Graph Neural Network for Symbol Detection on Document Images

Guillaume Renton, Pierre Héroux, Sébastien Adam, Benoit Gaüzère

Graph Neural Network for Symbol Detection on Document Images

Guillaume Renton, Pierre Héroux, Sébastien Adam
UNIVROUEN
LITIS
Rouen, France
guillaume.renton@univ-rouen.fr

Benoit Gaüzère
INSA de Rouen Normandie
LITIS
Rouen, France

***Abstract*—In this paper, we propose a new method to simultaneously detect and classify symbols in floorplan images. This method relies on the very recent developments of Graph Neural Networks (GNN). In the proposed approach, floorplan images are first converted into Region Adjacency Graphs (RAGs). Within those graphs, each node corresponds to a white region in the original image, and each edge indicates an adjacency relationship between two regions encoded by incident nodes. Nodes are attributed using Zernike moments, and edges are characterised using the distance between centers of gravity of connected components. Then, graphs are fed to a dedicated neural network which has been learned to classify the nodes of unknown graphs using both node attributes and topology. The method is evaluated on the ILPIso dataset and obtains very promising results. These results show the interest of using graphs for such a task, especially when input data are noisy.**

***Keywords*-Graph Neural Network, Graphs, Floorplans.**

I. Introduction

In pattern recognition community, most of existing approaches embed images to well-suited euclidean spaces to profit from efficient pattern recognition methods existing. These methods use the numerous mathematical properties associated to euclidean spaces, which allow to solve complex and challenging problems. However, despite euclidean spaces can encode numerical data, they fail to encode all the available information when the data include some structural information such as molecular compounds, networks, complex patterns, etc.

A nice and efficient way to encode such structural information is to use graphs. A graph G is defined as a pair $G = (V, E)$. The set of nodes $v_i \in V$ defines a set of elements which are connected or not through the set of edges $E \in V \times V$. Two elements are said to be connected if there exists an edge $(v_i, v_j) \in E$. This set of connections is used to define the neighbourhood N_{v_i} of each node v_i. This neighbourhood corresponds to all nodes connected to v_i, ie. $v_j \in N_{v_i} \Leftrightarrow (v_i, v_j) \in E$. Note that we talk here about undirected graphs, where edges have no direction. This leads to $(v_i, v_j) \in E \Leftrightarrow (v_j, v_i) \in E$. Nodes and edges can carry there own information. This information is carried out by a node labelling function $\mu : V \rightarrow L_v$ which associates each node of the graph to a particular label of the set L_v. Similarly, an edge labelling function $\xi : E \rightarrow L_e$ associates a label to each edge of a graph. Using such structure, we can now encode topological relationships between elements of a set. Graphs have been widely used to encode structured information such as networks, RAG, molecules and shapes.

Floorplans, as the ones included in the dataset ILPIso [10], [8], are originally encoded as images. These images are directly extracted from data produced by architects. However, since they represent a building floor, the underlying information includes some important structural relationships between the elements of the floorplan. Each building floor is composed of rooms and furniture elements which are interconnected by their topological relationships: A table is inside a kitchen, a bed is inside a bedroom, and bedroom and kitchen are connected together by a door. This structural information is not explicitly encoded within the original image, but it requires to be extracted from the former. Then, a graph representation of this floorplan may allow to encode this structural information.

A particular strength of euclidean based pattern recognition methods consists in using machine learning framework to perform the recognition and prediction task. However, most of machine learning methods are defined on euclidean spaces, where they benefit from well defined optimisation problems and efficient mathematical tools. Using such methods on graph data is not straightforward since graph space is more general than euclidean spaces, and thus more complex to manipulate. Using graph based methods, such as graph edit distance, we often face a complexity problem due to possible permutations of node order and different number of nodes. Considering a very basic problem such as determining if two graphs are an unique and same one is a NP-Hard problem, which forbids its exact solving in most of real world applications. To overcome this drawback, many methods are devoted to compute descriptors from graphs, and then embed graphs into an euclidean space to benefit from all machine learning methods. However, selecting and computing these descriptors reduce the versatility of graph representation and the encoded structural information. The set of descriptors is generally defined using an a priori expert choice, which doesn't allow to control the loss of information.

An alternative strategy is neural network based methods.

