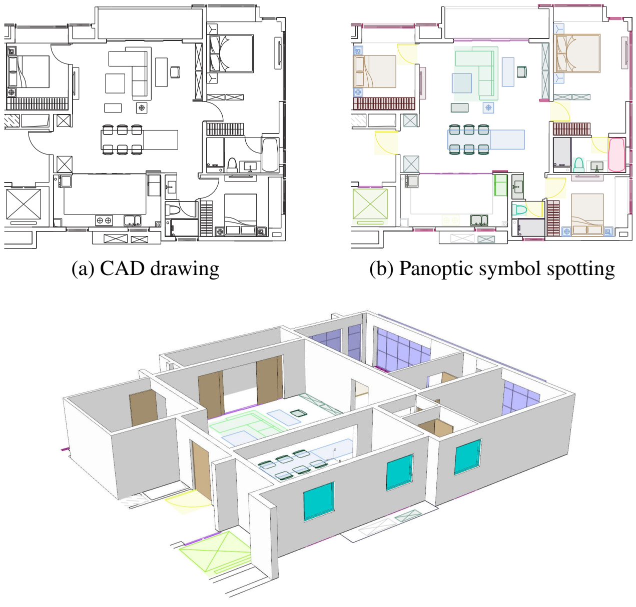
GAT-CADNet: Graph Attention Network for Panoptic Symbol Spotting in CAD Drawings

Zhaohua Zheng[1,2]*, Jianfang Li[2]*, Lingjie Zhu[2], Honghua Li[2],
Frank Petzold[1], Ping Tan[2,3]†
[1]Technical University of Munich, [2]Alibaba Group, [3]Simon Fraser University
{zhaohua.zheng,petzold}@tum.de,{wuhui.ljf}@alibaba-inc.com,
{lingjie.zhu.me, howard.hhli}@gmail.com, {pingtan}@sfu.ca

Abstract

Spotting graphical symbols from the computer-aided design (CAD) drawings is essential to many industrial applications. Different from raster images, CAD drawings are vector graphics consisting of geometric primitives such as segments, arcs, and circles. By treating each CAD drawing as a graph, we propose a novel graph attention network GAT-CADNet to solve the panoptic symbol spotting problem: vertex features derived from the GAT branch are mapped to semantic labels, while their attention scores are cascaded and mapped to instance prediction. Our key contributions are three-fold: 1) the instance symbol spotting task is formulated as a subgraph detection problem and solved by predicting the adjacency matrix; 2) a relative spatial encoding (RSE) module explicitly encodes the relative positional and geometric relation among vertices to enhance the vertex attention; 3) a cascaded edge encoding (CEE) module extracts vertex attentions from multiple stages of GAT and treats them as edge encoding to predict the adjacency matrix. The proposed GAT-CADNet is intuitive yet effective and manages to solve the panoptic symbol spotting problem in one consolidated network. Extensive experiments and ablation studies on the public benchmark show that our graph-based approach surpasses existing state-of-the-art methods by a large margin.


(a) CAD drawing

(b) Panoptic symbol spotting

(c) Reconstructed BIM model

Figure 1. A patch of floor plan (a) and its panoptic symbol spotting results (b), where line semantics are color coded and instances are presented by translucent rectangles. The BIM model (c) with complete semantic and accurate geometry can be reconstructed from such annotated floor plan. We only show 3D model of wall, windows and doors for the sake of clarity.

1. Introduction

Computer-aided design (CAD) is the use of computers to generate digital 2D or 3D illustrations of a product, aiding the creation, modification, analysis, or optimization process during designing and manufacturing. This technology has been widely used in modern architecture, engineering and construction (AEC) industries. The CAD drawings usually convey accurate geometry, rich semantic, and domain-specific knowledge of a product design, with basic geometric primitives, such as segments, arcs, circles, and ellipses, as illustrated in Figs. 1a and 1b.

Spotting and recognizing symbols from the CAD drawings is the first step towards understanding its content, which is crucial to many real-world industrial applications. For example, building information modeling (BIM) has growing demand in various architecture engineering ar-

*Equal contribution.
†Corresponding author.

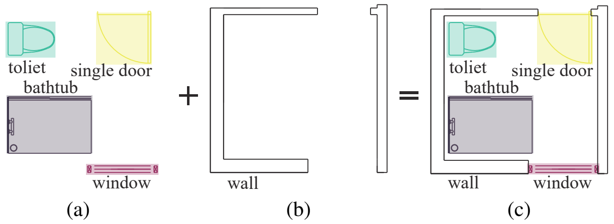
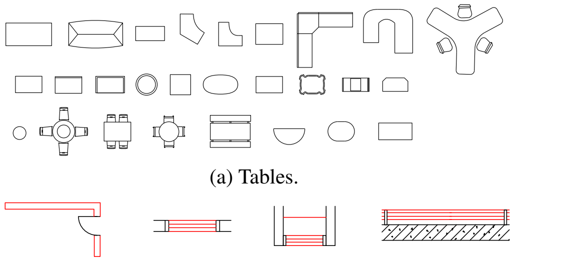


Figure 2. Illustration of the panoptic symbol spotting in a bathroom. Symbols of countable things (a), and uncountalbe stuff, e.g., wall (b). The panoptic symbol spotting proposed by [15] considers both types of symbols in a unified scheme (c).


(a) Tables.

(b) Wall, window, bay window, and curtain wall are high lighted in red.

Figure 3. The inter-class variance (a) and intra-class similarity (b) in the public FloorPlanCAD dataset.

eas such as pipe arrangement, construction inspection and equipment maintenance. A floor plan usually contains complete details of a storey in an orthogonal top-down view. Therefore, a BIM model can be precisely reconstructed from a group of 2D floor plans with accurate semantic and instance annotations, as demonstrated in Fig. 1c.

