

Chapter 1

Introduction

Detection of Objects in a Floor Plan and Architectural Images is a necessary task to understand the floor plan and Generating Descriptions. The major tasks accomplished in this project comprise of data collection and dataset preparation, training the deep learning model, and fine tuning of the model to improve the accuracy. The goal of the project is to detect and recognize the objects from floor plan images and generate a bounding box to localize the objects. We used existing models for the natural images and applied them to the document images of the floor plans. The models we used are 'Yolo', and 'Faster RCNN'. We collected our own data having many document images of floor plans developed our own dataset and improved the accuracy. We have also performed a comparative analysis of the models. We have developed a GUI tool which performs annotations for the DenseCap model and helps to generate region wise json annotations.

Abstract

Today there many models are available for doing object detection recognition in an image.

Like RCNN, fast RCNN, faster RCNN, mask RCNN, Yolo, SSD etc.

all of them are developed and configured for natural images. In this project we are working on document images of floor plans. In a floor plan image, we have objects like dining table, sofa, sink, etc. we used the yolo and faster RCNN for object detection.

Detecting a single object in an image is lot easier than detecting multiple objects in the images. But detection is not the only task we want to do. We also have to recognize the objects along with its location in the image. So, when we say multiple object detection in the image our primary task is to do detection, recognition, and localization of the image.

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 basic step for their conversion into other representations. in addition to this another contribution of this paper is the creation of dataset which is used for performing experiments and covers different types of floor plans.

we evaluate two object detection models originally designed and trained to recognize natural objects in images.

Chapter 2

Review of Literature

A basic method to detect object in the image is Convolution Neural Network (CNN). It passes the image to the network and it is then sent through various convolution and pooling layers. The final layers have the nodes equal to the number of different classes. Each node specifying one object class and we classify the object as class corresponding to the node having the higher value. MNIST data for recognizing handwritten digits is good example of recognizing single object in the image.

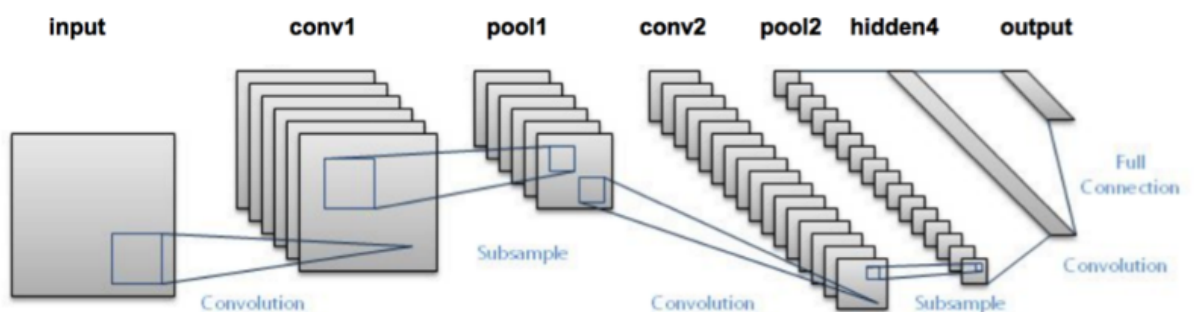


Fig 1: CNN

For recognizing multiple objects in the image, we can use the above technique to detect the multiple objects also.

The Basic algorithm to detect objects in an image -

1. First, we take an image as input.
2. Then divide the image into various regions.
3. Consider each region as a separate image.
4. Pass all the regions into CNN and classify them.
5. After classification we can combine all these regions to get the original images with the detected objects.

This task seems very easy, but it is not, the problem is how do you propose the regions.

Objects can have different sizes and special locations. In some cases, object may be small while in others may be large. To solve this problem, we can use various methods like region-based RCNN, YOLO, SSD etc.

2.1 Yolo (You Only Look Once)

Yolo is an object detection model targeted for real time processing of images

It divides the input images into an $S \times S$ grid and each grid cell predict only one object.

Each grid cell predicts fixed number of B bounding boxes with a confidence score.

