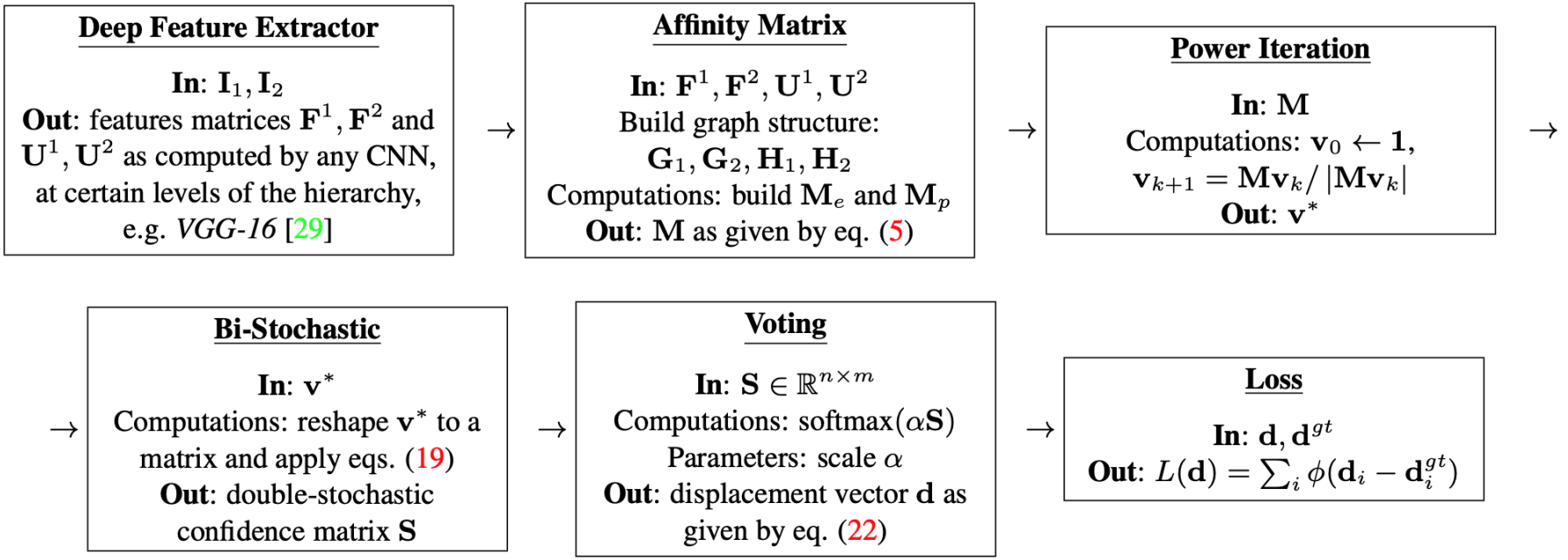
- The optimal $\mathbf{v}^*$ is given by the leading eigenvector of the matrix $\mathbf{M}$
- Since M has non-negative elements, the elements of $\mathbf{v}^*$ are in the interval $[0, 1]$, and we interpret $\mathbf{v}^*_{ia}$ as the confidence that i matches a.

End-to-End Graph Matching Learning

- Estimate the affinity matrix **M** parameterized in terms of unary and pair-wise point features computed over input images
- 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.

Outline

Pipeline


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]
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)
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: $\mathbf{v}^*$
Bi-Stochastic
In: $\mathbf{v}^*$
Computations: reshape $\mathbf{v}^*$ to a matrix and apply eqs. (19)
Out: double-stochastic confidence matrix $\mathbf{S}$
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)
Loss
In: $\mathbf{d}, \mathbf{d}^{gt}$
Out: $L(\mathbf{d}) = \sum_i \phi(\mathbf{d}_i - \mathbf{d}_i^{gt})$

Deep Feature Extractor

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]

Affinity Matrix Layer

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)

Affinity matrix encodes the similarities between unary and pairwise sets of nodes (points) in the two graphs

Affinity Matrix Factorization

- Factorization of the matrix $\mathbf{M}$: exposes the graph structure of the two graphs jointly (the unary and pairwise scores between nodes and edges, respectively)

$$\mathbf{M} = [vec(\mathbf{M}_p)] + (\mathbf{G}_2 \otimes \mathbf{G}_1)[vec(\mathbf{M}_e)](\mathbf{H}_2 \otimes \mathbf{H}_1)^\mathsf{T} \tag{1}$$

where $[x]$ is the diagonal matrix of x, and $\otimes$ is the Kronecker product

Affinity Matrix Layer

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

- $\mathbf{G}_1, \mathbf{H}_1, \mathbf{G}_2, \mathbf{H}_2$ describe the structure of each graph as node-edge incidence matrices, e.g. $\mathbf{G}_1, \mathbf{H}_1 \in \{0,1\}^{n \times p}$, where $g_{ic} = h_{jc} = 1$ if the c^{th} edge starts from the i^{th} node and ends at the j^{th} node.
- If we define the node-to-node adjacency matrices $\mathbf{A}_1 \in \{0,1\}^{n \times n}$, $\mathbf{A}_2 \in \{0,1\}^{m \times m}$, then $\mathbf{A}_1 = \mathbf{G}_1\mathbf{H}_1^\intercal, \mathbf{A}_2 = \mathbf{G}_2\mathbf{H}_2^\intercal$

