Article

Towards Robust Object detection in Floor Plan Images: A Data Augmentation Approach

Shashank Mishra [1], Khurram Azeem Hashmi [1,2,3,*], Alain Pagani[3], Marcus Liwicki[4], Didier Stricker [1,3] and Muhammad Zeshan Afzal [1,2,3*]

[1] Department of Computer Science, Technical University of Kaiserslautern, 67663 Kaiserslautern, Germany; s_mishra19@cs.uni-kl.de (S.M.); khurram_azeem.hashmi@dfki.de (K.A.H.); muhammad_zeshan.afzal@dfki.de (M.Z.A.); didier.stricker@dfki.de (D.S.);
[2] Mindgarage, Technical University of Kaiserslautern, 67663 Kaiserslautern, Germany
[3] German Research Institute for Artificial Intelligence (DFKI), 67663 Kaiserslautern, Germany; alain.pagani@dfki.de (A.P.);
[4] Department of Computer Science, Luleå University of Technology, 971 87 Luleå, Sweden; marcus.liwicki@ltu.se (M.L.);
* Correspondence: khurram_azeem.hashmi@dfki.de

Abstract: Object detection is one of the most critical tasks in the field of Computer vision. This task comprises identifying and localizing an object in the image. Architectural floor plans represent the layout of buildings and apartments. The floor plans consist of walls, windows, stairs, and other furniture objects. While recognizing floor plan objects is straightforward for humans, automatically processing floor plans and recognizing objects is a challenging problem. In this work, we investigate the performance of the recently introduced Cascade Mask R-CNN network to solve object detection in floor plan images. Furthermore, we experimentally establish that deformable convolution works better than conventional convolutions in the proposed framework. Identifying objects in floor plan images is also challenging due to the variety of floor plans and different objects. We faced a problem in training our network because of the lack of publicly available datasets. Currently, available public datasets do not have enough images to train deep neural networks efficiently. We introduce SFPI, a novel synthetic floor plan dataset consisting of 10000 images to address this issue. Our proposed method conveniently surpasses the previous state-of-the-art results on the SESYD dataset and sets impressive baseline results on the proposed SFPI dataset. The dataset can be downloaded from SFPI Dataset Link. We believe that the novel dataset enables the researcher to enhance the research in this domain further.

Keywords: Object Detection, Cascade Mask R-CNN, Floor Plan Images, Deep Learning, Transfer Learning, Dataset Augmentation, Computer Vision.

Citation: Mishra, S.; Hashmi, K.A.; Pagani, A.; Liwicki, M.; Stricker, D.; Afzal, M.Z. Object Detection in Floor Plan Images. *Preprints* **2021**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

1. Introduction

Object detection is one of the most elementary and important tasks in the field of computer vision. In object detection, we deal with the identification and localization of objects present in the image or video [1,2]. Architectural floor plans contain structural and semantic information, e.g., room size, type, location of doors and walls, and furniture. These images can be complex and challenging to interpret their semantic meanings. Floor plan analysis is not merely a general segmentation problem since floor plans contain not only floor plan elements like walls, furniture but how these individual objects are related to each other. Floor plan images have several applications, such as CAD model generation [3], 3D model creation for interactive walkthroughs [4] or similarity search [5]. Different floor plan images contain specific information within them and might not be similar. Figure 1 illustrates information about different rooms and their sizes in the floor plan. Different furniture objects are present in the individual rooms without any other specific information. Walls, windows, and doors are separated and can be identified easily. This type of floor plan can be used for interactive furniture fitting [4].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2021 doi:10.20944/preprints202110.0089.v1

Figure 1. Sample floor plan image with the specification of different room sizes and furniture objects. Each variety of floor plan layout has different floor plan elements, shapes, and sizes.

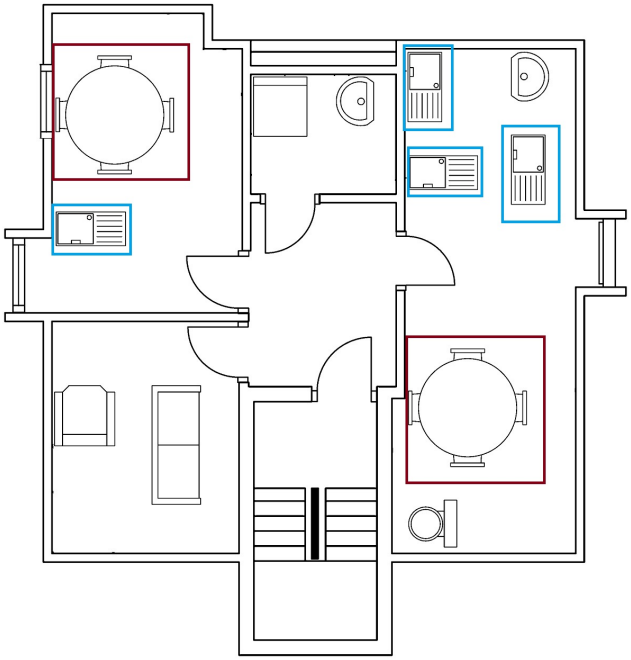

Figure 2. Sample floor plan image from SESYD [6] dataset containing various furniture object classes in different rooms. For example, blue boxes represent sink class, whereas red boxes highlight table class.

The floor plan image in Figure 2 is from the SESYD dataset [6]. This floor plan image contains various furniture objects placed in different rooms. No other information, like room size, is present in the image as it was available in Figure 1. These images can be used to identify the room type based on furniture objects present in it. Furthermore, it helps to generate a 3D model of floor plans [4]. Among these two pictures, we have few common

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2021 doi:10.20944/preprints202110.0089.v1

sets of elements: furniture objects, walls, windows, and doors. These are some common artifacts that must be present in every floor plan image. Identifying these objects is one of the preliminary steps toward the analysis of the floor plan images. Once these objects are identified, we can apply different analysis approaches for numerous applications.

There are only a few publicly available datasets for floor plan images [3]. Apart from fewer samples, less variation in floor plan layouts and furniture objects are also a concern for the floor plan datasets. It is challenging to train deep detectors using the currently available floor plan datasets. We have created our custom dataset to fill this gap, which contains 10000 images with various floor plans and furniture objects. This dataset will be available publicly for further enhancement and experiments.

Figure 3 represents a floor plan with ground truth where all these common artifacts we talked about are masked and highlighted with respective colors. Cascade Mask R-CNN [7], Faster R-CNN [8], SSD [9], YOLO [10], are some other state-of-the-art object detectors.

Figure 3. Sample floor plan image with ground truth. Various furniture class objects available in the image are highlighted.

Although object detection algorithms have been applied before on floor plan images [11], [12], [13], most of them employ Faster R-CNN [8] or YOLO [10]. Contrarily, we propose a framework that operates Cascaded Mask R-CNN [7] for object detection in floor plan images. We employ both conventional convolution and deformable convolutional networks (DCN) [14] on the backbone network, and compare the performance, comparing them with baseline methods. This paper presents an end-to-end approach for object detection in floor plan images. The main contributions of this paper are as follows:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2021 doi:10.20944/preprints202110.0089.v1

1. We present an end-to-end trainable framework that works on Cascade Mask R-CNN [7] with conventional and deformable [14] convolutional backbone network to detect various objects in floor plan images.
2. We publish SFPI (Synthetic Floor Plan Images), a novel floor plan dataset comprising 10000 images. This dataset includes ten different floor plan layouts and 16 different classes of furniture objects.
3. Our proposed method accomplishes state-of-the-art results on the publicly available SESYD dataset [6] and establishes impressive baseline results for the newly proposed dataset.

The rest of the paper is organized as follows. In Section 2, we describe the literature survey in the field of floor plan image object detection. Section 3 discusses the architecture of our proposed model. We discuss the architecture of Cascade Mask R-CNN [7], our backbone network [15] and deformable convolutions [14]. In Section 4, we talk about existing datasets and problems related to these datasets. Then, we analyze the peculiarities of our custom floor plan dataset. In Section 5, we explain our implementation configurations and different experiments. We also evaluate the results of these experiments and compare them with the previous state-of-the-art results. Our conclusions and pointers for future work are explained in Section 6.

