Symbols Detection and Classification using Graph Neural Networks

Guillaume Renton, Muhammet Balcilar, Pierre Héroux, Benoit Gaüzère, Paul Honeine, Sébastien Adam

Symbols Detection and Classification using Graph Neural Networks

Guillaume Renton[a,*], Muhammet Balcilar[a], Pierre Héroux[a], Benoit Gaüzère[a], Paul Honeine[a], Sébastien Adam[a]

[a]*LITIS Lab, Université de Rouen Normandie, 76100 Rouen, France*

Abstract

In this paper, we propose a method to both extract 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). In order to achieve both classification and extraction, two different GNNs are used. The first one aims at classifying each node of the graph while the second targets the extraction of clusters corresponding to symbols. In both cases, the model is able to take into account edge features. Each model is firstly evaluated independently before combining both tasks simultaneously, increasing the quickness of the results.

Keywords: Graph Neural Network, Graphs, Floorplans

1. Introduction

For a long time, the pattern recognition community has focused its contributions on data encoded in Euclidean space, taking benefit of the numerous mathematical properties of grid-based representations in order to solve complex and challenging learning problems. However, Euclidean spaces are not optimal to encode certain kinds of information when data contains structural information, such as molecular compounds, networks, and complex patterns.

Floorplans produced by architects are an example of data that implicitly integrate an underlying structural information. Each component of the plan can be connected to other ones. For example, a table is inside a kitchen or a living room, a bed lies in a bedroom, and living room and kitchen may be connected by a door, etc. Although we can see this information in a floorplan image, it is not explicitly encoded inside. However, many approaches that aim to analyse or understand floorplans or technical drawings try to reconstruct and exploit this high level structural information as it was initially encoded by CAD softwares [19, 21, 26].

*Corresponding author:
Email address: guillaume.renton@univ-rouen.fr (Guillaume Renton)

Graphs are mathematical objects that are efficient to encode such structural information. A graph G can be defined as a pair $G = (V, E)$, with V defining a set of elements called nodes. Two nodes v_i and v_j are said to be connected by an edge if there exists a corresponding edge (v_i, v_j) in the set of edges $E \subset V \times V$. Graphs can be directed or undirected. An undirected graph is a graph where the order of nodes in the pair forming the edge has no importance, meaning $(v_i, v_j) = (v_j, v_i)$. The neighborhood $N(v_i)$ of a node v_i is the set of nodes connected to v_i by an edge. Nodes and edges can be attributed, meaning they can also carry information individually. If $\mu\colon V \to L_v$ is a function representing the information carried by nodes and $\xi\colon E \to L_e$ is a function representing the information carried by edges, a graph can be redefined as a tuple $G = (V, E, \mu, \xi)$. This data structure allows to encode topological relationships between individual elements as well as features of both elements and binary relationships, such as atom properties and covalency in molecule for example.

Using such representations, graph-based methods such as subgraph isomorphisms or graph edit distance [8, 27, 30, 31, 20] have to be used to take decisions. Error-tolerant subgraph isomorphism methods exploiting graph representations have been applied for symbol symbol detection on floorplans [19, 21]. Unfortunately, those methods often induce a complexity problem. Hence, even fundamental problems such as determining if two graphs are similar are NP-Hard, which limits their use in real-world problems. To overcome this drawback, many methods have been proposed to compute descriptors from graphs, and then embed graphs into a Euclidean space in order to benefit from state-of-the-art machine learning methods [36, 14, 6]. However, selecting and computing these descriptors results in a loss of information with respect to the original graph representation.

Nowadays, most efficient machine learning methods are based on neural networks. These methods have broken many performance records by using deep architectures to learn meaningful features from thousands of examples. Thus, deep neural networks manage to be the state of the art on most of pattern recognition and computer vision tasks. However, as for most of the machine learning frameworks, deep neural networks have been for a long time restricted to Euclidean data such as vectors, matrices or tensors (e.g. images). The connection between graph representation and deep learning has recently became a challenge. This recent emergence has led to many Graph Neural Networks (GNN) that bridge the gap between neural networks and graphs [13, 28, 22, 12, 5, 37, 35]. Those methods tackle the problem of dimension and permutation invariance or equivariance through different ways, and are able to exploit the structural information included within graphs.

In this paper, we propose a GNN based approach to address the task of identifying the class and the location of different furniture symbols that appear frequently within floorplans. The proposed approach firstly transforms images into graphs where nodes represent parts of images and edges represent binary relationships between these parts. This allows our prediction model to use the topological information around each furniture symbol, and thus to include some contextual information during learning process.

