A NOVEL APPROACH FOR PART BASED OBJECT MATCHING USING DISTANCE METRIC LEARNING WITH GRAPH CONVOLUTIONAL NETWORKS

V. Kozlov[1,*], A. Maysuradze[1]

[1] Lomonosov Moscow State University - kozlov.vld@mail.ru, maysuradze@cs.msu.ru

KEY WORDS: Object tracking, Part based representation, Delaunay triangulation, Graph matching, Siamese network, Graph convolutional network, Distance metric learning.

ABSTRACT:

Part-based object representation and part matching problem often appear in various areas of data analysis. A special case of particular interest is when parts are not fully separated, but in relations with each other. The natural way to model such objects are graphs, and part matching problem becomes graph matching problem. Over the years, many methods to solve graph matching problems have been proposed, but it remains relevant due to its complexity. We propose a novel approach to solving graph matching problem based on learning distance metric on graph vertices. We empirically demonstrate that our method outperforms traditional methods based on solving quadratic assignment problem. We also provide an theoretical estimation of computational complexity of proposed method.

1. INTRODUCTION

Part-based object representation is often used in areas of image analysis and computer vision, and has applications in image detection and classification, object tracking, shape matching and more. Under part based approach the object is viewed as a set of meaningful primitive parts. One naturally arising problem for such representation is part matching, which is finding a correspondence between parts of two different objects.

Often, a relation between the parts within the object can be established. Such objects can be naturally modeled by graphs, with parts corresponding to vertices and relations corresponding to edges. In this case, part matching problem becomes graph matching problem, which is establishing correspondences between vertices with respect to edges. This problem, however, has been proven to be NP-hard. Over the years, many methods to solve this problem have beed proposed, but due to its complexity it remains highly relevant.

In this work, we concern ourselves with graph matching problem applied to matching objects on photos. We propose a novel graph matching method based on deep distance metric learning on graph vertices. We show empirically that our method achieves higher matching accuracy than graph matching methods based on traditional techniques. Moreover, it performs significantly better than these methods when actual match between graphs is low relative to their number of vertices.

The rest of this paper is structured as follows. In Section 2 we provide an overview of modern graph matching methods and discuss related works in the field. In Section 3 we describe our method in detail and provide theoretical estimates of computational complexity. In Section 4 we describe our experimental setup and provide empirical results.

2. RELATED WORK

In broad terms, graph matching problem for graphs G_1 and G_2 is finding some binary relation r between their vertices: $r \subseteq V_1 \times V_2$ (Conte et al., 2004). Often, the relation r is required to be mapping $r : V_1 \to V_2$ or even bijection. A possible additional constraint is for the mapping r to preserve edges. This special case is referred to as exact graph matching problem. However, edge preservation requirement usually contradicts high object variability that is common in the field of data analysis.

An attributed graph (Tsai and Fu, 1979) is an extension of the traditional notion of the graph. An attributed graph is a tuple $< V, E, \mu, \varepsilon >$, where $\mu = \{\mu_1, \dots, \mu_{|V|}\}$ are vertex attributes and $\varepsilon = \{\varepsilon_1, \dots, \varepsilon_{|E|}\}$ are edge attributes. This is particularly useful in data analysis, as vertex and edge attributes represent features extracted from object parts and relations between them. In this work, both vertex and edge attributes are numerical vectors.

Typically, inexact graph matching problem is formally defined as a discrete optimization problem of minimizing cost function (Yan et al., 2016). Under this approach, a pair of vertices $i \in V_1$ and $i' \in V_2$ are assigned unary matching cost $c_{ii'}$ and two pairs of vertices $(i,i'), (j,j') \in V_1 \times V_2$ are assigned pairwise matching cost $d_{ii',jj'}$ based on vertex and edge attributes and graph edge structure. The matching problem is then reduced to the binary quadratic programming problem (BQPP):

$$\sum_{i,i'} c_{ii'} r_{i,i'} + \sum_{(i,i'),(j,j')} d_{ii',jj'} r_{i,i'} r_{j,j'} \to \min_r, \qquad (1)$$

where r is a binary matrix of shape $|V_1| \times |V_2|$
r is constrained by matching requirements