2. Related Work

Recent advancements in deep learning methodologies have significantly affected the computer vision approaches like object detection [7,8]. We have quite a few detectors available as per the orientation of specific tasks. In one of the recent works [12], authors extract structural information from floor plan images to estimate the size of the rooms for interactive furniture fitting. In order to achieve this, first wall segmentation is done using a fully convolutional neural network; afterward, they detect objects using a Faster R-CNN, and finally, they perform optical character recognition to detect dimensions of a different room. Faster R-CNN was the main detector used for object detection in floor plan images. In [16], authors address the floor plans with different notations. They use segmentation of walls, doors, and windows to understand the floor plan images better. They tested on publicly available dataset CVC-FP [17] , which contains four different floor plans. It is good to have different floor plans in our dataset as it creates a variety of images and improves the performance of our model.

In [18], authors attempt to synthesize a textual description from a floor plan image. This is another good application for floor plan analysis. This can help visually impaired people to imagine the interiors of the house, and it is also helpful for potential buyers of the house who are located far. Authors detect walls by performing morphological closure on the input floor plan image, doors are detected using the scale-invariant feature, and then connected components are identified using the flood fill technique. Once this information was available, then text processing was applied.

In another work presented by Zeng et al. [19], authors proposed a method for floor plan recognition, with a focus on recognizing diverse floor plan elements, e.g., walls, doors, rooms, furniture. The authors used a shared VGG [20] encoder to extract features from the input floor plan images. It detects the room-boundary for walls, windows, and doors. It also detects the room type based on the elements in the room. The number of furniture items used to identify the room type is less. However, authors get good results in detecting walls, windows, and doors.

In one of the recent works on floor plans, authors [13] created a framework for floor plan recognition and reconstruction. The authors used text detection as well as symbol detection to identify room types. Symbol detection is identifying different furniture objects available in the room and, based on it, determine the type of the room. Authors use YOLO4 [21] as a base network for identifying symbols in different rooms. This is also supported by the information from the text detection. Once all the required information is present, vectorization is performed on the floor plan images to reconstruct a new floor plan image.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2021 doi:10.20944/preprints202110.0089.v1

In [12], authors presented a method to detect the elements in the floor plan images, wall, windows, as well as determining the text from floor plan images. The authors used a fully convolutional network (FCN) and optical character recognition (OCR) technique. Experiments were performed on CVC-FP [17] and self-collected datasets. The experiments were performed on the datasets were: wall segmentation, object detection, and text recognition. Although promising wall segmentation was reported, the number of testing samples to evaluate the object detection and text recognition performance was relatively low.

Another work for object detection in floor plan images was done by Ziran et al. [11], where authors analyzed different available datasets in the floor plan domain and created their own custom datasets. They used Faster R-CNN [8] with ResNet-101 [22] as backbone for detecting furniture objects in the floor plan images. The custom dataset used in this experiment has fewer furniture class objects and fewer samples. Thus, the work does not provide conclusive empirical evidence to verify the effectiveness of the proposed method. From the results, we can identify that the network, which was pre-trained on COCO [23] dataset, performs well.

Based on all these works, we can identify that furniture object detection is the preliminary step in processing floor plan images irrespective of the application. Whether we want to generate some text-based synthesis for floor plan images or we want to reconstruct the floor plan images, we must identify the objects available in different rooms of the floor plan and identify doors and windows correctly. Our work mainly focuses on identifying the furniture objects, windows, and walls in floor plan images creates a base for all the applications mentioned above.

3. Method

The presented approach is based on Cascade Mask R-CNN [7] equipped with backbone ResNeXt-101 [15]. We have implemented this model with conventional convolutional networks (CNN) as well as deformable convolutional networks (DCN) [14]. The Figure illustrates the complete pipeline of our proposed framework. In this section, we dive deep into the individual components of our proposed method.

3.1. Cascade Mask R-CNN

Cascade Mask R-CNN [7] was introduced by Cai and Vasconcelos, which is a multi-stage extension of Faster R-CNN [8]. Cascade Mask R-CNN [7] has a similar architecture as Faster R-CNN, but along with an additional segmentation branch, which is denoted as 'S' in Figure 4 for creating masks of the detected objects. Figure 4 display that the input image is passed through the ResNeXt-101 [15] backbone, which is explained in section 3.2. The backbone extracts the spatial features from the images and generates feature maps. The possible candidate regions where furniture objects might be present in the images are estimated by the region proposal network (RPN) head. These proposals are passed through the ROI pooling layer. The network head takes ROI features as input and makes two predictions - classification score (C) and bounding box regression (B). All three bounding box modules perform classification and regression. The output of one bounding box head is used as training input for the next head. These deeper detector stages are more selective against false positives even at higher IoU thresholds. Each regressor is optimized for the bounding box distribution generated by the previous regressor rather than the initial distribution. We get a bounding box of higher IoU thresholds when we train the bounding box regressor for a certain IoU threshold. We get refined bounding boxes and classification scores from B3, the segmentation head predicts the mask that contributes to the loss function to optimize the training further.

3.2. Backbone Network

We employ ResNeXt-101 [15] as backbone for our experiment. ResNeXt-101 [15] uses cardinality features as compared to its previous version, ResNets [22]. In ResNeXt, a layer is shown as the number of in channels, filter size, and the number of out channels. This

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2021 doi:10.20944/preprints202110.0089.v1

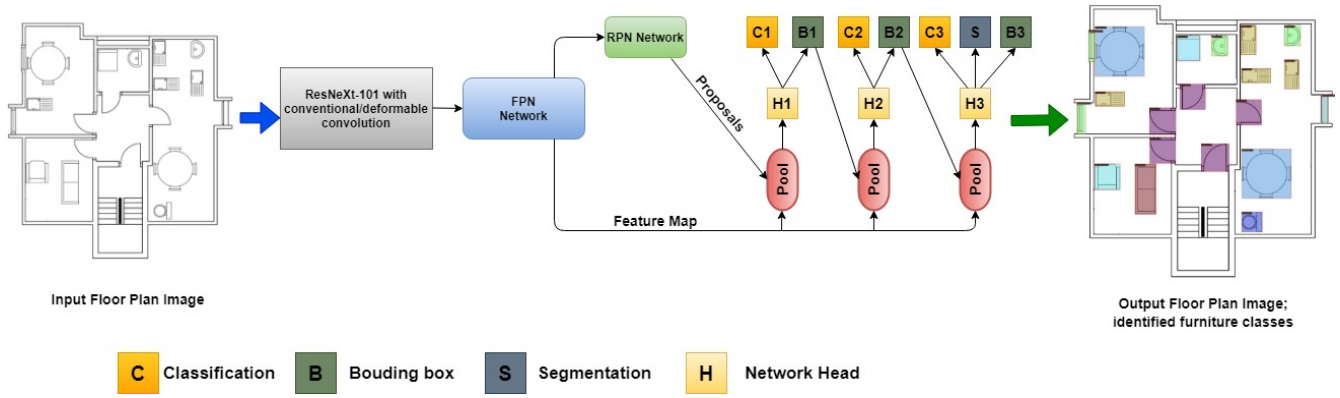

Figure 4. The presented framework is based on Cascade Mask R-CNN [7] equipped with ResNeXt-101 [15] backbone with conventional and deformable convolution applied on floor plan images. Modules B, C, and S represent bounding box, classification, and segmentation, respectively.

network stacks residual blocks. These blocks are subject to two simple rules: (i) if the same size spatial maps are produced, the blocks share the same hyperparameters, and (ii) every time when the spatial map is downsampled by a factor of 2, the width of the blocks is multiplied by a factor of 2. This ensures consistency in computation complexity in terms of FLOPs. In an artificial neural network, neurons perform inner product, which can be thought of as a form of aggregating transformation:

$$\sum_{i=1}^{D} w_i x_i \tag{1}$$

where x is the D-channel input vector to neuron and w_i is a filter's weight for the i-th channel. This has been updated in the ResNeXt [15] architecture with a more generic function, which can be a network in itself. Aggregated transformations are represented as:

$$f(x) = \sum_{i=1}^{C} \tau_i(x) \tag{2}$$

where $\tau_i(x)$ can be an arbitrary function that projects x into an (optionally lower dimension) embedding and then transforms it. The C is the size of transformations to be aggregated, referred to as cardinality. In equation 2 C looks the same as D in equation 1, but C need not equal D and can be an arbitrary number. This aggregated transformation serves as the residual function:

$$y = x + \sum_{i=1}^{C} \tau_i(x) \tag{3}$$

where y is the output which is then further propagated to the region proposal network of our Cascade Mask R-CNN as explained in Figure 4.