Each bounding box contain five parameters $xmin, ymin, xmax, ymax$ which are coordinates to define a bounding box and a confidence score. Confidence score tells the accuracy of the boundary of bounding box and how confidently the box contains an object.

Each cell has N conditional class probabilities which tell the probability that object belong to a particular class where N is the number of classes of objects we have.

So, prediction has a shape of $(S, S, B \times 5 + N)$. We keep only those bounding boxes which has box confidence score higher than some threshold value. So, class confidence score for each predicted box is multiplication of box confidence score and conditional class probability.

From each grid cell only one bounding box is chosen which has highest IoU (Intersection over Union) with the ground truth and responsible for the object.[10]

Yolo use sum squared error between the prediction and the ground truth to calculate loss.

The loss function comprises of the

- 1) classification loss
- 2) localization loss
- 3) confidence loss.

Yolo has 24 convolution layers followed by 2 fully connected layers. It uses CNN network to reduce the spatial dimension to $S \times S$ with 1024 output channel at each location.

Fully connected layer is used to generate bounding boxes and flattened the tensor into desired output format.[1]

2.2 Faster RCNN

Faster RCNN uses region proposal network (RPN) to generate the region of interest. It passes each image to a Convolution network which generate the feature map for the image. A region proposal network (RPN) is applied on these feature maps which return the object proposals. it uses a 3×3 sliding window and each window location generates some bounding boxes, based on fixed-ratio anchor boxes and “objectness” score for each box. We now have proposals with different shapes and sizes which are passed to a RoI pooling layer to make all the proposals of the same size. Finally, these proposals are passed to a fully connected layer which has SoftMax and regression layer at its top to classify and output the bounding boxes for objects.[3]

Problem with faster RCNN-

Algorithm need many passes through a single image to extract all the features from the image which take a lot of computational time and as there are different system working one after the other, the performance of system depends on how the previous system has performed. This problem can be tackled by yolo model which look only once on the image and propose proposals. It is much faster than faster RCNN.[6]

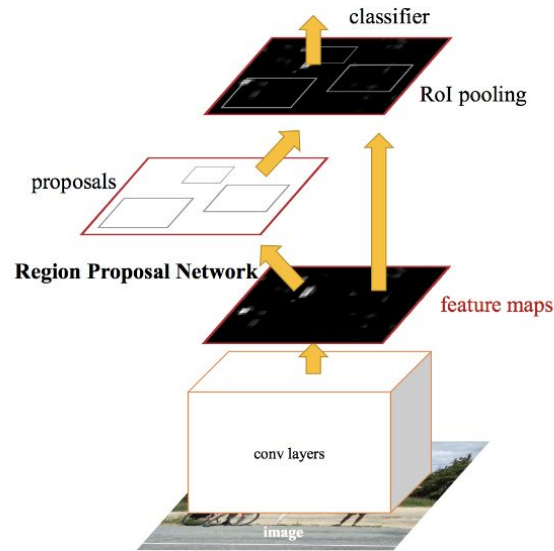


Fig-2. The internal architecture of faster RCNN

2.3 DenseCap

So far until now in object detection our focus was on three major task which we have already discussed detection, recognition and localization in our previous two models. Now we add one more task in object detection which is generating description regarding the objects.

So DenseCap perform object recognition and describe the salient regions in images in natural language. To address this task DenseCap uses a Fully Convolutional Localization Network (FCLN) architecture that processes an image with a single pass and requires no external regions proposals. The architecture is composed of a Convolutional Network, a novel dense localization layer, and Recurrent Neural Network language model that generates the label sequences.

this model requires different dataset format then our previous discussed method and of course dataset would be much more complex and bigger than yolo and faster RCNN.

DenseCap works on visual Genome dataset format. It stores the object annotation in json file format. It consists of seven main components. Region descriptions, objects, attributes of objects, relationship between two objects, region graph to describe region in graphical form, scene graph to describe entire image, and question-answer regarding regions and objects.[2]

Chapter 3

Data Collection

The most important step in any of the detection and recognition based deep learning models is perhaps the collection of data. We had to collect a large number of similar images of floor plans in order to get better accuracy. We also needed to keep in mind the structure of the data collected remained same, i.e., the data for one house should comprise of the floor plans, the description associated with the house plan and the specifications.