In the last decade, these methods helped the community to break a performance bottleneck by determining and computing optimal features thanks to deep architectures and thousands of examples. These particularities bring deep neural networks to the state of the art on most of pattern recognition and computer vision datasets. Nonetheless, one strong limitation of deep neural networks is that they are restricted, in their native definition, to euclidean data encoded as vectors, these vectors having a fixed and predefined size. To benefit from the expressive power of graphs, one needs to connect graph representations and deep neural networks. This challenge requires to define new architectures, but it may allow to automatically compute optimal features according to a particular task.

The recent emergence of different Graph Neural Networks [7], [12], [11], [6], [3], [17], [16] is dedicated to bridge the gap between neural networks and graphs. Through different approaches to tackle dimension and permutation problems, these methods have shown good prediction performances, hence proving they are able to exploit the structural information included within graphs.

In this paper, we propose to exploit graph neural networks methods to address the task proposed by ILPIso [8], [10] dataset. This task consists in identifying the class and the location of different furniture symbols which appear regularly within floorplan images. Transforming images to graphs will allow our prediction model to use the topological information around each furniture symbol, and thus to include some contextual information during learning process. The use of graph neural network approaches avoids the definition of ad-hoc features, and thus the hard choice of a predefined set of features. Combining these two approaches may lead to robust and accurate prediction of symbol classes.

This paper is structured as follows. First, we review in Section II the different neural network architectures proposed to process graphs within a classification scheme. Second, Section III-A details how we pass from floorplan images to graphs. Then, Section III-B presents our proper graph neural network model, adapted to the particular classification task associated to ILPIso dataset. Finally, our contribution is tested in section IV through some experiments on ILPIso dataset to evaluate its accuracy and robustness against some noise.

II. Related works

A. Original graph neural network

Graph Neural Networks (GNN) have firstly been theorised by Gori et al. in [7] and then extended in [12]. In those papers, authors want to update state of a node by aggregating information contained in its labelling, its neighbourhoods labelling and the labelling of the edges linking the node to its neighbourhood. This defines the very idea of graph neural network. More specifically, each node in the graph is updated according to equation 1. In this equation and for the rest of

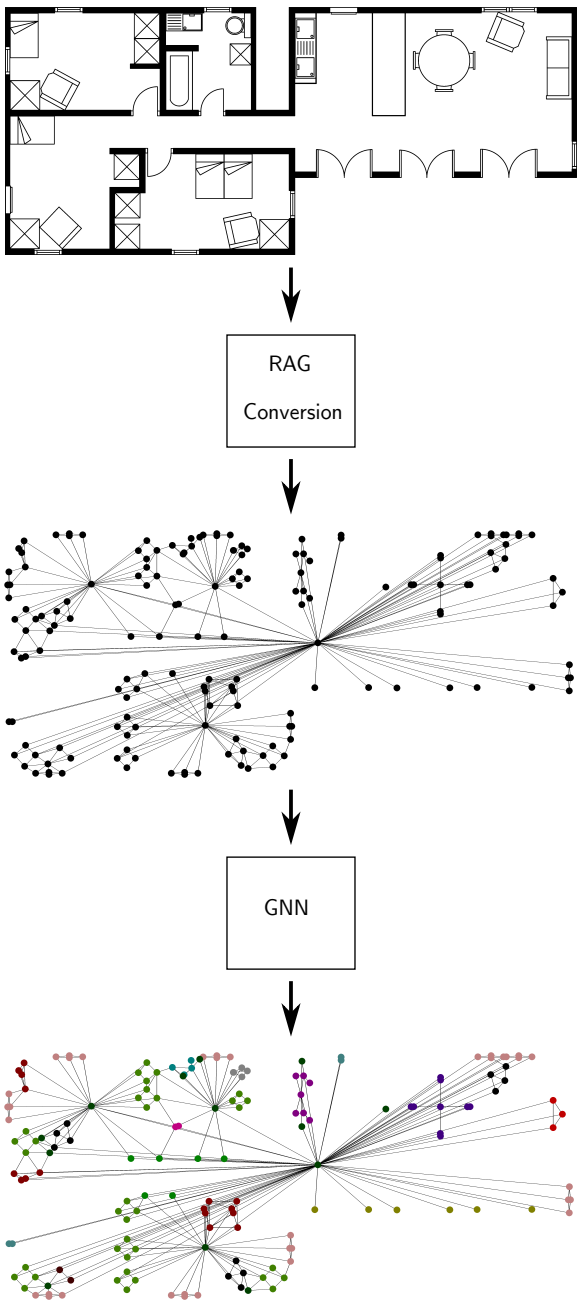


Figure 1: Summary of our method. First, a floorplan image is converted into a Region Adjacency Graph, where each node corresponds to a connected component. Then, a graph neural network is used to predict the label of each node.