3.3. Deformable Convolution

Apart from conventional convolution available in ResNeXt-101 [15], we incorporate deformable convolution filters [14]. A convolutional neural network uses local connections to extract spatial information effectively and shared weights. Convolutional layers at higher levels identify complete objects, whereas layers at the bottom look for fine features like edges and corners of the gradients. In standard 2D convolution, we apply 2D filter/kernels over the input at the fixed receptive field and spatial locations to generate the output feature map. The output feature map is generated by a convolution operation between

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2021 doi:10.20944/preprints202110.0089.v1

kernel w and the input x, which can be formulated as $y = w \times x$ and every element in feature map y can be calculated as:

$$y(p_0) = \sum_{p_i \in C} w(p_i).x(p_0 + p_i) \tag{4}$$

where p_0 is the center location of the sample in the input, and p_i enumerates the points in the collection of sampling points. Because different locations in the input feature maps may correspond to objects with different scales or deformation, adaptive determination of receptive field sizes is desirable for certain tasks.

Deformable convolution has a learnable shape to adapt to changes in features; this is explained in figure 5. Deformable convolution makes the sampling matrix learnable, allowing the shape of the kernel to adapt to the unknown complex transformations in the input. Instead of using the fixed sampling matrix with fixed offsets, as in standard convolution, the deformable convolution learns the sampling matrix with location offsets. The offsets are learned from the preceding feature maps via additional convolutional layers. Thus, the deformation is conditioned on the input features in a local, dense, and adaptive manner. To put this into the equation, in deformable convolution, the regular sampling matrix C is augmented with offsets $\triangle p_i | n = 1, \ldots, n$, where $N = |C|$. Equation 4 becomes:

$$y(p_0) = \sum_{p_i \in C} w(p_i).x(p_0 + p_i + \triangle p_1) \tag{5}$$

where p_0 is the center location of the sample in the input, and p_1 enumerates the points in the collection C of sampling/offset points. Now the sampling is on the irregular and offsets locations $p_i + \triangle p_i$.

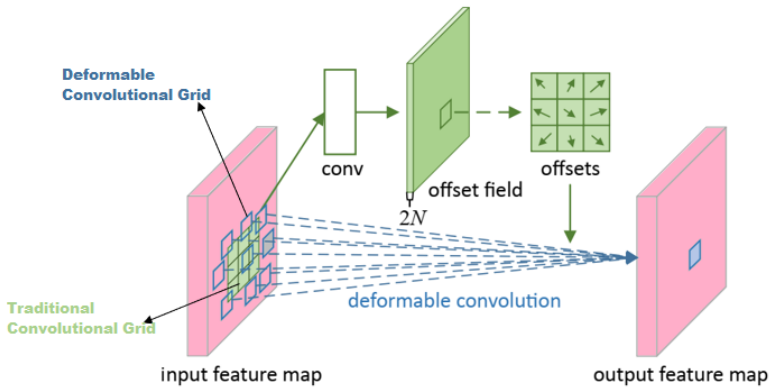

Figure 5. Internal working of deformable convolution [14]. The light green grid on the input feature map shows the conventional 3×3 convolutional operation, whereas the blue boxes on the input feature map show the effective receptive field of a 3×3 deformable convolution.

4. Dataset

The prior literature on floor plan images reflects that there is a scarcity of datasets in this area. SESYD [6], CVC-FP [17], and ROBIN [24] are the 3 most widely used publicly available datasets in this area. SESYD [6] contains synthetic floor plan images with furniture objects placed in different rooms randomly. It has 10-floor plan layouts and, in total, contains 1000 images. Although this idea is fascinating to put furniture objects randomly, the overall number of images is less. It has 16 different furniture classes.

In the CVC-FP [17] dataset, we only have 122-floor plan images. These floor plan images are distributed amongst four different floor plan layouts. Overall, the total number

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2021 doi:10.20944/preprints202110.0089.v1

of furniture classes is also less and limited to 4 classes. This dataset is not suitable for the training of the deep neural network.

In ROBIN [24] dataset, we have different hand-drawn as well as synthetically generated images. In this dataset also, we have only a limited number of images 510. Floor plan layouts and furniture object classes are also limited.

To train a deep detector, we need a dataset with sufficient images and variety in the floor plan layouts to generalize the network for realistic images. Also, the number of furniture classes should be large enough to identify different varieties of furniture objects. To address this, we create our custom dataset, which is based on SESYD [6] dataset. We named our custom dataset SFPI (Synthetic Floor Plan Images). From this point onward, we will describe our custom dataset with the name SFPI.

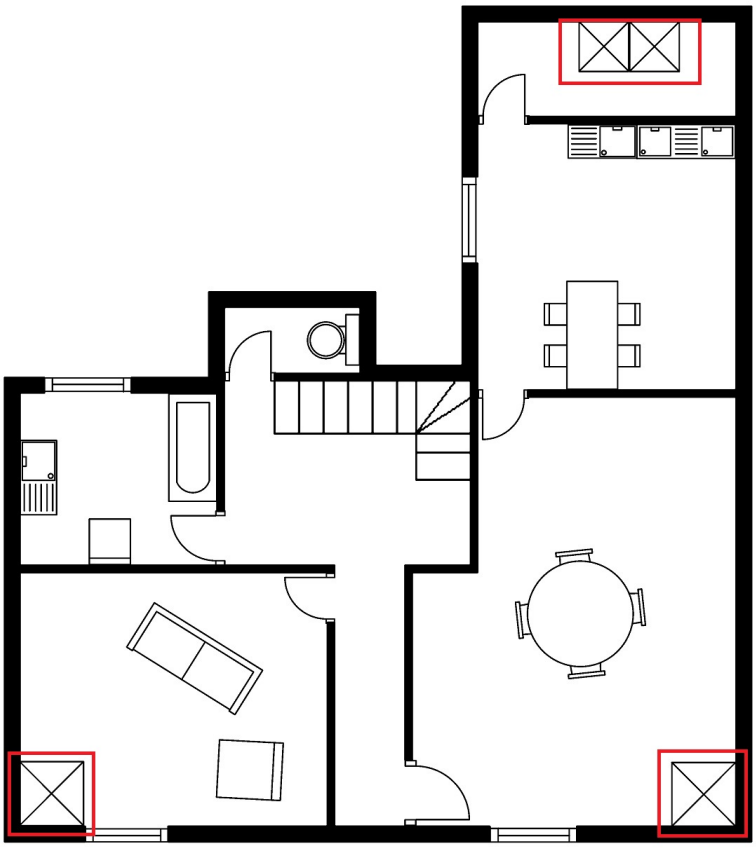

Figure 6. Sample image from SESYD [6] dataset. Minimal variation in the furniture objects. It is visible from the red highlighted objects of Table class that these objects do not vary in orientation or scale.