Traditional symbol spotting usually deals with instance symbols representing countable things [32], like table, sofa, and bed. Following the idea in [23], Fan *et al.* [15] extended the definition with recognizing semantic of uncountable stuff, and named it *panoptic symbol spotting*, as shown in Fig. 2. Therefore, all components in a CAD drawing are covered in one task altogether. For example, the wall represented by a group of parallel lines was properly handled by [15], which however was treated as background by [27, 28, 33, 35].

Large-scale dataset of high quality annotations is the fundamental ingredient to recent advances in supervised methods with deep learning, *e.g.*, ImageNet [10] for image classification, COCO [26] for image detection, and ShapeNet [2] for 3D shape analysis. Existing datasets for symbol spotting on floor plan, i.e., SESYD [9] and FPLAN-POLY [34], are either synthetic, or inaccurate, both with only a few hundreds of samples. Fan *et al.* [15] built the first large-scale real-world FloorPlanCAD dataset of over $10,000$ floor plans in the form of vector graphics, and provided line-grained panoptic annotations.

CAD drawings are composed of domain-specific items, which are usually represented by abstract symbols. Human perception of CAD drawings is usually a multi-modal cross-context reference process requiring strong domain related knowledge. Meanwhile, the large intra-class variance and small inter-class dissimilarity of symbols make it a more challenging task for computers, as shown in Fig. 3.

Representing a CAD drawing as a graph of primitives is an intuitive way to retain the property of vector graphics, and has been proven effective for the semantic symbol spotting task in [15]. In this work, we present a novel graph attention network GAT-CADNet to solve the panoptic symbol spotting problem. The network achieved state-of-the-art performance and our main contributions are:

- We formulate the instance symbol spotting task as a subgraph detection problem, and solve it by predicting the adjacency matrix.
- We explicitly encode the relative relation among vertices, using a relative spatial encoding (RSE) module, to enhance the vertex attention.
- We treat the vertex attention as edge encoding for predicting the adjacency matrix, and design a cascaded edge encoding (CEE) module to aggregate vertex attentions from multiple GAT stages.

2. Related Work

In this section we briefly summarize methods in related areas, including symbol spotting, panoptic segmentation, graph neural networks, and attention.

Symbol spotting. It is the process of finding target symbols from an image or a document [35, 37]. Optical character recognition (OCR) can be viewed as a specific case where symbols are from a standard character set. Traditional non-data-driven methods usually design hand-crafted descriptors [27, 28, 35], then the query symbol is matched to the document by sliding window or graph matching approaches [12–14]. With recent development in deep learning, data-driven approaches [15, 33] reported better results on various datasets [9, 34].

Panoptic segmentation. In the computer vision community, object detection often refers to identifying countable things from an image such as cats, dogs, and cars [17, 24, 25]. On the other hand, semantic segmentation is partitioning an image into multiple regions without distinguishing instances with the same semantic [5, 40]. However, there is uncountable stuff that has no instance but only semantic, such as sky, road, and pavement [4, 5, 36]. Panoptic segmentation is first introduced by Kirillov et al. [23], which treated

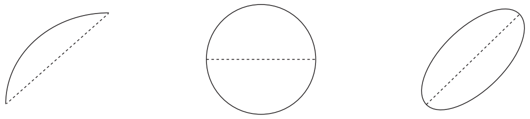


Figure 4. The segment approximation for graphic primitives (arc, circle, and ellipse) are shown as dash lines.

countable instance things and uncountable stuff as one visual recognition task [22, 23, 44]. Chen *et al*. [6] improved the panoptic segmentation quality with a bidirectional path between the semantic and instance segmentation branches. Wu *et al*. [43] constructed modular graph structure to reason their relations. Inspired by [23], Fan *et al*. [15] generalized the traditional symbol spotting problem and considered both countable things and uncountable stuff symbols as one recognition task. They also provided a reasonable evaluation metric and a well-annotated public dataset.

Graph neural networks. The graph convolutional networks (GCNs) proposed by Thomas *et al*. [20] operated directly on graphs via a local first-order approximation of spectral graph convolutions. To enable the training of traditional neural networks on the graphs, Zhang *et al*. [47] sorted graph vertices in a consistent order. Ying *et al*. [46] introduced a differentiable graph pooling module that can generate hierarchical representations of graphs. Some works [15, 16, 41] tried to fuse image features to enhance the GCNs. Thomas *et al*. [21] proposed graph auto encoders (GAE) and variational graph auto encoders (VGAE), where vertex features are used to restore adjacency matrix.

Attention. Transformers have brought the machine translation and natural language processing to a higher level [8, 19, 42, 45]. The success has stimulated the development of self-attention networks for various image perception tasks [11, 18, 29, 48]. Bello *et al*. [1] augmented CNN with relative self-attention to integrate global information to the network. Dosovitskiy *et al*. [11] cut images into grid patches and apply attention on the sequence. Vaswani *et al*. [39] proposed self attention which is permutation invariant for sequence data. In the same paper, they added positional embedding to the networks. In long sequence cases, Dai *et al*. [8] found attention matrix is usually sparse and local focused. Hence, they proposed a method to encode not absolute but relative position.

3. Methodology

Our GAT architecture takes CAD drawings of vector graphics as input and predicts the semantic and instance attributes of every geometric primitive in it.

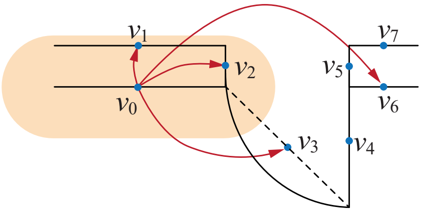


Figure 5. Graph construction: blue dots represent vertices v_i, and red arrows are edges staring from v_0. Note that v_1, v_2, v_3 are connected to v_0 due to their closeness (the orange area is the ϵ envelope of v_0), while v_6 is connected due to their collinearity.