Depending on matching requirements, various constraints can be used; if match is required to be a mapping, $\forall i \sum_{i'} r_{i,i'} = 1$ and $\forall i' \sum_i r_{i,i'} \leq 1$, and if match is required to be a bijection, r is a permutation matrix. In these cases, the cost function can be rewritten as $\sum_i c_{ir(i)} + \sum_{i,j} d_{ir(i),jr(j)}$, and the BQPP becomes a quadratic assignment problem (QAP, (Lawler, 1963)). It can be shown that many popular methods of graph matching based on cost optimization may be reduced to the BQPP;

[*] Corresponding author

for instance, the reduction for graph edit distance is provided in (Neuhaus and Bunke, 2007).

As QAP itself is NP-hard, multiple techniques for finding an approximate solution have been proposed over the years, including the ones based on finding primary eigenvector of cost matrix (Leordeanu and Hebert, 2005, Cour et al., 2006), projection onto convex sets-based method (van Wyk and van Wyk, 2004), modified gradient descent methods that use problem specifics to obtain better approximation of solution (Gold and Rangarajan, 1996, Leordeanu et al., 2009), and an interior point-like optimization procedure (Zhou and de la Torre, 2012). Typically, these techniques follow the same pattern:

1. approximating the original discrete problem with continuous one;
2. solving the continuous problem approximately;
3. performing some discretization procedure over the continuous solution.

To the best of our knowledge, machine learning methods in relation to graph matching have only been employed to calculate matching costs $c_{ii'}$ and $d_{ii',jj'}$ as a function of graphs (Caetano et al., 2009, Leordeanu et al., 2012, Zanfir and Sminchisescu, 2018, Nowak et al., 2018). In most cases, matching costs are treated as expert knowledge.

To sum up, the graph matching pipeline in most cases is the following:

1. for both objects a domain-specific feature extraction method is employed to provide attributed graphs;
2. from the graphs, a QAP is constructed;
3. the QAP is solved approximately, and the matching is produced.

In this pipeline, machine learning methods are typically used to train feature extraction step and potentially QAP construction step. The paper (Zanfir and Sminchisescu, 2018) in particular is very representative. The authors use deep learning-based feature extraction method to convert images to attributed graphs. Then two attributed graphs are converted to QAP. The key feature of this approach is that feature extraction, conversion and solution of QAP allow for joint back-propagation, so feature extraction and conversion steps may be trained. We use an extended version of the method provided in (Zanfir and Sminchisescu, 2018) as our baseline competitor.

3. PROPOSED GRAPH MATCHING MODEL

The obvious drawback of QAP-based approach to graph matching is high computational intensiveness inherent to it, as pairwise costs $d_{ii',jj'}$ form a 4-dimensional tensor with the size $|V_1|^2 \cdot |V_2|^2$. As such, we suggest a different approach that does not rely on solving QAP.

We propose a pipeline for graph matching based on siamese networks (Bromley et al., 1993) and distance metric learning between graph vertices. Under this approach, two graphs are processed in parallel and independently, their intermediate representations are produced, and matching is synthesized from these representations. The model can be divided in 3 consecutive parts: graph construction, graph processing and matching synthesis.

3.1 Graph Construction

Graph construction methods are domain-specific. In this work, we use a set of images with already specified keypoints with known coordinates, with every image transformed into a graph. Graph vertices correspond to keypoints of the image. We employ a pre-trained convolutional neural network to produce a feature map for the image, and use the values from that feature map taken at keypoints as vertex features. Graph edges correspond to the edges of Delaunay triangulation of keypoint set, with edge length used as edge attribute. The pipeline for attributed graph construction from images is presented on Fig. 1.

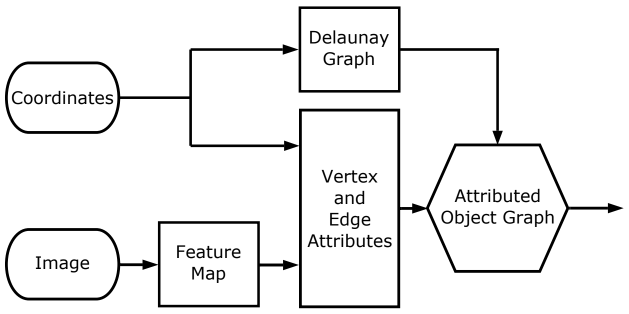

Figure 1. Flowchart of graph construction process used in the experiments.

