CADTransformer: Panoptic Symbol Spotting Transformer for CAD Drawings *(Supplementary Material)*

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

1. Introduction

We provide the implementation details of our neighbor-aware attention layer, the demonstration of class-wise quantitative evaluation, additional qualitative results and the detailed visualization method in supplementary material due to the space limitation in the main draft.

2. Transformer Layer Details

As presented in Figure 2 of the main draft, the neighbor-aware self attention generates the attention results using the k neighbors of each token. We summarize the full algorithm of attention function in Algorithm 5. Here, we first find the nearest k primitives using Equation 4 in the main draft. Then we generate the $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ using a MLP and an indexing function. The relative position encoding is obtained by the relative 2D coordinates of primitives with MLPs. The final results were achieved by attention over the nearest k neighborhoods.

3. Additional Quantitative Evaluations

3.1. Class-wise Evaluation

We provide the detailed evaluation of panoptic quality(PQ), segmentation quality(SQ) and recognition quality(RQ) in Table 2. Here, we provide the class-wise evaluation of CNN-GCN based method PanCADNet [3], the proposed CADTransformer and the CADTransformer with Random Layer data augmentation. Note that, we show the results of CADTransformer using the pre-trained ViT [2] as its Transformer backbone.

3.2. Impacts of Pretrained Models

We have trained CADTransformer using the pre-trained ViT by Moco v3 [1]. As is shown in Table 1, the accuracy of using self-supervised pretraining (SSL) is better than without pretraining but is still below the accuracy using supervised pretraining. It inspires us to potentially pursue a new SSL framework for better representations of graphical primitives.

Methods	PQ	SQ	RQ
ImageNet Pretrained [4]	0.6894	0.8832	0.7333
SSL Pretrained [1]	0.6744	0.8847	0.7106
No Pretraining	0.6552	0.8785	0.6996

Table 1. The proposed transformer with different pre-trained strategies.

4. Qualitative Evaluations

We provide more visualized results of our proposed method and previous PanCADNet [3] in Figure 1 and Figure 2. As can be seen, CADTransformer is more powerful in recognizing closely related symbols and the primitives within incomplete symbols near the border of CAD drawings. To generate a unique color for all instances with the same semantic category, we follow the cocopanopticapi [1], where we generate a random perturbation on the basic color value of each category. The visualization of basic color for each category is in Figure 3.

5. Futhur Works and Limitations

CADTransformer, the first transformer-based framework for panoptic symbol spotting task, can better reason the relationships among graphical primitives without handcrafted graph topology. Our solution can be applied to many other recognition tasks that depend on the similar input form of graphic primitives. For example, semantic sketch segmentation [5,6] aims to predict freehand sketching into semantic groups which can be solved via tokenizing the input stroke and using the plug-and-play k neighbors self-attention.

Although this work demonstrates the potentials of introducing transformer-based model for graphical data perception, some limitations still restrict its application: When the CADTransformer input with an extremely large drawing may consume tremendous amount GPU memory. The future work can focus on solving the memory bottleneck while keeping the accuracy for current models, one can apply a clustering algorithm on the drawing and only several clusters with fixed primitives numbers to be updated in each

[1] https://github.com/cocodataset/panopticapi

layer. Then the training process for a graphical drawing can be scheduled with bounded memory and made it feasible to apply current transformer-based models to arbitrary large drawings as well as other vector data.

```
def Attention(xy, tokens, nn_dist, nn_k, MLP):
    knn_idx = nn_dist[:, :, :nn_k]#find k nearest
    xy_knn = index_points(xy, knn_idx)
    qkv = get_qkv(tokens)
    q_feat, k_feat, v_feat = qkv.unbind(0)
    q = q_feat
    k = index_points(k_feat, knn_idx)
    v = index_points(v_feat, knn_idx)
    pos_enc = MLP(xy - xy_knn)
    attn = torch.sum(q * k, -1)
    attn = F.softmax(attn, dim=-1) # feature dim
    v = v + pos_enc # graphical entity PE
    x = torch.sum(attn * v, -2) # k dim
    return x
```