In Figure 6, we have a sample floor plan image from SESYD [6] dataset. It has various furniture objects present in different rooms. It is visible from the image that the scale of different furniture classes among them is almost the same and rarely varies; we observed this behavior in the full dataset as well. It is also noticeable that the orientation of different furniture objects also does not vary a lot. Also, we observe that few furniture objects are available in specific rooms, which is good if we want to identify room types but not for our purpose of detecting furniture objects. We want to generalize the model so that it can identify the objects available in any room. Keeping all these shortcomings of the SESYD [6] dataset in mind, we propose our dataset SFPI.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2021 doi:10.20944/preprints202110.0089.v1

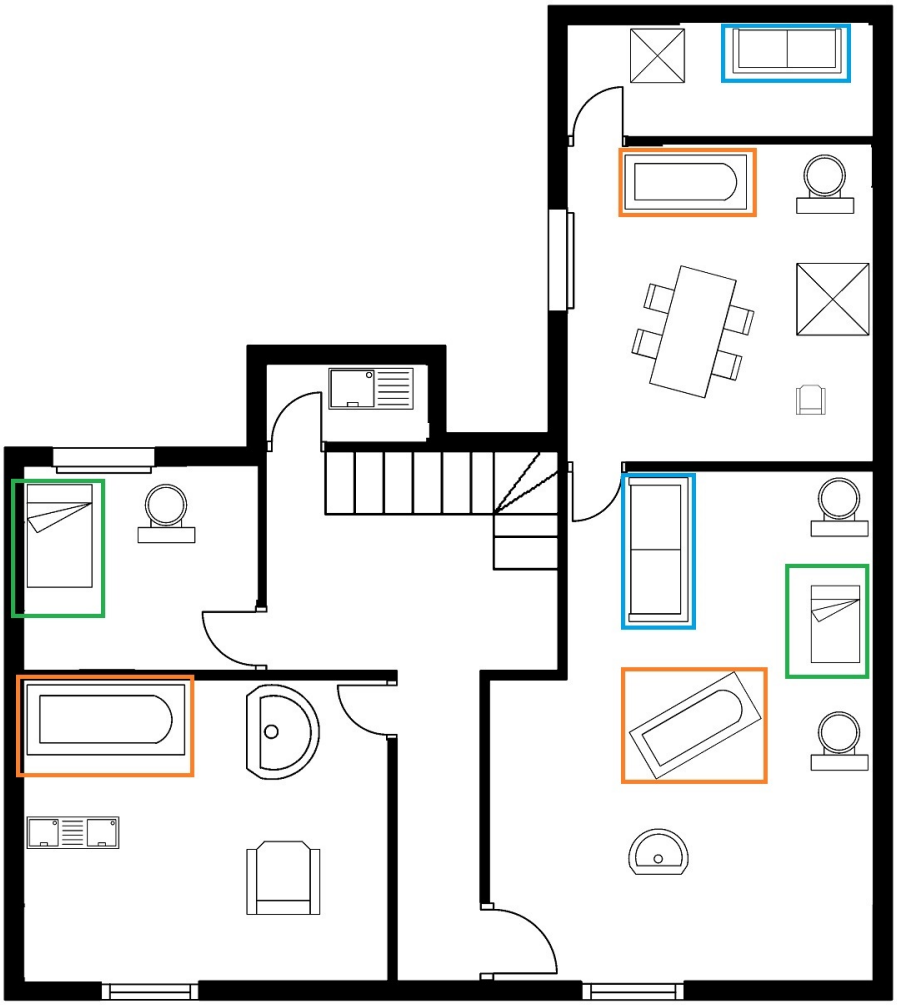

Figure 7. Sample image from the proposed SFPI dataset. Furniture object's augmentation is evident; i.e., highlighted Tub and Sofa classes have a couple of objects with different orientations and scales.

4.1. Dataset Creation

To generate the custom dataset SFPI, we took all floor plan layouts of SESYD [6] as the base and implanted the furniture objects to create different floor plan images. To overcome the shortcomings of SESYD [6], we take care of furniture object augmentations. The first augmentation we apply is rotation; we assign rotation randomly on furniture class objects. In this way, some objects will undergo rotation, and some will be the same as the original model. We use random angle choices between [0, 30, 45, 60, 75, 90, 120, 150, 180, 210, 250, 270, 300, 330] degrees, and rotate the image from its center. Figure 7 depicts the example of this augmentation; Sofa and Tub furniture class objects have different orientations based on randomly selected angels. Another augmentation choice we employ is scaling; this is an important step to ensure that the model generalizes well to identify natural furniture objects. We enforce scaling randomly to have both scaled and non-scaled furniture class objects in the same image. For scaling, we use a random resize factor between [40, 60, 75, 85, 100, 130, 145, 160, 185, 200] and scale the objects using this. We use the inter-are option for interpolation while resizing the objects. While resizing, we make sure to keep the original aspect ratio of the objects intact. In Figure 7, we have different scaling for bed and sofa objects of respective furniture classes.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2021 doi:10.20944/preprints202110.0089.v1

4.2. SFPI Statistics

Our SFPI dataset has ten different floor plan layouts and 16 different furniture classes, including armchair, bed, door1, door2, sink1, sink2, sink3, sink4, sofa1, sofa2, table1, table2, table3, tub, window1, window2. We also have multiple variants of the same class type to cover different varieties of furniture objects. It helps in generalizing the model for a more realistic output. We have created 10000 images. Each floor plan layout has 1000 images. Overall, we have 316160 objects of different furniture classes across these 10000 images. Figure 8 illustrates the furniture class distribution of our SFPI dataset.

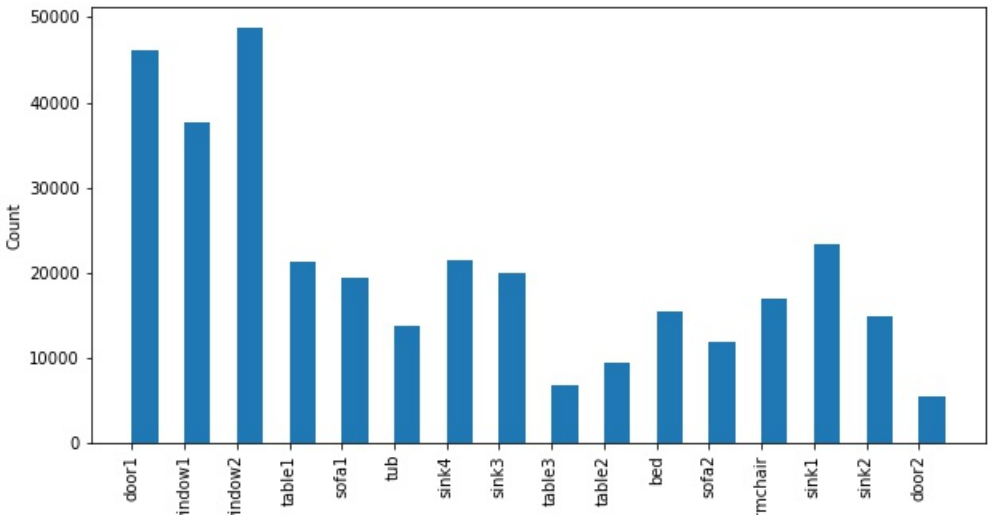

Figure 8. The distribution of different furniture class objects in the SFPI dataset.

Figure 8 depicts that doors and windows classes have the highest number of objects in the dataset. This is natural because, in each floor plan, the number of doors and windows is always higher than other individual furniture objects. We can identify that, at minimum, we have around 5000 representations of any class in our SFPI dataset.

Dataset	Objects	Train	Val	Test
SESYD [6]	20670	700	150	150
SFPI	316160	7000	1500	1500

Table 1: Statistical comparison between SESYD [6] and the proposed SFPI datasets. The distribution of images to train our base model on both datasets.