In the symbol detection problem, there may be several occurrences of the same symbol in one image. Moreover, different classes of symbols can be requested. We then decide to tackle this problem with a two-step strategy. The first step is used to classify nodes in the graph, according to the different classes of symbols. In a previous

paper [25], we already considered this task by using a state of the art approach . Here, we extend this first contribution by proposing a new model. The second step is used to group nodes into clusters with the aim that clusters are made of nodes representing the different parts of a symbol occurrence. As a consequence, this work can be seen as a first attempt to solve a multi-subgraph isomorphism problem using a GNN.

This paper is structured as follows. First, we review in Section 2 different neural network architectures proposed to process graphs within a classification scheme. Second, Section 3 details our contributions. It first describes how we build graph representations from floorplan images. Then, it presents a new GNN method able to take into account edge features and its application for both the classification and the clustering task. Section 4 details our experiments on ILPIso dataset. Results show that the proposed model performs similarly to state-of-the-art method using edge features, while being more robust to noise.

2. Related works

Historically, Graph Neural Networks (GNN) have been firstly theorised by [13] and extended by [28]. In those papers, each node has a state, and this state is updated iteratively by aggregating information contained in the node own labelling, its neighborhood labelling and the labelling of the edges linking the node to its neighborhood according to

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

where $\boldsymbol{h}_v$ is the hidden state of the node v, l_v encodes the labelling of node v, e_{vw} is the edge between node v and node w, l_{vw} represents the attributes of edge vw, $N(v)$ corresponds to the neighborhood of v, and thus $l_{N(v)}$ and $e_{N(v)}$ encode respectively the nodes and edges attributes of v's neighborhood. In this equation, f defines an arbitrary function. Finally, a decision $\boldsymbol{o}_v$ is taken for each node v as follows:

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

where g is an arbitrary function of the final node state $\boldsymbol{h}_v^t$ and the original node label l_v. Figure 1 shows an example of how a node is updated depending on its neighborhood. In GNN, both functions f and g are generally defined as neural networks.

Recently, [12] proposed a general framework called Message Passing Neural Network (MPNN) where a message m_v^{t+1} is computed through the following equation:

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

with M_t being an arbitrary function and e_{vw} the edge features between node v and node w. In MPNNs, the hidden state of each node is updated depending on an arbitrary function U_t, which takes as input the actual node state and the computed message:

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

It is shown in [12] that this model generalises multiple models, such as [10, 22, 4, 17, 29]

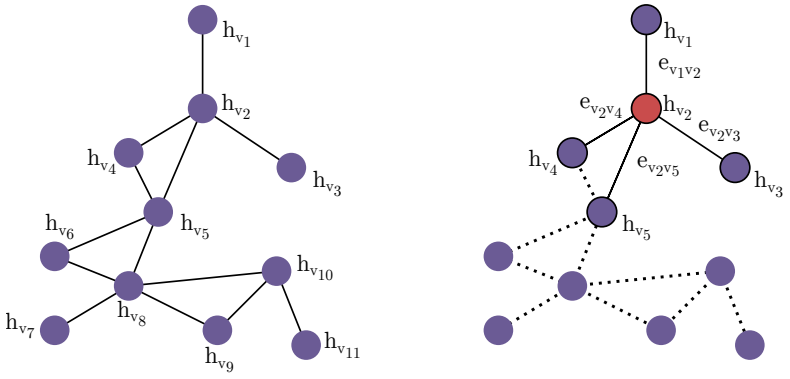


Figure 1: Example of how the hidden state of a node is updated depending on its neighborhood. On the left is the original graph. On the right, the hidden state of node v_2 is updated depending on its neighborhood v_1, v_3, v_4, v_5, and the edges between v_2 and its neighbor (in bold).

Even more recently, [3] shows that most of existing GNNs, including spectral ones, are MPNNs. It thus proposes a general model which also includes these spectral GNNs (see Table 1 in [3]). The proposed general form is given by

$$H^{(t+1)} = \sigma\Big(\sum_s C^{(s)} H^{(t)} W^{(t,s)}\Big), \tag{5}$$

where $C^{(s)} \in \mathbb{R}^{n\times n}$ is the s-th convolution support that defines how the node features are propagated to the neighboring nodes and $W^{(l,s)} \in \mathbb{R}^{f_l \times f_{l+1}}$ is the trainable matrix for the l-th layer and s-th support. With this notation, t can be seen as a layer index instead of an epoch number in [12].

Using such a model, the main question is the definition of the convolution kernels $C^{(s)}$. A natural method, named Vanilla ConvGNN, uses only one kernel defined by $C = A + I$ where A denotes the adjacency matrix of the graph and I the identity matrix. [10] used two convolution kernels defined by $C^{(1)} = A$ and $C^{(2)} = I$ to split the weights applied to own node features and the weights applied to neighborhood features. They also proposed to create $C^{(s)}$ convolution kernels by defining subset of adjacency matrix according to node types if it is given. [24] proposed to find cardinal reordering of each node neighborhood and create $C^{(s)}$ which shows the connection between the concerned node and its s-th order neighbor.

