Graph2Plan: Learning Floorplan Generation from Layout Graphs Supplemental Material

RUIZHEN HU, Shenzhen University
ZEYU HUANG, Shenzhen University
YUHAN TANG, Shenzhen University
OLIVER VAN KAICK, Carleton University
HAO ZHANG, Simon Fraser University
HUI HUANG*, Shenzhen University

CCS Concepts: • **Computing methodologies → Shape modeling**; *Neural networks.*

ACM Reference Format:
Ruizhen Hu, Zeyu Huang, Yuhan Tang, Oliver van Kaick, Hao Zhang, and Hui Huang. 2020. Graph2Plan: Learning Floorplan Generation from Layout Graphs Supplemental Material. *ACM Trans. Graph.* 39, 4, Article 118 (July 2020), 2 pages. https://doi.org/10.1145/3386569.3392391

1 TURNING FUNCTION BASED RETRIEVAL

During the layout graph retrieval, we measure the distance between turning functions of two buildings to retrieve buildings with similar boundaries from training data. We use a clustering-based approach to accelerate this process.

For all training data, we discretize their turning functions into M dimension vectors. Then, we used KNN algorithm to cluster them based on L2 distance and obtain N clusters. For each cluster, we reserve K nearest samples for retrieval. We show the some typical boundaries in Fig 1. Each sample is the closest one to its cluster center. For a test boundary, we first calculated its turning function and discretize it into an M dimension vector and find the n nearest clusters. Then we calculate the L2 distance between the test turning function vector and all samples in the selected n clusters, and retrieve top k nearest samples. In practice, We use $M = N = K = 1000, k \in [1, 5]$ so that the cluster samples can cover 95% of the training data. This make it 20 times faster than retrieval by directly ranking the distance between test data and all training data.

*Corresponding author: Hui Huang (hhzhiyan@gmail.com)

Authors' addresses: Ruizhen Hu, College of Computer Science & Software Engineering, Shenzhen University, ruizhen.hu@gmail.com; Zeyu Huang, Shenzhen University; Yuhan Tang, Shenzhen University; Oliver van Kaick, Carleton University; Hao Zhang, Simon Fraser University; Hui Huang, College of Computer Science & Software Engineering, Shenzhen University.

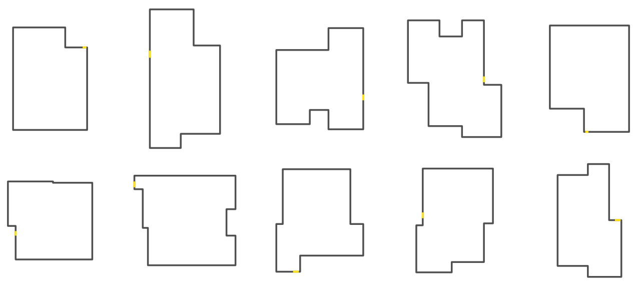


Fig. 1. Typical boundaries in the training data.

2 NETWORK ARCHITECTURES

We describe our Graph2Plan network architectures here for all component of our model. The graph neural network (GNN), is the same as the one used in [Ashual and Wolf 2019]. The architectures of the rest of our networks are listed in Tables 1, 2, 3 and 4. The Conv2d module has 4 arguments which are the input channel size, the output channel size, the convolution kernel size and the stride size. The RoIAlign module has two arguments which are the output feature shape and the spatial scale. The BoxToLayout module has 1 argument which is the output shape. The BoxToLayout module takes node feature vectors, node boxes, node categories as input, converts the boxes to dense grid coordinates, expands the feature vectors to 8*8 feature maps and outputs a new feature map by sampling from the expanded feature maps with the converted grid coordinates with bi-linear interpolation.

3 TRAINING DETAILS

We train all networks with an Nvidia TITAN V GPU under Ubuntu 18.04.3 and Pytorch 1.3.1. The training and testing is performed mainly in the GPU. We perform random $k\pi/2 (k \in [0, 1, 2, 3])$ rotation and left-right flip on the training data. For all experiments, we use Adam optimizer with learning rate 0.0001 and decay rate 0.0001 for training.