Table 1 explains the image distribution we use for training and testing our model. While working on the original SESYD [6] dataset, we take 700 images for training and 150 images for both validation and testing. Total 20670 different furniture objects are available in these 1000 images. While working on our custom dataset SFPI, we use the same 70-30 rule for splitting. We used 7000 images for training and 1500 images for both validation and testing. Now, as the number of images is increased, we have more furniture objects available. We performed multiple experiments using these two datasets; all information related to these experiments is available in section 5.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2021 doi:10.20944/preprints202110.0089.v1

5. Experimental Results

5.1. Implementation Details

We implement the proposed method using PyTorch and MMDetection's object detection pipeline [25]. Our backbone ResNeXt-101 [15] is pre-trained on MS-COCO dataset [23]. Using this pre-trained feature extraction backbone helps our architecture to adapt from the domain of natural scenes to floor plan images. We scale our input floor plan images to 1333×800, keeping the original aspect ratio. We use a batch size of one to train our network. The initial learning rate for training is 0.0025. We train the network for 12 iterations on our SFPI dataset. The IoU threshold value for cascaded bounding boxes is set to [0.5, 0.6, 0.7]. We use three different anchor rations of [0.5, 1.0, 2.0], strides of [4, 8, 16, 32, 64] and with only one anchor scale of [8] since FPN [26] itself performs multiscale detection because of its top-down architecture. We use Cross-Entropy loss for calculating network losses. Furthermore, we apply both traditional convolution, and DCN [14] backbone networks for different experiments. However, overall experiment settings for both are the same, apart from the choices of datasets. We trained on GeForce GTX 1080 GPU in coordination with 4 CPUs and with 25 GB memory.

5.2. Evaluation Criteria

As this is an object detection problem, we use the detection evaluation matrix of COCO [23]. The employed evaluation metrics are explained as follows:

5.2.1. Intersection Over Union

Intersection over Union (IoU) [27] is defined as the area of the intersection divided by the area of the union of a predicted bounding box (B_p) and a ground-truth box (B_{gt}):

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{6}$$

IoU is used as a criterion that determines whether detection is a true positive or a false positive.

5.2.2. Average Precision

Average precision [28] is based on the precision-recall curve. It is useful when we are comparing different detectors and the precision-recall curves intersect with each other. AP can be defined as the area under the interpolated precision-recall curve, which can be calculated using the following formula:

$$AP = \sum_{i=1}^{n-1} (r_{i+1} - r_i) p_{interp}(r_{i+1}) \tag{7}$$

where $r_1, r_2, \ldots, r_n$ are the recall levels at which the precision is first interpolated.

5.2.3. mAP

The calculation of AP only involves one class. However, in object detection, there are usually $K > 1$ classes. Mean average precision (mAP) [23] is defined as the mean of AP across all K classes:

$$mAP = \frac{\sum_{i=1}^{K} AP_i}{K} \tag{8}$$

5.2.4. Average Recall

Average recall (AR) [28] like AP can be used to compare detector performance. AR is the recall averaged over all $IoU \in [0.5, 1.0]$ and can be computed as two times the area under the recall-IoU curve:

$$AR = 2 \int_{0.5}^{1} recall(o) do \tag{9}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2021 doi:10.20944/preprints202110.0089.v1

where o is IoU and $recall(o)$ is the corresponding recall. For the COCO dataset, AR metric is calculated on a per-class basis, like AP.

5.2.5. Mean Average Recall (mAR)

Mean average recall [23] is defined as the mean of AR across all K classes:

$$mAR = \frac{\sum_{i=1}^{K} AR_i}{K} \tag{10}$$

5.3. Results and Discussion

We validate our proposed framework on both SESYD [6] and the SFPI dataset to demonstrate its effectiveness. In this section, we will discuss the quantitative and qualitative performance of our approach. We will discuss the strength and weaknesses of our model. Furthermore, we will compare our results with current state-of-the-art methods.

5.3.1. SESYD

For this dataset, we use a split of 70-30 as mentioned in Section 4. We use 700 random images out of 1000 for training the network, and from the remaining images, 150 for test and 150 for validation. We follow the evaluation protocol of COCO [23] performance metrics.

		Mean Average Precision (mAP)		**Mean Average Recall (mAR)**	
Model_Dataset	**Objects**	**Val**	**Test**	**Val**	**Test**
Our_SESYD	20670	0.981	0.982	0.986	0.987
Ziran et al. [11] - d1	1111	-	0.31	-	0.60
Ziran et al. [11] - d2	1111	-	0.39	-	0.69

Table 2: Quantitative analysis of our model with existing state-of-the-art methods.

All models mentioned in Table 2 are pre-trained on COCO [23] dataset. In the original dataset, where we have only 1000 images, our model can achieve good accuracy. We are able to achieve a 0.982 mAP score and 0.987 mAR score. Although we can not compare the result of Ziran et al. [11] directly with our results, as those experiments were performed on a different dataset, and they are not publicly available. However, from the domain perspective of furniture object detection, we can compare the methods. We can recognize that Cascade Mask R-CNN [7] outperforms Faster R-CNN [8] used by Ziran et al. [11].

5.3.2. SFPI Dataset

We perform multiple experiments with the SFPI dataset by dividing the dataset in different ways. Before we dive deeper into different experiments and their details, first, we lay down some experiment labels for better understanding in Table 3. These labels will be used throughout the paper.In general, the used naming convention is model_dataset_train_dataset_test.

	Experiment Label	**Train**	**Val**	**Test**
Experiment-1	Our_SFPI_train_test	SFPI	SFPI	SFPI
Experiment-2	Our_SFPI_train_SESYD_test	SFPI	SESYD [6]	SESYD [6]
Experiment-3	Our_SFPI_SESYD_train_SESYD_test	SFPI + SESYD [6]	SESYD [6]	SESYD [6]

Table 3: Explanation of different experiments and dataset associated with it.

In our SFPI dataset, we have 10000 images to perform experiments. First, we will present the results between the SESYD [6] dataset and our SFPI dataset in Table 4.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2021 doi:10.20944/preprints202110.0089.v1

		Mean Average Precision (mAP)		Mean Average Recall (mAR)	
Method	**Object**	**Val**	**Test**	**Val**	**Test**
Our_SESYD_train_test	20670	0.981	0.982	0.986	0.987
Our_SFPI_train_test	316160	0.995	0.995	0.997	0.997

Table 4: Quantitative analysis of our proposed model on SESYD [6] dataset and SFPI dataset.

For our SFPI dataset, we followed the 70-15-15 rule to split the dataset. We take 7000 images for training and 1500 images for validation and testing. With the number of increased images and objects, we can see the improvement in the results of our proposed model. We achieve a 0.995 mAP score and 0.997 mAR score. This clearly shows that our model performs better on the SFPI dataset where we have sufficient images to train a model as compared to less number of images we have in SESYD [6].

We further execute more experiments, including SFPI and SESYD [6] dataset, to get more generalized results from our end-to-end model.

		Mean Average Precision (mAP)		Mean Average Recall (mAR)	
Method	**Object**	**Val**	**Test**	**Val**	**Test**
Our_SESYD_train_test	20670	0.981	0.982	0.986	0.987
Our_SFPI_train_test	316160	0.995	0.995	0.997	0.997
Our_SFPI_train_SESYD_test	336830	0.751	0.750	0.775	0.775
Our_SFPI_SESYD_train_SESYD_test	336830	0.997	0.997	0.998	0.998

Table 5: Quantitative analysis of different experiments performed on our proposed model.

In the second experiment Our_SFPI_train_SESYD_test mentioned in Table 5, we use the full SFPI dataset for training, which means all 10000 images are used to train our end-to-end model. We use SESYD [6] dataset for validation and testing. We perform a random split on SESYD [6] dataset and use 500 images for validation and 500 for testing. In this way, we can compare how our network performs with a generalized dataset. Also, we can establish similarities and dissimilarities between our SFPI dataset and SESYD [6] dataset. We can achieve good results in this experiment if we compare it to the results of Ziran et al. [11].