Listing 1. The Attention module in a PyTorch-like style

Class	Property	CADTransformer+RL			CADTransformer			PanCADNet [3]		
	#Entity($\times 10^3$)	PQ	SQ	RQ	PQ	SQ	RQ	PQ	SQ	RQ
single door	171.30	0.8049	0.9431	0.8535	0.797	0.9396	0.8482	0.5622	0.7945	0.7077
double door	172.20	0.8451	0.9386	0.9004	0.8317	0.9378	0.8868	0.6912	0.8191	0.8438
sliding door	79.22	0.8504	0.949	0.8961	0.8676	0.958	0.9056	0.7987	0.9071	0.8805
folding door	3.27	0.9054	0.9779	0.9259	0.7515	0.9619	0.7812	0.8159	0.9746	0.8372
window	197.08	0.7214	0.9155	0.788	0.7097	0.9168	0.7741	0.5558	0.7839	0.709
bay window	6.01	0.4058	0.945	0.4294	0.5195	0.9296	0.5588	0.0803	0.5645	0.1423
blind window	36.09	0.7436	0.9283	0.8011	0.831	0.9464	0.878	0.5827	0.8377	0.6955
opening symbol	9.01	0.3745	0.9551	0.3922	0.3204	0.9058	0.3537	0.2926	0.8945	0.3271
sofa	44.25	0.8405	0.9616	0.874	0.8469	0.9631	0.8794	0.6746	0.9278	0.7271
bed	568.36	0.7821	0.9201	0.8499	0.7915	0.9118	0.868	0.6654	0.8906	0.7472
chair	251.67	0.8028	0.9553	0.8404	0.7895	0.9465	0.8341	0.7602	0.9653	0.7876
table	64.53	0.7688	0.9606	0.8004	0.7812	0.9483	0.8237	0.6002	0.8631	0.6954
TV cabinet	27.67	0.8572	0.9548	0.8977	0.857	0.9488	0.9032	0.7665	0.9475	0.809
Wardrobe	183.66	0.8361	0.9503	0.8798	0.8384	0.9468	0.8855	0.8201	0.8816	0.9302
cabinet	90.36	0.6627	0.9109	0.7276	0.6963	0.918	0.7584	0.5998	0.8565	0.7002
gas stove	93.16	0.8877	0.9795	0.9063	0.8465	0.974	0.8691	0.9283	0.9916	0.9361
sink	352.96	0.8113	0.9503	0.8538	0.7763	0.9358	0.8295	0.7926	0.9332	0.8494
refrigerator	30.30	0.7778	0.942	0.8257	0.7331	0.9338	0.7851	0.6841	0.8726	0.784
airconditioner	31.44	0.6613	0.9775	0.6765	0.6533	0.9508	0.6871	0.6041	0.9379	0.6441
bath	80.84	0.628	0.928	0.6767	0.5568	0.9267	0.6008	0.3813	0.7785	0.4898
bath tub	43.50	0.8035	0.9329	0.8613	0.7359	0.9049	0.8132	0.5494	0.7713	0.7122
washing machine	68.66	0.7543	0.9529	0.7916	0.6747	0.9288	0.7264	0.7428	0.9213	0.8062
squat toilet	126.21	0.8777	0.9535	0.9205	0.8581	0.9404	0.9125	0.8792	0.9434	0.9319
urinal	94.25	0.8648	0.9542	0.9062	0.7871	0.9343	0.8425	0.8492	0.9595	0.885
toilet	275.67	0.8724	0.9625	0.9064	0.8331	0.9322	0.8936	0.8482	0.9567	0.8865
stairs	222.24	0.7382	0.9105	0.8108	0.7117	0.9084	0.7834	0.5404	0.8084	0.6685
elevator	102.05	0.8635	0.9564	0.9029	0.8655	0.9549	0.9064	0.7477	0.8875	0.8425
escalator	22.14	0.4567	0.84	0.5437	0.4861	0.8251	0.5891	0.3936	0.8153	0.4828
row chairs	540.56	0.5229	0.9678	0.5403	0.4626	0.9667	0.4786	0.4603	0.953	0.483
parking	271.24	0.7177	0.9652	0.7436	0.6366	0.9689	0.657	0.5271	0.9275	0.5683
wall	1362.03	0.6205	0.792	0.7835	0.607	0.7771	0.7811	0.6231	0.7902	0.7886
curtain wall	139.00	0.3808	0.897	0.4245	0.4356	0.9212	0.4729	0.4224	0.9203	0.459
total	5760.93	0.6894	0.8832	0.7333	0.6732	0.8754	0.7226	0.5953	0.8258	0.6693

