CADTransformer: Panoptic Symbol Spotting Transformer for CAD Drawings

Zhiwen Fan[1], Tianlong Chen[1], Peihao Wang[1], Zhangyang Wang[1]
[1]The University of Texas at Austin
{zhiwenfan,tianlong.chen,peihaowang,atlaswang}@utexas.edu

Abstract

Understanding 2D computer-aided design (CAD) drawings plays a crucial role for creating 3D prototypes in architecture, engineering and construction (AEC) industries. The task of automated panoptic symbol spotting, i.e., to spot and parse both countable object instances (windows, doors, tables, etc.) and uncountable stuff (wall, railing, etc.) from CAD drawings, has recently drawn interests from the computer vision community. Unfortunately, the highly irregular ordering and orientations set major roadblocks for this task. Existing methods, based on convolutional neural networks (CNNs) and/or graph neural networks (GNNs), regress instance bounding boxes in the pixel domain and then convert the predictions into symbols. In this paper, we present a novel framework named ***CADTransformer****, that can painlessly modify existing vision transformer (ViT) backbones to tackle the above limitations for the panoptic symbol spotting task. CADTransformer tokenizes directly from the set of graphical primitives in CAD drawings, and correspondingly optimizes line-grained semantic and instance symbol spotting altogether by a pair of prediction heads. The backbone is further enhanced with a few plug-and-play modifications, including a neighborhood aware self-attention, hierarchical feature aggregation, and graphic entity position encoding, to bake in the structure prior while optimizing the efficiency. Besides, a new data augmentation method, termed Random Layer, is proposed by the layer-wise separation and recombination of a CAD drawing. Overall, CADTransformer significantly boosts the previous state-of-the-art from 0.595 to 0.685 in the panoptic quality (PQ) metric, on the recently released FloorPlanCAD dataset. We further demonstrate that our model can spot symbols with irregular shapes and arbitrary orientations. Our codes are available in* `https://github.com/VITA-Group/CADTransformer`.

1. Introduction

1.1. A Primer for CAD Panoptic Symbol Spotting

Symbol spotting [42, 43, 45, 47] refers to a particular application of pattern recognition, in which symbols with domain-specific semantics are localized and recognized to predefined symbol types. Symbols with simple line segment groups with an engineering, electronics or architectural flair, which constitute some examples of symbols. Therefore, symbol spotting plays a crucial role for document image analysis community [42] and architecture, engineering and construction (AEC) industries [17]. In architecture, a 2D computer-aided design (CAD) drawing typically contains accurate geometric and rich semantic information of a cross-section of a 3D design [17]. With the perception of such CAD drawings, 3D prototypes and the according 3D model can be efficiently and precisely reconstructed.

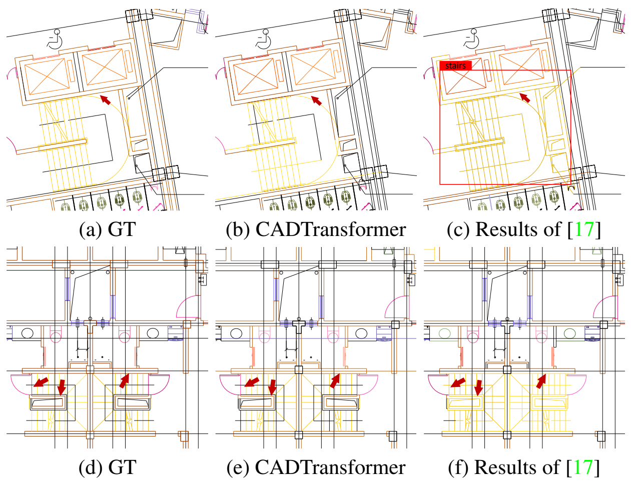

Figure 1. CADTransformer makes predictions on graphic entities, without converting the predicted 2D bounding boxes on pixel images to the label on graphical entities. (a) and (d): Panoptic annotations. (b) and (e): Results from the proposed CADTransformer. (c) and (f): Results from PanCADNet [17]. Red arrows indicate spotting results of adjacent symbols, bounding box in (c) shows one of predicted boxes of [17] for stair symbol.

However, unlike images which are arranged on regular pixel grids, a CAD drawing is composed of graphical primitives (*e.g.*, arc, circle, polyline, etc.). It is non-trivial to spot each symbol (set of *graphical primitives*) in CAD drawing due to the presence of occlusion, cluster, appearance variations, and large unbalanced distribution of the categories. Traditional symbol spotting methods are typically carried out as query-by-example [31, 32, 45], and are impractical for real-world datasets since they can not cope with the tremendous graphical notation variability [17, 43] caused by the producer. Recent learning-based symbol spotting methods [19, 43] proposed to apply convolutional neural networks (CNNs) to real-world symbol spotting datasets. However, they formulate symbol spotting as symbol detection [19, 41, 43, 65] and simply treat vector CAD data as pixel images, leaving the gap between images and vector graphics and leading to inaccurate predictions for real-world applications.

The latest work [17] proposed a large-scale FloorPlan-CAD dataset from industry with annotations for graphical entities (visualization on several CAD drawing examples from the dataset, with irregular shapes and slanting orientation, can be seen in Figure 1.). Similar to the *panoptic segmentation* task [7] which integrates instance and semantic segmentation as one visual recognition task, the *panoptic symbol spotting* task was formulated to unify the spotting of countable instances ($e.g.$, a single door, a window, a table, etc.) and the recognition of uncountable stuff ($e.g.$, wall, railing, etc.) in [17] . The authors [17] address the panoptic symbol spotting task by introducing Graph Convolutional Networks (GCNs) [26] to reason the stuff semantics. Parallel to the GCN head, another CNN-based detection head also predicts the 2D box information of each countable instance. However, the CAD graphical primitives come with irregular ordering, arbitrary scales and orientations, that remain to challenge the standard CNNs. Moreover, the GCN module requires a pre-specified graph topology for information propagation; [17] uses an ad-hoc graph manually crafted by rules, but that might be inaccurate and subject to structural noise.