Affinity Matrix Factorization

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

- $\mathbf{M}_p \in \mathbb{R}^{n \times m}$ represents the 1^{st}-order terms (node-to-node similarities)
- $\mathbf{M}_e \in \mathbb{R}^{p \times q}$ represents the 2^{nd}-order potentials (edge-to-edge similarity), where p and q are the numbers of edges in G_1, G_2
- One simple way to build $\mathbf{M}_p$ and $\mathbf{M}_e$ is:

$$\mathbf{M}_p = \mathbf{U}_1 \mathbf{U}_2^\mathsf{T}, \mathbf{M}_e = \mathbf{X}\mathbf{\Lambda}\mathbf{Y} \tag{2}$$

where $\mathbf{X} \in \mathbb{R}^{p \times 2d}$ and $\mathbf{Y} \in \mathbb{R}^{q \times 2d}$ are per-edge feature matrices, and $\mathbf{X}_c = [\mathbf{F}_i^1 | \mathbf{F}_j^1], \mathbf{Y}_c = [\mathbf{F}_i^2 | \mathbf{F}_j^2]$ represent the c^{th} edge b/w node i and j

Affinity Matrix Layer

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

Power Iteration Layer

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: $\mathbf{v}^*$

Power Iteration Layer

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

$$\mathbf{v}_{k+1} = \frac{\mathbf{M}\mathbf{v}_k}{||\mathbf{M}\mathbf{v}_k||_2} \tag{3}$$

  where we initialize $\mathbf{v}_0 = 1$
- We run the assignment from Eq 3 for N iterations, and output the vector $\mathbf{v}^* = \mathbf{v}_N$.

Bi-Stochastic Layer

Bi-Stochastic

In: $\mathbf{v}^*$

Computations: reshape $\mathbf{v}^*$ to a matrix and apply eqs. (19)

Out: double-stochastic confidence matrix $\mathbf{S}$

Bi-Stochastic Layer

- Make the result of the power iteration layer bi-stochastic which means mapping the eigenvector $\mathbf{v}^*$ onto the L1 constraints of the matching problem $\forall i \sum_a \mathbf{a}_{ia} = 1$ and $\forall a \sum_i \mathbf{v}_{ia} = 1$.

Bi-Stochastic Layer

- Make the result of the power iteration layer bi-stochastic which means mapping the eigenvector $\mathbf{v}^*$ onto the L1 constraints of the matching problem $\forall i \sum_a \mathbf{a}_{ia} = 1$ and $\forall a \sum_i \mathbf{v}_{ia} = 1$.
- Takes as input vector $\mathbf{v}^* \in \mathbb{R}^{nm}$, reshaped to a matrix of size $n \times m$, and outputs the bi-stochastic matrix $\mathbf{S}$
- Given a starting matrix $\mathbf{S}_0 = (\mathbf{v}^*)_{n \times m}$, we run the following assignments for a number of iterations

$$\mathbf{S}_{k+1} = \mathbf{S}_k [\mathbf{1}_n^\intercal \mathbf{S}_k]^{-1}, \mathbf{S}_{k+2} = [\mathbf{S}_{k+1} \mathbf{1}_m^{-1}] \mathbf{S}_k \tag{4}$$

Voting Layer

Voting

In: $\mathbf{S} \in \mathbb{R}^{n \times m}$

Computations: $\text{softmax}(\alpha \mathbf{S})$

Parameters: scale α

Out: displacement vector $\mathbf{d}$ as given by eq. (22)

Voting Layer

Converting Confidence-maps to Displacement

- Given $\mathbf{S} \in \mathbb{R}^{n \times m}$, normalize for each assigned point i, its corresponding candidate scores given by the i^{th} 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

$$\mathbf{d}_i = \frac{\exp \alpha \mathbf{S}(i, 1...m)}{\sum_j \exp \alpha \mathbf{S}(i, j)} \mathbf{P} - \mathbf{P}_i \tag{5}$$

Loss

Loss

In: $\mathbf{d}, \mathbf{d}^{gt}$

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

Outline

CUB Dataset

- 11,788 images of 200 bird categories, with bounding box object localization and 15 annotated parts
- The number of points in the two graphs are maximum $n = 15$ and $m = 256$

CUB Dataset

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.

CUB Dataset

Pascal-VOC Dataset

- Annotations of body parts for 20 semantic classes. There are 7,000 annotated examples for training, and 1,700 for testing

Pascal-VOC Dataset

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.

Pascal-VOC Dataset




Conclusion

- End-to-end learning framework for graph matching
- The main challenges are the calculation of backpropagated derivatives through complex matrix layers and the implementation of the entire framework in a computationally efficient manner
- Limited to small graphs