3.1. Graph Construction

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is constructed for one input CAD drawing, where vertex $v_i \in \mathcal{V}$ is the segment approximation of a geometric primitive. The segment approximation of an arc is the line connecting its start and end points, while the horizontal diameter or major axis are approximations for a circle and ellipse respectively, see Fig. 4 for illustrations. Such simplifications are acceptable, because segments are the majority in CAD drawings.

An edge connecting two vertices v_i and v_j is added if their distance d_{ij} is below certain threshold ϵ, where:

$$d_{ij} = \min_{\boldsymbol{p} \in v_i, \boldsymbol{q} \in v_j} \|\boldsymbol{p} - \boldsymbol{q}\|. \tag{1}$$

Since CAD drawings are usually drawn by professionals to depict man-made objects with strong regularity, we add extra edges for collinear primitives. To keep the graph complexity low, at most K edges are allowed for every vertex by random dropping. Fig. 5 demonstrates the graph construction around a door symbol, where only edges starting from v_0 are illustrated. In the following experiments, we set $\epsilon = 300\text{mm}$ and $K = 30$.

Instance and subgraph. An instance symbol of countable things, e.g., tables or doors, usually consists of a set of locally connected primitives. Naturally, an instance corresponds to an connected subgraph $\mathcal{G}_k \subset \mathcal{G}$. Therefore we formulate the instance symbol spotting task as a subgraph detection problem, which can be solved by predicting the adjacency matrix.

Vertex feature. We define the vertex features $\boldsymbol{v}_i \in \mathbb{R}^7$ as:

$$\boldsymbol{v}_i = [cos(2\alpha_i), sin(2\alpha_i), l_i, \boldsymbol{t}_i]\,, \tag{2}$$

where $\alpha_i \in [0, \pi)$ is the clockwise angle from the x positive axis to v_i, and l_i measures the length of v_i. Note that our direction features are continuous when α jumps between 0 and π. We encode the primitive type (segment, arc, circle, or ellipse) into a one hot vector $\boldsymbol{t}_i \in \mathbb{R}^4$ to make up the missing information of segment approximations.

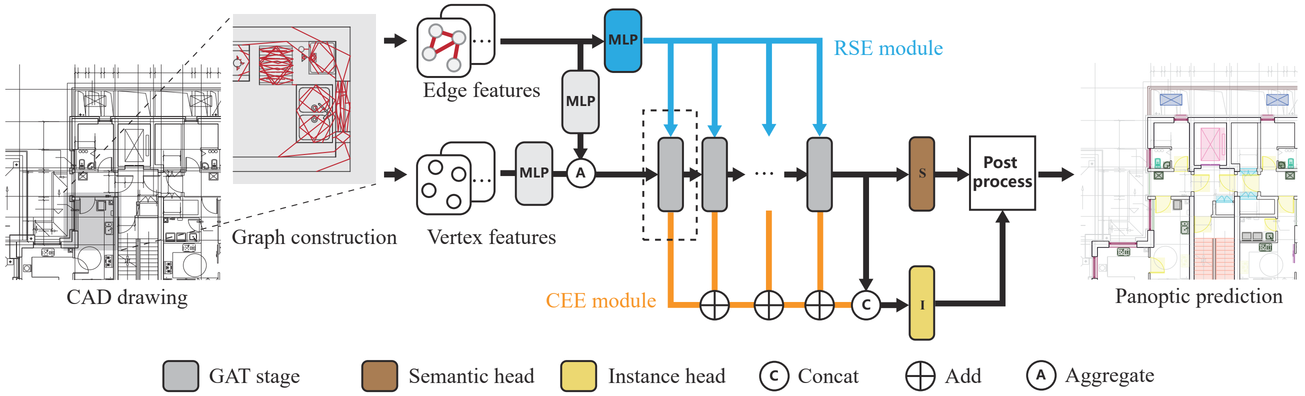


Figure 6. Architecture of the proposed GAT-CADNet. The middle branch includes the main GAT stages of gray blocks followed by the semantic and instance symbol spotting heads. The upper blue branch is the RSE module, and the lower orange branch is the CEE module.

Edge features. Besides vertex features, we explicitly encode relation between two vertices as edge features. The positional offset $\boldsymbol{\delta}_{ij}$ from v_i to v_j is defined as:

$$\boldsymbol{\delta}_{ij} = \boldsymbol{m}_j - \boldsymbol{m}_i, \tag{3}$$

where $\boldsymbol{m}_i$ is the middle point of v_i. The directional offset $\measuredangle_{ij}$ is defined as the acute angle between v_i and v_j. The length ratio between v_i and v_j is computed as:

$$r_{ij} = \frac{l_i}{l_i + l_j}. \tag{4}$$

As illustrated in Fig. 1 and reported in [15], the parallelism and orthogonality between two line segments are common and play crucial role in CAD drawings. We add three binary indicators to emphasize such regularities:

$$\boldsymbol{g}_{ij} = [\|_{ij}, \perp_{ij}, \neg_{ij}], \tag{5}$$

where $\|_{ij}$ and $\perp_{ij}$ indicates whether v_i is parallel or orthogonal to v_j, and $\neg_{ij}$ is used to indicate whether v_i and v_j share a same end point. Putting the aforementioned terms together, we obtain the edge features $\boldsymbol{e}_{ij} \in \mathbb{R}^7$ as:

$$\boldsymbol{e}_{ij} = \left[\boldsymbol{\delta}_{ij}, \measuredangle_{ij}, r_{ij}, \boldsymbol{g}_{ij}\right]. \tag{6}$$

In our experiments, the angle and distance threshold used in $\boldsymbol{g}_{ij}$ are set to 5° and 100mm respectively.

