Object Detection in Floor Plan Images

Zahra Ziran(✉) and Simone Marinai

Dipartimento di Ingegneria dell'Informazione (DINFO),
Università degli Studi di Firenze, Florence, Italy
{zahra.ziran,simone.marinai}@unifi.it

Abstract. In this work we investigate the use of deep neural networks for object detection in floor plan images. Object detection is important for understanding floor plans and is a preliminary step for their conversion into other representations.

In particular, we evaluate the use of object detection architectures, originally designed and trained to recognize objects in images, for recognizing furniture objects as well as doors and windows in floor plans. Even if the problem is somehow easier than the original one in the case of this research the datasets available are extremely small and therefore the training of deep architectures can be problematic. In addition to the use of object detection architectures for floor plan images, another contribution of this paper is the creation of two datasets that have been used for performing the experiments covering different types of floor plans with different peculiarities.

Keywords: Floor plan analysis · Object detection
Convolutional neural networks · Transfer learning

1 Introduction

Detecting and recognizing objects in floor plans is an essential task for the understanding of these graphical documents. Our research on this topic is part of the overall task of understanding of graphical documents for generating accessible graphical documents for visually impaired people [4,13]. A comprehensive perception of a floorplan is crucially important for blind people, allowing them to find their path as they face a new building. It is important to clarify that floorplans available in real estate websites or other floorplans in other websites are nearly always in image format even if they have been produced with CAD tools. CAD files are in general only available to their authors and not distributed. Object detection in natural images is basically defined as finding the location of objects in one image and labeling them. In many cases, the object location is based on the identification of the bounding box surrounding it. Also in this application the identification of the object bounding box is sufficient for our purposes. Starting from widely studied architectures based on convolutional neural networks, a few object detectors have been recently proposed, such as: Faster R-CNN [17], R-FCN, Multibox, SSD [11] and YOLO [16].

1.1 Previous Work

As in several domains also the document analysis community faced a growing use of deep learning in recent research. When looking to the use of deep learning in the area of graphics recognition there are a limited, but interesting research works. Among various techniques object detectors have been used to address various problems in document analysis. Symbol detection in on-line graphical documents is proposed in [9] where the authors use Faster R-CNN do address the task. In particular, the work addresses the recognition of mathematical expressions and flowcharts in handwritten documents by using the Tensorflow Object Detection API [8]. Another application of the latter API is related to handwritten music object detection [14] where the Faster R-CNN is used to recognize musical symbols. In both papers the number of training items is relatively high and the results are evaluated only considering the accuracy of the model without taking into account the recall. Other authors used Faster R-CNN for page layout identification [18], for comic character face detection [15], and for arrow localization on handwritten industrial inspection sheets [5].

One recent effort to extract structural information from floor plan images is described in [2] where the authors parse floor plan images to estimate the size of the rooms for interactive furniture fitting. They first perform wall segmentation by using a fully convolutional neural network, subsequently they detect objects using a Faster R-CNN, and finally, they do optical character recognition to obtain the rooms dimensions. One interesting feature of this work is the combination of three methods to achieve the overall floor plan understanding. Unfortunately, very few details are provided in the paper about the use of Faster R-CNN for object location. Moreover, the floor plan dataset created by the authors only contains the ground-truth about the wall position.

In the work described in [6] the authors address the floor plan understanding by segmenting walls, windows, and doors. One of the main focuses of the paper is to address images with different notations (e.g. for walls or for furniture objects). The proposed techniques are tested on four floor plan datasets (named CVC-FP) which are freely accessible to the public. As discussed also in Sect. 3 the CVC-FP dataset only contains objects of 6 classes: `sink`, `toilet`, `shower`, `bath`, `door`, and `window`, without include furniture objects.

From the point of view of the neural architecture one important paper for this work is [7] where the authors evaluate and compare different object detection architectures. The goal of [7] is to identify the most successful architectures and support users when choosing one architecture on the basis of various perspectives: speed, memory, and accuracy. To this end, the authors in [7] implement some modern convolutional detectors: Faster R-CNN, R-FCN and SSD in a unified framework, as a part of the Tensorflow Object Detection API [8]. The authors pre-trained the architectures on several datasets, but the best performance were achieved by pre-training with the COCO dataset [10].