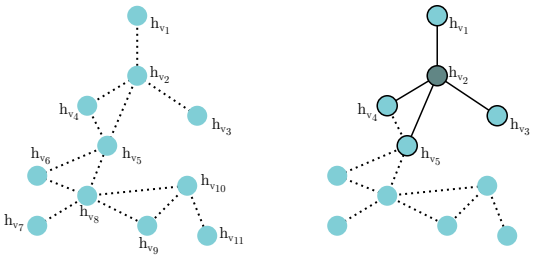


Figure 2: Example of how a node is updated depending on its neighbourhood. On the left is the original graph. On the right, the node h_{v_2} is updated depending on its neighbourhood $h_{v_1}, h_{v_3}, h_{v_4}, h_{v_5}$, and the edges between h_{v_2} and its neighbour (in bold).

this paper, the following notation are defined : $\boldsymbol{h}_v$ the hidden state of the node v, l_v is still the attributes of node v, e_{vw} the edge between node v and node w, $N(v)$ is also still the neighbourhood of v, and thus $l_{N(v)}$ and $e_{N(v)}$ respectively the nodes and edges attributes of v neighbourhood. Finally, f defines an arbitrary function. This function is iterated T times, and thus, each $\boldsymbol{h}_v^{t+1}$ is updated depending on $\boldsymbol{h}_v^t$.

$$\boldsymbol{h}_v^{t+1} = f(l_v, l_{N(v)}, \boldsymbol{h}_v^t, e_{N(v)}) \tag{1}$$

Finally, a decision $\boldsymbol{o}_v$ is taken for each node v following equation 2:

$$\boldsymbol{o}_v = g(\boldsymbol{h}_v^t, l_v) \tag{2}$$

where g_w is an arbitrary function which take into account the actual node state $\boldsymbol{h}_v^t$ and the original node label l_v. Figure 2 shows an example of how a node is updated depending on its neighbourhood.

Finally, both functions f and g are neural network.

B. General graph neural network model

More recently, [6] proposed a more general approach, where a message m_v^{t+1} is computed following equation 3 where M_t is an arbitrary function.

$$m_v^{t+1} = \sum_{w \in N_{(v)}} M_t(h_v, h_w^t, e_{vw}) \tag{3}$$

Finally, each node is updated depending on an arbitrary function U_t, which take as input the actual node state and the computed message, as expressed in equation 4.

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1}) \tag{4}$$

As shown in [6], this model generalises multiple models, such as [5], [11], [2], [9], [13], and both functions M_t and U_t are generally neural networks.

III. Model

A. From images to graphs

Region Adjacency Graphs are well suited for representing symbols and technical drawings since they enable to model the adjacency relations between the regions extracted by a segmentation process. Working on technical documents, the digital images are mainly binary images where white components denote the background while black components stand for the drawings. Segmenting such kind of images can be achieved using component labelling [4]. However, aiming at finely modelling adjacency relations between two regions, a binary image can be firstly thinned [1]. The obtained image is then morphologically the same than the initial image of the document but the thickness of the drawing components is reduced to a single pixel. Using this image, each white component is mapped to a vertex of the graph. Then, the skeleton branches represent frontiers and adjacency relations between two regions. An edge is then build between two vertices representing regions separated by a skeleton branch. Figure 3 illustrates the overall process on an extract of a document image.

To enrich such a description, attributes have to be assigned to each vertex and edge. Many features have been proposed to characterize shapes and spatial relations [15]. Among them, Zernike Moments (ZM) [14] yield interesting results for pattern recognition tasks when invariance to affine transforms and robustness to degradations are required. Hence, a feature vector corresponding to a set of ZM is assigned to each vertex in order to characterize shapes. Concerning the edges, a single attribute is used : the *relative distance* between gravity centers of adjacent regions which is computed with respect to the overall area of the two regions.

$$\frac{d_e(g_{source}, g_{target})}{\sqrt{A(source) + A(target)}}$$

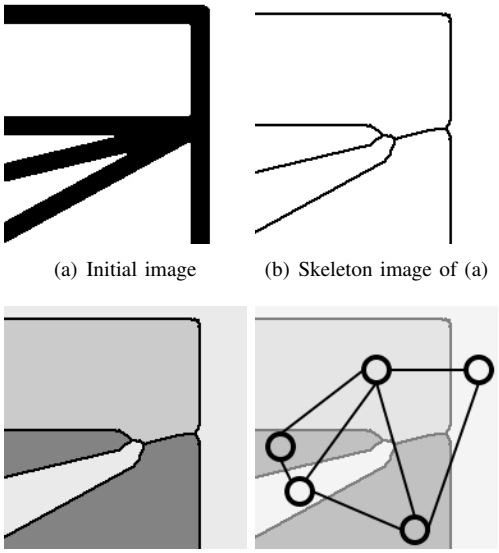
where d_e denotes the Euclidean distances between gravity centers.