Figure 9 is the output of the experiment Our_SFPI_train_SESYD_test. Few classes are misclassified; majorly network is confused between an armchair, sofa, and bed classes. We see many instances where sofa or armchair classes are recognized as the bed. This might be because of the data augmentation we put in the SFPI dataset, whereas in SESYD [6] furniture objects are not that much augmented, and in some scenarios sofa and armchair resembles a bed. To improve the results, we perform our next experiment Our_SFPI_SESYD_train_SESYD_test, where we use close domain fine-tuning. In Close domain fine-tuning, we fine-tune models using datasets that are closer to the domain of our problem rather than using natural images, which we do when we apply fine-tuning.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2021 doi:10.20944/preprints202110.0089.v1

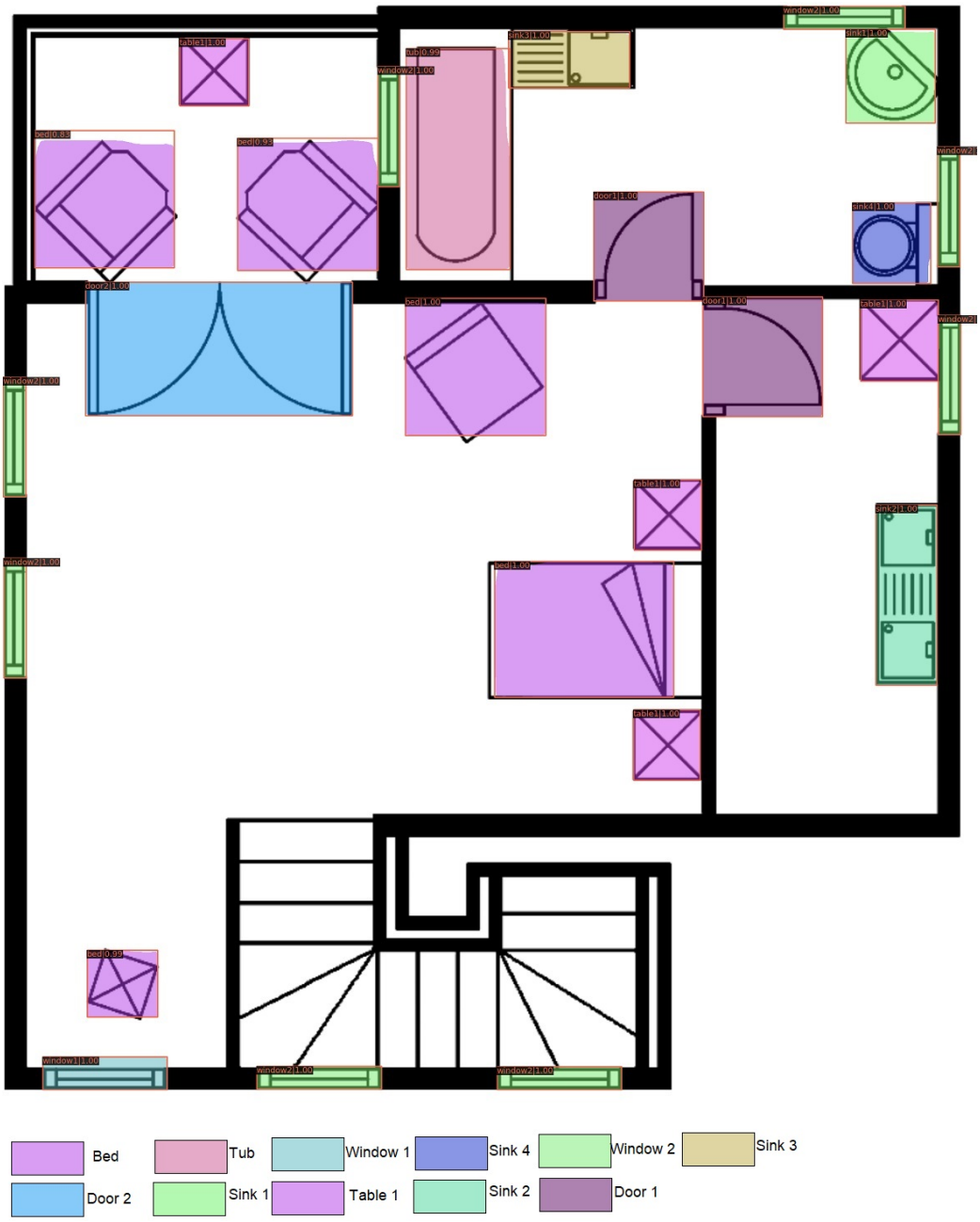

Figure 9. Qualitative results of our proposed model from experiment Our_SFPI_train_SESYD_test. Many miss-classified classes are visible in the image, like Bed, Sofa.

In our next experiment Our_SFPI_SESYD_train_SESYD_test we combine both the SFPI dataset and SESYD [6] dataset. For training, we use 10500 images, out of which 10000 images are from the SFPI dataset, and we pick 500 random images from SESYD [6] dataset. Out of the remaining 500 images of the SESYD [6] dataset, 250 are used for validation, and 250 are used for testing. With the close domain fine-tuning, our model improves, and we get better results. We can achieve a 0.997 mAP score and 0.998 mAR score, which is even better than our experiment Our_SESYD_train_test, where we used the SFPI dataset only for training, validation, and testing. This indicates the advantages of using closed domain fine-tuning.

Figure 10 is the final output of our proposed model in the case of experiment Our_SFPI_SESYD_

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2021 doi:10.20944/preprints202110.0089.v1

The image shows that all furniture objects are correctly classified with a good confidence score. In image 10 we can observe good furniture augmentation, as discussed earlier. Our proposed model can generalize well given the context of two datasets SFPI, and SESYD [6] object detection, and localization worked perfectly.

Figure 10. Sample output from experiment Our_SFPI_SESYD_train_SESYD_test. It is evident that all furniture objects are identified and localized correctly.

In Table 6, we described class-wise average precision score achieved in our experiment Our_SFPI_SESYD_train_SESYD_test. It is visible from the Table 6 that for few classes, we have reached the average precision of one like Door, Bed, and Tub; whereas for the other remaining classes score is high, and except Window1 class, all other classes are already reached above .90 average precision. This class-wise AP result gives us more clarity about model performance. We can identify where our model works well and what all classes are causing problems.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2021 doi:10.20944/preprints202110.0089.v1

Category	AP	Category	AP	Category	AP
Armchair	0.998	Bed	1.000	Door1	1.000
Door2	1.000	Sink1	0.997	Sink2	0.994
Sink3	0.994	Sink4	0.999	Sofa1	0.997
Sofa2	0.996	Table1	0.999	Table2	0.996
Table3	1.000	Tub	1.000	Window1	0.987
Window2	0.994	-	-	-	-

Table 6: Class-wise average precision (AP) from Our_SFPI_SESYD_train_SESYD_test experiment results. Few classes have reached the score of one, whereas there is some space for improvement in other classes.

For completeness of the paper, we computed the mAP score on various IoU thresholds ranging from 0.5 to 1.0. We performed this for all of our three experiments. Figure 11 illustrates the performance of our approach in terms of mean average precision. We can see that we can achieve mAP score of one for Our_SESYD_train_test, 0.861 in the case of Our_SFPI_train_SESYD_test, and 0.936 for Our_SFPI_SESYD_train_SESYD_test when the IoU is set to 0.5. From this point onwards, as we increase, the IoU mAP is decreasing for the latter mentioned two experiments. For experiment Our_SFPI_train_SESYD_test and experiment Our_SFPI_SESYD_train_SESYD_after the IoU threshold of 0.8, mAP is equal. mAP score eventually reaches zero when we set the IoU to 1 for all experiments.

