Graphs in pattern recognition: successes, shortcomings and new perspectives

Donatello Conte

Graphs in pattern recognition: successes, shortcomings and new perspectives

Donatello Conte[a,*]
[a]Université de Tours, Laboratoire d'Informatique Fondamentale et Appliquée de Tours (LIFAT - EA6300), Tours, France

Abstract. Graphs in Pattern Recognition have been studied starting from the late seventies with alternating events of successes and failures. Far from wanting to propose a new survey on the subject, this paper aims to present the areas in which graphs have been shown to be effective, and to illustrate the problems still open in this research domain, with a focus on recent developments moving in the direction of Deep Learning. The paper intends to arouse the interest of neophytes to this interesting domain that is the representation of the world through graphs.

Keywords: Graphs in Pattern Recognition, Graph matching, Graph Topology, Graph Embedding, Graph Neural Networks.

*Donatello Conte, donatello.conte@univ-tours.fr

1 Introduction

Graphs are powerful data structures that represent mainly relationships between entities.

Graph representation is very common in many application contexts, from social networks to chemo-informatics and transportation systems. But graphs have been effectively used in many other contexts, in particular image processing and computer vision.

Pattern Recognition methods using graphs have been proposed since early 90's (see[1] for a survey on first papers discussing graph-based representation in Pattern Recognition). In this early period, until about 2010, researchers focused on problems of graph matching,[2] graph embedding ([3–5]), and graph topology ([6,7]). After there was an explosion of interest in deep learning techniques, and the use of graphs in neural networks ([8,9]). Recent trends can be found in some recent surveys[10,11] and a dozen other surveys are present in the literature.

However the goal of this paper is not to write a one more survey, rather a historical excursion of the use of graphs in pattern recognition. And even more than a historical excursion, to point out

some method that was a pillar in this topic.

The intention here is to propose a generic view of the different possibilities of the use of graphs in Pattern Recognition, possibly for a non-expert audience (which might differentiate it from other surveys). By presenting the best successes, and also what did not succeed, the reader can already get an idea of where to start when beginning research in this field. The reader will then be able to realize where graphs can be used effectively, and possibly jump into the fray to propose new solutions to open problems.

The rest of the paper is organized as follows: in the Section 2 we present some of the main methods that have had successful results in the use of graphs in pattern recognition; graph-based representation presents some drawbacks that we illustrate in the Section 3; we cannot pass over in silence, of course, a discussion of recent propositions of graph neural networks (GNN) which we do in the Section 4; in the Section 5 we describe the main applications and benchmarks used for graph problems in pattern recognition; and we discuss of main open problem in the Section 6; finally we draw some conclusions in the Section 7.

2 Some successes on the use of graphs in pattern recognition

2.1 Graph Matching

Graph matching is the problem of finding an optimal correspondence between the vertices and edges of two graphs. The Graph Matching problem can be divided into two general categories: exact matching and inexact matching. Exact matching aims to find a strict correspondence, or at least among their substructures. In the inexact matching, this constraint is relaxed to find the bijection between the vertex that optimizes a certain affinity or distortion criterion.

Graph Matching is a fundamental problem in computer science and relates to many areas such as combinatorics, pattern recognition, multimedia and computer vision.

Referring back to other surveys for details, we would like to highlight here some of the most effective methods of graph matching.

Graph Matching can be formulated as quadratic assignment problem (QAP) that is known to be NP-hard. One of the most successfully proposal for finding a solution in a reasonable time is the one proposed by Zhou et al.[12] Following[12] graph with n nodes and m directed edges are represented by a 4-tuple $G = \{\mathbf{P}, \mathbf{Q}, \mathbf{G}, \mathbf{H}\}$, where $\mathbf{P}^{dp \times n}$ contains features for nodes and $\mathbf{Q}^{dg \times m}$ features for edges (dp and dq are the dimensions of the features space). The topology of the graph is encoded by two node-edge incidence matrices $\mathbf{G}$, $\mathbf{H}$ where $g_{ic} = h_{jc} = 1$ if the c^{th} edge starts from the i^{th} node and ends at the j^{th} node. Given a pair of graphs, G_1 and G_2, an affinity matrix $\mathbf{K} \in \mathbb{R}^{n_1 n_2 \times n_1 n_2}$ is defined encoding the node and edge affinities, to measure the similarity of each node and edge pair respectively. Therefore, the problem of Graph Mathing consists in finding the optimal correspondence $\mathbf{X}$ between nodes, such that the sum of the node and edge compatibility is maximized. Using K, Graph Mathing can be formulated as the QAP of Eq. 1.Given $\mathbf{X}$, which is a matrix in which each element (i, j) is one if node i of graph G_1 is to be associated with node j of graph G_2 and zero otherwise, the formula expresses the fact that, among all possible combinations of values for the matrix X, the algorithm provides as a solution the one that maximizes the affinities between the nodes.

$$\max_{\mathbf{X} \in \Pi} vec(\mathbf{X})^T \mathbf{K} vec(\mathbf{X}) \tag{1}$$

where $\mathbf{X}$ is constrained to be a one-to-one mapping, and Π is the set of partial permutation

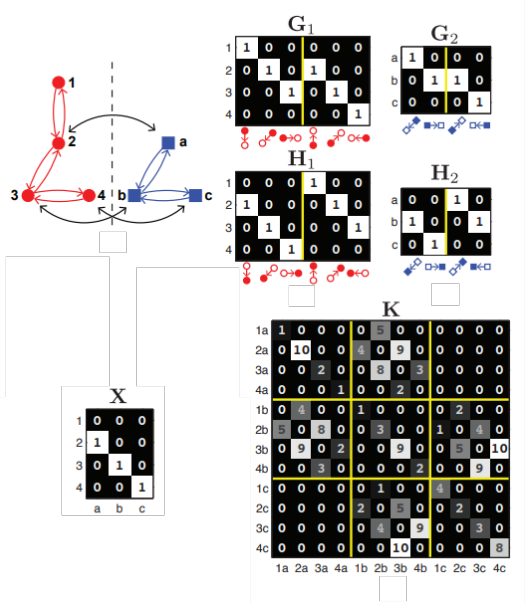


Fig 1 Example of Graph Matching defition as defined in[12] (Figure modified from the paper). Note that the assignement of X corresponds to the node associations for which in the $\mathbf{K}$ matrix the affinities have the highest values.

matrices. In Fig. 1 there is an illustrative example of these definitions.