3.2. Network Architecture

Based on the graph constructed from the CAD drawing in Sec. 3.1, we propose a novel GAT-CADNet to solve the panoptic symbol spotting problem, as shown in Fig. 6. The network 1) formulates the instance symbol spotting task as an adjacency matrix prediction problem, 2) enhances the vertex attention with edge feature encoding, 3) aggregates vertex attentions from multiple GAT stages for predicting the sparse adjacency matrix.

The initial vertex features $\boldsymbol{v}_i$ and edge features $\boldsymbol{e}_{ij}$ are embedded to $\hat{\boldsymbol{v}}_i$ and $\hat{\boldsymbol{e}}_{ij}$ with two separate multilayer perceptron (MLP) blocks. For each vertex v_i, we enhance its features by its connected edges as:

$$\boldsymbol{v}_i^0 = \text{Concat}(\hat{\boldsymbol{v}}_i, \text{MaxPooling}(\{\hat{\boldsymbol{e}}_{ij}\})). \tag{7}$$

Vertex features are stacked to $V^0 \in \mathbb{R}^{N \times 128}$, $N = |\mathcal{V}|$, as the input for the following GAT stages.

Relative spatial encoding (RSE). When processing point cloud [49] or natural language [39], researchers often use relative position encoding to make the network invariant to translation and aware of distance. Similarly, we pass the initial edge features through another MLP block to encode the relative spatial relations among vertices:

$$R = \text{MLP}(E), \tag{8}$$

where $E \in \mathbb{R}^{N \times N \times 7}$ is the edge features by expanding $|\mathcal{E}|$ edges to $N \times N$. The RSE encoding $R \in \mathbb{R}^{N \times N \times H}$ is then fed to every stage of the main GAT branch, where H is the number of heads in the GAT statge.

Graph attention stage. The stem of our network is the GAT branch of S stages, as illustrated in Fig. 7. The s-th stage takes vertex features V^{s-1} from previous stage and outputs vertex features V^s of the same dimension. In the h-th head of the GAT block, we project V^s to a query matrix $Q_h \in \mathbb{R}^{N \times d}$, a key matrix $K_h \in \mathbb{R}^{N \times d}$, and a value matrix $V_h \in \mathbb{R}^{N \times d}$. Then the multihead attention score $A^s \in \mathbb{R}^{N \times N \times H}$ can be formulated as:

$$A_h^s = Q_h K_h^T, \tag{9}$$

$$A^s = \text{Concat}(A_1^s, \ldots, A_H^s). \tag{10}$$

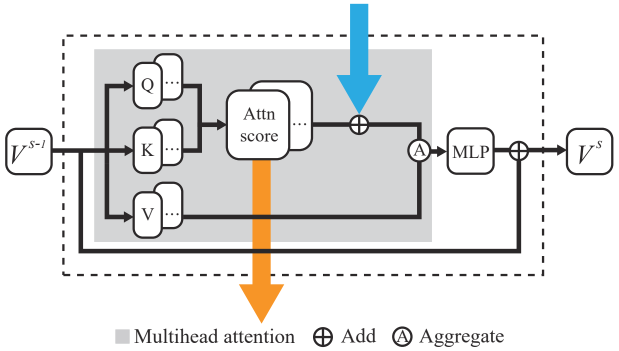


Figure 7. A GAT stage in Fig. 6, where the gray area contains the multihead attention [39]. The vertex attention score is fed to the CEE module as edge encoding (orange arrow), and then enhanced with edge encoding from the RSE module (blue arrow).

Note that A^s expresses the relation among vertices in the embedding space. Similar to the relative position encoding in [39,49], we add our relative spatial encoding R to A^s to enhance their attention explicitly. Therefore the aggregated value matrix $V'_h \in \mathbb{R}^{N\times d}$ is obtained by:

$$V'_h = \text{Softmax}(A^s + R)V_h, \tag{11}$$

which is passed through a MLP block and added to V^{s-1}, producing the output vertex features V^s of current stage. The semantic symbol spotting head maps vertex features from the final stage to the classification prediction:

$$Y = \text{Softmax}(\text{MLP}(V^S)), \tag{12}$$

with the semantic loss as:

$$loss_{\text{sem}} = \text{CrossEntropy}(Y, Y^{gt}). \tag{13}$$

Cascaded edge encoding (CEE). Recall that vertex attentions A^s can be viewed as relational intensity among vertices, which are good choice for predicting the adjacency matrix. Therefore, we cascade attention scores from all GAT stages $\{A^s\}$ as implicit edge encoding to capture local and global vertex connectivity:

$$C = \sum_{s=1}^{S} A^s. \tag{14}$$

Each valid edge encoding $\boldsymbol{c}_{ij}$ in $C \in \mathbb{R}^{N\times N\times H}$ is then concatenated with vertex features of its two endpoints from the last GAT stage to form the final edge feature:

$$\tilde{\boldsymbol{e}}_{ij} = \text{Concat}(\boldsymbol{c}_{ij}, \boldsymbol{v}_i^S, \boldsymbol{v}_j^S). \tag{15}$$

Finally, the adjacency matrix prediction $Z \in \mathbb{R}^{N\times N}$ is given by the instance symbol spotting head:

$$Z = \text{Sigmod}(\text{MLP}(\tilde{E})), \tag{16}$$

where $\tilde{E} \in \mathbb{R}^{N\times N\times (H+256)}$ denotes the stacked final edge features $\{\tilde{\boldsymbol{e}}_{ij}\}$. The loss for instance symbol spotting is defined as:

$$loss_{\text{ins}} = \text{BinaryCrossEntropy}(Z, Z^{gt}, \boldsymbol{w}), \tag{17}$$

where weights $\boldsymbol{w}$ for punishing incorrect predictions are defined as:

$\boldsymbol{w}$	$Z_{ij}^{gt} = 0$	$Z_{ij}^{gt} = 1$
$Y_i^{gt} = Y_j^{gt}$	20	2
$Y_i^{gt} \neq Y_j^{gt}$	1	0

.