1.2. Why Transformer, and Our Contributions

Transformers [10, 12, 23, 40, 51, 54, 55, 59, 61, 64] reason global relationships across tokens without pre-defined graph connectivity, by instead learning with self-attention. That makes transformer a promising replacement for GCN to tackle the panoptic symbol spotting task. However, a standard transformer is not immediately ready for this task due to the following challenges: **1). Tokenization and position encoding of graphical symbols.** The standard Vision Transformers (ViT) [13] splits each image into 14×14 or 16×16 patches ($a.k.a.$, tokens) with fixed length over entire dataset. But line segments, as the highly structured minimum units in CAD drawings, are an unordered set of vectors represented in the continuous coordinate space, which differ drastically from raster images. **2). Immense set of primitives in certain scenes.** ViT conducts global self-attention among tokens which leads to quadratic complexity with respect to the token number. That becomes intractable for processing CAD drawings, whose maximum primitive number can be explosively high, $e.g.$, up to 5×10^4. The number of primitives also varies a lot across drawings. **3). Training data limitation.** ViTs are freed from the inherent inductive biases to CNNs. While that leads to more flexibility, it also makes the ViT training particularly data-hungry [13]. Although the latest dataset [17] contains over 10,000 floor plans, it is still unclear whether that suffices for training ViTs to generalize.

In view of those roadblocks, we propose the first transformer-based framework for panoptic symbol spotting, called **CADTransformer**. CADTransformer is designed to be **a general framework** that can be painlessly plugged into existing ViT backbones. It is well motivated since transformers can reason the hidden relationships among graphical primitives without any handcrafted graph topology.

Firstly, in contrary to the common token embeddings from image pixel patches, CADTransformer tokenizes directly from the set of graphical primitives in CAD drawings. Correspondingly, to optimize line-grained semantic and instance predictions, we design a semantic head that predicts the categories of graphical primitives, in parallel with another offset head to shift to their respective ground-truth instance centroids. Further, to enhance the efficiency-accuracy trade-off, CADTransformer embraces a few "plug-and-play" improvements over the standard transformer backbone, including injecting neighborhood awareness to self-attention to bake-in the structure drawing prior; and hierarchical feature aggregation, along with graphic entity position encoding. Additionally, we introduce a novel *Random Layer* data augmentation approach, which leverages CAD domain knowledge to augment new drawings.

We perform the panoptic symbol spotting via CAD-Transformer, by plugging our proposed design into two ViT backbones: ViT [13] and Point Transformer [62]. CAD-Transformer significantly boosts the previous state-of-the-art from 0.595 to 0.689 in the panoptic quality (PQ) metric, on the FloorPlanCAD dataset. We also report a comprehensive set of ablation studies to demonstrate the benefit of each proposed design component.

2. Related Works

2.1. CAD Panoptic Symbol Spotting

Traditional Symbol Spotting Symbol spotting task [42, 45, 47] refers to the retrieval and recognition of symbols from images or documents. Classical symbol spotting algorithms can be roughly categorized as pixel-based methods [31, 32, 45] where algorithms make use of statistical properties of pixels, and vector-based methods [14–16] where structural properties of symbol primitives are considered by the proposed methods. However, the retrieval and recognition of symbols remain challenging in real-world cases as these handcrafted symbol descriptors cannot cope

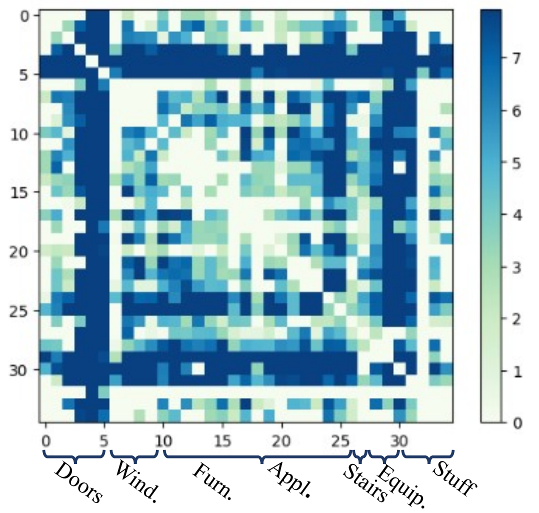

Figure 2. We visualize the minimum L_2 distance between symbols of all categories. Nearly all symbols are closely located with at least one symbol of different kinds, which means inaccurate predicted bounding boxes on pixel images degrade the accuracy of panoptic symbol spotting task. Note that, the captions in axes indicate category ID and the belonging super-class, visualized values are measured by pixel and are normalized via $log_{10}(1+L_2(\cdot))$

with the graphical notation variability of all kinds [43].

Recently, deep models have been adapted to the symbol spotting field where the popular detection models like Faster-RCNN [41] or YOLO [48] are used to detect the bounding box of symbols of interest.

Panoptic Symbol Spotting Although impressive accuracy is obtained by the power of neural networks, these methods mainly focus on synthetical vectorized graphic documents [11, 44] or collected images dataset from the internet [19]. Therefore, these methods are unable to tackle symbols that are closely positioned or with semantics of uncountable stuff ($e.g.$, wall and rails) which play a main role in the architecture, engineering and construction industries. A real-world large-scale floor plan CAD dataset [17] is recently published, containing residential buildings, schools, hospitals, and large shopping malls with complicated structures. Up to 35 object classes of interest, including 30 countable thing classes and 5 uncountable stuff classes are listed and labeled with line-grained annotations.

Panoptic Symbol Spotting with Deep Networks FloorplanCAD [17] also proposes a CNN-GCN-based framework for solving panoptic symbol spotting. They integrate CNN features into both a GCN head and a detection head for semantic symbol spotting and instance symbol spotting tasks, respectively. The final predictions for panoptic symbol spotting are fused by: **(1)** projecting lines in the CAD drawing onto the predicted boxes from the detection head to acquire the instance indexes of countable things; **(2)** followed by recognizing the categories of uncountable stuff from the GCN head. Adapting the GCN model into panoptic symbol spotting tasks leads to a significant accuracy improvement, but the detection head for spotting countable object instances struggles with the inaccurate predicted boxes, and imprecise mapping introduced by the extra post-processing step to project each graphic primitive onto the predicted 2D boxes. In addition, their GCN head operates over a manually designed initial topology, composed with sophisticated handcrafted rules [17].