The main idea of this paper is to propose a new factorization of the pairwise affinity matrix $\mathbf{K}$. This factorization provides a light-weight representation for Graph Matching problems, allows a unification of Graph Matching methods (subgraph isomorphism, graph edit distance, etc.), and it allow a good performance in terms of time. For details on this factorization please refer to.[12] Results are very competitive in terms of time and quality of assignment, with respect many of state-of-the-art methods.[13–15]

A particular Graph Matching (GM) problem which has been addressed by many researchers is the so-called Graph Edit Distance (GED). The latter can be stated as follows: given two graphs, and some edit operations with an edit cost, graph edit distance is the problem of finding the minimum sum of edit cost to transform a graph into the other; Graph Edit Distance provides also a node-to-node correspondence between the nodes of the two graphs. Actually in 2021, Raveaux[16] shows that the GED problem can be equivalent to the GM problem under certain permissive conditions.

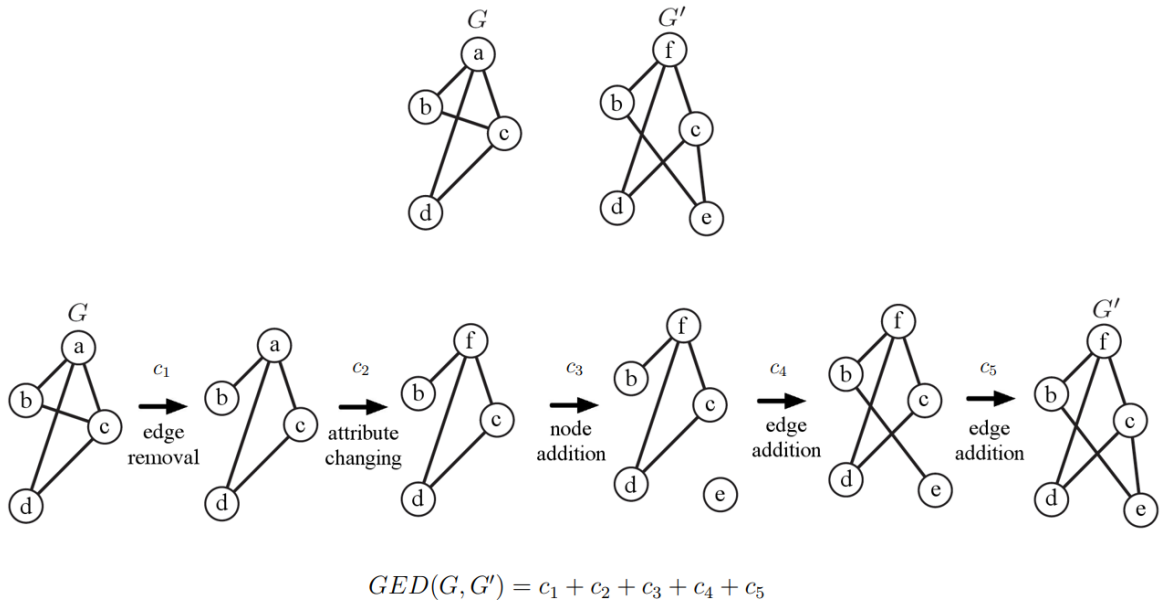
Nevertheless, as GED was widely addressed in literature we propose here some of key papers on this specific problem.

We have seen a quadratic formulation of Graph Matching problem, and therefore equivalently there are some quadratic formulation of GED. But as quadratic formulation is NP-hard, many papers proposed a linear formulation of the GED problem, at the expense of a less precise but still acceptable final solution. However some researcher have tried to find techniques for solving the linear formulation that are both fast and accurate. Darwiche et al.[17] propose the use of some research operational methods, in particular here is the Local Branching technique, to solve efficacy the GED problem. Here we present the general principle of this proposal.

First an attributed graph is defined as a 4-tuple $G = (V, E, \mu, \xi)$ where, V is the set of vertices, E is the set of edges, such that $E \subseteq V \times V$, $\mu : V \rightarrow L_V$ (resp. $\xi : E \rightarrow L_E$) is the function that assigns attributes to a vertex (resp. an edge), and L_V (resp. L_E) is the label spaces for vertices (resp. edges). Next, given two graphs $G = (V, E, \mu, \xi)$ and $G' = (V', E', \mu', \xi')$, GED is the task of transforming one graph source into another graph target. To accomplish this, GED introduces the vertices and edges edit operations: $(u \rightarrow v)$ is the substitution of two nodes, $(u \rightarrow \epsilon)$ is the deletion of a node, and $(\epsilon \rightarrow v)$ is the insertion of a node, with $u \in V, v \in V'$ and ϵ refers to the empty node. The same logic goes for the edges. In mathematical formula the Graph Edit Distance between two graphs G and G' is defined by:

$$d_{\lambda_{min}}(G, G') = \min_{\lambda \in \Gamma(G,G')} \sum_{e_i \in \lambda} c(e_i) \tag{2}$$

where $\Gamma(G, G')$ is the set of all complete edit paths, λ_{min} represents the set of operations with the minimal cost, and c is the cost function that assigns the costs to elementary edit operations. See



$$GED(G, G') = c_1 + c_2 + c_3 + c_4 + c_5$$

Fig 2 An intuitive explanation of the GED: Given two graphs G and G', the figure shows the edit operations to transform G in G', each operation having a cost c_i. The sum of the costs is the GED measure between the two graphs.

at the Fig. 2 for an intuitive explanation of the GED.