3.2 Graph Processing

Now we observe two attributed graphs constructed from objects. We propose a machine learning model that learns to produce a matching between object graphs by learning distance metric on graph vertices. The model can be divided into the following main stages:

1. embedding stage that, given attributed graphs, constructs a new representation for each vertex and edge using provided graph;
2. similarity computation stage that produces pairwise similarity matrix between components from these representations.

For embedding stage, we propose graph convolutional networks, and for similarity computation stage, we propose metric learning on vertices. The pipeline is presented on Fig. 2.

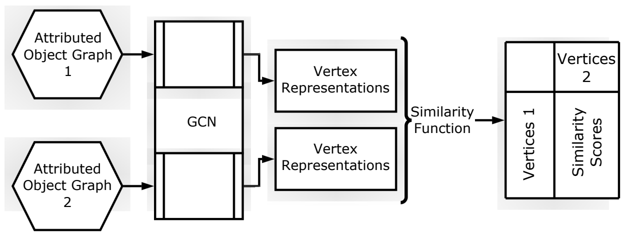

Figure 2. Flowchart of processing a pair of graphs to produce similarity scores (distance metrics) between vertices.

On the embedding stage, both graphs are processed independently and in parallel using the same model. To obtain secondary representations of the vertices, we employ a graph convolutional network (GCN) (Kipf and Welling, 2019), also known as message passing neural network. Under this approach, each layer of the network calculates new features for graph vertex using both features of the vertex itself and its neighbors. Conventional GCN does not make use of edge attributes; we, however, propose an extended version that also incorporates that information. In this model, each layer accepts an attributed directed graph $G =< V, E, \mu, \varepsilon >$ and recalculates vertex and edge attributes by formulas:

1. vertex attribute transformation:

$$\mu'_k = \sigma_v \left(W_{vself} \mu_k + \mu_k^{in} + \mu_k^{out} + b_v \right) \qquad (2)$$

$$\mu_k^{in} = \frac{1}{|N_{in}(k)|} \sum_{i \in N_{in}(k)} (W_{vin} \mu_i + W_{ein} \varepsilon_{ik})$$
$$\mu_k^{out} = \frac{1}{|N_{out}(k)|} \sum_{i \in N_{out}(k)} (W_{vout} \mu_i + W_{eout} \varepsilon_{ki})$$

2. edge attribute transformation:

$$\varepsilon'_{ij} = \sigma_e \left(W_{eself} \varepsilon_{ij} + W_{vfrom} \mu_i + W_{vto} \mu_j + b_e \right) \quad (3)$$

where μ_k and μ'_k are input and output vectors of features for vertex $k \in V$ respectively, ε_{ij} and ε'_{ij} are input and output vectors of features for edge $(i, j) \in E$ respectively, $N_{in}(k)$, $N_{out}(k)$ are neighbors of vertex k that have an edge going from them to k and to them from k respectively, W and b denote trained model parameters, σ denotes some activation function. If $N_{in}(k)$ or $N_{out}(k)$ are empty, the corresponding member simply is not computed; that means that for an isolated vertex, the layer is identical to the dense layer, and if the graph has no edges at all (is simply a set of vertices), GCN is equivalent to applying MLP to each component. The purpose of this part is to produce representations for vertices to match; therefore, we discard the edge features in the end.

As distance metric between vertices of graphs G_1 and G_2, we suggest using conventional distance between vertex embeddings. We use Mahalanobis metric, as is typically learned in contemporary metric learning problems (Bellet et al., 2013). To that end, we simply apply a linear transformation to vertex representations for both graphs and calculate pairwise Euclidean distance. The result is numeric matrix D of shape $|V_1| \times |V_2|$ of pairwise distances between graph vertices.