We use a three step strategy to warm up the training process. In the first epoch, we only train the BoundaryCNN and BoxRegregrssionNet to let the network learn a legal box distribution. We start training the LayoutCNN from the second epoch and increase the weight of loss for the layout images from 0.1 to 1.0 in 3 epochs to avoid sudden difference on the gradient. We use the same method to

Table 1. BoundaryCNN architecture

Index	Inputs	Operation	Output Shape
1	-	Boundary input image	[3,128,128]
2	1	Conv2d(3,32,3,2)	[32,63,63]
3	2	Batch Normalization	[32,63,63]
4	3	LeakyReLU(0.01)	[32,63,63]
5	4	Conv2d(32,64,3,2)	[64,31,31]
6	5	Batch Normalization	[64,31,31]
7	6	LeakyReLU(0.01)	[64,31,31]
8	7	Conv2d(64,128,3,2)	[128,15,15]
9	8	Batch Normalization	[128,15,15]
10	9	LeakyReLU(0.01)	[128,15,15]
11	10	Conv2d(128,256,3,2)	[256,7,7]
12	11	Global Average Pooling	[256]

Table 2. BoxRegressionNet architecture

Index	Inputs	Operation	Output Shape
1	-	GNN node output	[128]
2	-	Boundary CNN output	[256]
3	1,2	Concatenate	[384]
4	3	Linear(384,512)	[512]
5	4	LeakyReLU(0.01)	[512]
6	5	Linear(512,4)	[4]
7	6	LeakyReLU(0.01)	[4]

Table 3. LayoutCNN architecture

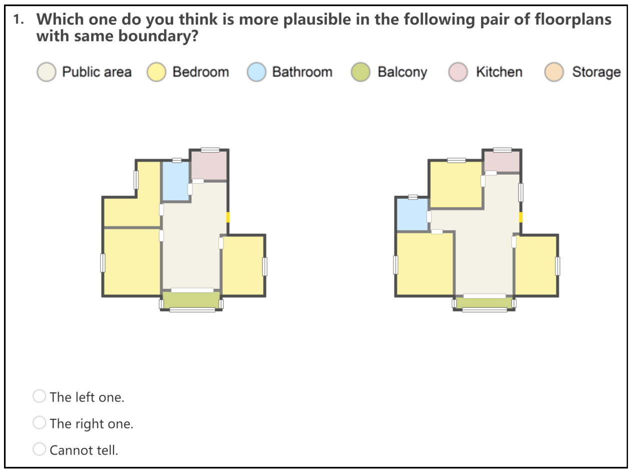
Index	Inputs	Operation	Output Shape
1	-	Predicted box	[4]
2	-	GNN node output	[128]
3	-	Boundary CNN output	[256]
4	2,3	Concatenate	[384]
5	1,4	BoxToLayout(128)	[384,128,128]
6	5	Conv2d(384,128,3,1)	[128,128,128]
7	6	Batch Normalization	[128,128,128]
8	7	LeakyReLU(0.01)	[128,128,128]
9	8	Conv2d(128,64,3,1)	[64,128,128]
10	9	Batch Normalization	[64,128,128]
11	10	LeakyReLU(0.01)	[64,128,128]
12	11	Conv2d(64,15,3,1)	[15,128,128]

start training the BoxrefineNet from the third epoch and gradually increase the weight of loss for the refined boxes.

Then we reduce the learning rate by 0.1 when the total loss on the validation set stops improving with a threshold 0.001 relative the minimum loss and a patience 10. We train 100 epochs and it takes about 30 hours. The inference time for one forward propagation is about 50ms.

Table 4. BoxRefineNet architecture

Index	Inputs	Operation	Output Shape
1	-	Predicted box	[4]
2	-	LayoutCNN output	[15,128,128]
3	2	Conv2d(15,64,3,2)	[15,63,63]
4	3	Batch Normalization	[15,63,63]
5	4	LeakyReLU(0.01)	[15,63,63]
6	5	Conv2d(64,128,3,2)	[128,31,31]
7	6	Batch Normalization	[128,31,31]
8	7	LeakyReLU(0.01)	[128,31,31]
9	8	Conv2d(128,256,3,2)	[256,15,15]
10	1,9	RoIAlign(8,1/8)	[256,8,8]
11	10	Global Average Pooling	[256]
12	11	Linear(256,512)	[512]
13	12	Linear(512,4)	[4]


1. Which one do you think is more plausible in the following pair of floorplans with same boundary?

○ The left one.
○ The right one.
○ Cannot tell.

Fig. 2. An example task in our user study.

4 USER STUDY

We show an example graphical interface of our user study tasks in Fig.2. Each task contains the same question, a legend for identifying different room types, two images from ground truth and outputs of our method with random order, and three options.

REFERENCES

Oron Ashual and Lior Wolf. 2019. Specifying Object Attributes and Relations in Interactive Scene Generation. In *Proc. Int. Conf. on Computer Vision*.