The method adopted for data collection was searching the websites which had similar structure for the data and allowed scraping the data, scraping the data, converting the images in proper format and organizing the collected data into folders.

The websites which were completely scraped were “architecturalhouseplans.com” and “houseplans.com”. Web scraping was done on using Python. The libraries used included ‘requests’, ‘urllib’, ‘Beautifulsoup’ and ‘Selenium’ with PhantomJs as webdriver for scraping,

‘image’ and ‘os’ for converting the image formats and csv for writing the specs in csv files moreover ‘os’ was also used for organization of data into proper folders.

Problems encountered during scraping processes included permissions for scraping (robots.txt) and in the event of internet connection failure the process had to be restarted from the beginning for organizational continuity. The way out to overcome this problem is writing robust codes which can help in resume scraping from the last link. An important etiquette was to keep the number of requests per second in check which is implemented by ‘wait’.

Floor plan data of 2906 houses with average of 2 images of different floor plans for different floors was collected from ‘architecturalhouseplans.com’ and floor plan data of 3681 houses with an average of 2 images of floor plans for different floors was collected from ‘houseplans.com’.

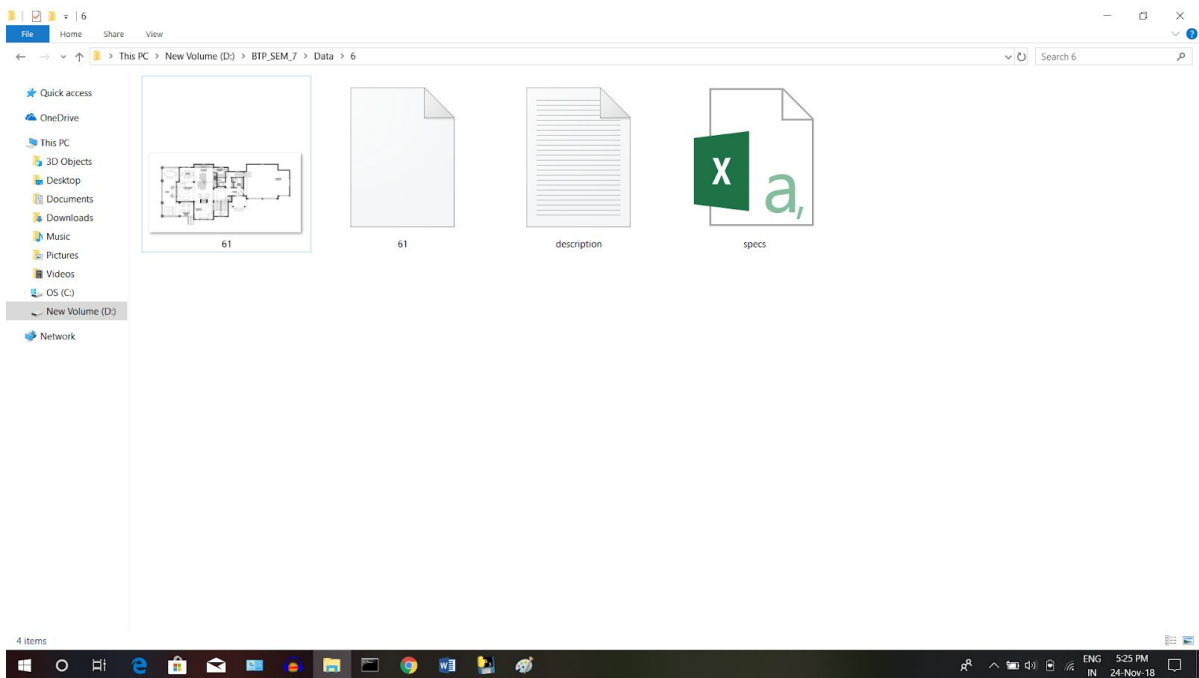


Fig 3: Collected data format for a house

Other sources for the collected dataset include ROBIN [12] and SESYD [11]. The dataset collected from ROBIN have 510 images with their annotations. The dataset collected from SESYD gave access to 1000 images.