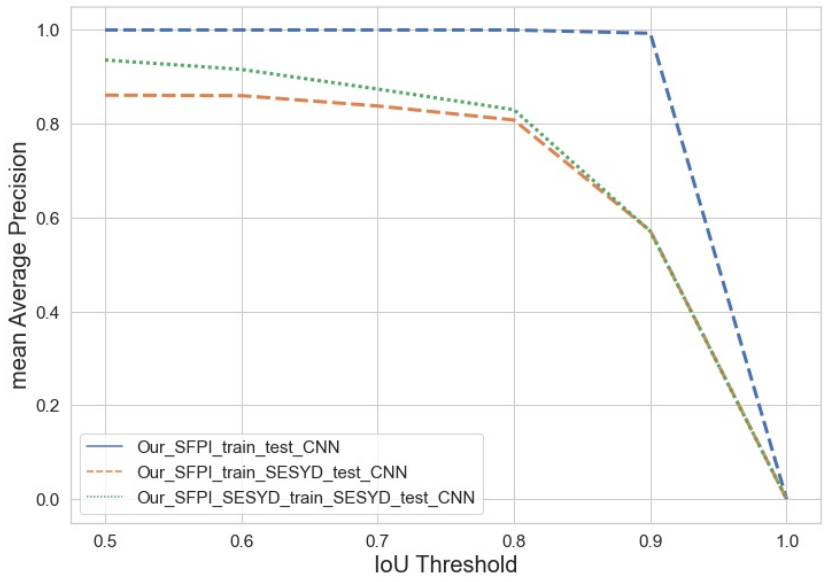

Figure 11. Mean Average Precision achieved over varying IoU thresholds for different experiments on the proposed method with conventional convolution (CNN).

Until this point, we were only deploying conventional convolutional networks, but we want to apply our model with deformable convolution network (DCN) [14] as well. DCN [14] could be useful for our datasets as they can easily adapt the shape of unknown complex transformations in the input. DCN's [14] are helpful when we have huge data augmentation in the dataset; to identify different transformations of the same objects. We performed all three experiments with backbone ResNeXt-101 [15] along with deformable convolutions [14]. All other specifications of the experiments like dataset split, Cascade

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2021 doi:10.20944/preprints202110.0089.v1

Mask R-CNN [7] remains the same as it is for backbone ResNeXt-101 [15] with traditional convolution (CNN).

		Mean Average Precision (mAP)		Mean Average Recall (mAR)	
Method	**Type**	**Val**	**Test**	**Val**	**Test**
Our_SESYD_train_test	CNN	0.981	0.982	0.986	0.987
Our_SFPI_train_test	CNN	0.995	0.995	0.997	0.997
	DCN [14]	0.998	0.998	0.999	0.999
Our_SFPI_train_SESYD_test	CNN	0.751	0.750	0.775	0.775
	DCN [14]	0.768	0.763	0.788	0.783
Our_SFPI_SESYD_train_SESYD_test	CNN	0.997	0.997	0.998	0.998
	DCN [14]	0.998	0.998	0.999	0.999

Table 7: Performance analysis of the proposed model on different experiments performed with conventional convolution (CNN) and deformable convolution (DCN) [14] backbone.

In Table 7, we present the quantitative analysis of all experiments we have performed with our end-to-end model. We can identify that using deformable convolution [14] enhances the results of our model. In our experiment Our_SESYD_train_test with conventional convolution (CNN), we can achieve mAP of 0.995 and mAR of 0.997, whereas when we changed the backbone to use deformable convolution [14] the overall score improves 0.998 for mAP and 0.999 for mAR, which is close to the perfect score. For our experiment, our_SFPI_train_SESYD_test where we are using SFPI dataset for training and SESYD [6] dataset for testing and validation; with CNN backbone, we get the score of 0.750 for mAP and 0.775 for mAR, whereas when we used a deformable convolution [14], the score is improved, and we get 0.763 for mAP and 0.783 for mAR. This indicates that deformable convolution can be helpful to get more generalized object detection. In our experiment Our_SFPI_SESYD_train_SESYD_test where we have performed closed domain fine-tuning on our model and achieved our best result till now, which was 0.997 score for mAP and 0.998 score for mAR. This further improves when we use a deformable convolution [14] for closed domain fine-tuning; we achieve a score of 0.998 for mAP and 0.999 for mAR. This is the best result among all experiments we performed on the SFPI dataset, as well as other experiments we came across during the literature survey for object detection in floor plan images.

We perform all our experiments with backbone ResNeXt-101 [15] combining deformable convolutions (DCN) [14] on different IoU thresholds. Figure 12 depicts the performance of our model during each experiment on different IoU thresholds. Our_SFPI_train_test and Our_SFPI_SESYD_train_SESYD_test are resulting in the same mAP score, whereas for Our_SFPI_train_SESYD_test we start with a mAP score of 0.881 for 0.5 IoU threshold. Our_SFPI_train_test and Our_SFPI_SESYD_train_SESYD_test gives constant mAP score till IoU threshold 0.9, whereas we see a constant decrease in the mAP score of Our_SFPI_train_SESYD_test. Eventually, all three experiments will end up on a mAP score of zero when we set the IoU to 1. The final output of experiment Our_SFPI_SESYD_train_SESYD_test is available in the Figure 13.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2021 doi:10.20944/preprints202110.0089.v1

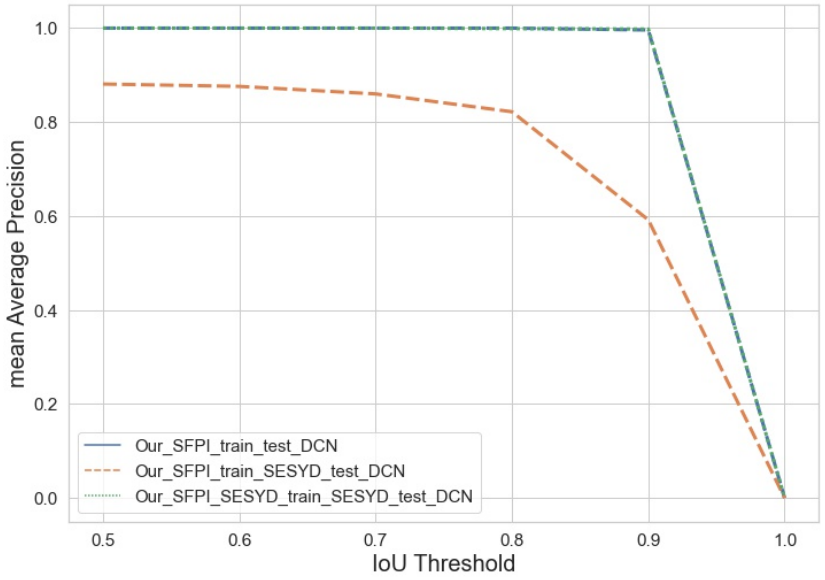

Figure 12. Mean Average Precision achieved over varying IoU thresholds for different experiments on the proposed method with DCN [14].

Figure 13. The qualitative result of our experiment Our_SFPI_SESYD_train_SESYD_test with deformable convolutions [14] on SFPI dataset.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 October 2021 doi:10.20944/preprints202110.0089.v1

Category	AP	Category	AP	Category	AP
Armchair	0.997	Bed	1.000	Door1	1.000
Door2	1.000	Sink1	0.996	Sink2	0.997
Sink3	0.999	Sink4	1.000	Sofa1	0.999
Sofa2	0.999	Table1	1.000	Table2	0.998
Table3	1.000	Tub	1.000	Window1	0.994
Window2	0.995	-	-	-	-

Table 8: Class-wise average precision (AP) from the results of our experiment Our_SFPI_SESYD_train_SESYD_test_DCN.

We can take a better look at individual furniture classes with respective accuracy in Table 8. Comparing this result 8 with the class-wise result we have obtained in Table 6 improvements are clearly visible. Comparison of these two class-wise results is available in the Figure 14. Figure illustrates that scores for Sink2 and Sink3 furniture classes have been improved. When we verify the images of these two classes, we can recognize that these classes have many similarities, and using DCN helps our model differentiate and recognize each class more precisely. We can also see the major improvement in Window1 and Window2 classes; these classes are also difficult to distinguish, and that is where we take advantage of deformable convolution [14] to improve the overall score. In conclusion, we can see that most of the furniture classes either have a score of one or close to one.