2.2. Transformer and self-attention

Recent advances [28, 34, 51] in natural language processing have demonstrated the attention mechanisms can learn the attentions to *soft-search* relevant inputs that are important to make the prediction [1]. The works of BERT [12] and GPT [3, 38, 39] parallelize the transformer model via multi-head self-attention for efficient training and inference and also lead to superior performance. Inspired by the success of transformer applied in natural language processing field, a growing interests have shown in exploring the use of transformer framework for computer vision tasks, including image generation [5, 35], image classification [5, 6, 9, 13, 21, 33, 50, 54], object detection [2, 4], semantic segmentation [56, 63], point cloud processing [20, 62], 3D reconstruction [27], generative adversarial networks [25] and line segmentation [57]. The emergence of transformers unveils the potential to reason global relationships across tokens without pre-defined graph connectivity [60], by instead learning with self-attention. Vector graphics are essentially sets of 2D vectors with positional information which are particularly suitable to use self-attention mechanism without any handcraft initial features and topology [17].

3. Methodology

3.1. Overview

The general panoptic symbol spotting task can be formulated as a mapping $F_p : e_k \mapsto (l_k, z_k) \in \mathcal{L} \times \mathcal{N}$, where e_k denotes a graphical entity serves as the basic building block of a CAD drawing, l_k and z_k are its semantic label and instance index. For the input vector drawing, we apply a decomposition at first to split it into basic graphical primitives ($e.g.$, arc, circle, or polyline) and convert the drawing into a rasterized image. Token ($a.k.a$ primitive) embeddings are obtained via projecting the each graphical primitive to the extracted 2D feature maps using a pre-trained CNN. Input with the graphical tokens, a standard Vision Transformer with our proposed "plug-and-play" improvements can be applied to reason relationships across graphical tokens. The transformer layers are followed by the *Two-Branch Heads* by which we can optimize the line-grained predictions of vector drawings. Our framework is shown in Figure 3 (a).

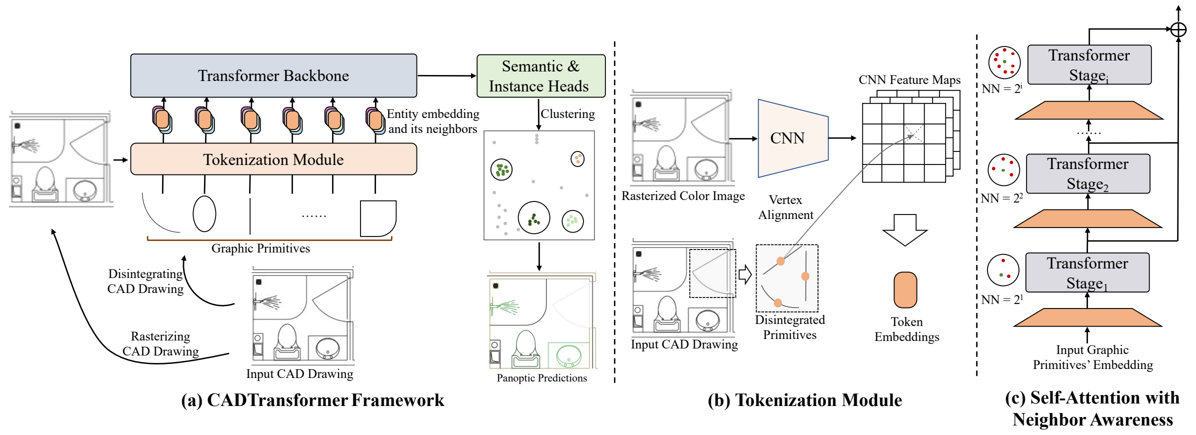

Figure 3. Network architecture of CADTransformer and its components. The input graphic document is decomposed into graphical primitives and also rasterize the document into a color image. Pre-trained CNN is used to extract rich multi-resolution representations from the rasterized image. With the extracted feature maps and isolated graphical primitives, the primitives ($a.k.a.$ token) embeddings are acquired by projecting their middle coordinates onto the feature spaces. A multi-level neighbor-aware ViT takes tokens embeddings as input and jointly optimizes line-grained semantic and instance predictions with the designed *Two-Branch Heads*.

Tokenizing with Graphical Primitives: The vanilla tokens by image patches [13] apparently cannot capture the strong geometric structures of CAD drawings. Note that, besides being treated as a rasterized color image, a CAD drawing could also be painlessly decomposed into its graphical primitives, since the basic building blocks such as arc, circle, and polyline are readily available in CAD drawings.

Leveraging this unique structure information, given a CAD drawing, we employ HRNetV2-W48 [49] pre-trained on ImageNet classification [46] to extract hierarchical image feature maps from the rasterized color image. Next, we decompose the CAD drawing into the graphical primitive domain, project the coordinates of the primitive's middle position onto the image plane, and compute the bilinearly interpolated feature vectors [18, 24]. To be more specific, we upsample and concatenate the feature from four resolution branches of HRNetV2-W48, formulating a $(48+96+192+184)\times H\times W$ feature tensor. After projecting each primitive, linear transformation followed by batch normalization and ReLU are used to reduce the token embedding dimension. By applying *Tokenization Module*, we obtain embeddings $\boldsymbol{f}^{token} \in \mathbb{R}^{N\times C}$ from the input CAD drawing, where N represents the number of graphical primitives and C means the embedding dimension. We draw the workflow of the Tokenization Module in Figure 3 (b).

3.2. Two-Branch Heads: Instance and Semantic Symbol Spotting

To optimize the line-grained predictions, we propose the *Two-Branch Heads*, it takes token with strong geometric embedding from *Transformer Backbone* as input. We will describe Transformer Backbone in section 3.3

Semantic Symbol Spotting Head Applying a MLP on the output feature from aggregated transformer feature vectors $\boldsymbol{f}^{trans}$, it can produce the semantic scores $\boldsymbol{S} = (c_1, ..., c_N) \in \mathbb{R}^{N\times l_{class}}$ for the N entities. Cross-entropy loss L_{sem} is applied to regularize the results between prediction and ground truth labels. The predicted semantic label for entity e is the class with maximum score $i.e.$, argmax(c_i).