[9] proposed to use Chebyshev kernels in ConvGNN where it was shown that all spectral filters on graph can be written by weighted sum of these kernels in [16]. According to [9], the first two Chebyshev kernels are $C^{(1)} = I$ and $C^{(2)} = 2L/\lambda_{\max} - I$ and the remaining kernels are defined by

$$C^{(k)} = 2C^{(2)}C^{(k-1)} - C^{(k-2)}, \tag{6}$$

where L is the graph Laplacian and $\lambda_{\max}$ is the maximum eigenvalue of the graph Laplacian. One major simplification of the previous Chebyshev kernels, named Graph Convolution Network (GCN) is proposed in [18]. Their single convolution matrix is

defined by:

$$C = \widetilde{D}^{-1/2}\widetilde{A}\widetilde{D}^{-1/2}, \tag{7}$$

where $\widetilde{D}_{i,i} = \sum_j \widetilde{A}_{i,i}$ and $\widetilde{A} = (A+I)$ is the adjacency matrix with added self-connections.

So far, those methods use fixed convolution kernels and cannot use edge features. There are also some other methods that use trainable convolution kernels in order to make the convolutions more productive such as graph attention networks [34, 15]. The attention mechanism tunes each element of the convolution kernel by a function of connected nodes and/or edge features and some trainable parameters $W_{AT}^{(s)}$ by

$$C_{i,j}^{(s)} = f\left(h_i^t, h_j^t, e_{ij}, W_{AT}^{(s)}\right). \tag{8}$$

Graph Attention Network (GAT) creates convolution kernels by the concatenation of feature vectors of connected nodes [34].

3. Proposed approach

3.1. From images to graphs

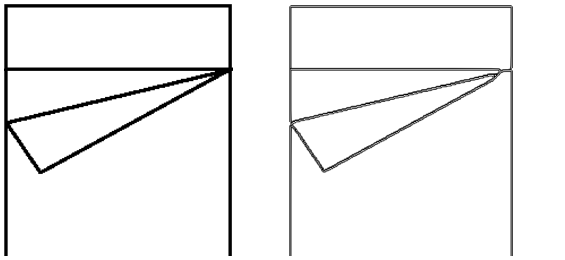
Region Adjacency Graphs (RAGs) are well suited to encode symbols and technical drawings since they allow to model the adjacency relationships between the regions extracted by a segmentation process. Working on technical documents, the digital images are mainly binary images where white components correspond to the background and black components to drawings. Segmenting such kind of images can be achieved using component labelling [7]. However, aiming at finely modelling adjacency relationship between two regions, a binary image can be firstly thinned [2]. 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 node of the graph. Then, the skeleton branches represent frontiers and adjacency relationships between two regions. An edge is then built between two nodes representing regions separated by a skeleton branch.

To enrich such a description, attributes have to be assigned to each node and edge. Many features have been proposed to characterize shapes and spatial relations [33]. Among them, Zernike Moments (ZM) [32] 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. Attribute of an edge connecting two adjacent regions (source and target) is defined by the *relative distance* between their gravity centers, computed with respect to the overall area of the two regions:

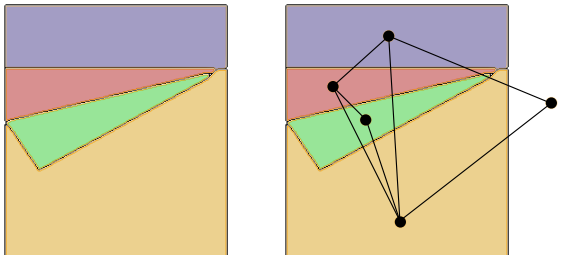
$$e_{source,target} = \frac{d_e(g_{source}, g_{target})}{\sqrt{\text{Area}(source) + \text{Area}(target)}}, \tag{9}$$

where d_e denotes the Euclidean distance between gravity centers.

We finally define a graph-based representation $G = (V, E, \mu, \xi)$ of a document where V encodes the white connected components (regions) and E corresponds to adjacency relationships between regions. Labelling function on nodes $\mu\colon V \to \mathbb{R}^{24}$ encodes the morphology of each region according to its ZM and labelling function $\xi\colon E \to \mathbb{R}$ expresses the geometrical properties of an adjacency relationship, as defined in Equation 9.



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



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

Figure 2: From initial image to Region Adjacency Graph.