In this work we explore the use and adaptation of the Tensorflow Object Detection API [8] to identify floor plan objects in two datasets that have been built to address this task.

The rest of the paper is organized as follows. In Sect. 2 we describe the architecture of the Faster R-CNN model considered and provide some information about how we modified it in order to obtain information about the recall of the system. In Sect. 3 we analyze the peculiarities of the floor plan datasets that we built and used in the experiments discussed in Sect. 4. Our conclusions and pointers for future work are in Sect. 5.

2 The Architecture

In this research we work with the widely used Tensorflow Object Detection API [8] for an easy comparison of alternative architectures. We initially evaluated one COCO-pre-trained Single Shot Detector with MobileNets that we fine-tuned with floor plan images. We selected this architecture because it is a small and flexible model that has the benefit of fast training times compared to larger models, while it does not sacrifice much in terms of accuracy. In these preliminary tests we also compared the SSD with Faster R-CNN with ResNet 50 and with ResNet 101. After these preliminary experiments it turned out that Faster R-CNN performs significantly better than SSD. Moreover, comparing the performance of ResNet 50 with ResNet 101 on the floor plan datasets, there was no real difference. We therefore used Faster R-CNN with ResNet 50 as a basis model for our work.

Faster R-CNN is one of the most accurate and fast neural object detectors proposed so far. The internal structure of the network is as follows (see Fig. 1): first, the image is passed through some convolutional layers to produce several feature maps. Then, the main component of Faster R-CNN, the region proposal network (RPN), uses a 3×3 sliding window and takes the previous feature maps as input. The output of the RPN is a tensor in a lower dimension. At this stage, each window location generates some bounding boxes, based on fixed-ratio anchor boxes (e.g. 2.0, 1.0, 0.3) and an "objectness" score for each box. These are the region proposals for the input image which provide approximate coordinates of the objects in the image. The "objectness" scores, if above a given threshold, determine which region proposal can move forward in the network. Subsequently, the good regions pass through a pooling layer, then a few fully-connected layers, and finally a softmax layer for classification and a regressor for bounding box refinement.

As previously mentioned to perform our experiments we use the Tensorflow Object Detection API. This is an open source framework, built on top of the widely used Tensorflow library, that takes care of the training and evaluation of the different architectures implemented. One interesting feature of the API is that it makes it easy to train different models on the same dataset and compare their performance. In addition to the average precision performance per category, we extended the API to calculate the number of false negatives as well as the average recall per class.

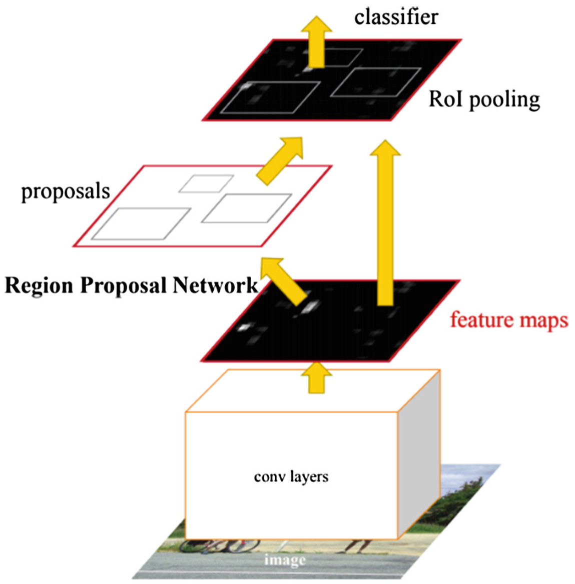

Fig. 1. The internal architecture of the Faster R-CNN as a single, unified network for object detection (Image from [17]).

2.1 False Negative Calculation

By default, the Tensorflow Object Detection API supports the PASCAL Visual Object Classes (VOC) 2007 [3] detection metric. This metric is designed to evaluate visual object detection and recognition models, which helps machine learning researchers have standard evaluation procedures.