3.3 Matching Synthesis

Matching synthesis stage takes pairwise distance matrix and produces the binary matrix R, obtained as binarization of matrix D. We use a simple threshold rule: if the distance is less than a threshold, the vertices match, if the distance is greater, they do not. Matching stage is only used to produce matching itself, and is not used during learning process.

3.4 Model Learning

The nature of our method allows us to combine embedding stage and similarity computation stage into single pipeline that allows for backpropagation. It should be noted that in our particular case graph feature extractor based on convolutional neural network can be included into the pipeline as well, allowing for fine-tuning feature selection.

Training set consists of pairs of objects. Each pair has an associated binary matrix R that represents actual target relation between the parts. For each object we construct a directed attributed graph of parts as explained in 3.1.

For a pair of graphs from the training set, we perform forward pass up to the distance matrix D. As the surrogate loss function, we suggest a MSE-inspired loss $L(R, D) = ||R - \exp(-D^2)||_F^2 / (|V_1| \cdot |V_2|)$. This finishes the model definition.

3.5 Computational Efficiency

We stress here that our model differs dramatically from a traditional QAP-based method. During matching itself, we do not deal with pairs of pairs of vertices and their associated 4-D cost tensor $d_{ii',jj'}$. In fact, we discard edge attributes and all edge information after producing vertex representations. We expect this fact to positively affect the computational efficiency of our method. Here, we discuss the matter of computational complexity of our model.

Suppose we have two graphs $G_t =< V_t, E_t, \mu^t, \varepsilon^t >, t = 1, 2$. Each graph G_t is defined by its adjacency matrix A_t, a binary matrix of shape $|V_t| \times |V_t|$, its vertex features matrix μ^t, a continuous matrix of shape $m \times |V_t|$, and its edge features matrix ε^t, a continuous matrix of shape $n \times |E_t|$. From these, we can calculate the following auxiliary matrices:

1. G_t, H_t — incidence matrices, binary matrices of shape $|V_t| \times |E_t|$: if edge $e \in E_t$ begins in vertex $i \in V_t$ and ends in $j \in V_t$, then $(G_t)_{ie} = 1, (H_t)_{je} = 1$, otherwise it's 0. These matrices are hugely sparse, with only one 1 in each column. They can be obtined from A_t in $O(|V_t|^2)$, and $G_t H_t^T = A_t$. We note that our baseline competitor (Zanfir and Sminchisescu, 2018) makes use of these matrices as well, as inspired by (Zhou and de la Torre, 2012).

2. $c_t = (\frac{1}{\sum_i (A_t)_{ij}}, j = 1, \ldots, |V_t|)$, $r_t = (\frac{1}{\sum_j (A_t)_{ij}}, j = 1, \ldots, |V_t|)$ — inverse column and row rates of adjacency matrix. If one of the sums in denominator is 0, the respective value is considered 0.

We will also denote $[x]$ a diagonal matrix with vector x on its main diagonal.

Let us investigate the computational complexity of the forward pass of GCC vertex layer 2:

1. $W_{vself} \mu_k$ in matrix form can be computed as $W_{vself} \mu^t$, and its complexity is $O(|V_t|)$ (we consider all feature dimensionalities constant);

2. $\frac{1}{|N_{in}(k)|} \sum_{i \in N_{in}(k)} W_{vin} \mu_i$ in matrix form can be computed as $W_{vin} \mu^t A_t [c_t]$, and its complexity is $O(|V_t|^2)$;

3. $\frac{1}{|N_{in}(k)|} \sum_{i \in N_{in}(k)} W_{ein} \varepsilon_{ik}$ in matrix form can be computed as $W_{ein} \varepsilon^t H_t^T [c_t]$, with complexity $O(|E_t| \cdot |V_t| + |V_t|^2)$;

4. similarly to previous parts, μ_k^{out} in matrix form can be calculated as $W_{vout} \mu^t A_t^T [r_t] + W_{eout} \varepsilon^t G_t^T [r_t]$, and its complexity is $O(|E_t| \cdot |V_t| + |V_t|^2)$ as well;

5. complexity of other parts of 2 is negligible compared to the ones above.