Instance Symbol Spotting Head One major limitation of detect-and-spot approaches [17, 19, 43] that is: they rely on the assumption that each graphical entity can be unambiguously encircled by a certain bounding box. However, this assumption is problematic in most cases of CAD drawing as symbols are usually closely located (shown in Figure 2). For example, one side of a single door always overlaps with the wall, which indicates the predicted box is not always accurate enough to split the closely related line-grained symbols.

Instead of predicting a 2D bounding box in pixel images, we propose to predict an offset vector per graphic entity to gather the instance entities around a common instance centroid. To better cluster relevant primitives into the same instance, we apply MLPs to encode token embeddings, producing N offset vectors $\boldsymbol{O} = \{\boldsymbol{o_1}, \boldsymbol{o_2}, ..., \boldsymbol{o_N}\} \in \mathbb{R}^{N\times 2}$. We constrain the learned offset to the belonging instance centroids by applying a L_1 regression loss as

$$L_{reg} = \frac{1}{\sum_i m_i} \sum_i \|\boldsymbol{o_i} - (\boldsymbol{c_i} - \boldsymbol{p_i})\| \cdot m_i \qquad (1)$$

where $m \in \{0,1\}^N$ is a binary value to mask uncountable stuff primitives out, $\boldsymbol{p_i}$ means the 2D coordinates of each graphic primitive of its middle position, $\boldsymbol{c_i}$ indicates the centroid of each instance that i_{th} entity belongs to, $\boldsymbol{c_i} = \frac{1}{N_I}\sum_{i\in I}\boldsymbol{p_i}$, where I indicates indices of each countable thing instance and i is each graphic element belonging to I, N_I counts the number of all elements within each instance symbol. We make the clustering more robust against outliers by modifying a simple and fast clustering algorithm ($e.g.$, the mean-shift algorithm [8]) to a class-wise clustering algorithm with class-specific bandwidths.

3.3. Improving Transformer Backbone Design for Better Spotting

Typical Transformer layers equipped with the proposed improvements can be applied for graphical primitives' embeddings aggregation. Note that, the transformer backbone takes the token embeddings from the Tokenization Module as input, performs self-attention and provides rich representations for the Two-Branch Heads.

Self-Attention with Neighbor Awareness: Typical Vision Transformer architecture [13] consists of several consecutive transformer layers. Each layer contains a self-attention module and a feed-forward-network(FFN). The self-attention is computed using the scaled-dot product:

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{C}}\right)\boldsymbol{V} \quad (2)$$

where $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{N \times C}$ are the query, key and value matrix; N and C indicate the token number and the hidden dimension.

However, the primitive number of a single drawing is up to 5×10^4 in FloorplanCAD dataset [17] and the global attention complexity will be quadratic to the token number, preventing us from applying typical attention mechanism over CAD documents. To overcome this issue, we propose several improvements over vanilla ViT to make it applicable.

Self-attention within k Neighbors We adopt the recent self-attention network [13] in applying the attention operation within its nearest k neighbors [36, 37, 62] of each token. The local neighbor aggregation strategy optimizes each graphic entity using its local features, making it scalable for CAD document of arbitrary size. The minimum distance is determined by checking the start and end points among all entities.

$$\begin{gathered}
\boldsymbol{e}_i = \{\boldsymbol{e}_i^{start} = (x_i^{start}, y_i^{start}), \boldsymbol{e}_i^{end} = (x_i^{end}, y_i^{end})\} \\
D(\boldsymbol{p}_i, \boldsymbol{p}_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \\
D_{min} = \min_{\boldsymbol{p}_i \in \{e_i^{start}, e_i^{end}\}, \boldsymbol{p}_j \in \{e_j^{start}, e_j^{end}\}} |D(\boldsymbol{p}_i, \boldsymbol{p}_j)|
\end{gathered} \quad (3)$$

where e^{start} and e^{end} represent the start and end points of each graphical entity e. i indicates the queried graph entity and j indicates all other entities within a CAD document.

In order to increase the model capacity to *soft-search* [27] the relevant tokens in a more global perspective as well as keep the framework efficient, we propose to enlarge the number of nearest tokens. Specifically, we take the vanilla transformer layers from [13] which allow our improved design to reuse the pre-trained weights. The architecture includes four stages with layer number = $\{2, 2, 6, 2\}$ [29]. Each stage operates on progressively increasing neighborhoods for a larger receptive field and we set the number of neighborhoods as 2^i for each stage where i indicates stage number. The visualization of the design is shown in Figure 3 (c).

Multi-resolution Feature Fusion We fuse features across stages to define a nonlinear local-to-global representation which has been proven effective in dense prediction tasks [22, 29, 30]. We represent the fused embedding as the mean of previous ones from different scales $\boldsymbol{f}^{trans} = \text{mean}(\sum_i \boldsymbol{f}_i^{trans})$ since they have the same embedding dimension. Here, i means the stage number.

Graphic Entity Position Encoding In computing self-attention, we introduce a trainable relative position bias [29, 62] as the 2D coordinates of primitive's middle position are natural candidate for position encoding. We formulate the learned position encoding as $\boldsymbol{b} \in \mathbb{R}^{N \times k \times C}$ for each graphical entity to compute similarity matrix:

$$\begin{gathered}
\boldsymbol{b} = \text{MLP}(\boldsymbol{p_i} - \boldsymbol{p_k}) \\
\text{Attention}(\boldsymbol{Q}, \boldsymbol{K_k}, \boldsymbol{V_k}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K_k}^T}{\sqrt{C}}\right)(\boldsymbol{V_k} + \boldsymbol{b})
\end{gathered} \quad (4)$$

where $\boldsymbol{p_i} \in \mathbb{R}^{N \times 1 \times 2}$ is the query graphical entity coordinate, $\boldsymbol{p_k} \in \mathbb{R}^{N \times k \times 2}$ indicates the k neighbors of $\boldsymbol{p}_i$, MLP represents two linear layers with one ReLU nonlinearity.