In the detection metric, for a detection bounding box to be a true positive, three conditions must be true:

- The area of the intersection of the detected bounding box B_d and the ground truth bounding box B_{gt} over the union area of the two bounding boxes must be greater than 0.5, according to the following equation:

$$r_i = \frac{area(B_d \cap B_{gt})}{area(B_d \cup B_{gt})} > 0.5 \tag{1}$$

- The class label of the detection bounding box and the ground truth bounding box must be the same.
- The probability of the object's recognition must be greater than some specific thresholds. In most cases, and also in this work, we consider the object as found if the probability is higher than 0.50.

To find false negative detections we first matched all the detections to objects in the ground truth. True/false positives are determined and detections matched to difficult boxes are ignored. In the next stage the ground truth objects that have not been detected are determined as false negatives.

After computing the true positives and false negatives number for each category it is easy to calculate the average recall in addition to the average precision computed by the API: $Recall = \frac{TP}{TP+FN}$.

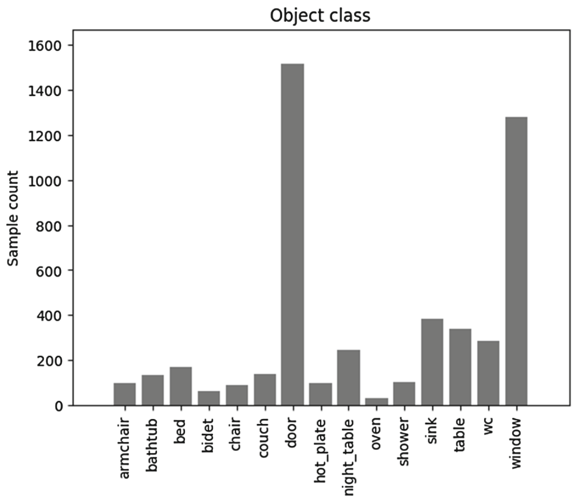

Fig. 2. Object class distribution of the *d1* dataset.

3 The Floor Plan Datasets

In order to evaluate the object detection in floor plans we obviously need one or more labeled datasets. In the past decade, some datasets have been proposed for evaluating research on floor plan analysis. The SESYD dataset [1] contains synthetic floor plans where furniture objects are randomly placed on a few fixed floor plan layouts. Even if this approach for dataset generation is very interesting, the actual dataset contains only ten floor plan layouts and the objects come from a limited number of categories (for instance, there is only one model for the bed). Moreover, the generated floor plans are somehow unrealistic with very small beds or similar mistakes. Another dataset widely used for this research has been proposed in [12]. This dataset contains 90 actual floor plans provided by one architectural firm. While more realistic than the others, these floor plans contain only few objects and therefore are not suitable for the research carried out in this work.

In order to work with realistic images we first created one small dataset (referred to as *d1*) using the images that show up in Google's image search. This dataset consists of 135 images of variable size containing objects in 15 classes and a total of 4973 objects (in the experiments we considered 2697 objects in the training set, 1165 objects in the validation set, and 1111 objects in the test set). In Fig. 4 we show one example of floor plan in this collection, while Fig. 2 shows the distribution of objects in the different classes. Of course some types of objects are not present in all the images, for instance the floor plan of an office might not have any bed in it. Among all the classes, the `oven` is the rarest one and the `door` is the most frequent one.

The second dataset that we gathered is called *d2*. This dataset contains Middle Eastern floor plans, with object shapes different from the ones in *d1*. Another important feature is that the floor plans in *d2* come from one architectural firm and are therefore more homogeneous in their content. The *d2* dataset consists of 300 images, but only 160 images have been labeled so far. The 160 images contain objects in 12 classes and a total of 7788 (in the experiments we considered 4535 objects in the training set, 1457 objects in the validation set, and 1796 objects in the test set). In Fig. 5 we show one example of floor plan in this collection, while Fig. 3 shows the distribution of objects in the different classes. As a particular property of these floor plan datasets, it is worth to note that the images are mostly grayscale and contain simple shapes. As we will see in the experimental part this property has a positive effect on the performance of the model, compared to datasets that contain images with more complex features and more noise. The dataset *d1* has the greatest imbalance in the number of objects in each class. Moreover, images in *d1* have more diversity. For example, almost none of the objects in *d2* are filled with color, while in *d1* all the floor plans are painted, for presentation purposes.