Note that an edge connecting two vertices with the same semantic label ($Y_i^{gt} = Y_j^{gt}$) but belong to different instances ($Z_{ij}^{gt} = 0$) has largest weight of 20.

Panoptic loss. The panoptic symbol spotting loss of our network is the linear combination of the semantic and instance loss terms:

$$loss_{\text{pan}} = loss_{\text{sem}} + \lambda loss_{\text{ins}}. \tag{18}$$

In our implementation, the attention is conducted within the one-ring neighbors and our $N \times N$ matrices are sparse.

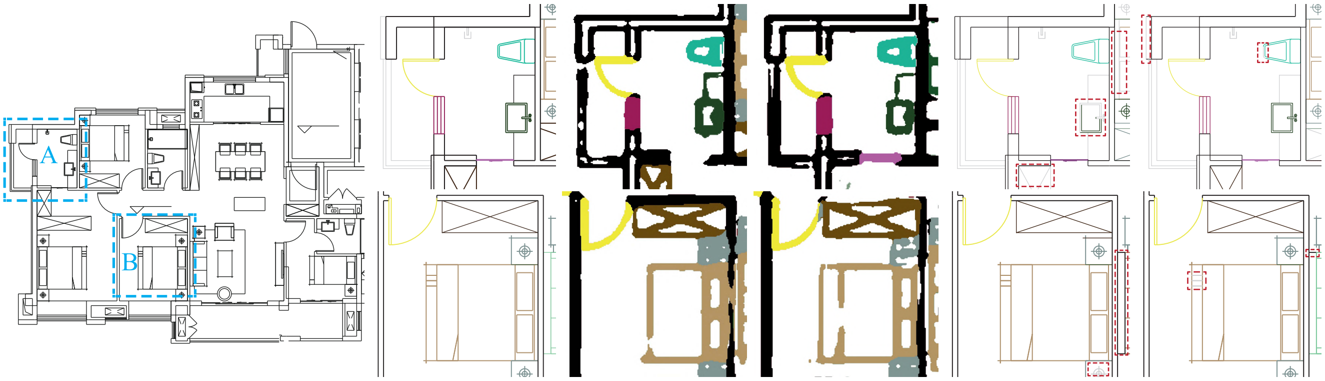
4. Experiment

Qualitative and quantitative evaluations of our GAT-CADNet are conducted for the panoptic symbol spotting task on the public CAD drawing dataset. We also compare our method with typical image-based instance detection [30,31,38] and semantic segmentation methods [5,40]. Extensive ablation study is performed to validate the design choice of our network.

Dataset and panoptic metric. Although there are several small vector graphics datasets [9, 34] for traditional symbol spotting, we use the latest large-scale FloorPlanCAD [15] dataset in our experiment, which has $11,602$ CAD drawings of various floor plans with segment-grained panoptic annotation. The dataset consists of 10m × 10m squared blocks covering 30 things and 5 stuff classes. Similar to [23], it also provides a panoptic metric defined on vector graphics:

$$\begin{aligned} PQ &= RQ \times SQ \\ &= \frac{\sum_{(s^p,s^g)\in TP} \text{IoU}(s^p, s^g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}, \end{aligned} \tag{19}$$

where RQ is the F_1 score measuring the recognition quality and SQ is the segmentation quality computed by averaging IoUs of matched symbols. For the detailed IoU evaluation of a predicted symbol s^p and the ground truth symbol s^g at primitive level, please refer to [15].



(a) Input CAD drawing (b) GT (c) DeepLabv3 [5] (d) HRNetV2 [40] (e) PanCADNet [15] (f) Ours

Figure 8. Qualitative comparison of semantic symbol spotting results on the FloorPlanCAD [15] dataset. Two close-ups of region A (upper row) and B (lower row) are listed from left to right.

Methods	F1	length-weighted F1
HRNetsV2 W18 [40]	0.656	0.683
HRNetsV2 W48 [40]	0.666	0.693
DeepLabv3+R50 [4]	0.680	0.705
DeepLabv3+R101 [4]	0.688	0.714
PanCADNet [15]	0.806	0.798
Ours	**0.850**	**0.823**

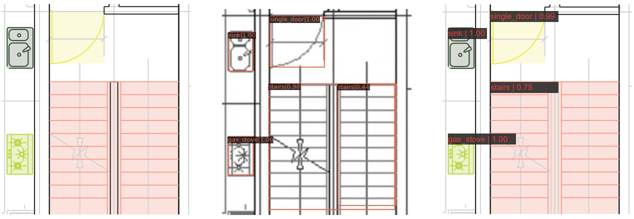
Table 1. Statistical results of different image semantic segmentation models and our GAT-CADNet.

Implementation. In the following experiments, our GAT-CADNet is configured with 8 GAT stages and $H = 8$, $\lambda = 2$ if not specified. We use the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.99$, $lr = 0.001$ and set the decay rate to 0.7 for every 20 epochs. We train our GAT-CADNet for 100 epochs and take the best model on the validation split. The number of graph vertices and their neighbours are limited to 4096 and 30 respectively for each CAD drawing to fit graphics card memory. All other image-based networks are trained with the latest release of OpenMMLab [3, 7].