We finally get a graph-based representation $\mathcal{G} = (V, E, \mu, \xi)$ of a document where V stands for the set of vertices (regions) and $E \subseteq V \times V$ stands for the edges (adjacency relations). μ and ξ are mapping functions :

- $\mu : V \to \mathbb{R}^{24}$ describing the morphology of a region (a vertex),
- $\xi : E \to \mathbb{R}$ expressing the geometrical properties of an adjacency relation (an edge).

B. Graph Neural Network Model

As stated in introduction, our method relies on the recent graph neural network developments which have been introduced in the literature. In section 2, we presented a general model of a GNN. In this subsection, we focus our definition on the particular GNN we use.



(a) Initial image (b) Skeleton image of (a)

(c) Components labels of (b) (d) Corresponding RAG

Figure 3: Raster to Region Adjacency Graph.

One of the simplest way to compute a GNN is probably the one defined in equation 5, where a node and its neighbourhood are updated independently. In this equation, W is a neural network.

$$h_v^{t+1} = m_v^{t+1} = (h_v^t + \sum_{w \in N_{(v)}} h_w^t) \cdot W \tag{5}$$

More particularly, this GNN is interesting by the way it is computed when working with tensors (equation 6). One can notice the similarity with neural networks.

$$H^{t+1} = (A + I) \cdot H^t \cdot W \tag{6}$$

One problem with this simple way comes with the fact that it makes no difference between the central node and its neighbourhood. For example, in Figure 4, the simplest GNN will update the node v similarly in both cases, while intuitively, one would probably update them in a different way.

To tackle this limitation, one solution could be to use two different neural networks, one for the central node, and another one for the neighbourhood. This is defined by equation 7.

$$h_v^{t+1} = h_v^t \cdot W_0 + (\sum_{w \in N_{(v)}} h_w^t) \cdot W_1 \tag{7}$$

This last equation defines our general model. However, it does not use the possible edges attributes. For this case, we use the edge network proposed in [6], which is defined by equation 8, where W_A is a neural network applied to the edges attributes which computes a matrix.

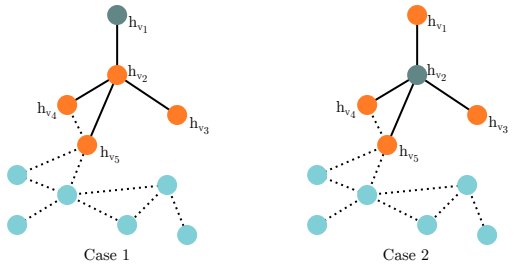


Figure 4: During the hidden state update of vertex h_v, the simplest GNN that considers the central node the same way as neighbouring node can not distinguish Case 1 and Case 2 since this update is based on 3 orange nodes and 1 green node in both cases. These two cases can be distinguished when the central node is considered differently from the neighbouring nodes

$$h_v^{t+1} = h_v^t \cdot W_0 + (\sum_{w \in N_{(v)}} (h_w^t \cdot e_{vw} \cdot W_A)) \cdot W_1 \tag{8}$$

In our experiments, models defined by equation 7 and 8 are both tested.

IV. Experiments

A. Dataset presentation

To evaluate our model, we experiment it on the ILPIso dataset, a dataset made of 200 floorplans like the plan of Figure1. We apply our graph conversion strategy to each of the floorplan images, and thus get 200 graphs. After this conversion, we count 24281 nodes and 105110 edges, which give us a mean of 121 nodes and 525 edges per graph. Each node is labelled with the 24 first Zernike moments of the connected component he represents. Each edge is labelled with the distance between the center of gravity of the two nodes the edge is linking. Finally, each node is associated to one of the 17 classes corresponding to the different objects in the original floorplan images. The 17 classes are the 16 kinds of symbols to be found in floorplans (Figure 5), plus a dummy class, for the case where the node do not correspond to any of the 16 symbols.

B. Evaluated models

We evaluated 3 different models: the first model, which corresponds to our baseline, is a neural network which takes as input and classify each node independently and only using its original features. The neural network is made of two dense layers, with the second one used for decision. Thus, with this instance, no topological relationships between any node is taken into account (see Figure 6).