3.2. Node classification

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

As said before, one of the simplest GNN expressed with a tensor notation is given by:

$$H^{t+1} = \sigma((A + I) \cdot H^t \cdot W), \tag{10}$$

where a node and its neighborhood are updated independently with the W corresponding to a neural network.

With Equation 10 no difference is made between the central node and its neighborhood. For example, in Figure 3, 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 is to use two different neural networks, one for the central node, and another one for the neighborhood. This can be obtained by modifying Equation 10 to get

$$H^{t+1} = \sigma(I \cdot H^t \cdot W^0 + A \cdot H^t \cdot W^1). \tag{11}$$

This last equation defines our general model. However, it does not exploit edge attributes. For this case, one common way is to use the Edge Network (EN) model proposed by [12], defined as:

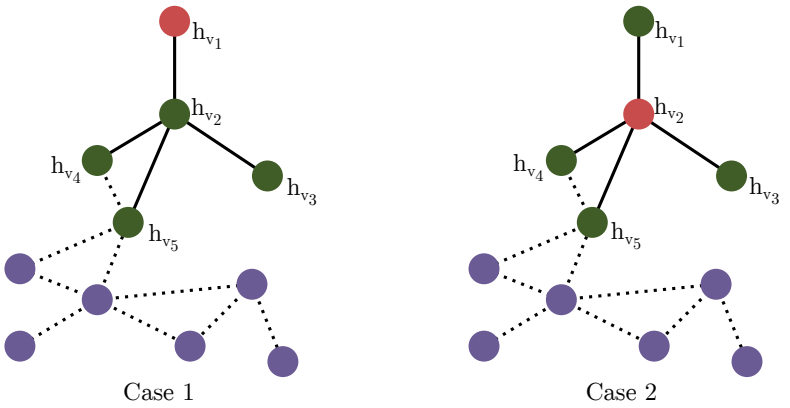


Figure 3: During the hidden state update of vertex v, the simplest GNN that considers the central node the same way as neighboring node cannot distinguish Case 1 and Case 2 since this update is based on 3 orange nodes and 1 green node in both cases. These two cases can be distinguished when the central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

$$H^{t+1} = \sigma(I \cdot H^t \cdot W^0 + H^t \cdot W^1 \cdot \text{ReLu}(A_E \cdot W_A)), \tag{12}$$

where $A_E \in \mathbb{R}^{N \times N \times |\xi|}$ is the tensor of edge features and W_A is a neural network applied to the edges attributes which computes a $f_n \times f_n$ matrix, where f_n is the number of hidden cells computed by W_0 and W_1.

However, this model presents multiple drawbacks. First, the shape of W_A is imposed by f_n. Second, W_A requires f_n^2 parameters, which greatly increases the number of parameters required by the model. Finally, this model computes tensors of shape $N \times N \times f_n \times f_n$, which induces a huge spatial complexity.

In order to overcome those drawbacks, we propose a new model able to take into account edge features. In this new model, edge features vectors are embedded into a new space of f_e dimensions, where f_e is an hyperparameter. A new diffusion matrix $C \in \mathbb{R}^{N \times N \times f_e}$ is computed as follows

$$C = \text{ReLU}(AW^A) \tag{13}$$

We call this new model Edge Embedding (EE). Following the general model defined by [3], this matrix is then used to update the nodes hidden states H using the following update rule:

$$H^{(l+1)} = \sigma\Big(\sum_{i=0}^{f_e} C^{(i)} H^l W^{(l,i)}\Big) \tag{14}$$

Our EE model can also be defined using the general framework defined as a combination of update and aggregate functions by [15]. Following this general framework, we can define our aggregate function as :

$$h^l_{N(v_i)} = \sum_{v_j \in N(v_i)} C_{i,j} h^{l-1}_{v_j} W^{(l,i)} \tag{15}$$

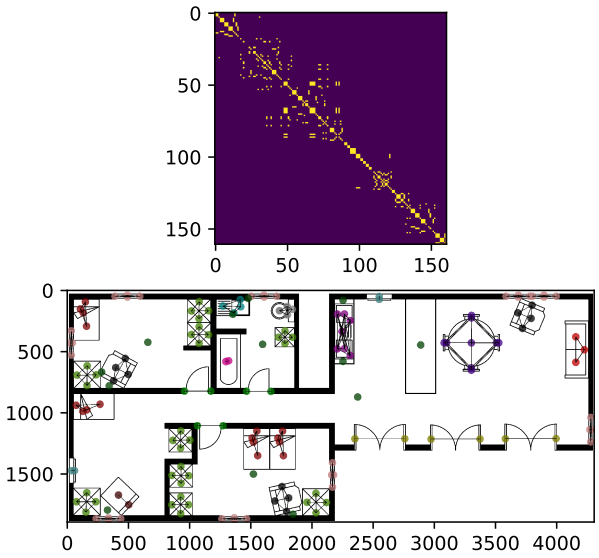


Figure 4: Example of the output matrix and its representation over the image

The update function can then simply be defined following :

$$h_{v_i}^{l} = \sigma(h_{N(v_i)}^{l} + h_{v_i}^{l-1} W^{l,i}) \tag{16}$$

In our experiments, models defined by Equations 11, 12 and 14 are tested.