During inference, we prune the resulted adjacency matrix by a threshold of 0.7, producing a directed graph. Vertices of the same semantics are grouped first, and then instances are found by searching connected components within each group. Please refer to the supplementary material for more results and feel free to zoom in since they are vector graphics.

4.1. Quantitative Evaluation

Semantic symbol spotting. To compare with existing image segmentation methods, the CAD drawings are rendered as images with line width of 2 pixels. The semantic of a primitive in $\mathcal{G}$ is then retrieved by sampling on the predicted mask with a majority voting strategy. PanCADNet [15] is a GCN architecture for semantic symbol spotting and relies on image features from a CNN backbone. Tab. 1 compares the results of popular segmentation methods [4, 40] with different configurations. Qualitative comparion are shown in Fig. 8 where DeepLabv3 [5] and HRNetV2 [40] are with the W48 and R01 configuration in Tab. 1 respectively. While our GAT-CADNet is built on the graph entirely and requires geometric features only, it manages to outperform other image-based methods.

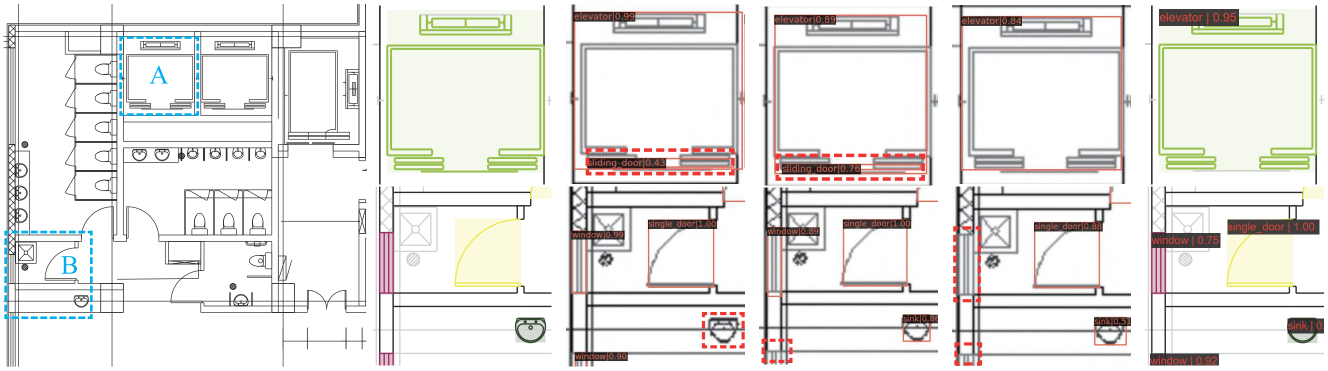


(a) GT (b) FRCNN [31] (c) Ours

Figure 9. Prediction quality comparison. Our primitive-level prediction produces clearer boundary and can exclude background (grey lines) in an instance symbol.

Instance symbol spotting. As reported in [15, 33], traditional symbol spotting algorithms [27, 28, 35] have lower generalization ability and are omitted in the comparison. By rendering CAD drawings into images, our GAT-CADNet is compared with various image detection methods, including the two stage Faster-RCNN [31], the one stage YOLOv3 [30] and the more recent FCOS [38]. Note that the instance head in PanCADNet [15] is from Faster-RCNN and is not listed here.

The image based detection methods [30, 31, 38] predict bounding boxes directly, while we predict instance labels for each geometric primitive. For a fair comparison, we compute the bounding box of each instance symbol and use its averaged connection intensity as the confidence score. Quantitative comparison are listed in Tab. 2 and our GAT-CADNet surpasses other methods by a large margin.



(a) Input CAD drawing (b) GT (c) FRCNN [31] (d) YOLOv3 [30] (e) FCOS [38] (f) Ours

Figure 10. Instance symbol spotting comparison with image detection methods. (a) The input CAD drawing with close-ups of regions in blue rectangles. Two close-ups of region A (upper row) and B (lower row) are listed from (c) to (f). Wrong predictions are marked by red rectangle with dash lines. Faster R-CNN [31] and YOLOv3 [30] mistakenly recognize two more sliding doors in region A. Both YOLOv3 [30] and FCOS [38] fail to recognize some windows at bottom left in region B. Compared to the image based methods, our GAT-CADNet gives closer bounding boxes to ground truth.

Methods	AP50	AP75	mAP
Faster R-CNN [31]	0.693	0.631	0.568
YOLOv3 [30]	0.656	0.431	0.395
FCOS [38]	0.648	0.572	0.525
Ours	**0.735**	**0.680**	**0.690**

Table 2. Comparison on instance symbol spotting with typical image detection methods.

One thing noteworthy is that our average precision (AP) does not drop dramatically when increasing the IoU threshold and has a much higher mAP score. Since CNNs rely on local patch texture for recognition and may ignore features at border, it is not a surprise that their box predictions are less accurate due to the low texture in CAD drawings. Such phenomenon can be observed in Figs. 9 and 10 where our primitive-level prediction has clearer bounding boxes.

Panoptic symbol spotting. Converting CAD drawings into images and applying panoptic segmentation algorithms on them is a straightforward approach. However, as demonstrated in the aforementioned comparison sections, the image based methods are less capable of recognizing abstract symbol at geometric primitive level. PanCADNet [15] provides a CNN-GCN architecture for the panoptic symbol spotting. It constructs a graph on the CAD drawing first, then fetches CNN multi-layer features to each vertex and uses a simple GCN structure for recognition. Since PanCADNet [15] adopts Faster-RCNN as its backbone and detection head, there is no surprise that it has much lower recognition quality than our model, second and last row in Tab. 3. In addition, it does not encode inter-vertex relation explicitly and even has lower recognition and segmentation than our baseline model, third row in Tab. 3.