The second model is a graph neural network (GNN), made of two layers. Similarly to the baseline, the second layer is used for decision. With this model, we are only able to use

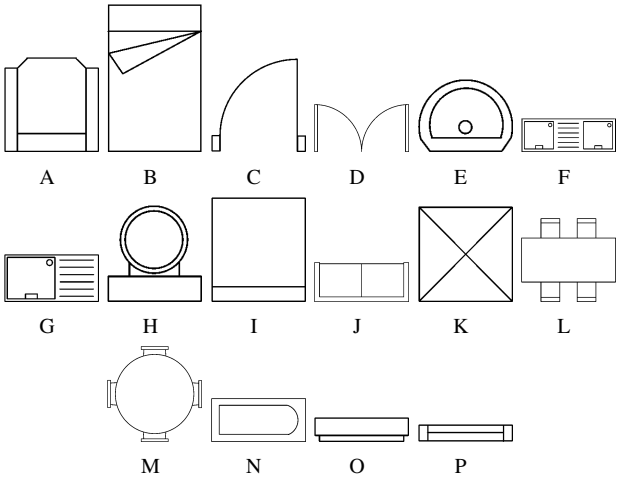


Figure 5: Symbol models

Model	Result
Base	95.2 ± 1.4
GNN	99.8 ± 0
EN	100.0 ± 0

Table I: Results obtained (in percentage of accuracy of nodes classification) by the different experimented models.

the existence of an edge between a pair of nodes, but not the labels associated to this edge (see Figure 6).

The third experimented model uses the edge network (EN) defined in section III-B. This model uses both the existence of an edge and its labels (see Figure 6).

To be able to compare the three models, we make them as similar as possible. Thus, both the neural networks of the baseline and the two GNN produce 200 features for each node.

C. Experimental protocol and results

In this section, we describe the protocol used to conduct our experiments. First, our 200 graphs are randomly divided into 3 sets. 160 graphs are used in training, 20 in validation and 20 in test. Each node's and edge's attribute is normalized to have a value either between 0 and 1 or -1 and 1, depending on the minimum value of the attribute. Each model is then trained using the Adam optimizer with an initial learning rate of 5.10^{-3}. Each model is trained for 100 epochs using the crossentropy loss and the best model in validation is used for test. We run this experiment 10 times per model and compute the mean and standard deviation for the node classification of those 10 experiments. We computed the results obtained in table I.

As one can see, results for both GNN and EN are pretty similar and close to 100%. The difference between the two results comes from one run which failed on GNN, while the 9 others obtained perfect scores, as for EN.

Gaussian noise of variance 0.05	
Base	68.3 ± 2.44
GNN	99.7 ± 0
EN	97.4 ± 1
Gaussian noise of variance 0.10	
Base	46.4 ± 3.7
GNN	98.9 ± 0
EN	94.8 ± 2.2
Gaussian noise of variance 0.20	
Base	27.2 ± 2.44
GNN	81.3 ± 3.8
EN	89.1 ± 2.2

Table II: Results obtained with noised test data.

To be able to compare the methods more rigorously, and evaluate their ability to generalize, we decided to add some noise on the tested data. We added a Gaussian noise centered in 0 with different variances. We tried 3 different values for this variance : 0.05, 0.1 and 0.2. Results are given in table II. First thing we notice is that results for the baseline collapse quickly. This shows how useful the structural information is in decision. Second is that GNN behave better than EN with low variance (0.05 and 0.1), but are dropping quickly when the variance becomes higher (0.2), while EN results are dropping in a more regular way. This can be explained by the fact that at first, the existence or not of edges are not influenced by the noise, but when the noise on nodes is too important, the presence of attributes on edges, even if they are noised, helps the network for the decision.

V. Conclusion

In this paper, a new approach for floorplans segmentation is proposed. The approach firstly transform the floorplans images into Region Adjacency Graphs. Graph Neural Network are then applied on those graphs to take a decision on each node.

Two different models of graph neural networks are studied. Both of them presents great results on the evaluated dataset, and improve results obtained without structural information. Especially, we show that this structural information is really important with noised data. Future works will explore more flexible ways to convert images to graphs and try different GNN models on new data. Another outlook would be to find subgraph symbols inside the floorplans, instead of classifying nodes.

References

[1] G. Sanniti Di Baja and E. Thiel. Skeltonization algorithm running on path-based distance maps. *Image and Vision Computing*, 14:47–57, 1996.