3.4. Random Layer Data Augmentation

Given that transformers typically require large datasets, it would be beneficial if CADTransformer is trained with more training data. Commonly used data augmentation approaches ($i.e.$, add noise and blur color images) will disrupt the rasterized image. It is because the rasterization process assigns a fixed value for the coordinate occupied by graphical primitives based on a white canvas; Random perturbing each primitive in vector space breaks the precise topology of its originality. Random dropping primitives may lose key structures for spotting algorithms ($e.g.$, the arc of a single door). Contrary to existing methods, we start from a floor plan drawing consists of multiple layers of different functionality ($e.g.$, furniture, equipment, and appliance) and the annotation of FloorplanCAD dataset [17] is conducted layer-wise. Therefore, layers are the primary method for organizing the graphical symbols and annotations. The availability of "layer" metadata provides extra insight for us, to leverage the domain knowledge and augment the labeled data "for free".

Specifically, we propose the *Random Layer* data augmentation. Given an annotated CAD drawing of multiple layers, we traverse and classify all layers into three categories based on their functionalities ($i.e.$, thing layers, stuff layers and background layers). We select new layers over the three layers groups with certain probabilities

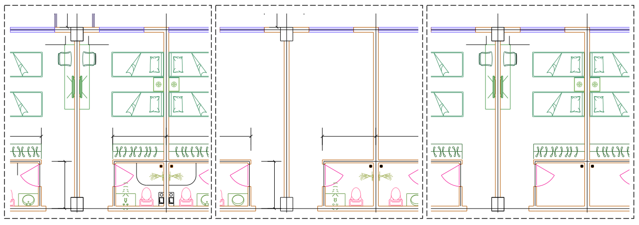

Figure 4. Examples of Random Layer applied on FloorplanCAD dataset [17]. (a): Original CAD drawing with full layers, (b): Randomly chose layers with probabilities of $\{p_{th} = 0.8, p_{st} = 0.8, p_{bg} = 0.5\}$ from (a), (c): Randomly chose layers with the same probabilities of (b) but with a different seed.

$\{p_{th}, p_{st}, p_{bg}\}$. Finally, we re-organize the selected layers to generate a new drawing with its original annotations and topology. A visualization example is provided in Figure 4.

4. Experiments

Implementation Details During training, we use Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The initial learning rate is set to 0.00025, and halved after epoch 20, 40. All models are trained for 40 epochs. The batch size is fixed to 4, and we train our method with 4 NVIDIA RTX A6000 GPUs with 1 training sample on each GPU. Our CADTransformer and previous PanCADNet [17] are evaluated on the first version of FloorPlanCAD dataset [1](released in August 13, 2021). AM-Softmax [52] loss is introduced to replace the original softmax loss to learn large-margin features. Both the Vision Transformer Backbone [13] and HRNet [49] are pre-trained and jointly fine-tuning with the proposed Two-Branch Heads during training process. Our overall objective is written as $L = L_{sem} + \alpha \times L_{reg}$. where α is a scalar value to balance the two-loss items. We set α=0.3 in our experimental setting.

4.1. FloorPlanCAD Datasets

The first real-world, large-scale floor plan CAD drawings datasets with line-wise annotations are proposed in [17]. The proposed datasets are collected from different sources covering residential buildings, schools, hospitals, underground parking, and shopping malls with real structures. The released dataset contains 35 object classes which is enriched compared with the initial version mentioned 30 object classes in [17]. The released version includes 30 countable thing classes, categorized by super-classes including doors, windows, stairs, home appliances, furniture, and equipment. In addition, row chairs, parking spots, wall, curtain wall, and railing are categorized as uncountable stuff classes. For a graphical element that belongs to thing classes, it is annotated with semantic class and instance index while an element that belongs to stuff classes contains solely semantic labeling. Each floor plan drawing is cut into squared blocks with the dimension of 14m × 14m in real-world size. The training, testing, and validation set contains 6965, 3827, and 810 pieces of floor plan drawing, respectively.

[1]https://floorplancad.github.io/

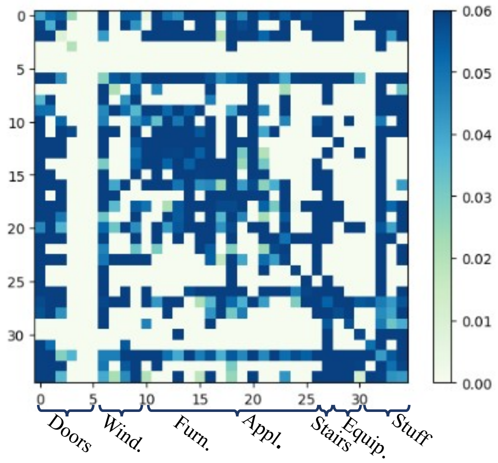

Figure 5. Visualized attention map among all categories. The x and y-axis correspond to different categories and the belonging super classes. Darker color indicates stronger attention. We can observe primitives **1).** with smaller distance, **2).** within the same class or super class, **3).** belongs to wall class (semantic Id: 33), usually have strong attention with other categories.