The main idea in the linear formulation of GED consists in determining the permutation matrix minimizing the L_1 norm of the difference between adjacency matrix of the input graph and the permuted adjacency matrix of the target one. The model is as follows:

$$\min_{P,S,T \in \{0,1\}^{N \times N}} \sum_{i=1}^{N} \sum_{j=1}^{N} c\left(\mu(u_i), \mu'(v_j)\right) P^{ij} + \left(\frac{1}{2} \times const \times (S+T)^{ij} \right) \tag{3}$$

such that

$$(AP - PA' + S - T)^{ij} = 0 \; \forall i, j \in \{1, N\} \tag{4}$$

$$\sum_{i=1}^{N} P^{ik} = \sum_{j=1}^{N} P^{kj} = 1 \; \forall k \in \{1, N\} \tag{5}$$

where A and A' are the adjacency matrices of graphs G and G' respectively, $c : (\mu(u_i), \mu'(v_j)) \rightarrow \mathbb{R}^+$ is the cost function that measures the distance between two vertices attributes. As for P, S

and T, they are the permutation matrices of size $N \times N$, and of type boolean. P represents the vertices matching e.g. $P^{ij} = 1$ means a vertex $i \in V \cup \{v_\phi\}$ is matched with vertex $j \in V' \cup \{v_\phi\}$. While S and T are for edges matching. Local Branching heuristic is a local search approach, that makes use of MILP solver to explore the neighborhood of solutions through a branching scheme. In addition, it involves mechanisms such as intensification and diversification. In[17] they adapt the branching scheme and intensification and diversification phases in the case of Graph Matching. We refer to the paper for details. Authors show that for classical benchmarks for GED (see Section 5 for details on benchmarks) the method is in average more accurate than others by one or two orders of magnitude while remaining competitive in terms of execution time.

The difficulty of solving the GED and the quality of the result also depends on the definition of the edit cost. For this reason, some researchers, instead of focusing on the search for new resolution techniques, propose to learn which are the best edit costs, based on the problem at hand, and in some precise application contexts where it is possible to know some instances of correspondence between graphs from which to learn. Some examples of this technique can be found in.[18–21] More recently there some deep learning technique to learn edit costs for GED, we will discuss about that in Secion 4. In Fig. 3 we illustrate the basic principle of edit cost learning: some ground truth data are available that indicate the true correspondences between a set of graph couples; theses correspondences are used in some learning schema (neural networks or other regression schema) to learn the best edit cost that give the true correspondence between graphs as solution of GED problem; then these cost are used in an application context. In[19] authors show that, learning costs, given a fixed dataset in which some graph correspondences are known, provide better results than using some fixed edit costs.

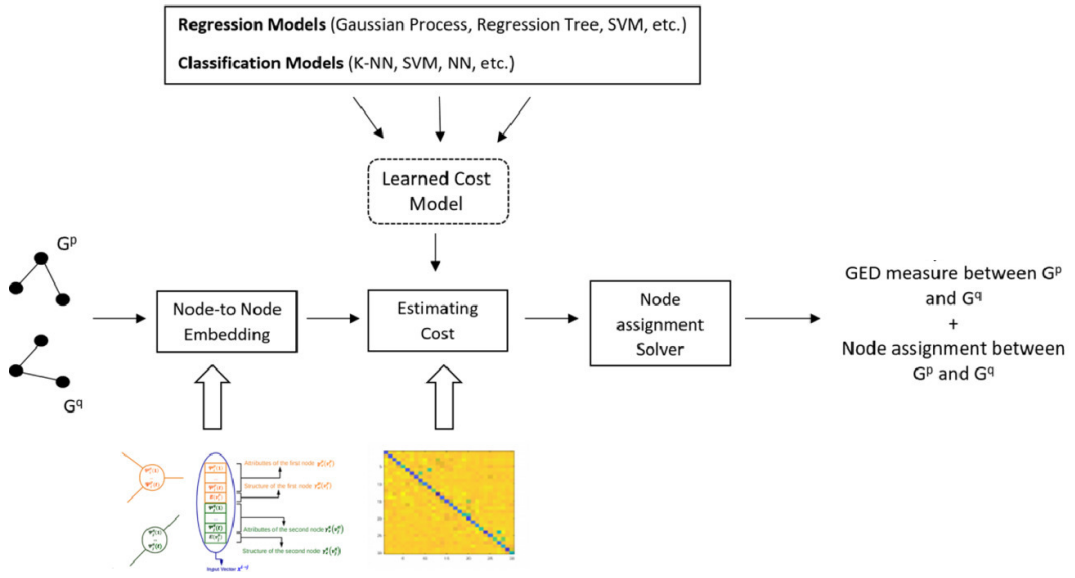


Fig 3 A general framework for edit cost learning (Figure modified from the paper[19]).

2.2 Graph Embedding

In some contexts, particularly in classification and regression applications, the node-to-node correspondence between graphs is not necessary, and what is useful is to be able to compare graphs with each other in order to have a measure of distance for use with machine learning tools for regression and classification. In these cases graph embedding has emerged as a promising solution. Graph embedding methods map either explicitly or implicitly graphs into high dimensional spaces hence allowing to perform the basic mathematical computations required by various statistical pattern recognition techniques. Graph embedding methods appear thus as an interesting solution to address graph clustering and classification problems.

The implicit graph embedding methods are based on graph kernels. A graph kernel is a function that can be thought of as a dot product in some implicitly existing vector space. Explicit graph embedding methods explicitly embed an input graph into a feature vector and thus enable the use of all the methodologies and techniques devised for vector spaces.

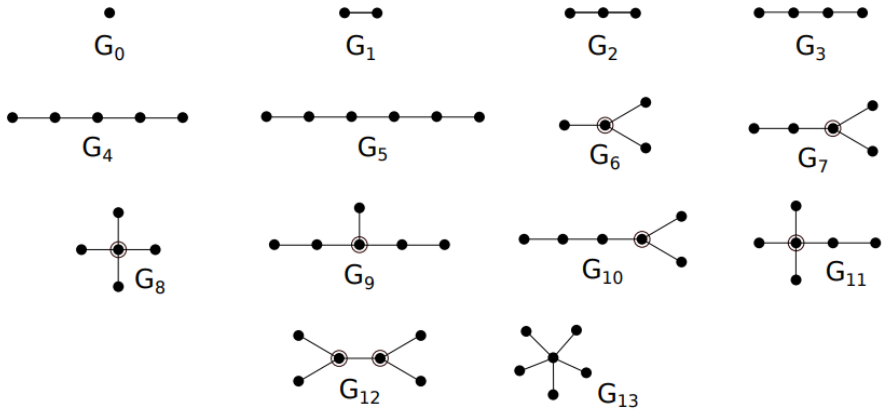


Fig 4 The set of treelets having a size lower than or equals to 6 nodes (taken from[22]).

Among the proposals that have been made in recent years, we choose two in particular that have proven particularly effective in classification contexts.