Now, we investigate the computational complexity of the forward pass of GCC edge layer 3:

1. $W_{eself}\varepsilon_{ij}$ in matrix form can be computed as $W_{eself}\varepsilon^t$, and its complexity is $O(|E_t|)$
2. $W_{vfrom}\mu_i + W_{vto}\mu_j$ in matrix form can be computed as $W_{vfrom}\mu^t G_t + W_{vto}\mu^t H_t$, and its complexity is $O(|E_t| \cdot |V_t|)$.
3. complexity of other parts of 3 is negligible compared to the ones above.

As each layer has same asymptotic complexity, the combined complexity of embedding phase is $O(|E_t| \cdot |V_t| + |V_t|^2)$. Finally, the complexity of computing pairwise distances between vertex representations is $O(|V_1| \cdot |V_2|)$.

Therefore, the total computational complexity of forward pass of our model is $O(|E_1| \cdot |V_1| + |E_2| \cdot |V_2| + |V_1|^2 + |V_2|^2 + |V_1| \cdot |V_2|)$. At worst, in case of fully connected graphs, $|E_t| = O(|V_t|^2)$, and this becomes $O(|V_1|^3 + |V_2|^3 + |V_1| \cdot |V_2|)$. In case of Delaunay triangulation, however, $|E_t| = O(|V_t|)$ (specifically, $|E_t| \leq 3|V_t| - 6$, and the same applies to any other planar graph), and total complexity of forward pass is $O(|V_1|^2 + |V_2|^2 + |V_1| \cdot |V_2|)$, which is significantly better than $O(|V_1|^2 \cdot |V_2|^2)$ that is inherent to solving QAP. Additionally, less complex forward pass is likely to result in less complex backwards pass during model training, significantly reducing training time.

4. EXPERIMENTS

4.1 Dataset

To test our method, we apply it to matching points in images. We use dataset CUB_200_2011 (Wah et al., 2011). The dataset contains almost 12000 photographs of birds of different species and in different poses. On each photo, no more than 15 keypoints are marked, each annotated with its type. There are 15 types of points in total, denoting different parts of the bird's body. In a photo, no two points have the same type.

For each image, the bounding box of a bird is provided in the dataset. We use this information to normalize the images. We cut out the bird from the image using bounding box information and reshape it to the size of 224 × 224. The annotated point coordinates are transformed accordingly.

The original dataset provides no graphs. We construct the graph edges using Delaunay triangulation of keypoint set. For vertex feature extraction, we use MobileNetV2 convolutional network (Sandler et al., 2018). As our feature map, we use the output of `block_5_expand_relu`. Then, we use the elements of the feature map in positions that correspond to the coordinates of the keypoints as vertex features. This means the size of vertex feature vectors in object graph is 192. We use Euclidean distance between points as the only edge feature. We do not fine-tune our feature extraction model.

The original dataset comes already split in non-overlapping train and test subsets, almost 6000 images each. We make use of this and select training and test pairs from respective sets. This ensures that images used for training are never used for testing, and vice versa. Unlike paper (Zanfir and Sminchisescu, 2018), we use arbitrary test pairs, so we can not expect that birds are in similar poses in each pair to be tested.

The original dataset has no target relations. We assume in our matching problem that same body parts on images match. If a body part is visible on one image in a pair but not on other, it does not have any match.

4.2 Models and Metrics

We consider 2 models: baseline model, based on (Zanfir and Sminchisescu, 2018), and our model. Both models begin with the same feature extraction step and embedding step. After that, baseline model pipeline performs learning and approximately solving QAP, and our model performing distance metric learning and fast matching.

Our baseline competitor is not identical to the one described in (Zanfir and Sminchisescu, 2018). First, we use our own vertex and edge features. Second, we extend the model by adding the embedding like in our approach. This change, however, is expected to actually make our baseline stronger because we allow vertex and edge features to incorporate information from the neighborhood, unlike in the original. Only when we get representations for both vertices and features from GCN, we proceed with the pipeline from (Zanfir and Sminchisescu, 2018) to construct and solve QAP. In this case, we do not discard edge information. The baseline competitor still contains a large 4D tensor in its pipeline.