3.3. Graph clustering

The second step of our model is the symbol detection part. There are several ways to consider symbol detection. For example, it can be considered as an object detection problem. In the proposed approach, we explore the symbol detection problem under the spectrum of pairwise classification [1].

In [11], one of the studied methods consists in computing pairwise predictions to decide if two elements belong or not to the same cluster. Therefore, our approach aims to predict a $N \times N$ binary matrix M for a graph having N nodes. M is a symmetric matrix, with $M(i, j)$ equals to 1 if nodes v_i and v_j represent parts belonging to the same symbol occurrence, and 0 otherwise. The matrix construction is shown in Equation 17, where S denotes the set of symbols in the graph, and $S(k)$ one particular symbol. Such a matrix can be seen as the adjacency matrix of a disconnected graph, where each connected component of the graph correspond to a symbol. The symbols are then represented by the different connected parts of the whole graph, each of them being a clique. Figure 4 shows an example of a desired output, and its interpretation over the input image.

$$M(i, j) = \begin{cases} 1, & \text{if } (v_i, v_j) \in S(k) \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

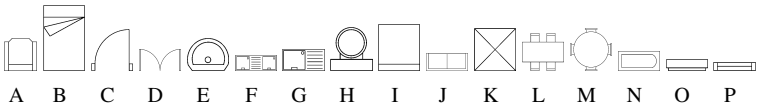


Figure 5: Symbol models

4. Experiments

4.1. Dataset presentation

To evaluate our models, we experiment them on the ILPIso dataset, a dataset made of 200 floorplans (see Figure **??** for an example). We apply the RAG transformation strategy to each floorplan image, thus leading to 200 graphs. This process results in a total of 24 281 nodes and 105 110 edges, with 121 nodes and 525 edges per graph on average. Nodes and edges are labelled according to Section 3.1.

For the classification task, each node is associated to one of the 17 classes corresponding to the different objects in original floorplan images. These 17 classes correspond to the 16 symbol types (Figure 5), plus a dummy class associated to nodes which are not associated to any symbol.

For the clustering task, each pair of nodes is classified either as 1, if end nodes represent image parts belonging to the same symbol occurrence, or 0 otherwise. A pair made of twice the same node is classified as one if this node is a part of a symbol occurrence, and 0 otherwise.

4.2. Classification task

In this section, we focus on the classification problem of nodes into the 16 symbols described previously.

4.2.1. Evaluated models for classification

For the classification task, we evaluated 4 different models. The first model is a neural network that aims at classifying each node using only its own features, and then discarding all information encoded within the graph representation. This model serves as a baseline in order to measure the information brought by the structural information for node classification. It is made of three dense layers, with the third one used for decision.

The second model is a Graph Neural Network (GNN) made of three layers. Similarly to the baseline, the third layer is used for decision. With this model, we are only able to use the existence of an edge between a pair of nodes, but not the label associated to this edge.

The third and fourth experimented models use respectively the Edge Network (EN) and the Edge Embedding (EE) defined in Section 3.2. Those models use both the existence of an edge and its label.

In order to make a fair comparison of the four models, we make them as similar as possible. Thus, both the neural networks of the baseline and the three GNN produce 200 features for each node. For our edge embedding model, we studied 4 different values for f_e: 2, 4 , 8 and 16. Since $f_e = 8$ obtained the best results on validation and test, in each experiment, we only show results for this value.

Table 1: Results obtained (in percentage of accuracy of nodes classification) by the different experimented models.

Model	**Gaussian Noise Variance**			
	0.0	0.05	0.1	0.2
MLP	95.57 ± 0.83	60.37 ± 1.82	39.99 ± 0.98	23.54 ± 0.79
GNN	99.54 ± 0.17	99.27 ± 0.30	96.36 ± 0.82	79.67 ± 3.03
EN	99.67 ± 0.12	99.54 ± 0.18	98.19 ± 0.36	85.77 ± 1.94
EE	99.64 ± 0.12	99.53 ± 0.17	98.08 ± 0.57	88.33± 2.16

4.2.2. Experimental protocol for classification task

In this section, we describe the protocol used to conduct the experiment for the classification task. First, the 200 graphs are randomly divided into 3 sets. 160 graphs are used for training, 20 for validation and 20 for test. Node and edge attributes are 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. In order to evaluate the robustness of each model, a Gaussian noise is added on the node features for the test dataset. This allows to evaluate the models behavior when test data are slightly different from training and validation data. We ran those experiments using a 10-fold cross-validation in order to limit split bias. Mean and standard deviation of accuracy for each model are reported in Table 1.