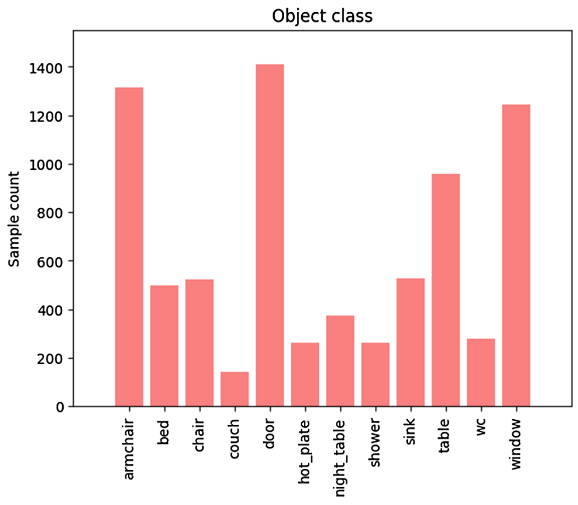

Fig. 3. Object class distribution of the *d2* dataset (Color figure online)

4 Experiments

The Faster R-CNN is first trained on the *d1* dataset, which contains 135 images. As mentioned earlier, this dataset is substantially diverse and contains random non-standard images collected from the Internet.

Table 1. Final evaluation results

Dataset	Objects	False negatives		Mean average precision		Mean average recall	
		Val	Test	Val	Test	Val	Test
d1	1111	411	445	0.32	0.31	0.60	0.56
d2	1796	87	102	0.83	0.86	0.92	0.92

Final evaluation results of the d1 *dataset after 46916 training steps, and the* d2 *dataset after 18550 training steps.*

In the first experiments performed on a smaller dataset with the default configuration of the API the results on the validation set were not satisfying with a maximum mean average precision of about 0.26. To aid generalization, we threw in a few of data augmentation options. In particular we considered `random horizontal flip`, `random vertical flip`, `random rotation 90`, and `random RGB to gray`. These options are provided by the Tensorflow Object Detection API. In addition to data augmentation we also changed the scales and aspect ratios of anchor generator in order to take into account the peculiarities of the floor plan objects.

With the above mentioned modified configuration we also ran our experiments on the *d2* dataset. After stopping the training considering the validation set, the mean average precision and recall on the test set for both datasets are shown in Table 1. Details about the performance achieved for each class are reported in Table 2.

Taking into account the features of the two datasets it is not surprising that the best results are achieved on the *d2* dataset with a mean average precision of 0.86, and a mean average recall of 0.92. Part of the difference in performance is probably related to the special nature of the dataset: compared to *d1*, the objects are cleaner, less diverse and not different across images other than rotation and scale. As it turns out, the performance of the model is not too much affected by an imbalanced dataset (*d2*). For instance the model achieves 0.80 average precision for the `couch` class that is the less frequent one. At the same time the model achieved 0.93 average precision for the `door` class which has at least 10 times more samples than the `couch` class. Concerning the average recall it is very interesting to notice that the test images had zero false negatives for the `hot-plate` and `bed` classes. On the basis of the superior performance of the network on the *d2* dataset we wanted to explore more in details the possibility of doing some transfer learning of this network to improve the performance on the *d1* dataset. As we can see in Table 3 by finetuning on the *d1* dataset one network previously trained on the *d2* dataset we achieve a 0.39 mean average precision and 0.69 mean average recall. This is better than the previously mentioned results obtained by finetuning on the *d1* dataset one network previously trained on the COCO dataset.

Table 2. Performance by category

Class	Average recall(Val)		Average recall(Test)		Average precision(Val)		Average precision(Test)	
	d1	d2	d1	d2	d1	d2	d1	d2
armchair	0.92	0.97	0.50	0.95	0.21	0.92	0.21	0.92
bathtub	0.72	N/A	0.80	N/A	0.57	N/A	0.57	N/A
bed	0.56	1.00	0.70	1.00	0.47	0.98	0.47	0.98
bidet	0.57	N/A	0.54	N/A	0.11	N/A	0.11	N/A
chair	0.58	0.68	0.44	0.76	0.12	0.50	0.12	0.62
couch	0.75	0.80	0.52	0.88	0.46	0.52	0.48	0.80
door	0.63	0.97	0.63	0.95	0.60	0.94	0.60	0.93
hot_plate	0.67	0.97	0.52	1.00	0.40	0.92	0.39	0.96
night_table	0.65	0.93	0.34	0.81	0.37	0.86	0.37	0.79
oven	0.12	N/A	0.55	N/A	0.01	N/A	0.01	N/A
shower	0.55	0.97	0.71	0.93	0.19	0.90	0.16	0.90
sink	0.57	0.95	0.47	0.94	0.28	0.80	0.28	0.85
table	0.58	0.94	0.46	0.98	0.33	0.86	0.33	0.93
wc	0.45	1.00	0.65	0.95	0.13	0.79	0.13	0.78
window	0.58	0.91	0.54	0.88	0.49	0.87	0.48	0.85