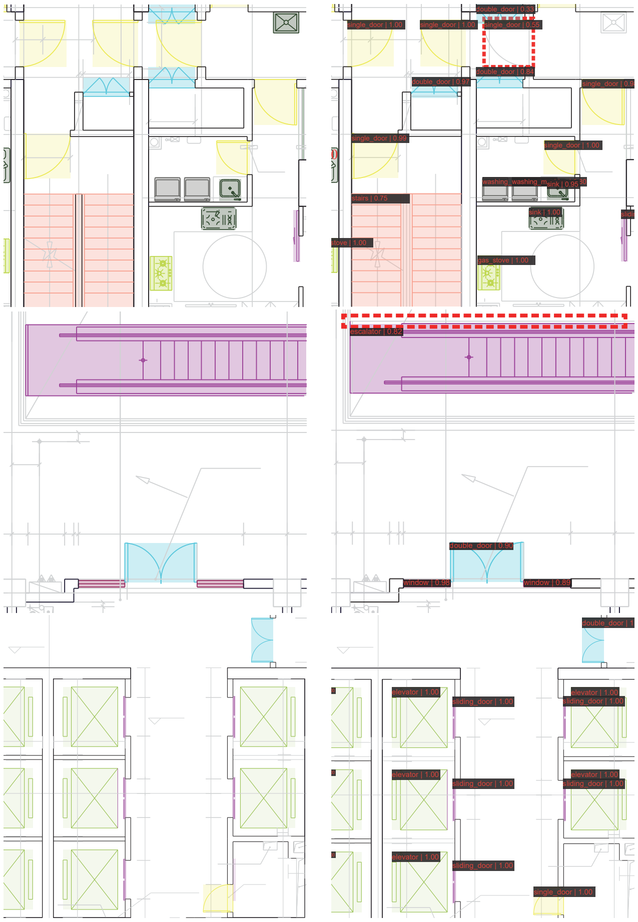
Model	RSE	CEE	RQ	SQ	PQ
PanCADNet [15]	-	-	0.660	0.838	0.553
baseline			0.687	0.875	0.602
b. + RSE	✓		0.734	0.891	0.654
b. + CEE		✓	0.749	0.896	0.671
	✓	2nd	0.761	0.903	0.687
	✓	4th	0.768	0.903	0.694
	✓	6th	0.768	0.904	0.695
	✓	8th	0.786	0.908	0.714
Ours	✓	✓	**0.807**	**0.914**	**0.737**

Table 3. Ablation study of different network configurations. Numbers in the CEE column represent the nth GAT stage.

4.2. Ablation study

Various controlled experiments are conducted to verify specific design decisions in our GAT-CADNet architecture. Discussion about initial geometric feature selection and the number of GAT stages are also included.

The RSE module. The baseline architecture of our model is the multi-stage GAT branch in the middle of Fig. 6. Following the black arrows in Fig. 6, it takes initial vertex and edge features and maps to the semantic and instance heads. The blue branch in Fig. 6 is the RSE module that attaches relative spatial relation to the vertex attention in every GAT stage. Adding the RSE module to the baseline shows clear improvement in both recognition and segmentation quality by 4 and 5 percentage points respectively, as shown in the third row in Tab. 3. It is evident that the explicitly encoded primitive spacial relations, *e.g.* parallelism and orthogonality, enhances vertex attention and thus yields better performance in the panoptic recognition.



(a) GT (b) Panoptic prediction.

Figure 11. Visual results of our network on various scenes. Missing symbols are highlighted with rectangles of red dash lines. For more results, please refer to the supplementary materials.

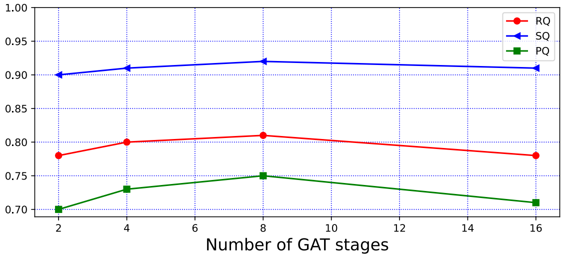


Figure 12. Evaluation on different numbers of GAT stages.

The CEE module. Our CEE module is the orange branch in Fig. 6, which views attention among vertices as affinity in feature space and cascades them to predict instance adjacency matrix. Adding the CEE module to the baseline boosts the RQ metric up to 6 percentage points as shown in the fifth row in Tab. 3. It proves that the CEE module is able to gather connections between vertices effectively and assist in collecting primitives of the same instance. If we add both RSE and CEE modules to the baseline, our method achieves state-of-the-art performance, which exceeds PanCADNet [15] in RQ, SQ and PQ metrics by 14.7, 7.6 and 18.4 percentage points respectively.

To further verify the cascaded structure in CEE, we take attention score from only one GAT stage and test their performance. Specifically, the attention in the 2nd, 4th, 6th and 8th GAT stage are fed to the instance head separately. Statistics listed in Tab. 3 (sixth to eighth row) show steady improvement in the RQ metric, indicating the higher level information is gathered form deeper GAT stage. Our cascaded structure is able to merge multi-stage local and global features for instance symbol spotting.

Edge regularity features. Theoretically, the parallel and orthogonal indicators in Eq. (6) are redundant if we have the angle between two vertices. However, if we drop the regularity term in the initial edge features, the RQ, SQ and PQ metrics decrease to 0.58, 0.85 and 0.49 respectively. This suggests that the regularities in CAD drawings are essential to recognizing symbols and our extra geometric regularity properties help the network to find a better solution.