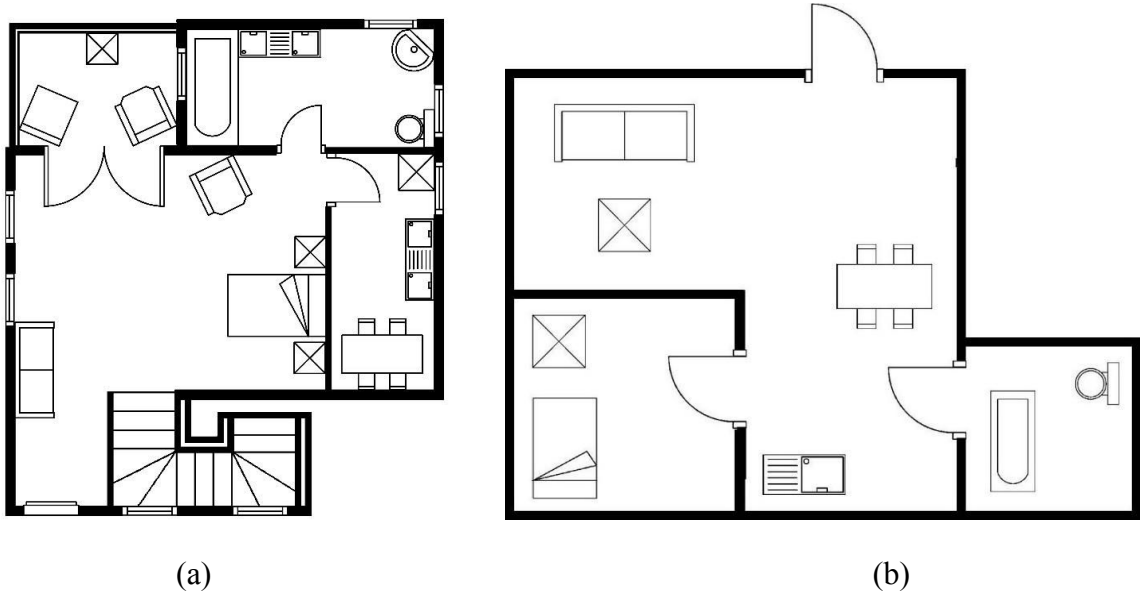


Fig 3.2: Floor plans from SESYD (a) and ROBIN (b)