As one can see, results for GNN, EN and EE are pretty similar and close to 100%, widely increasing results obtained by the base model. However, adding noise allows to show disparities between the different models. Indeed, EN and EE models perform better than GNN when noise is added. Moreover, when Gaussian noise variance is set to 0.2, our model EE obtains better result than EN.

4.3. Clustering task

In this section, we focus on the task of clustering nodes into subgraphs corresponding to symbol occurrences.

4.3.1. Evaluated models for clustering

For the clustering task, we also compare the 4 different models described in Section 4.2.1. First model consists of 2 dense layers. A tensor $P \in \mathbb{R}^{N \times N \times f_n * 2}$ is then computed, where $P(i, j, :)$ is the concatenation of hidden state vector h_i^t of node v_i and hidden state vector h_j^t of node v_j Finally, a dense layer takes the decision whether the pair of nodes v_i and v_j belong to the same cluster or not. This model does not use any topological information.

The second evaluated model is composed of 2 GNN layers. The output node states are then used to compute a tensor related to node pairs. A dense layer whose architecture is similar to the one used in the first model finally predicts whether the nodes of the pair belongs to the same cluster (symbol occurrence) or not.

Table 2: Results obtained following the previous metric by the different experimented models.

Model	**Gaussian Noise Variance**			
	0.0	0.05	0.1	0.2
MLP	45.65 ± 4.51	0.46 ± 0.32	0.0 ± 0.0	0.0 ± 0.0
GNN	90.13 ± 1.28	75.12 ± 4.37	44.22 ± 4.57	17.18 ± 3.62
EN	92.59 ± 1.02	76.54 ± 2.00	39.88 ± 3.25	12.31 ± 1.81
EE	92.22 ± 1.11	79.63 ± 3.03	49.72 ± 5.15	21.21 ± 2.71

Finally, the third and fourth model are similar to the second except that they also uses either the edge network model or the edge embedding model, in order to take into account edge features during the hidden state computation.

4.3.2. Experimental protocol for clustering task

In this section, we describe the protocol used to conduct the experiment for the clustering task. Similarly to the previous experiment, the 200 graphs are randomly divided into 3 sets following the same protocol. Node and edge attributes are also normalized in the same way than before, and we keep using the Adam optimizer with an initial learning rate of 5.10^{-3}. The model is trained for 300 epochs. To tackle the huge class imbalance, the focal loss is used [23], with $\gamma = 2$ and $\alpha = 0.3$, where α is a weights both classes (class 0 is weighted by α and class 1 is weighted by $(1-\alpha)$) and γ is an hyperparameter that controls the contribution of each sample in the loss computation depending on how hard is the sample to be classified. A hard sample correctly classified will thus have more importance than an easy sample correctly classified. $\alpha = 0.3$ allows us to weights the class 0, which is over-represented, with a lower weight than the class 1, which is under-represented. $\gamma = 2$ is a standard value for focal loss. The evaluated metric is the ratio of symbols detected. We define a symbol as detected as a connected component of the graph consisting of all the nodes describing a symbol occurrence and that does not contain any extra node. Results obtained are given in Table 2.

Results show that integrating edge features definitively improves the symbol detection when no noise is added. This seems to show that edge features carry new information that help the models in decision, compared to GNN that only rely on structural information. The same result occurs with a Gaussian noise of variance equal to 0.5. However, in the case of EN, those results drop quickly and obtain worst results with Gaussian noise variance above 0.1. This could be explained by the huge number of parameters of EN, preventing the model to generalize on few noisy data. The poor results obtained by the different models can be explained by two arguments. First, the edge prediction task is a very imbalanced task. Most of the pairs of nodes do not belong to the same symbol, leading to more than 96% of pairs that are labeled as 0. Second, the importance of the noise. Since each feature is normalized, adding a Gaussian noise of variance 0.2 leads to important noising.

4.4. Experimental protocol for combination of both classification and clustering

In this last experiment, we examine the effect of combining both clustering and classification in the same model. Again, the same four models that were presented in Section 4.2.1 are compared.

Table 3: Results obtained by the different experimented models using the symbol detection accuracy.