4.2. Evaluation Metrics

In the setting of panoptic symbol spotting, a generalized symbol is a set of graphical entities (primitives), representing either a thing instance (*e.g.*, a single door or a toilet) or particular stuff (*e.g.*, wall, railing). Therefore, we follow previous paper [17] to denote a graphical entity (*e.g.*, arc, circle or polyline) $e = (l, z)$ by a semantic label l and an instance index z. A symbol is represented by a collection of entities and is defined as $s = \{e_{i \in J} \mid l = l_i, z = z_i\}$, where J is set of primitives. The panoptic symbol spotting task requires a map $F_p : e_k \mapsto (l_k, z_k) \in \mathcal{L} \times \mathcal{N}$,
Following the definition of *Panotic Segmentation* [7], the *Panoptic Quality* metric in symbol spotting is defined as:

$$\begin{aligned} PQ = RQ \times SQ &= \frac{\sum_{(s_p,s_g)\in TP} \mathrm{IoU}(s_p,s_g)}{|TP|+\frac{1}{2}|FP|+\frac{1}{2}|FN|}. \\ RQ &= \frac{|TP|}{|TP|+\frac{1}{2}|FP|+\frac{1}{2}|FN|}. \\ SQ &= \frac{\sum_{(s_p,s_g)\in TP} \mathrm{IoU}(s_p,s_g)}{|TP|}. \end{aligned} \tag{5}$$

where the metric of RQ can be regarded as F_1 score to measure the symbol matching ability, SQ represents the averaging IoUs of matched symbols. The predicted symbol $s_p = (l_p, z_p)$ is considered as true positive (TP) if it can be matched to a ground truth symbol $s_g = (l_g, z_g)$, otherwise false positives (FP). Instead, symbol is false negatives

Models	Total			Countable Thing Classes				Uncountable Stuff Classes		
	PQ	SQ	RQ	PQ^{Th}	SQ^{Th}	RQ^{Th}	mAP	PQ^{St}	SQ^{St}	RQ^{St}
PanCADNet [17]	0.5953	0.8258	0.6693	0.6557	0.8614	0.7611	0.5630	**0.5872**	0.813	**0.7222**
CADTransormer	0.6732	0.8754	0.7226	0.7713	0.9355	0.8245	-	0.5793	0.8102	0.7151
CADTransormer + RL	**0.6894**	**0.8832**	**0.7333**	**0.7849**	**0.9404**	**0.8346**	-	0.5855	**0.8188**	0.7151

Table 1. Panoptic symbol spotting results on FloorplanCAD dataset [17]. **Top row:** the CNN-GCN-based model proposed in [17] with the claimed optimal setting. **Second row** and **last row:** our proposed CADTransformer without and with Random Layer (RL) data augmentation. Note that, mAP is used for instance symbol spotting branch evaluation in PanCADNet [17]

.

(FN) if it misses ground truth to match. A certain predicted symbol is considered as matched if it finds a ground truth symbol, with $l_p = l_g$ and $\mathrm{IoU}(s_p, s_g) > 0.5$, where the intersection over union (IoU) between predicted and ground truth symbols are computed based on:

$$\mathrm{IoU}(s_p, s_g) = \frac{\Sigma_{e_i \in s_p \cap s_g} \log(1 + L(e_i))}{\Sigma_{e_j \in s_p \cup s_g} \log(1 + L(e_j))}. \tag{6}$$

4.3. Main Results on Panoptic Symbol Spotting

We mainly compare with the previous method in [17] which is the first framework designed for panoptic symbol spotting task. We re-implemented the method since the source code is not available. To validate the effectiveness of our CADTransformer architecture and the benefits brought by Random Layer data augmentation, we apply HRNetV2-W48 [53] as our CNN for tokenization process which is pre-trained on 1000 class image classification task of ImageNet [46]. The $\{p_{th}, p_{st}, p_{bg}\}$ in Random Layer data augmentation set as 0.8, 0.8 and 0.5, respectively. The quantitative results on FloorPlanCAD testing set [17] are shown in Table 1. We can see that CADTransformer improves from previous state-of-the-art 0.5953 to **0.6732**. When more training samples are generated using Random Layer, the performance is further boosted to **0.6894**. The qualitative results are shown in Figure 6, we can see CADTransformer generates better results for the adjacent primitives belonging to different symbols. The fact that CADTransformer works well demonstrates that **1).** A standard vision transformer backbone equipped with proposed improvements demonstrates its potential for panoptic symbol spotting task. **2).** It benefits the predictions on vector data without projecting pixel predictions, *i.e.*, bounding boxes, to the label on graphical entities. **3).** A stronger transformer model can be obtained by enriching the diversity of floor plan drawings.

4.4. Ablation Study

Study the Effect of Nearest Neighbors' Numbers We have conducted experiments on the *Self-attention within k Neighbors* module to support our design: we try different neighborhoods k in each stage i. As shown in Table 2, the accuracy increases with a larger k, and gets saturated when k comes to [16,16,16,16]. Our multi-scale setting achieves similar accuracy while reducing ∼30% FLOPs.

Model setting	PQ	SQ	RQ	FLOPs
$k_1=2, k_2=2, k_3=2, k_4=2$	0.6722	0.8798	0.7185	**24.791G**
$k_1=4, k_2=4, k_3=4, k_4=4$	0.6796	0.8809	0.7272	28.348G
$k_1=8, k_2=8, k_3=8, k_4=8$	0.6865	0.8817	0.7322	35.463G
$k_1=16, k_2=16, k_3=16, k_4=16$	**0.6897**	**0.8849**	0.7314	49.693G
$k_1=2, k_2=4, k_3=8, k_4=16$ (Ours)	0.6894	0.8832	**0.7333**	34.870G

Table 2. Analysis on the effect of attention neighborhood k, where i is the stage of the model, k is attention neighborhoods.

Methods	PQ	SQ	RQ
CADTransformer-ViT [13]	**0.6732**	0.8754	**0.7226**
CADTransformer-PointT [62]	0.6678	**0.8786**	0.7117

Table 3. The proposed transformer design applied to both ViT [13] and Point Transformer [62] for panoptic symbol spotting task.

Visualizations To understand the learned attention map over classes, we visualize the attention maps from the last layer of CADTransformer-ViT, by sampling all floor plan drawings from the test set. The visualization is shown in Figure 5 where each grid indicates the intensity of self-attention. It can be observed that the neighbor-aware self-attention learns the relationship among classes that are aligned well with the real-world layout (*e.g.*, wall usually distributed throughout the drawing and most elements are wall-mounted or closed to the wall).