Number of GAT stages. We also test the effect on different number of GAT stages. The number of GAT stage is configured from 2 to 16 and the results are plotted in Fig. 12. As the number of stages increases, the performance gets better. However, if the number of stages reaches to 16, our network does not benefit from it.

5. Conclusion

In this work we present an intuitive yet effective architecture named GAT-CADNet for panoptic symbol spotting on CAD drawings. It formulates the instance symbol spotting task as an adjacency matrix prediction problem. The relative spatial encoding module explicitly encodes the relative relation among vertices to enhance their attention. The cascaded edge encoding module extracts vertex attentions from multiple GAT stages capturing both local and global connectivity information. With the help of the RSE and CEE modules, our GAT-CADNet surpasses other approaches by a large margin.

Limitation and future work. It is undeniable that our method is still far from perfection, and the panoptic symbol spotting remains an open problem. One shortcoming of our network is that it can only process drawings with a limited number of primitives, otherwise it will suffer from GPU memory shortage. A possible solution is cutting the drawing into smaller blocks and fuse the results. We will keep exploring more efficient networks to alleviate such issue.

References

[1] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3286–3295, 2019. 3

[2] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 2

[3] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 6

[4] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 2, 6

[5] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 2, 5, 6

[6] Yifeng Chen, Guangchen Lin, Songyuan Li, Omar Bourahla, Yiming Wu, Fangfang Wang, Junyi Feng, Mingliang Xu, and Xi Li. Banet: Bidirectional aggregation network with occlusion handling for panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3793–3802, 2020. 3

[7] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 6

[8] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, 2019. 3

[9] Mathieu Delalandre, Ernest Valveny, Tony Pridmore, and Dimosthenis Karatzas. Generation of synthetic documents for performance evaluation of symbol recognition & spotting systems. *International Journal on Document Analysis and Recognition (IJDAR)*, 13(3):187–207, 2010. 2, 5

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 2

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020. 3

[12] Anjan Dutta, Josep Lladós, Horst Bunke, and Umapada Pal. Near convex region adjacency graph and approximate neighborhood string matching for symbol spotting in graphical documents. In *2013 12th International Conference on Document Analysis and Recognition*, pages 1078–1082. IEEE, 2013. 2

[13] Anjan Dutta, Josep Lladós, and Umapada Pal. Symbol spotting in line drawings through graph paths hashing. In *2011 International Conference on Document Analysis and Recognition*, pages 982–986. IEEE, 2011. 2

[14] Anjan Dutta, Josep Lladós, and Umapada Pal. A symbol spotting approach in graphical documents by hashing serialized graphs. *Pattern Recognition*, 46(3):752–768, 2013. 2

[15] Zhiwen Fan, Lingjie Zhu, Honghua Li, Xiaohao Chen, Siyu Zhu, and Ping Tan. Floorplancad: A large-scale cad drawing dataset for panoptic symbol spotting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 2, 3, 4, 5, 6, 7, 8

[16] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9785–9795, 2019. 3

[17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 2

[18] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3464–3473, 2019. 3

[19] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019. 3

[20] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 3

[21] Thomas N Kipf and Max Welling. Variational graph autoencoders. *arXiv preprint arXiv:1611.07308*, 2016. 3

[22] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019. 3

[23] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9404–9413, 2019. 2, 3, 5

[24] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 2

[25] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 2

[26] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In

European conference on computer vision, pages 740–755. Springer, 2014. 2

[27] Thi-Oanh Nguyen, Salvatore Tabbone, and Alain Boucher. A symbol spotting approach based on the vector model and a visual vocabulary. In *2009 10th International Conference on Document Analysis and Recognition*, pages 708–712. IEEE, 2009. 2, 6

[28] Thi Oanh Nguyen, Salvatore Tabbone, and O Ramos Terrades. Symbol descriptor based on shape context and vector model of information retrieval. In *2008 The Eighth IAPR International Workshop on Document Analysis Systems*, pages 191–197. IEEE, 2008. 2, 6

[29] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. *Advances in Neural Information Processing Systems*, 32, 2019. 3

[30] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 5, 6, 7

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016. 5, 6, 7

[32] Alireza Rezvanifar, Melissa Cote, and Alexandra Branzan Albu. Symbol spotting for architectural drawings: state-of-the-art and new industry-driven developments. *IPSJ Transactions on Computer Vision and Applications*, 11(1):2, 2019. 2

[33] Alireza Rezvanifar, Melissa Cote, and Alexandra Branzan Albu. Symbol spotting on digital architectural floor plans using a deep learning-based framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 568–569, 2020. 2, 6

[34] Marçal Rusiñol, Agnés Borràs, and Josep Lladós. Relational indexing of vectorial primitives for symbol spotting in line-drawing images. *Pattern Recognition Letters*, 31(3):188–201, 2010. 2, 5

[35] Marçal Rusiñol, Josep Lladós, and Gemma Sánchez. Symbol spotting in vectorized technical drawings through a lookup table of region strings. *Pattern Analysis and Applications*, 13(3):321–331, 2010. 2, 6

[36] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 2

[37] KC Santosh. Document image analysis: Current trends and challenges in graphics recognition. 2018. 2

[38] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9627–9636, 2019. 5, 6, 7

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 3, 4, 5

[40] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 2, 5, 6

[41] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *ECCV*, 2018. 3

[42] Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. In *International Conference on Learning Representations*, 2018. 3

[43] Yangxin Wu, Gengwei Zhang, Yiming Gao, Xiajun Deng, Ke Gong, Xiaodan Liang, and Liang Lin. Bidirectional graph reasoning network for panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9080–9089, 2020. 3

[44] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8818–8826, 2019. 3

[45] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019. 3

[46] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 4805–4815, 2018. 3

[47] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 3

[48] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10076–10085, 2020. 3

[49] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021. 4, 5