Model	**Gaussian Noise Variance**			
	0.0	0.05	0.1	0.2
MLP	40.69 ± 3.19	1.04 ± 0.46	0.08 ± 0.10	0.0 ± 0.0
GNN	86.30 ± 2.31	66.09 ± 3.12	37.51 ± 2.55	12.42 ± 1.48
EN	90.99 ± 0.64	75.29 ± 2.65	37.64 ± 4.29	12.10 ± 1.66
EE	87.22 ± 1.74	70.88 ± 2.53	41.00 ± 3.37	14.22 ± 3.02

The network is trained for 300 epochs, with a combination of both standard crossentropy and focal loss. The computed loss is thus the mean of both losses. The computed metric is the ratio of symbols detected, where a symbol is considered as detected as a connected component of the graph consisting of all the nodes correctly classified describing a symbol occurrence and that does not contain any extra node. This task is thus harder than before, due to the node classification condition. Results are shown in Table 3.

Conclusions drawn from these results are similar to the previous ones: EN obtained best results but are not robust to noise. Conversely, our method performs below EN with unnoised data, but performs much better with noised data, the gap increasing with the noise.

A fair comparison with the results given by subgraph isomorphism methods that are not based on machine learning reported in [19, 21] is not easy for several reasons. Even if those methods are evaluated on the same dataset, the reported results are expressed with the precision-recall metrics when a symbol is considered as detected when half of the graph nodes are correctly matched. The method described in [21] which is an improvement of the one of [19] achieves an F1-score between 0.95 and 1.0 according to the value of an hyperparameter. In our experiment we consider the accuracy measure with a correct detection when all the nodes describing a symbol occurrence are found without any extra node. Our criterion is more stringent. The noise model applied to evaluate the method is also different. Besides, the experimental evaluation conducted in [21] has been limited to the search of 13 among the 16 symbols that can be represented by connected graphs whereas our proposition is able to handle symbols described with disconnected graphs.

Methods based on subgraph isomorphism such as those described in [21] do not need any training step. As a consequence, a new symbol can be directly used as a query, whereas methods based on machine learning such as our proposition need the knowledge of these symbols at the training step. On the other hand, machine learning based methods operate in polynomial time whereas traditional graph matching methods have an exponential complexity and can be intractable for large graphs.

5. Conclusion

In this paper, a new approach for floorplans segmentation is proposed. Floorplans images are firstly transformed into Region Adjacency Graph by the approach. Graph Neural Network are then applied on those graphs to take two different decisions: one decision for node classification, and another for clustering.

We compared three different models of graph neural network with a baseline who does not use structural information. Each GNN models present great results for each task, largely improving result obtained with the baseline model. Moreover, the baseline model was not able to predict a satisfactory output for the clustering task, providing cliques that do not correspond to symbols.

Even imperfect, we argue that these results provide a first empirical evidence that machine learning can be used to solve a subgraph isomorphism problem when learning data are available.

Future work will consider using graph pooling strategy to compute a new graph where a node corresponds to a whole symbol. Another perspective would be to use Siamese networks to detect a single symbol specified in input.

References

[1] Awasthi, P., Zadeh, R.B., 2010. Supervised clustering, in: Advances in neural information processing systems, pp. 91–99.

[2] Baja, G.S.D., Thiel, E., 1996. Skeltonization algorithm running on path-based distance maps. Image and Vision Computing 14, 47–57.

[3] Balcilar, M., Renton, G., Héroux, P., Gaüzère, B., Adam, S., Honeine, P., 2021. Analyzing the expressive power of graph neural networks in a spectral perspective, in: International Conference on Learning Representations. URL: `https://openreview.net/forum?id=-qh0M9XWxnv`.

[4] Battaglia, P., Pascanu, R., Lai, M., Rezende, D.J., et al., 2016. Interaction networks for learning about objects, relations and physics, in: Advances in neural information processing systems, pp. 4502–4510.

[5] Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al., 2018. Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261 .

[6] Cai, H., Zheng, V.W., Chang, K.C.C., 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. IEEE Transactions on Knowledge and Data Engineering 30, 1616–1637.

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

[8] Conte, D., Foggia, P., Sansone, C., Vento, M., 2004. Thirty years of graph matching in pattern recognition. IJPRAI 18, 265–298.

[9] Defferrard, M., Bresson, X., Vandergheynst, P., 2016. Convolutional neural networks on graphs with fast localized spectral filtering, in: Advances in Neural Information Processing Systems 29, pp. 3844–3852.

[10] Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., Adams, R.P., 2015. Convolutional networks on graphs for learning molecular fingerprints, in: Advances in Neural Information Processing Systems 28, pp. 2224–2232.

[11] Finley, T., Joachims, T., 2005. Supervised clustering with support vector machines, in: Proceedings of the 22nd international conference on Machine learning, pp. 217–224.