Average precision and recall calculated by category, for the d1 *and* d2 *dataset after 46916 and 18550 training steps, respectively.*

Table 3. More transfer learning

Dataset	Objects	False negatives	Mean average precision	Mean average recall
COCO - *d1*	1111	445	0.31	0.60
d2 - *d1*	1111	368	0.39	0.69

The final evaluated results of the pre-trained models that is fine tuned on dataset *d1*.

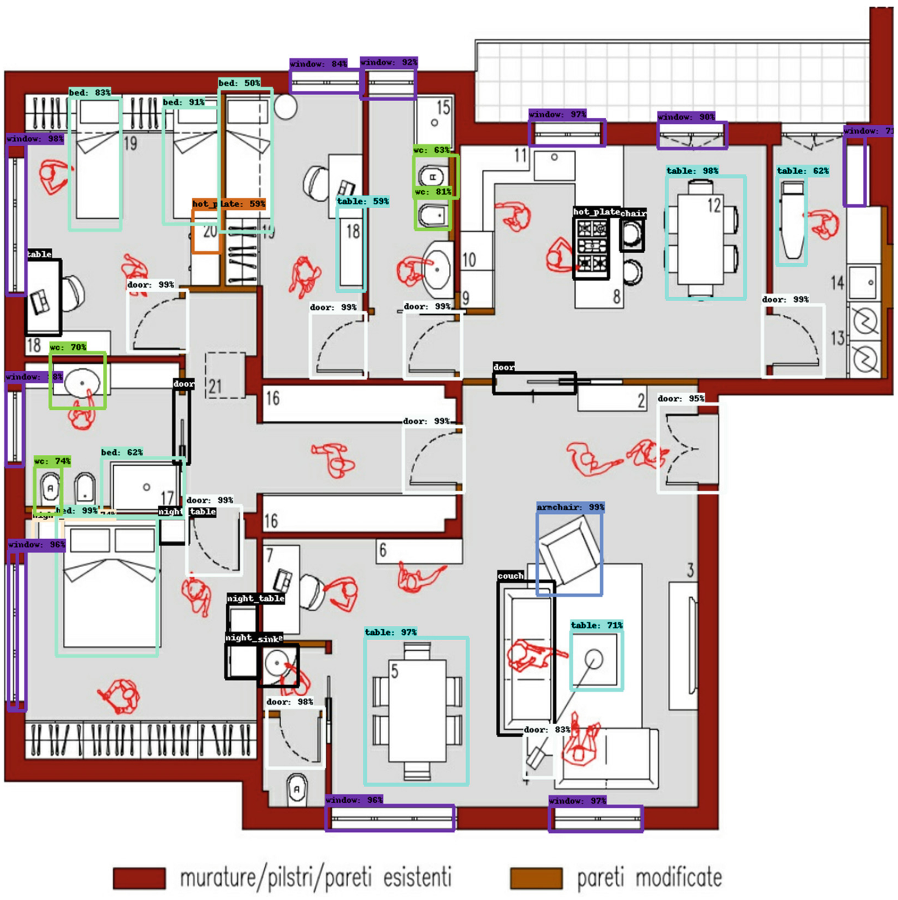

Fig. 4. An inference result of the model trained on the *d1* dataset. False negatives have black bounding boxes and detection bounding boxes are colorful. Note how new shapes and colors in this test image damage the performance of the model. (Color figure online)

4.1 Discussion

From the experiments performed on the two datasets we can notice that by using convolutional object detectors, the recognition performance are not too much influenced from class imbalance in the training set.