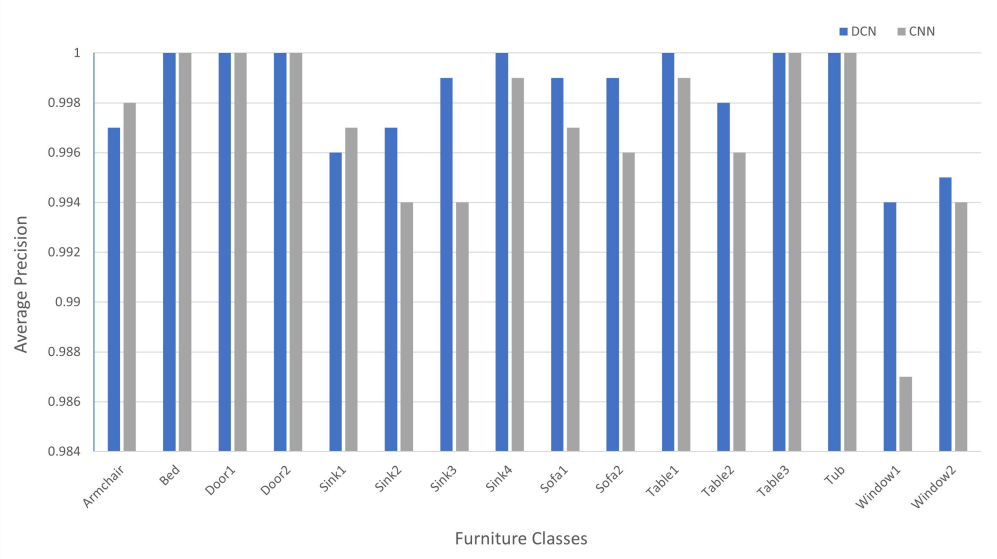

Figure 14. Class-wise average precision comparison Conventional Convolutional Network (CNN) and Deformable convolutional Network (DCN) [14]. The results have been taken from experiments Our_SFPI_SESYD_train_SESYD_test_CNN and Our_SFPI_SESYD_train_SESYD_test_DCN.

6. Conclusion and Future Work

We introduce an end-to-end trainable network for detecting furniture objects in floor plan images. Our proposed method incorporates the high-level architectural principle of traditional object detection approaches. Specifically, we exploit and compare traditional convolution and deformable convolution approaches to detect furniture objects in the floor plan images using Cascade Mask R-CNN [7]. Our different experiments and modifications will help to achieve better generalization and detection performance. We achieve state-of-the-art performance on COCO primary challenge matrices (AP at IoU=.50:.05:.95) with the mAP score of 0.998 on our SFPI dataset. With our proposed method, we achieved a mAP score of 0.982 on the publicly available SESYD [6] dataset. Our literature survey identified

no significant public dataset available for floor plan images that can be used to train deep learning detectors. We try filling this gap by creating a custom dataset SFPI containing 10000 floor plan images with 316160 object instances available in these images. There are 16 different furniture classes and ten different floor plans. This dataset can be further extended using our scripts and can quickly adapt to new furniture classes. Moreover, the presented work empirically establishes that it is possible to achieve state-of-the-art object detection in floor plan images.

For future work, we expect our SFPI dataset to be embedded with more floor plan layouts and different furniture objects to make a more generalized floor plan dataset. A deeper backbone would be able to improve the performance without using deformable convolution. Moreover, these experiments can be used in different floor plan applications like interactive image-fitting, floor plan text generation, helping visually impaired people with floor plans. Earlier, all these applications were using Faster R-CNN [8], but now with our experiments, it is evident that Cascade Mask R-CNN [7] will perform better in these applications.

Author Contributions: Conceptualization, S.M. and K.A.H.; writing—original draft preparation, S.M. and K.A.H.; writing—review and editing, S.M., K.A.H., and M.Z.A.; supervision and project administration, M.L., A.P., D.S. All authors have read and agreed to the submitted version of the manuscript.

Funding: The work leading to this publication has been partially funded by the European project INFINITY under Grant Agreement ID 883293.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kang, K.; Ouyang, W.; Li, H.; Wang, X. Object detection from video tubelets with convolutional neural networks. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 817–825.
2. Ahmed, M.; Hashmi, K.A.; Pagani, A.; Liwicki, M.; Stricker, D.; Afzal, M.Z. Survey and Performance Analysis of Deep Learning Based Object Detection in Challenging Environments. *Sensors* **2021**, *21*, 5116.
3. Gimenez, L.; Robert, S.; Suard, F.; Zreik, K. Automatic reconstruction of 3D building models from scanned 2D floor plans. *Automation in Construction* **2016**, *63*, 48–56.
4. Gimenez, L.; Hippolyte, J.L.; Robert, S.; Suard, F.; Zreik, K. reconstruction of 3D building information models from 2D scanned plans. *Journal of Building Engineering* **2015**, *2*, 24–35.
5. Ahmed, S.; Liwicki, M.; Weber, M.; Dengel, A. Automatic room detection and room labeling from architectural floor plans. 2012 10th IAPR international workshop on document analysis systems. IEEE, 2012, pp. 339–343.
6. Delalandre, M.; Valveny, E.; Pridmore, T.; Karatzas, D. Generation of synthetic documents for performance evaluation of symbol recognition & spotting systems. *International Journal on Document Analysis and Recognition (IJDAR)* **2010**, *13*, 187–207.
7. Cai, Z.; Vasconcelos, N. Cascade R-CNN: High Quality Object Detection and Instance Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2019**, p. 1–1. doi:10.1109/tpami.2019.2956516.
8. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence* **2016**, *39*, 1137–1149.
9. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. European conference on computer vision. Springer, 2016, pp. 21–37.
10. Redmon, J.; Farhadi, A. YOLO9000: better, faster, stronger. Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 7263–7271.
11. Ziran, Z.; Marinai, S. Object detection in floor plan images. IAPR workshop on artificial neural networks in pattern recognition. Springer, 2018, pp. 383–394.
12. Dodge, S.; Xu, J.; Stenger, B. Parsing floor plan images. 2017 Fifteenth IAPR international conference on machine vision applications (MVA). IEEE, 2017, pp. 358–361.
13. Lv, X.; Zhao, S.; Yu, X.; Zhao, B. Residential Floor Plan Recognition and Reconstruction. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 16717–16726.

14. Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; Wei, Y. Deformable Convolutional Networks. Proceedings of the IEEE international conference on computer vision, 2017.
15. Xie, S.; Girshick, R.; Dollar, P.; Tu, Z.; He, K. Aggregated Residual Transformations for Deep Neural Networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
16. de las Heras, L.P.; Ahmed, S.; Liwicki, M.; Valveny, E.; Sánchez, G. Statistical segmentation and structural recognition for floor plan interpretation. *International Journal on Document Analysis and Recognition (IJDAR)* **2014**, *17*, 221–237.
17. de las Heras, L.P.; Terrades, O.; Robles, S.; Síanchez, G. CVC-FP and SGT: a new database for structural floor plan analysis and its groundtruthing tool. *International Journal on Document Analysis and Recognition* **2015**.
18. Goyal, S.; Chattopadhyay, C.; Bhatnagar, G. Plan2Text: A framework for describing building floor plan images from first person perspective. 2018 IEEE 14th International Colloquium on Signal Processing Its Applications (CSPA), 2018, pp. 35–40. doi:10.1109/CSPA.2018.8368681.
19. Zeng, Z.; Li, X.; Yu, Y.K.; Fu, C.W. Deep Floor Plan Recognition Using a Multi-Task Network With Room-Boundary-Guided Attention. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019.
20. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition, 2015, [arXiv:cs.CV/1409.1556].
21. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934* **2020**.
22. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
23. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. European conference on computer vision. Springer, 2014, pp. 740–755.
24. GitHub, I. Open Source Survey. https://github.com/gesstalt/ROBIN, 2017.
25. Chen, K.; Wang, J.; Pang, J.; Cao, Y.; Xiong, Y.; Li, X.; Sun, S.; Feng, W.; Liu, Z.; Xu, J.; others. MMDetection: Open mmlab detection toolbox and benchmark. arXiv 2019. *arXiv preprint arXiv:1906.07155*.
26. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2117–2125.
27. Everingham, M.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The pascal visual object classes (voc) challenge. *International journal of computer vision* **2010**, *88*, 303–338.
28. Powers, D.M.W. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation, 2020, [arXiv:cs.LG/2010.16061].