Table 2. Graphical entities number and quantitative results for each category. We demonstrate the comparisons of panoptic symbol spotting metric between our methods and previous CNN-GCN-based method [3]. Here, RL in the third column indicates the proposed Random Layer data augmentation. Note that, we skip the classes that contain less than $1k$ graphical entities.

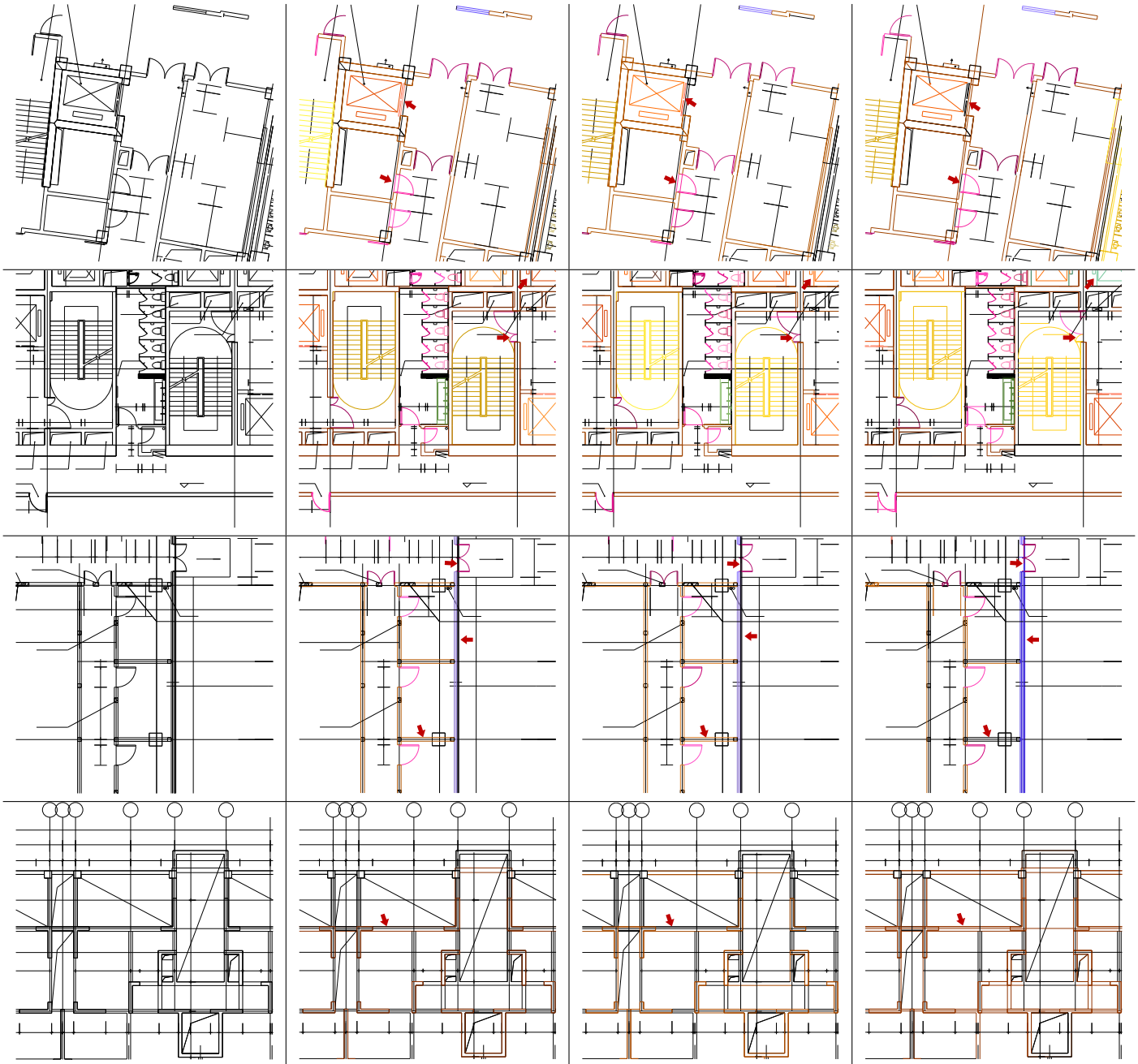

Figure 1. Qualitative comparisons between CADTransformer and GCN-CNN based method [3] on FloorPlanCAD dataset. Red arrows indicate representative graphical primitives. Visualization colors of each category can be found in Figure 3. Best view in color and zoom in.

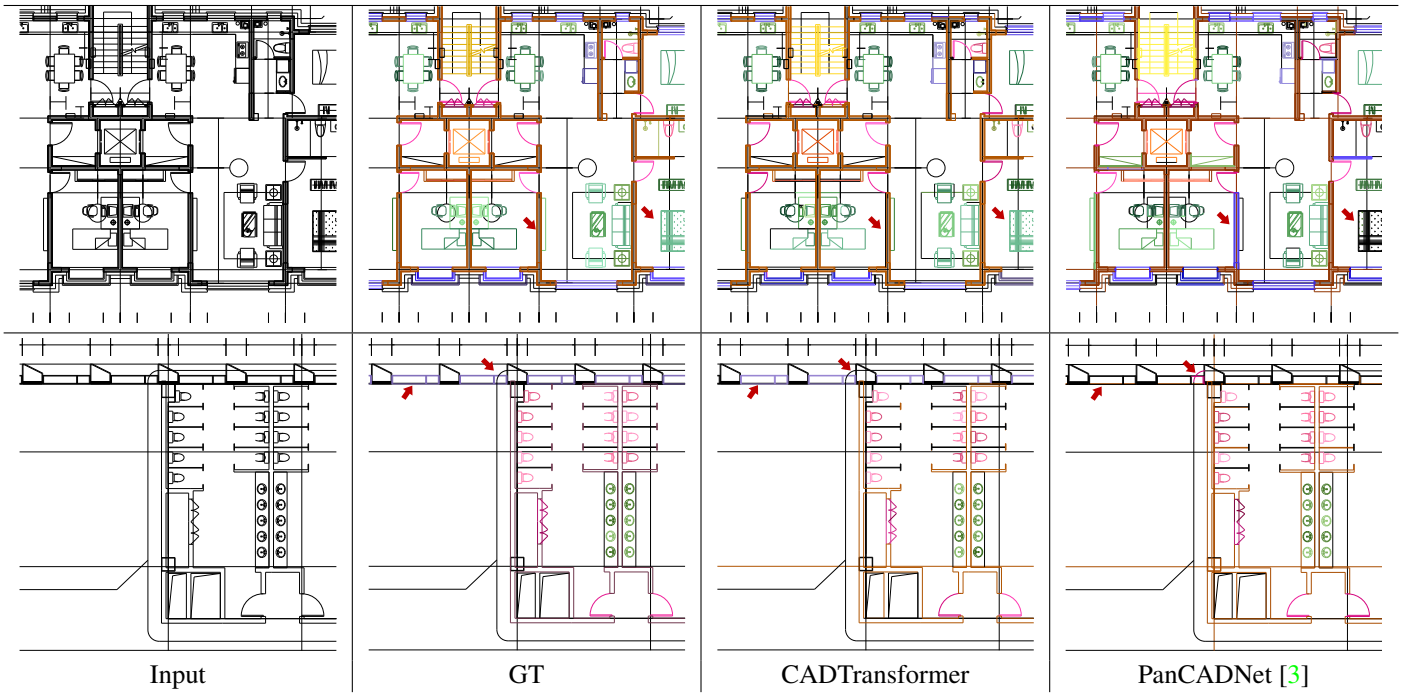

Figure 2. Qualitative comparisons between CADTransformer and GCN-CNN based method [3] on FloorPlanCAD dataset. Red arrows indicate representative graphical primitives. Visualization colors of each category can be found in Figure 3. Best view in color and zoom in.