[12] Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E., 2017. Neural message passing for quantum chemistry. arXiv preprint arXiv:1704.01212 .

[13] Gori, M., Monfardini, G., Scarselli, F., 2005. A new model for learning in graph domains, in: Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on, IEEE. pp. 729–734.

[14] Grover, A., Leskovec, J., 2016. node2vec : a scalable feature learning for networks, in: proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, ACM.

[15] Hamilton, W., Ying, Z., Leskovec, J., 2017. Inductive representation learning on large graphs, in: Advances in Neural Information Processing Systems, pp. 1024–1034.

[16] Hammond, D.K., Vandergheynst, P., Gribonval, R., 2011. Wavelets on graphs via spectral graph theory. Applied and Computational Harmonic Analysis 30, 129–150.

[17] Kearnes, S., McCloskey, K., Berndl, M., Pande, V., Riley, P., 2016. Molecular graph convolutions: moving beyond fingerprints. Journal of computer-aided molecular design 30, 595–608.

[18] Kipf, T., Welling, M., 2017. Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations.

[19] Le Bodic, P., Héroux, P., Adam, S., Lecourtier, Y., 2012. An integer linear program for substitution-tolerant subgraph isomorphism and its use for symbol spotting in technical drawings. Pattern Recognition 45, 4214–4224.

[20] Lerouge, J., Abu-Aisheh, Z., Raveaux, R., Héroux, P., Adam, S., 2016a. Exact Graph Edit Distance Computation Using a Binary Linear Program, in: S+SSPR 2016, Mérida, Mexico, November 29 - December 2, 2016, Proceedings, pp. 485–495. doi:10.1007/978-3-319-49055-7_43.

[21] Lerouge, J., Hammami, M., Héroux, P., Adam, S., 2016b. Minimum cost subgraph matching using a binary linear program. Pattern Recognition Letters 71, 45–51. URL: https://www.sciencedirect.com/science/article/pii/S0167865515004250, doi:https://doi.org/10.1016/j.patrec.2015.11.026.

[22] Li, Y., Zemel, R., Brockschmidt, M., Tarlow, D., 2016. Gated graph sequence neural networks, in: Proceedings of ICLR'16.

[23] Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P., 2017. Focal loss for dense object detection, in: Proceedings of the IEEE international conference on computer vision, pp. 2980–2988.

[24] Niepert, M., Ahmed, M., Kutzkov, K., 2016. Learning convolutional neural networks for graphs, in: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, pp. 2014–2023.

[25] Renton, G., Héroux, P., Gaüzère, B., Adam, S., 2019. Graph neural network for symbol detection on document images, in: 2019 International Conference on Document Analysis and Recognition Workshops (ICDARW), IEEE. pp. 62–67.

[26] Rica, E., Moreno-García, C.F., Álvarez, S., Serratosa, F., 2020. Reducing human effort in engineering drawing validation. Computers in Industry 117, 103198. URL: https://www.sciencedirect.com/science/article/pii/S0166361519310826, doi:https://doi.org/10.1016/j.compind.2020.103198.

[27] Riesen, K., Bunke, H., 2009. Approximate graph edit distance computation by means of bipartite graph matching. Image and Vision Computing 27, 950–959. doi:10.1016/j.imavis.2008.04.004.

[28] Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G., 2009. The graph neural network model. Trans. Neur. Netw. 20, 61–80. doi:10.1109/TNN.2008.2005605.

[29] Schütt, K.T., Arbabzadah, F., Chmiela, S., Müller, K.R., Tkatchenko, A., 2017. Quantum-chemical insights from deep tensor neural networks. Nature communications 8, 13890.

[30] Serratosa, F., 2014. Fast computation of bipartite graph matching. Pattern Recognition Letters 45, 244–250. doi:10.1016/j.patrec.2014.04.015.

[31] Solnon, C., 2010. Alldifferent-based filtering for subgraph isomorphism. Artif. Intell. 174, 850–864.

[32] Teague, M., 1980. Image analysis via the general theory of moments. Journal of the Optical Society of America 70, 920–930.

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

[34] Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., 2017. Graph attention networks. arXiv preprint arXiv:1710.10903 1.

[35] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S., 2019. A comprehensive survey on graph neural networks. arXiv preprint arXiv:1901.00596 .

[36] Yan, S., Xu, D., Zhang, B., jiang Zhang, H., Yang, Q., Lin, S., 2007. Graph embedding and extensions : a general framework for dimensionality reduction. IEEE Trans. on Pattern Analysis and Machine Intelligence 29, 40–51.

[37] Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Sun, M., 2018. Graph neural networks: A review of methods and applications. arXiv preprint arXiv:1812.08434 .