Agnostic Transformer Design The proposed set of designs, including tokenization, neighbor aware self-attention module, hierarchical feature fusion as well as graphic entity position encoding, is agnostic to the choice of transformer backbones for panoptic symbol spotting. To verify the effectiveness, we introduce the representative vector-based attention, *i.e.*, Point Transformer Block [62], equipped with our designs which is formulated as:

$$\textstyle\sum_{j \in k} \mathrm{Softmax}(\mathrm{MLP}_1(\varphi(\boldsymbol{t_i}) - \psi(\boldsymbol{t_j}))) \odot (\alpha(\boldsymbol{t_j}) + \boldsymbol{b})$$
$$\boldsymbol{b} = \mathrm{MLP}_2(\boldsymbol{p_i} - \boldsymbol{p_k})$$

where $\boldsymbol{t_i}$ and $\boldsymbol{t_j}$ are the queried token and its k neighboring token feature, respectively. φ, ψ and α are pointwise feature transformation. $\boldsymbol{b}$ is graphical position encoding. $\boldsymbol{p_i}$ and $\boldsymbol{p_k}$ are the queried entity coordinate and its k neighbors. Here, we adopt eight vanilla Point Transformer layers without transition down and transition up operations as CAD drawings may contain extremely sparse segments. We formulate them into four stages (2 layers for each stage), set k as 2, 4, 8, and 16 for each stage. The feature fusion is conducted by averaging the output features from the output

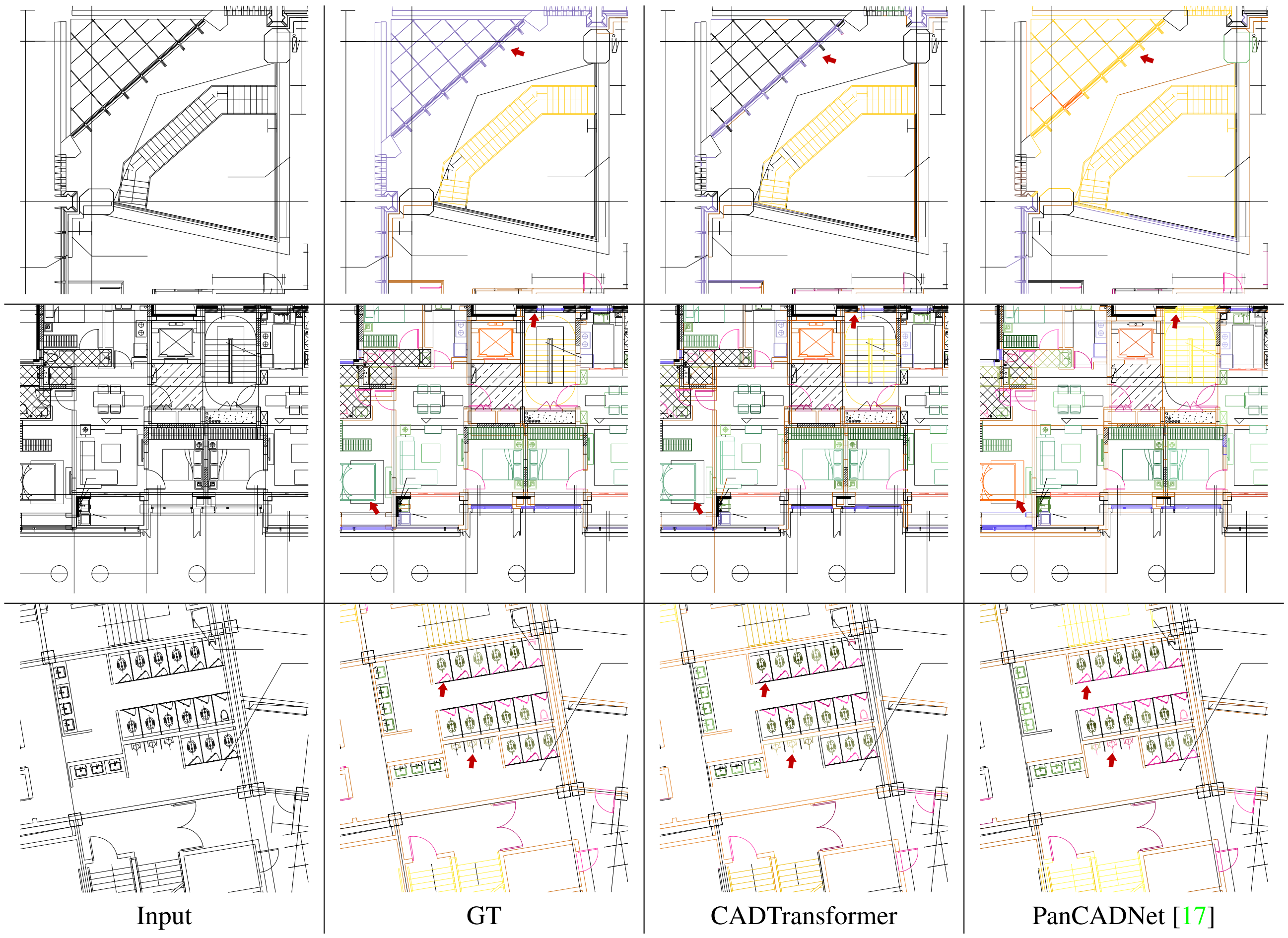

Figure 6. Qualitative comparisons between CADTransformer and CNN-GCN based method [17] on FloorPlanCAD dataset. Red arrows indicate regions of adjacent symbols. CADTransformer distinguishes segments that belongs to adjacent symbols and is applicable to CAD data with arbitrary shapes and orientations. Best view in color and zoom in. See the appendix for annotation details.

of the four stages. We demonstrate the results in Table 3, by applying the proposed techniques into ViT [13] and Point Transformer [62]. Both of them achieve promising accuracy, validating our module design for existing transformers in addressing the panoptic symbol spotting task.

Random Layer Data Augmentation As shown in Table 4, when implementing Random Layer in training CADTransformer, Random Layer consistently improves the panoptic quality metric. Specifically, when we add additional 25% to 50% training samples, the PQ improves from 0.6732 to 0.6894 without any cost during testing. The experimental results indicate that such a simple setup, without bells and whistles, benefits the transformer model and achieves better accuracy.