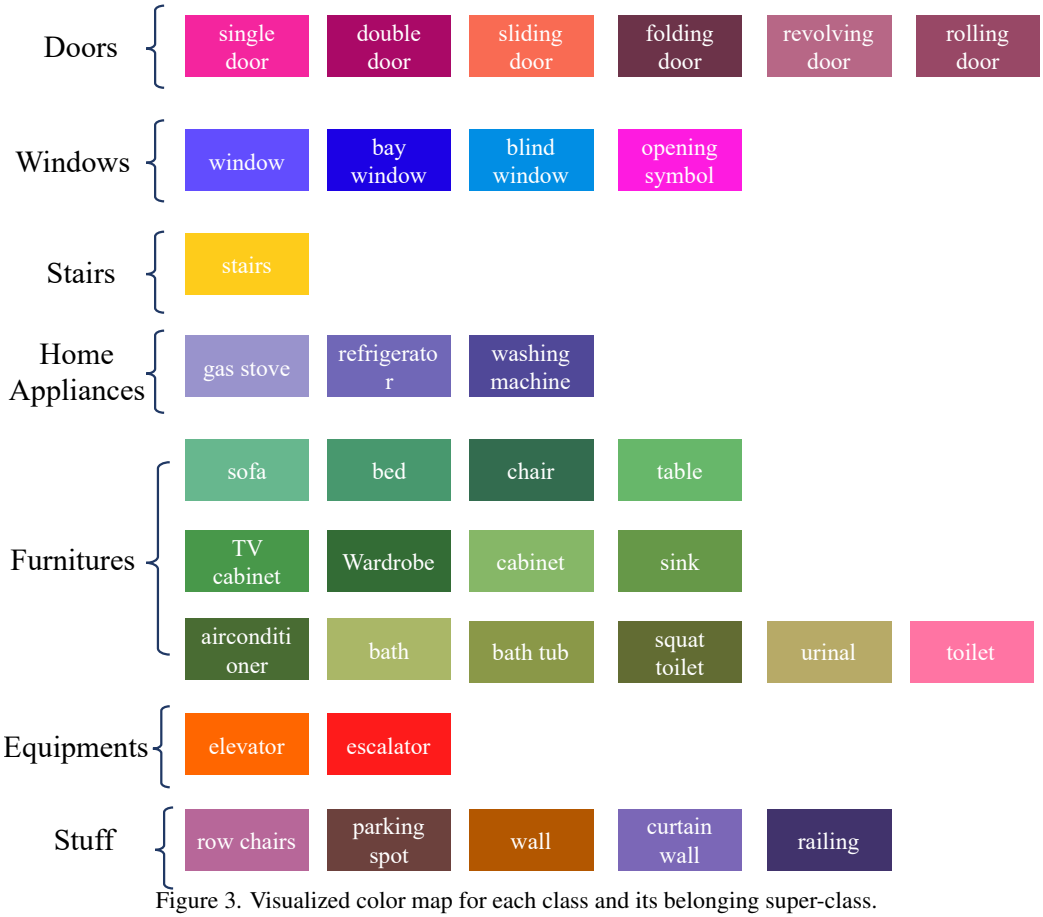

Figure 3. Visualized color map for each class and its belonging super-class.

References

[1] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9640–9649, 2021. 1

[2] 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. 1

[3] 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, 3, 4, 5

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 1

[5] Fang Liu, Xiaoming Deng, Yu-Kun Lai, Yong-Jin Liu, Cuixia Ma, and Hongan Wang. Sketchgan: Joint sketch completion and recognition with generative adversarial network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5830–5839, 2019. 1

[6] 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. 1