Treelet kernel[22] is a graph kernel based on a bag of non linear patterns which computes an explicit distribution of each pattern within a graph. This method explicitly enumerates the set of treelets included within a graph. The set of treelets, denoted $\mathcal{T}$, is defined as the 14 trees having a size lower than or equals to 6 nodes (Fig. 4, taken from the paper, shows the set of treelets). Thanks to the limited number of different patterns encoding treelets, an efficient enumeration of the number of occurrences of each labeled pattern within a graph can be computed by algorithm defined in the paper.[22] Treelet kernel between graphs is defined as a sum of sub kernels between common treelets of both graphs (Eq. 6)

$$K_{\mathcal{T}}(G, G') = \sum_{t \in \mathcal{T}(G) \cap \mathcal{T}(G')} k(f_t(G), f_t(G')) \tag{6}$$

where $\mathcal{T}(G)$ encodes the set of treelets included within G, $f_t(G)$ encodes the number of occurrences of each treelet $t \in \mathcal{T}$ and $k(.,.)$ defines any positive definite kernel between real numbers

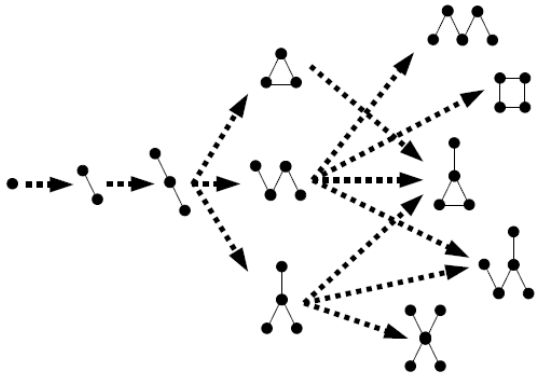


Fig 5 The non-isomorphic graph network used to embed the topology.[4]

such as linear, Gaussian or polynomial kernel. Each sub kernel $k(.,.)$ encodes the similarity of the number of occurrences for each treelet t common to both graphs to be compared.

The topological embedding method proposed in[23] uses a generic lexicon of topological structures that could be enumerated in graphs during the computation of the vectorial signature of the graphs. This lexicon must be comprehensive enough to ensure discrimination from a graph to another. They have therefore decided to take as a baseline the non-isomorphic graphs network presented in.[24] The network presents all graphs composed of n edges up to N (where N is the maximum number of edges) (see Fig. 5 for an example). The vectorial representation of a graph topology will be built by counting the occurrences of each pattern of the lexicon: each element of the vector is the frequency of apparition of a pattern, which represents a descriptor of a part of the graph. This vectorial representation needs now to be enriched by encapsulating the information provided by attributes that can be associated to the edges and vertices, or by discretizing numerical attributes or by performing a clustering in the label space using attributes as feature vectors. More details about this topological embedding method can be found in.[23]

Just to get an idea of the efficiency of these methods, we present in Table 1 an extract of the results of both methods, presented in,[4] on known graph bases for classification.

Table 1 Classification results for different graph embedding methods and datasets (extracted from[4]).

	AIDS	MAO
Treelet kernel[22]	99.1	91.2
Topological Embedding[23]	99.4	91.2

2.3 Graph Topology

Graph representation is particularly useful when the data encodes information in which topology is important, notably image processing. Many classical image processing have been efficiently handled by a graph-based representation: image segmentation,[25] LBP coding,[26] Connected Components Labeling.[27]

Graph representation is particularly efficient when we want to represent an image at different resolution in a multi-level representation of an image called pyramid.[28] This hierarchy may be encoded using Irregular Pyramids.[29,30] These data structures encode each image as a graph whose nodes and edges respectively correspond to regions and region's adjacencies. Irregular pyramids are a stack of successively reduced graphs where each graph is constructed from the graph below by selecting a specific subset of vertices and edges. For generation of irregular pyramids, two basic operations on graphs are needed: edge contraction and edge removal. The former merge two connected nodes in one, removing the edge connecting them. All edges that were incident to the joined vertices will be incident to the resulting vertex after the operation. Edge removal removes an edge from the graph, without changing the number of vertices or affecting the incidence relationships of other edges. In each level of the pyramid, the vertices and edges disappearing in level above are called *non-surviving* and those appearing in the upper level *surviving* ones.

There are different structures to build the irregular pyramid such as simple graphs,[31] dual graphs[32] and combinatorial maps.[33] Combinatorial Maps are most efficient data structure to built and represent irregular pyramid:[25] within the combinatorial maps the dual graphs may be implicitly

encoded and thus updated, this property allows to decrease both the memory and computational time requirements; combinatorial maps preserve the local orientation of edges around vertices and faces; combinatorial map formalism may be easily extended to higher dimensions.

3 Drawbacks of graph-based representation

The main drawback in the use of graph data structures is the processing times. In all the problems described above, the execution time remains a challenge to solve. The GED and GM problems have been proven to be NP-hard.[34,35] So, solving the problem to optimality cannot be done in polynomial time with respect to the size of the input graphs. On the other hand, heuristics are used when the demand for low computational time dominates the need to obtain optimal solutions. Graph embedding is also a costly operation. For example, treelet kernel requires to enumerate all labeled treelets from a graph with an overall complexity required equals to $O(nd^5)$ where n is the number of nodes and d is the maximum degree of the graph.[22] Construction of Irregular Pyramid requires also many operations,[36] and for this reason some parallel algorithms are proposed to overcome this problem.[6]

Another drawback of using graph-based representation, is that in several application context this kind of representation is not unique and the way in which the graphs are defined to represent the data strongly influences the results. If, for example, a graph-based representation of chemical data, or social networks, is quite straightforward, in domains such as image processing or computer vision, there are many possible alternatives of graph-based representations. Attributes on nodes and edge contribute to the variety of possible data representations. Data should be represented as nodes, or attributes on it, as edges or edges attributes. For example, when you deal with spatio-temporal data, you should choose to represent temporal data by a sequence of nodes at the different

time instances, or you can choose to represent a time series data as an attribute of a node. This variety of representations makes it difficult to generalise the algorithms to all application domains. Even more, in the contexts of learning, the solutions proposed are very specific to the domain under consideration.