The only exception is related to the `oven` class from dataset *d1* whose extremely low performance are probably due to the very low sample size and the variability of the appearance of this object in the dataset. On the other hand, the `door` and `window` classes are responsible for 52% of the false negatives in d1 validation set, while they make up 58% of the object samples. In these two classes the model performs relatively well in terms of average precision, but it

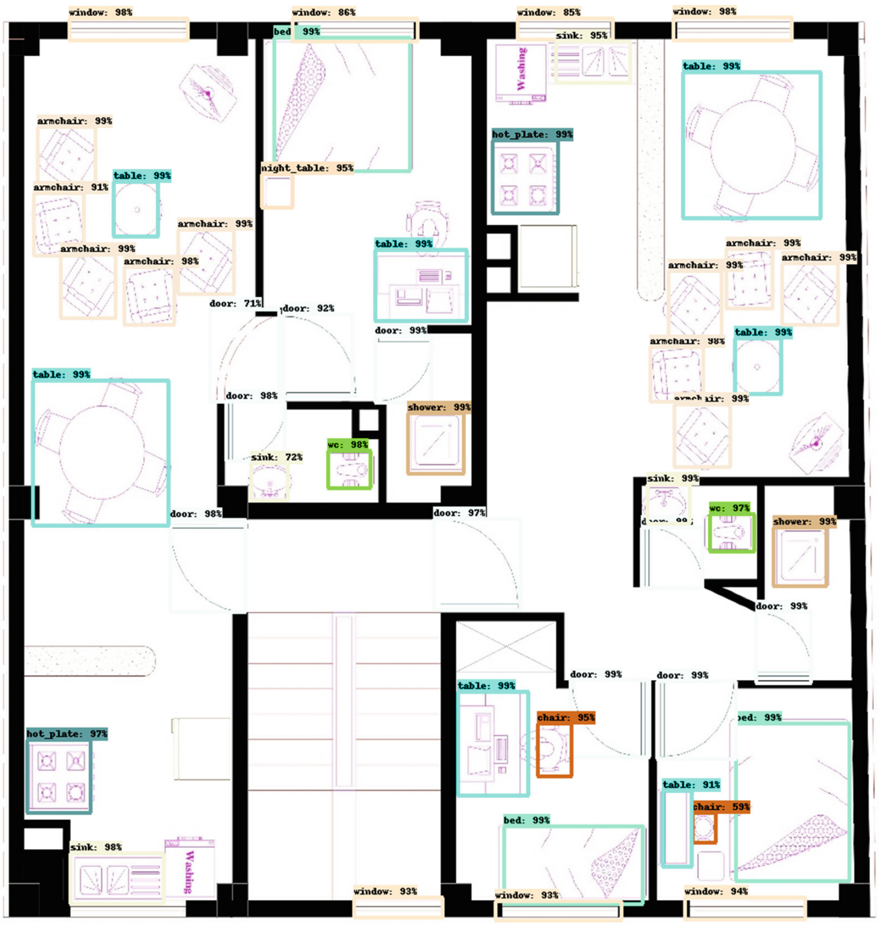

Fig. 5. Example of results of the model trained on the *d2* dataset.

is not capable of detecting many objects. At first this result might contradict the intuition that more samples led to better performance. However it is important to recall that the performance of the model in one class heavily depends on the diversity of object samples. It is useful to remark that in the case of doors and walls the objects are connected to the walls while other objects are usually more isolated in the rooms. The diversity of walls and doors is therefore higher with respect to other classes because of the variable context. Another source of errors for windows is the higher variability of the aspect-ratio with respect to other objects that in most cases are simply scaled and rotated, in particular in dataset *d2* where reasonable performance on the dataset is obtained. Regarding the three most frequent classes in *d2* `armchair`, `door`, and `window`, it can be seen from Table 2 that the model is nearly perfect in terms of average precision and the class bed (whose items are more regular) achieves an average precision of 98%.