5. Conclusion and Broad Impact

We present the first transformer-based framework for the panoptic symbol spotting task. By tokenizing graphical primitives, injecting standard vision transformers with neighbor-aware self-attention module, hierarchical feature aggregation as well as graphic entity position encoding, we can easily apply a standard vision transformer with vector CAD drawings as its input. We design two-branch heads that perform semantic and instance predictions directly on vector data which significantly advances the state-of-the-art performance on the FloorPlanCAD dataset. Random Layer, an intuitive data augmentation scheme, consistently improves the spotting performance without additional labor cost. One limitation is that we only perform studies on floor plan drawings. We will extend the investigation to other graphical data recognition tasks (e.g. semantic sketch segmentation [58]). We do not see that this work will directly impose any negative social risk.

Methods	Train Samples	PQ	SQ	RQ
Baseline	6,965	0.6732	0.8754	0.7226
Aug. 0.25 ×	8,796	0.6862	0.8772	**0.7335**
Aug. 0.50 ×	10,448	**0.6894**	**0.8832**	0.7333

Table 4. Comparisons on Random Layer data augmentation settings. Augmenting training samples during training improves model accuracy. Evaluations are on the FloorPlanCAD test set.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 3

[2] Josh Beal, Eric Kim, Eric Tzeng, Dong Huk Park, Andrew Zhai, and Dmitry Kislyuk. Toward transformer-based object detection. *arXiv preprint arXiv:2012.09958*, 2020. 3

[3] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. 3

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 3

[5] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International Conference on Machine Learning*, pages 1691–1703. PMLR, 2020. 3

[6] Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. *Advances in Neural Information Processing Systems*, 34, 2021. 3

[7] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab. *arXiv preprint arXiv:1910.04751*, 2019. 2, 6

[8] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995. 5

[9] Xiangxiang Chu, Bo Zhang, Zhi Tian, Xiaolin Wei, and Huaxia Xia. Do we really need explicit position encodings for vision transformers? *arXiv e-prints*, pages arXiv–2102, 2021. 3

[10] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019. 2

[11] 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. 3

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2, 3

[13] 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. *arXiv preprint arXiv:2010.11929*, 2020. 2, 3, 4, 5, 6, 7, 8

[14] 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

[15] 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

[16] 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

[17] Zhiwen Fan, Lingjie Zhu, Honghua Li, Xiaohao Chen, Siyu Zhu, and Ping Tan. Floorplancad: A large-scale cad drawing dataset for panoptic symbol spotting. *arXiv preprint arXiv:2105.07147*, 2021. 1, 2, 3, 4, 5, 6, 7, 8

[18] 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. 4

[19] Shreya Goyal, Vishesh Mistry, Chiranjoy Chattopadhyay, and Gaurav Bhatnagar. Bridge: building plan repository for image description generation, and evaluation. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1071–1076. IEEE, 2019. 2, 3, 4

[20] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, 2021. 3

[21] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021. 3

[22] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 447–456, 2015. 5

[23] 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. 2

[24] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28:2017–2025, 2015. 4

[25] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two pure transformers can make one strong gan, and that can scale up. *Advances in Neural Information Processing Systems*, 34, 2021. 3

[26] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 2

[27] Kevin Lin, Lijuan Wang, and Zicheng Liu. End-to-end human pose and mesh reconstruction with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1954–1963, 2021. 3, 5

[28] Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133, 2016. 3

[29] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021. 5

[30] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 5

[31] 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

[32] 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

[33] Bowen Pan, Rameswar Panda, Yifan Jiang, Zhangyang Wang, Rogerio Feris, and Aude Oliva. Ia-red ^{2}: Interpretability-aware redundancy reduction for vision transformers. *Advances in Neural Information Processing Systems*, 34, 2021. 3

[34] Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*, 2016. 3

[35] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR, 2018. 3

[36] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 5

[37] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. 5

[38] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018. 3

[39] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 3

[40] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019. 2

[41] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015. 2, 3

[42] 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. 1, 2

[43] 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. 1, 2, 3, 4

[44] 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. 3

[45] 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. 1, 2

[46] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 4, 7

[47] KC Santosh. Document image analysis: Current trends and challenges in graphics recognition. 2018. 1, 2

[48] Mohammad Javad Shafiee, Brendan Chywl, Francis Li, and Alexander Wong. Fast yolo: A fast you only look once system for real-time embedded object detection in video. *arXiv preprint arXiv:1709.05943*, 2017. 3

[49] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5693–5703, 2019. 4, 6

[50] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 3

[51] 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. 2, 3

[52] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu. Additive margin softmax for face verification. *IEEE Signal Processing Letters*, 25(7):926–930, 2018. 6

[53] 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. 7

[54] Peihao Wang, Wenqing Zheng, Tianlong Chen, and Zhangyang Wang. Anti-oversmoothing in deep vision transformers via the fourier domain analysis: From theory to practice. In *International Conference on Learning Representations*, 2022. 2, 3

[55] Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*, 2019. 2

[56] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *arXiv preprint arXiv:2105.15203*, 2021. 3

[57] Yifan Xu, Weijian Xu, David Cheung, and Zhuowen Tu. Line segment detection using transformers without edges. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4257–4266, 2021. 3

[58] Lumin Yang, Jiajie Zhuang, Hongbo Fu, Xiangzhi Wei, Kun Zhou, and Youyi Zheng. Sketchgnn: Semantic sketch segmentation with graph neural networks. *ACM Transactions on Graphics (TOG)*, 40(3):1–13, 2021. 8

[59] 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. 2

[60] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34, 2021. 3

[61] 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. 2

[62] 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. 2, 3, 5, 7, 8

[63] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6881–6890, 2021. 3

[64] Wenqing Zheng, Qiangqiang Guo, Hao Yang, Peihao Wang, and Zhangyang Wang. Delayed propagation transformer: A universal computation engine towards practical control in cyber-physical systems. *Advances in Neural Information Processing Systems*, 34, 2021. 2

[65] Zahra Ziran and Simone Marinai. Object detection in floor plan images. In *IAPR workshop on artificial neural networks in pattern recognition*, pages 383–394. Springer, 2018. 2