4 The revival of graphs: Graph Neural Networks (GNN) and other machine learning tools for graphs

For the reasons mentioned in the previous section 3, research in the domain of graphs has always remained a somewhat niche research, without crossing the boundaries of a more or less restricted community. This is not the case nowadays: with the advent of Deep Learning, many researchers have proposed graph-based neuron network solutions. The positive and surprising thing is that these solutions have provided excellent results in many application contexts, particularly in the domain of image processing and computer vision, in which graphs had not been very successful until now (apart from a few happy exceptions). Today, however, there is great interest in graph-based neural networks and the major conferences in the field of pattern recognition, computer vision, and machine learning always contain many papers on this topic.

In this section, we give a brief overview of the principles behind these techniques and a brief excursion of the possible proposals in the domain of GNNs, referring to the numerous papers in the scientific literature to explore the subject further. We are inspired here mainly by the work of Wu et al.[8] which is a good survey to introduce the topic.

4.1 Basics on Graph Neural Networks

The basic principle of a Graph Neural Network is to update node features, in the different level of the network, based on an information diffusion mechanism, i.e. by exchanging node neighborhood information. This exchange is made by means of convolution operators defined on graph nodes. Convolution operators fall into two categories, spectral-based and spatial-based. Spectral based approaches define graph convolutions by introducing filters from the perspective of graph signal processing.[37] Spatial-based approaches defines graph convolutions by information propagation. Since GCN[38] bridged the gap between spectral-based approaches and spatial-based approaches, spatial-based methods have developed rapidly recently due to its attractive efficiency, flexibility, and generality.[8]

The basic convolution operator expressed in mathematical formulas is as follows. For this purpose we define here a graph as $G = (V, E)$ where V is the set of vertices and E is the set of edges. The neighborhood of a node i is defined as $N(i) = \{j \in V | (i, j) \in E\}$. $\mathbf{A}$ is the adjacency matrix of G ($A_{ij} = 1$ if $e_{ij} \in E$, 0 otherwise) and $\mathbf{F} \in \mathbf{R}^{n \times d}$ is node feature matrix where $f_i \in \mathbf{R}^d$ represents the feature vector of a node i. Analogously graph may have an edge attributes matrix $\mathbf{F}^e \in \mathbf{R}^{m \times e}$. So a generic convolution operator for a node i in a layer k is defined by Eq. 7, where $a(\cdot)$ is an activation function and $\mathbf{h}^{(0)} = 0$. $\mathbf{W}$ and $\mathbf{\Theta}$ are learnable model parameters that are learned with classical tools of Deep Learning.

$$\mathbf{y}_i^{(k)} = a(\mathbf{W}^{(k)}\mathbf{f}_i + \sum_{j \in \mathbf{N}(i)} \mathbf{\Theta}^{(k)}\mathbf{h}^{(k-1)}) \tag{7}$$

The main intuition behind the formula is that, at each convolution level, the new features of a node results from the features of the node itself, plus the weighted average of all neighbors' node

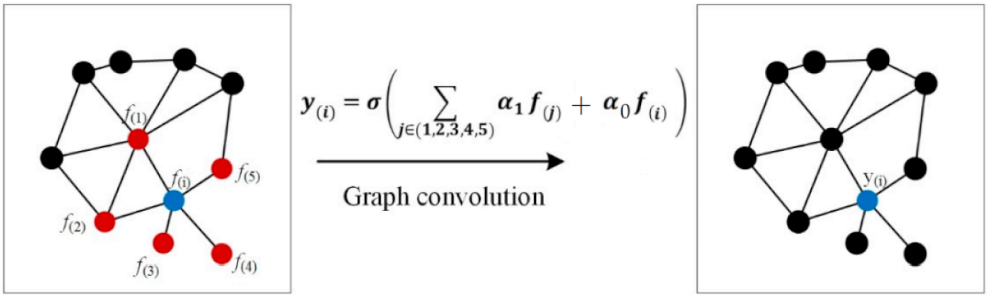


Fig 6 An illustration of graph convolution on a graph node: here $f(i)$ is the input features vector of the node i, $f(j)$ are the features vectors of the neighboring nodes j of i, $\alpha_0, \ldots, \alpha_5$ are learnable parameters, and $y(i)$ is the output feature vector (figure adapted from[39]).

features. In the Fig. 6 this principle is visually explained for a node of an example graph.

4.2 A brief survey on GNN models

The general formula presented above, declines in various versions in the various propositions of GNN models. Diffusion Graph Convolution (DGC)[40] add a transition probability matrix in the convolution to measure the contribution of each neighbors to a central node. As the number of neighbors of a node can vary from one to a thousand or even more, it is inefficient to take the full size of a node's neighborhood. GraphSage[41] adopts sampling to obtain a fixed number of neighbors for each node. Graph Attention Network (GAT)[42] adopts attention mechanisms to learn the relative weights between connected nodes. Mixture Model Network (MoNet)[43] introduces node pseudo-coordinates to determine the relative position between a node and its neighbor. In such a way, the parameters of a graph filter can be shared across different locations. PATCHY-SAN[44] orders neighbors of each node according to their graph labelings and selects the top q neighbors. As each node now has a fixed number of ordered neighbors, graph-structured data can be converted into grid-structured data.

Based on the convolution operator, many Neural Networks architectures have been developed for different tasks: Convolutional graph neural networks for node or graph classification, graph