5 Conclusions

In this work two different floor plan datasets have been created to cover different architectures and drawing conventions of floor plans from all over the world. The performance of an object detector which is originally designed for detecting objects in natural images was tested to identify objects in the floor plans in these datasets. The floor plan has an essential misrepresentation issue in terms of the sample size of objects. To better analyze the performance, false negative objects of each class have individually been counted in order to find out whether the detection results suffer from differences in the number of samples for each class. We noticed that the performance of the model in a class heavily depends on the diversity of object samples, object rotation and scale somehow outweighing the role of sample size. It is interesting also to notice how a network that pre-trained from another domain (COCO pre-trained Faster-RCNN with Res Net 50) can perform well on the floor plan datasets using just a one hundred images. To further improve the results on this task, it is recommended to either collect larger datasets to cover different graphical conventions, or implement data augmentation techniques more suitable for the object detection in floor plans.

References

1. Delalandre, M., Valveny, E., Pridmore, T., Karatzas, D.: Generation of synthetic documents for performance evaluation of symbol recognition & spotting systems. Int. J. Doc. Anal. Recognit. (IJDAR) **13**(3), 187–207 (2010)
2. Dodge, S., Xu, J., Stenger, B.: Parsing floor plan images. In: 2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA), pp. 358–361, May 2017. https://doi.org/10.23919/MVA.2017.7986875
3. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object classes (VOC) challenge. Int. J. Comput. Vis. **88**(2), 303–338 (2010). https://doi.org/10.1007/s11263-009-0275-4
4. Goncu, C., Madugalla, A., Marinai, S., Marriott, K.: Accessible on-line floor plans. In: Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, 18–22 May 2015, pp. 388–398 (2015). http://doi.acm.org/10.1145/2736277.2741660
5. Gupta, G., Sharma, S.M., Vig, L.: Information extraction from hand-marked industrial inspection sheets. In: 14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, 9–15 November 2017, pp. 33–38 (2017)
6. de las Heras, L.P., Ahmed, S., Liwicki, M., Valveny, E., Sánchez, G.: Statistical segmentation and structural recognition for floor plan interpretation. Int. J. Doc. Anal. Recognit. (IJDAR) **17**(3), 221–237 (2014). https://doi.org/10.1007/s10032-013-0215-2

7. Huang, J., et al.: Speed/accuracy trade-offs for modern convolutional object detectors. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3296–3297, July 2017. https://doi.org/10.1109/CVPR.2017.351
8. Huang, J., et al.: TensorFlow object detection API (2018). https://github.com/tensorflow/models/tree/master/research/object_detection
9. Julca-Aguilar, F.D., Hirata, N.S.T.: Symbol detection in online handwritten graphics using faster R-CNN. In: Proceedings of the 13th International Workshop on Document Analysis Systems, pp. 151–156 (2018)
10. Lin, T., et al.: Microsoft COCO: common objects. In: Proceedings of Computer Vision - ECCV 2014 - 13th European Conference Part V, Zurich, Switzerland, 6–12 September 2014, pp. 740–755 (2014)
11. Liu, W., et al.: SSD: single shot multibox detector (2016, to appear). http://arxiv.org/abs/1512.02325
12. Macé, S., Locteau, H., Valveny, E., Tabbone, S.: A system to detect rooms in architectural floor plan images. In: Proceedings of the 9th IAPR International Workshop on Document Analysis Systems, DAS 2010, pp. 167–174. ACM, New York (2010). http://doi.acm.org/10.1145/1815330.1815352
13. Madugalla, A., Marriott, K., Marinai, S.: Partitioning open plan areas in floor plans. In: 14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, 9–15 November 2017, pp. 47–52 (2017). https://doi.org/10.1109/ICDAR.2017.17
14. Pacha, A., Eidenberger, H., Kwon-Young Choi, B.C., Ricquebourg, Y., Zanibbi, R.: Handwritten music object detection: open issues and baseline results. In: Proceedings of the 13th International Workshop on Document Analysis Systems, pp. 163–168 (2018)
15. Qin, X., Zhou, Y., He, Z., Wang, Y., Tang, Z.: A faster R-CNN based method for comic characters face detection. In: 14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, 9–15 November 2017, pp. 1074–1080 (2017)
16. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017, pp. 6517–6525 (2017). https://doi.org/10.1109/CVPR.2017.690
17. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(6), 1137–1149 (2017). https://doi.org/10.1109/TPAMI.2016.2577031
18. Yi, X., Gao, L., Liao, Y., Zhang, X., Liu, R., Jiang, Z.: CNN based page object detection in document images. In: 14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, 9–15 November 2017, pp. 230–235 (2017)