For embedding step in both baseline and our model, we use a GCN with of 4 layers with output dimensions of 128, 64, 32 and 16 for vertex features and 1 for edge features. As activation, we use hyperbolic tangent for vertex features and ReLU for edge features. We also apply a dense layer with linear activation, square weight matrix and no bias to final vertex features. This layer performs linear transform of vertex representations, which is equivalent to learning Mahalanobis distance.

To compare our models, we use two performance measures:

- accuracy: $Acc(R, M) = \frac{\sum_{i \in V_1, i' \in V_2} [R_{ii'} = M_{ii'}]}{|V_1||V_2|}$;
- Jaccard measure: $Jacc(R, M) = \frac{|R \cap M|}{|R \cup M|}$.

Here R (actual matching) and M (predicted matching) are interpreted as both binary matrices of size $|V_1| \times |V_2|$ and as subsets of $V_1 \times V_2$. Accuracy is a standard quality metric for graph matching. We also decided to use Jaccard measure because number of pairs in R is small compared to size of $V_1 \times V_2$. During the experiments, the model is trained on random pairs of images drawn from training subset, and performance metrics are averaged over random pairs of images drawn from test subset.

4.3 Results

We have trained both our model and our baseline competitor on the same dataset and compared their average matching accuracies and Jaccard measures. The results are provided in Table 1. This shows that our model clearly outperforms the competitor in general.

In addition, we have conducted the study of method robustness. Namely, we wanted to know how the methods would behave if actual matching is small, that is, when the objects have not many parts in common. For that, we have recorded average matching accuracies and Jaccard measures for various sizes of

Model	Acuracy	Jaccard
Matching model	0.91	0.84
Baseline	0.87	o.64

Table 1. Testing results for matching methods under consideration.

actual match from 6 to 13 (as the others sizes were too rare to draw conclusions). The results are presented on Fig. 3 and 4. It can clearly be seen that our matching model outperforms the baseline for every size.

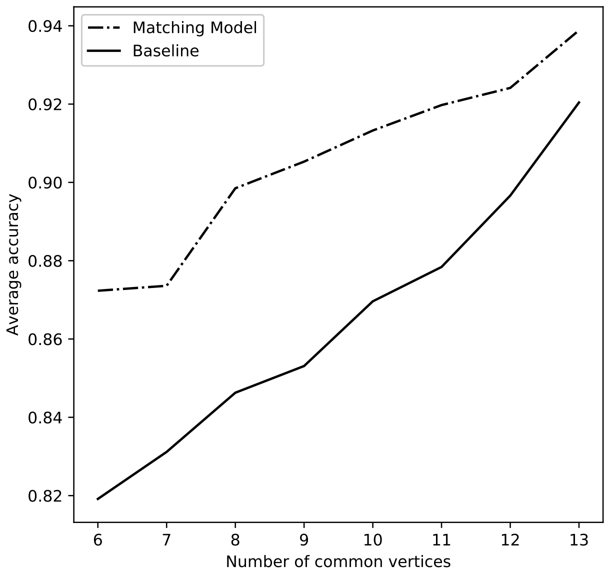

Figure 3. Average matching accuracies for various sizes of actual match

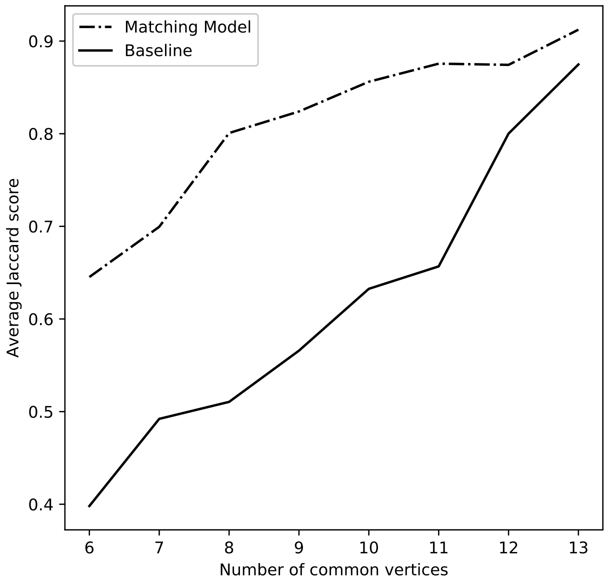

Figure 4. Average Jaccard mesures for various sizes of actual match

5. CONCLUSION

We have presented and examined a novel machine learning-based approach to graph matching that abandons typical method of solving a quadratic assignment problem and instead uses a siamese graph convolutional network that perform distance metric learning on graph vertices. We have demonstrated empyrically that our approach outperforms traditional QAP-based graph matching approache. We have also provided a theoretical estimation of computational complexity of the approach, showing that it under many circumstances less computationally intensive than QAP-based ones.