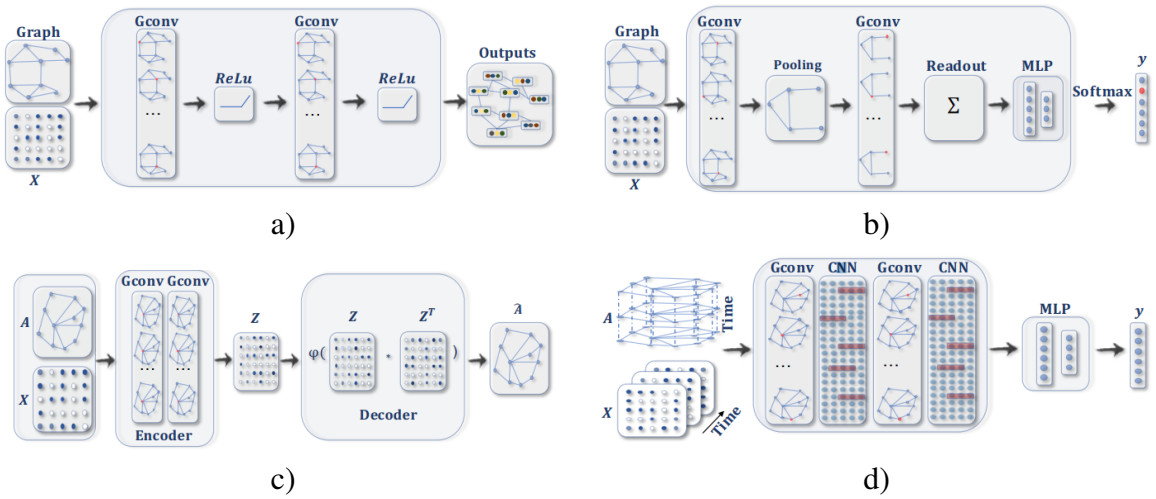


Fig 7 Figure extracted from paper of Wu et al.:[8] different models of GNN: a) a Convolutional GNN for node classification, b) a Convolutional GNN for graph classification, c) a Graph Auto Encoder and d) a Spatio-Temporan GNN.

autoencoders, spatio-temporal graph neural networks.

5 Applications and Datasets

Many application domain have exploited graph-based representation. We present here some of the main domains in which graph was successfully used. In these contexts we also present the typical benchmarks used for evaluating algorithms based on graph representation.

5.1 Chemistry

Graph theory provides a very natural representation of a 2D chemical structure, with the nodes and edges of a graph denoting the atoms and bonds of a molecule, and enables the exploitation of previously developed algorithms for the manipulation of graphs.[45] There are many properties of chemical compounds that are dependent on the structure of the components; problems related to these properties (searching for molecules that have similar properties, or searching for chemical components that have a particular action such as carcinogenicity, etc.) are therefore solved

through graph matching[46–48] or even graph embedding techniques.[4] Some of classical datasets in this domain are briefly described below.

- **AIDS**:[49] This dataset consists of two classes (active, inactive) of 2000 graphs representing molecules with activity against HIV or not.

- **Mutagenicity**:[49] This dataset is divided in two classes regarding the mutagenicity (one of the numerous adverse properties of a compound that hampers its potential to become a marketable drug) of 4337 molecules.

- **Predictive Toxicology Challenge (PTC)**:[50] This dataset deals with the predicting of the outcome of biological tests for the carcinogenicity of chemicals using information related to chemical structure only (positive or negative) on four catgories of animals : female rats (FR), male rats (MR), female mice (FM), male mice (MM) with about 240 graphs per set.

- **Monoamine oxidase dataset (MAO)**:[22] This problem is defined on a set of 68 molecules divided into two classes: the molecules that inhibit the monoamine oxidase (antidepressant drugs) and those that do not.

With respect to these datasets we would like to emphasize here one very important thing. While from the application point of view the use of these databases still makes sense to solve some problems in chemo informatics, using these benchmarks to test algorithms on graphs (graph matching, etc.) does not make much sense nowadays. It has proved that these data have known defects for benckmarking graph matching algorithms: for example the dataset Mutagenicity is known[51] to have an error, all vertices of graphs with more than 99 vertices are isolated, so it is not appropriate to evaluate graph matching problems; Solnon[52] shows also that graph size is not the only

parameter to take into consideration for evaluating graph matching problem, in fact there are still small but hard instances which cannot be solved within a reasonable amount of time by any of state-of-the-art methods.

In conclusion, our opinion is that the above described datasets are somehow out to date and it is important to evaluate solvers on other hard instances and more recent benchmarks.

5.2 Social Networks

Another application domain in which is immediate to represent the data in terms of graphs is social or web network analysis. In this context nodes of graphs can represent people, web pages, papers, etc. and edges represent interactions between people, or citation between papers, hyperlinks.

In this context there are many problems dealt with graph-based algorithms: Community detection[53] or interaction,[54] recommendation systems.[55]

A good collection of benchmarks in this context can be found on the website of Stanford Large Network Dataset Collection[56] (**SNAP**).

5.3 Image Processing

The representation of images by graphs is less immediate than in other areas. Nevertheless several image processing problems were addressed through graph-based representations. We have seen (Section 2) that irregular pyramids have been successfully used for representing images at differents resolution levels and for dealing with some classical problem: image segmentation, connected component labeling, and so on.

Thus, in this field we can mention the following databases: the **YACCLAB**,[57] a dataset for

comparing Connected Components Labeling Algorithms; the dataset for evaluating image segmentation problem proposed by **Martin et al.**[58]

In the context of image processing, several databases exist also as benchmark for graph matching problems. The **CMU house/hotel** image sequence[59] was commonly used to test the performance of graph matching algorithms. This dataset consists of 111 frames of a house, each of which has been manually labeled with 30 landmarks. The **car and motorbike image dataset** was also created in.[59] This dataset consists of 30 pairs of car images and 20 pairs of motorbike images taken from the PASCAL challenges. The **UCF shape** dataset[60] has also been widely used for comparing graph matching algorithms.

5.4 Computer Vision

Deep Learning with graph-based representation has been widely used in the field of Computer Vision. The notably graph representations are based on the extraction of features points from object of interest in videos, that will represent the node of graphs, and connecting them by edge based on some rules (e.g. the proximity of points). In this category are widely used the following datasets for skeleton-based action recognition. The **Kinetics** dataset[61] is a large-scale, high-quality dataset for human action recognition in videos. The dataset consists of around 500,000 video clips covering 600 human action classes with at least 600 video clips for each action class. The **Human3.6M**[62] dataset is one of the largest motion capture datasets, which consists of 3.6 million human poses and corresponding images captured by a high-speed motion capture system. **NTU RGB+D**[63] is a large-scale dataset for RGB-D human action recognition. It involves 56,880 samples of 60 action classes collected from 40 subjects.

But some other graph-based representation was proposed for addressing computer vision prob-

lem. Thus, the very famous image datasets **ImageNet**,[64] **Coco**,[65] **Pascal VOC**,[66] used for object detection and recognition, are also used by graph-based Deap Learning techniques.

6 To go further: open problems

As we have seen, graph-based techniques have evolved greatly in recent years, with many effective proposals. However, some open problems remain, and can be addressed in the coming years.

The first main problem, which we described earlier but which still remains unresolved, is that of **execution time**. Graph-based algorithms still spend a lot of resources, in terms of time, compared to equivalent algorithms that are based on a statistical representation of data. Many efforts have been made in this direction, but there is still room for improvement.