[2] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pages 4502–4510, 2016.

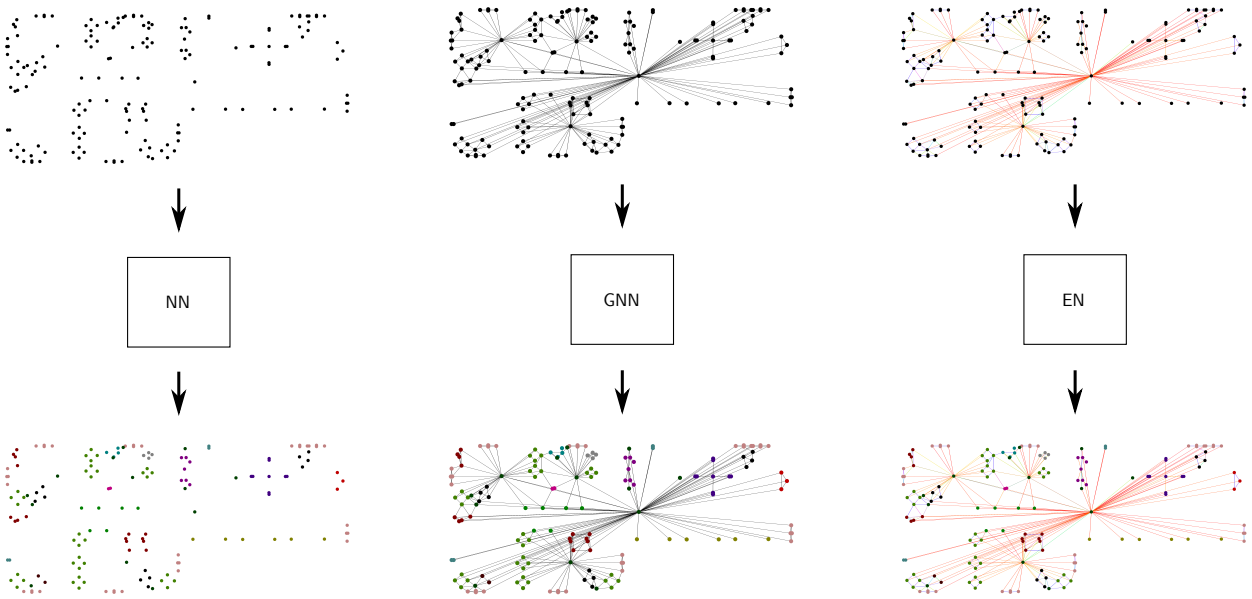


Figure 6: Examples of the 3 different models experimented. First (left) model do not take into account any edges, second model (middle) only takes into account the existence of an edge and third one (right) use edge's attributes.

[3] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[4] C.-J. C. Fu Chang and C.-J. Lu. A linear-time component-labeling algorithm using contour tracing technique. *Computer Vision and Image Understanding*, 93:206–220, 2004.

[5] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems 28*, pages 2224–2232, 2015.

[6] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

[7] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 2, pages 729–734. IEEE, 2005.

[8] Pierre Héroux, Pierre Le Bodic, and Sébastien Adam. Datasets for the Evaluation of Substitution-Tolerant Subgraph Isomorphism. In *IAPR International Workshop on Graphics Recognition*, pages 240 – 251, Bethlehem, United States, August 2013.

[9] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

[10] Pierre Le Bodic, Pierre Héroux, Sébastien Adam, and Yves Lecourtier. An integer linear program for substitution-tolerant subgraph isomorphism and its use for symbol spotting in technical drawings. *Pattern Recognition*, 45(12):4214–4224, 2012.

[11] Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. Gated graph sequence neural networks. In *Proceedings of ICLR'16*, 2016.

[12] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *Trans. Neur. Netw.*, 20(1):61–80, January 2009.

[13] Kristof T Schütt, Farhad Arbabzadah, Stefan Chmiela, Klaus R Müller, and Alexandre Tkatchenko. Quantum-chemical insights from deep tensor neural networks. *Nature communications*, 8:13890, 2017.

[14] M. Teague. Image analysis via the general theory of moments. *Journal of the Optical Society of America*, 70(8):920–930, 1980.

[15] O. R. Terrades, S. Tabbone, and E. Valveny. A review of shape descriptors for document analysis. In *Proceedings of the ninth International Conference on Document Analysis and Recognition*, pages 227–231, 2007.

[16] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

[17] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.