ACKNOWLEDGEMENTS

The study was partially supported by RFBR (project no. 20-01-00664-a) and state-financed research work no. 5.1.21 of Lomonosov Moscow State University.

REFERENCES

Bellet, A., Habrard, A., Sebban, M., 2013. A Survey on Metric Learning for Feature Vectors and Structured Data. http://arxiv.org/abs/1306.6709.

Bromley, J., Bentz, J. W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Säckinger, E., Shah, R., 1993. Signature Verification Using a "Siamese" Time Delay Neural Network. *International Journal of Pattern Recognition and Artificial Intelligence*, 07(04), 669–688.

Caetano, T. S., McAuley, J. J., Cheng, L., Le, Q. V., Smola, A. J., 2009. Learning graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(6), 1048–1058.

Conte, D., Foggia, P., Sansone, C., Vento, M., 2004. Thirty Years of Graph Matching in Pattern Recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(03), 265–298.

Cour, T., Srinivasan, P., Shi, J., 2006. Balanced Graph Matching. *Advances in Neural Information Processing Systems (NIPS)*, MIT Press, Canada, 313—-320.

Gold, S., Rangarajan, A., 1996. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4), 377–388. http://ieeexplore.ieee.org/document/491619/.

Kipf, T. N., Welling, M., 2019. Semi-supervised classification with graph convolutional networks. *5th International Conference on Learning Representations, ICLR 2017 — Conference Track Proceedings*.

Lawler, E. L., 1963. The Quadratic Assignment Problem. *Management Science*, 9(4), 586–599. http://pubsonline.informs.org/doi/abs/10.1287/mnsc.9.4.586.

Leordeanu, M., Hebert, M., 2005. A spectral technique for correspondence problems using pairwise constraints. *Proceedings of the IEEE International Conference on Computer Vision*, II, IEEE, 1482–1489.

Leordeanu, M., Hebert, M., Sukthankar, R., 2009. An Integer Projected Fixed Point Method for Graph Matching and MAP Inference. *Proceedings Neural Information Processing Systems*, 1number 3, 1114–1122.

Leordeanu, M., Sukthankar, R., Hebert, M., 2012. Unsupervised learning for graph matching. *International Journal of Computer Vision*, 96(1), 28–45.

Neuhaus, M., Bunke, H., 2007. A quadratic programming approach to the graph edit distance problem. *International Workshop on Graph-Based Representations in Pattern Recognition*, Springer, 92–102.

Nowak, A., Villar, S., Bandeira, A. S., Bruna, J., 2018. Revised Note on Learning Quadratic Assignment with Graph Neural Networks. *2018 IEEE Data Science Workshop, DSW 2018 - Proceedings*, IEEE, 229–233.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L. C., 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 4510–4520. http://arxiv.org/abs/1801.04381.

Tsai, W.-H., Fu, K.-S., 1979. Error-correcting isomorphisms of attributed relational graphs for pattern analysis. *IEEE Transactions on systems, man, and cybernetics*, 9(12), 757–768.

van Wyk, B. J., van Wyk, M. A., 2004. A POCS-based graph matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11), 1526–1530. http://ieeexplore.ieee.org/document/1335455/.

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

Yan, J., Yin, X.-C., Lin, W., Deng, C., Zha, H., Yang, X., 2016. A short survey of recent advances in graph matching. *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, ICMR ’16, ACM, New York, NY, USA, 167–174.

Zanfir, A., Sminchisescu, C., 2018. Deep Learning of Graph Matching. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2684–2693.

Zhou, F., de la Torre, F., 2012. Factorized graph matching. *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, IEEE, 127–134.