We also talked about a second, still open, problem which we want to call the **representation gap** here. In itself this is not a problem, but it is inherent in graph-based representation: data can be represented as graphs by many different ways. This means that algorithms cannot always be generalized to all application contexts because performance depends very much on how the data has been represented. This fact actually, rather than a problem, may also prove to be a good opportunity to propose new graph-based representations that provide surprising results in various application domains, such as image processing and computer vision.

7 Conclusions

In this paper we wanted to discuss pattern recognition techniques that make use of graphs. Deferring a full presentation to other survey papers, our proposed objective was rather to show the successes of using graphs in pattern recognition in some applications and the drawbacks yet to be overcome.

The research domain of graph-based pattern recognition is increasingly in vogue with the advent of deep learning, and there is much room to research and propose increasingly effective solutions.

We hope that this paper will fuel the desire of researchers to delve into this wonderful world of graphs.

References

1 D. Conte, P. Foggia, C. Sansone, *et al.*, "Thirty years of graph matching in pattern recognition," *International journal of pattern recognition and artificial intelligence* **18**(03), 265–298 (2004).

2 J. Yan, X.-C. Yin, W. Lin, *et al.*, "A short survey of recent advances in graph matching," in *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, 167–174 (2016).

3 H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering* **30**(9), 1616–1637 (2018).

4 D. Conte, J.-Y. Ramel, N. Sidère, *et al.*, "A comparison of explicit and implicit graph embedding methods for pattern recognition," in *International Workshop on Graph-Based Representations in Pattern Recognition*, 81–90, Springer (2013).

5 N. M. Kriege, F. D. Johansson, and C. Morris, "A survey on graph kernels," *Applied Network Science* **5**(1), 1–42 (2020).

6 L. Brun and W. Kropatsch, "Contains and inside relationships within combinatorial pyramids," *Pattern Recognition* **39**(4), 515–526 (2006).

7 D. Batavia, R. Gonzalez-Diaz, and W. G. Kropatsch, "Image= structure+ few colors," in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, 365–375, Springer (2021).

8 Z. Wu, S. Pan, F. Chen, *et al.*, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems* **32**(1), 4–24 (2020).

9 J. Zhou, G. Cui, S. Hu, *et al.*, "Graph neural networks: A review of methods and applications," *AI Open* **1**, 57–81 (2020).

10 L. Brun, P. Foggia, and M. Vento, "Trends in graph-based representations for pattern recognition," *Pattern Recognition Letters* **134**, 3–9 (2020).

11 A. H. Osman and O. M. Barukub, "Graph-based text representation and matching: A review of the state of the art and future challenges," *IEEE Access* **8**, 87562–87583 (2020).

12 F. Zhou and F. De la Torre, "Factorized graph matching," *IEEE transactions on pattern analysis and machine intelligence* **38**(9), 1774–1789 (2015).

13 M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," (2005).

14 M. Leordeanu, M. Hebert, and R. Sukthankar, "An integer projected fixed point method for graph matching and map inference," *Advances in neural information processing systems* **22** (2009).

15 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).

16 R. Raveaux, "On the unification of the graph edit distance and graph matching problems," *Pattern Recognition Letters* **145**, 240–246 (2021).

17 M. Darwiche, D. Conte, R. Raveaux, *et al.*, "A local branching heuristic for solving a graph edit distance problem," *Computers & Operations Research* **106**, 225–235 (2019).

18 M. Neuhaus and H. Bunke, "Automatic learning of cost functions for graph edit distance," *Information Sciences* **177**(1), 239–247 (2007).

19 X. Cortés, D. Conte, and H. Cardot, "Learning edit cost estimation models for graph edit distance," *Pattern Recognition Letters* **125**, 256–263 (2019).

20 M. Neuhaus and H. Bunke, "A probabilistic approach to learning costs for graph edit distance," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, **3**, 389–393, IEEE (2004).

21 A. Solé-Ribalta, F. Serratosa, and A. Sanfeliu, "On the graph edit distance cost: properties and applications," *International Journal of Pattern Recognition and Artificial Intelligence* **26**(05), 1260004 (2012).

22 B. Gaüzére, L. Brun, and D. Villemin, "Two new graphs kernels in chemoinformatics," *Pattern Recognition Letters* **33**(15), 2038 – 2047 (2012).

23 N. Sidere, P. Héroux, and J.-Y. Ramel, "Vector representation of graphs: Application to the classification of symbols and letters," in *2009 10th International Conference on Document Analysis and Recognition*, 681–685, IEEE (2009).

24 J. Jaromczyk and G. Toussaint, "Relative neighborhood graphs and their relatives," *In Proceedings of the IEEE* (1992).

25 L. Brun, M. Mokhtari, and F. Meyer, "Hierarchical watersheds within the combinatorial pyramid framework," in *International Conference on Discrete Geometry for Computer Imagery*, 34–44, Springer (2005).

26 M. Cerman, R. Gonzalez-Diaz, and W. Kropatsch, "Lbp and irregular graph pyramids," in *International Conference on Computer Analysis of Images and Patterns*, 687–699, Springer (2015).

27 M. Banaeyan and W. G. Kropatsch, "Parallel computation of the adjacency of connected components," in *International Conference on Pattern Recognition and Artificial Intelligence*, 102–113, Springer (2022).

28 S. L. Horowitz and T. Pavlidis, "Picture segmentation by a tree traversal algorithm," *Journal of the ACM (JACM)* **23**(2), 368–388 (1976).

29 A. Montanvert, P. Meer, and A. Rosenfeld, "Hierarchical image analysis using irregular tessellations," in *European Conference on Computer Vision*, 28–32, Springer (1990).

30 L. Brun and W. Kropatsch, "Introduction to combinatorial pyramids," in *Digital and image geometry*, 108–128, Springer (2001).

31 L. Brun and W. Kropatsch, "Hierarchical graph encodings," *Image processing and analysis with graphs: theory and practice* (2012).

32 W. G. Kropatsch, "Building irregular pyramids by dual-graph contraction," *IEE Proceedings-Vision, Image and Signal Processing* **142**(6), 366–374 (1995).

33 L. Brun and W. Kropatsch, "Combinatorial pyramids," in *Proceedings 2003 International Conference on Image Processing (Cat. No. 03CH37429)*, **2**, II–33, IEEE (2003).

34 Z. Zeng, A. K. Tung, J. Wang, *et al.*, "Comparing stars: On approximating graph edit distance," *Proceedings of the VLDB Endowment* **2**(1), 25–36 (2009).

35 J. Hartmanis, "Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson)," *Siam Review* **24**(1), 90 (1982).

36 K. S. Camilus and V. Govindan, "A review on graph based segmentation," *International Journal of Image, Graphics and Signal Processing* **4**(5), 1 (2012).

37 D. I. Shuman, S. K. Narang, P. Frossard, *et al.*, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine* **30**(3), 83–98 (2013).

38 T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907* (2016).

39 Y. Han, S. Wang, Y. Ren, *et al.*, "Predicting station-level short-term passenger flow in a citywide metro network using spatiotemporal graph convolutional neural networks," *ISPRS International Journal of Geo-Information* **8**(6), 243 (2019).

40 Y. Li, R. Yu, C. Shahabi, *et al.*, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," *arXiv preprint arXiv:1707.01926* (2017).

41 W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems* **30** (2017).

42 P. Veličković, G. Cucurull, A. Casanova, *et al.*, "Graph attention networks," *arXiv preprint arXiv:1710.10903* (2017).

43 F. Monti, D. Boscaini, J. Masci, *et al.*, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5115–5124 (2017).

44 M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International conference on machine learning*, 2014–2023, PMLR (2016).

45 P. Willett, "Chemoinformatics: a history," *Wiley Interdisciplinary Reviews: Computational Molecular Science* **1**(1), 46–56 (2011).

46 M. Fuchs and K. Riesen, "Matching of matching-graphs-a novel approach for graph classification," in *2020 25th International Conference on Pattern Recognition (ICPR)*, 6570–6576, IEEE (2021).

47 Z. Abu-Aisheh, B. Gaüzere, S. Bougleux, *et al.*, "Graph edit distance contest: Results and future challenges," *Pattern Recognition Letters* **100**, 96–103 (2017).

48 M. Martineau, R. Raveaux, D. Conte, *et al.*, "Graph matching as a graph convolution operator for graph neural networks," *Pattern Recognition Letters* **149**, 59–66 (2021).

49 K. Riesen and H. Bunke, "Iam graph database repository for graph based pattern recognition and machine learning," in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, 287–297, Springer (2008).

50 H. Toivonen, A. Srinivasan, R. D. King, *et al.*, "Statistical evaluation of the predictive toxicology challenge 2000–2001," *Bioinformatics* **19**(10), 1183–1193 (2003).

51 D. B. Blumenthal, "New techniques for graph edit distance computation," *arXiv preprint arXiv:1908.00265* (2019).

52 C. Solnon, "Experimental evaluation of subgraph isomorphism solvers," in *International Workshop on Graph-Based Representations in Pattern Recognition*, 1–13, Springer (2019).

53 K. S. Phyu and M. M. Min, "Graph-based community detection in social network," in *2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS)*, 12–17, IEEE (2019).

54 S. Kumar, W. L. Hamilton, J. Leskovec, *et al.*, "Community interaction and conflict on the web," in *Proceedings of the 2018 world wide web conference*, 933–943 (2018).

55 Z. Wang, Y. Tan, and M. Zhang, "Graph-based recommendation on social networks," in *2010 12th International Asia-Pacific Web Conference*, 116–122, IEEE (2010).

56 J. Leskovec and A. Krevl, "SNAP Datasets: Stanford large network dataset collection." http://snap.stanford.edu/data (2014).

57 C. Grana, F. Bolelli, L. Baraldi, *et al.*, "Yacclab-yet another connected components labeling benchmark," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, 3109–3114, IEEE (2016).

58 D. Martin, C. Fowlkes, D. Tal, *et al.*, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, **2**, 416–423, IEEE (2001).

59 T. S. Caetano, J. J. McAuley, L. Cheng, *et al.*, "Learning graph matching," *IEEE transactions on pattern analysis and machine intelligence* **31**(6), 1048–1058 (2009).

60 H. Chui and A. Rangarajan, "A new point matching algorithm for non-rigid registration," *Computer Vision and Image Understanding* **89**(2-3), 114–141 (2003).

61 J. Carreira, E. Noland, C. Hillier, *et al.*, "A short note on the kinetics-700 human action dataset," *arXiv preprint arXiv:1907.06987* (2019).

62 C. Ionescu, D. Papava, V. Olaru, *et al.*, "Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **36**, 1325–1339 (2014).

63 A. Shahroudy, J. Liu, T.-T. Ng, *et al.*, "Ntu rgb+d: A large scale dataset for 3d human activity analysis," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1010–1019 (2016).

64 J. Deng, W. Dong, R. Socher, *et al.*, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, 248–255, Ieee (2009).

65 T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft coco: Common objects in context," in *European conference on computer vision*, 740–755, Springer (2014).

66 M. Everingham, S. M. A. Eslami, L. Van Gool, *et al.*, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision* **111**, 98–136 (2015).

Donatello Conte received his Ph.D. degree in 2006 by a joint supervision between LIRIS laboratory of the INSA of Lyon (France) and MIVIA laboratory of the University of Salerno (Italy). He is Associate Professor at the Computer Science Laboratory of the University of Tours. He is currently head of the Computer Science Department at Polytech Tours School of Engineering. His main research fields are: structural pattern recognition (graph matching, graph kernels, combinatorial maps), video analysis (objects detection and tracking, trajectories analysis, behavioral analysis, etc.), and affective computing (emotion recognition, multimodality analysis for affective analysis, physiological measures by video analysis, etc.). He is the author of more than 70 publications and reviewers in the main journals in his research field (PAMI, PR, CVIU, TIP, etc.). He is member of the Editorial Board of the Elsevier Journal Internet of Things, MDPI Journal of Imaging and he is Guest Editor for the Pattern Recognition Letters journal and IEEE Transactions on Emerging Topics in Computing journal. He has been co-chair of the International Workshop on Graph-basd Representation in Pattern Recognition (GbR2019) and he is the chairman of the International IAPR

Technical Committee 15 dedicated to the promotion of graphs in the Pattern Recognition.

List of Figures

List of Tables