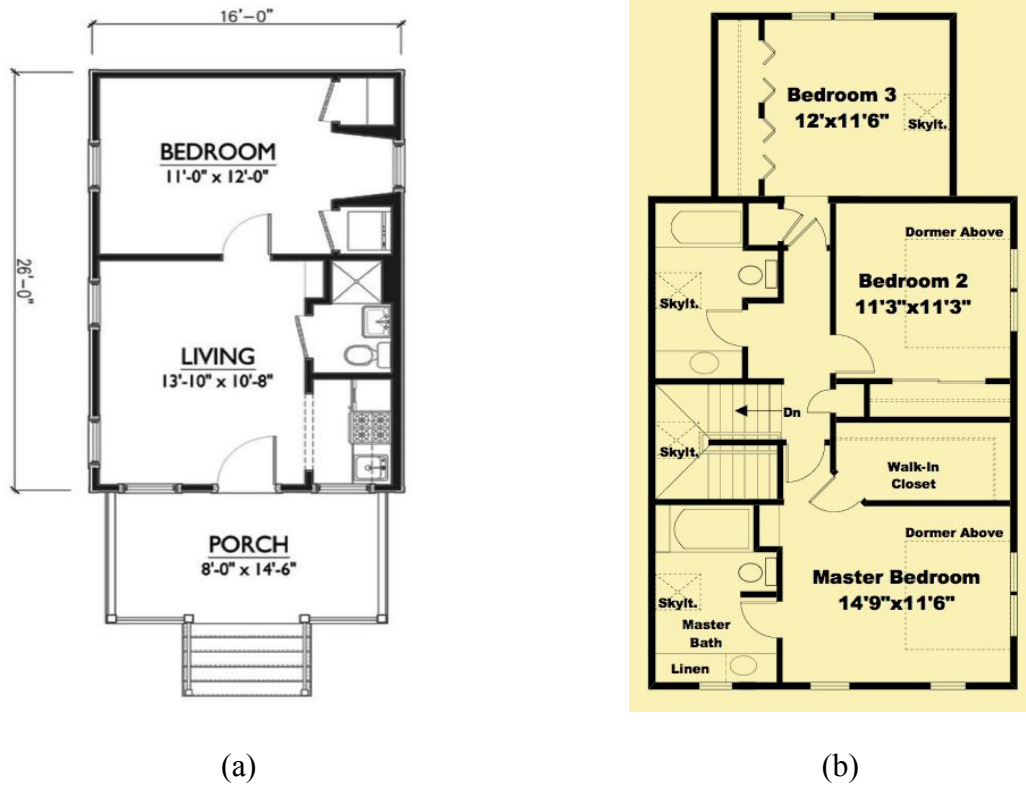


Fig 3.3: Floor plans from houseplans.com (a) and architecturalhouseplans.com (b)

Chapter 4

Building the Dataset

After collecting the data, the next step was to build the dataset for the model. Building of the dataset involved conversion of images to jpg format, annotating the images to generate xml files and then editing the generated xml files to point to the associated image.

The annotation process involved the categorization of objects into classes, drawing bounding boxes around every object in the image and labeling them according to the classes they fell into. The tool used for this was “labelimg”. The annotation was performed on 105 images in which the objects were labeled into ‘sink’, ‘bed’, ‘sofa’, ‘tub’, ‘door’, ‘window’, etc. and 50 images are annotated for generating region descriptions for denseCap model using the self developed tool explained section 4.3.

naming to each image is given according to the folder in which they reside. like if a folder number 23 contain 3 images then name of images would be 231,232,233 where last digit specifying image number in folder 23.

regions id in the images are generated in the same manner. if image 231 contain 3 regions then region ids would be 23101,23102,23103. 0 stand for separating region number from image name so 0 will always be there. if an image contains 12 regions then region id would be 231012.



Fig 5: Annotation Process using labeling

4.1 Data Preprocessing for Yolo Model

Yolo requires image and an xml file of the corresponding image containing the different objects and their bounding box coordinates in the image dataset.

Input format –

```
<annotation>
  <filename>000016.jpg</filename>
  <object>
    <name>dining_table</name>
    <bndbox>
      <xmin>686</xmin>
      <ymin>421</ymin>
      <xmax>1089</xmax>
      <ymax>778</ymax>
    </bndbox>
  </object>
</annotation>
```

SESYD dataset has images in tiff format so we first converted them to jpg format using ‘PIL’ library of python moreover it does not have the structure of xml file same as that of ROBIN. We also needed to perform a processing on sesyd xml files and convert them into desired structure for this task we used ‘lxml’ and ‘os’ library of python.

4.2 Data Preprocessing for Faster RCNN

Faster RCNN require image annotation input in a txt file and the format should be as-

Filepath,xmin,ymin,xmax,ymax,class_name

Since we had the data already in the xml format, so we had to convert it into the csv and then to txt format for input to the Faster RCNN model which we achieved using ‘BeautifulSoup’ and ‘csv’ libraries of python.

we modify the testing file in RCNN to store predicted classes , confidence and bounding box in a json file.

4.3 Data Preprocessing for DenseCap

For densecap we would be requiring region descriptions for each region stored in a json file along with its regions bounding box coordinates.

Format of this file is shown -

```
[
  {
    "Image_id": 212,
    "regions": [
      {
        "region_id": 21201,
        "image_id": 212,
        "x": 117,
        "y": 79,
        "width": 249,
        "height": 107,
        "phrase": "a cat sitting on a table.",
      }
      { So on .....
    ]
  }
  { So on....
}
```

We built our own dataset and created our own region descriptions file for the images we scrapped from various sites. To generate the descriptions of various regions in the images we created a tool Fig 6.

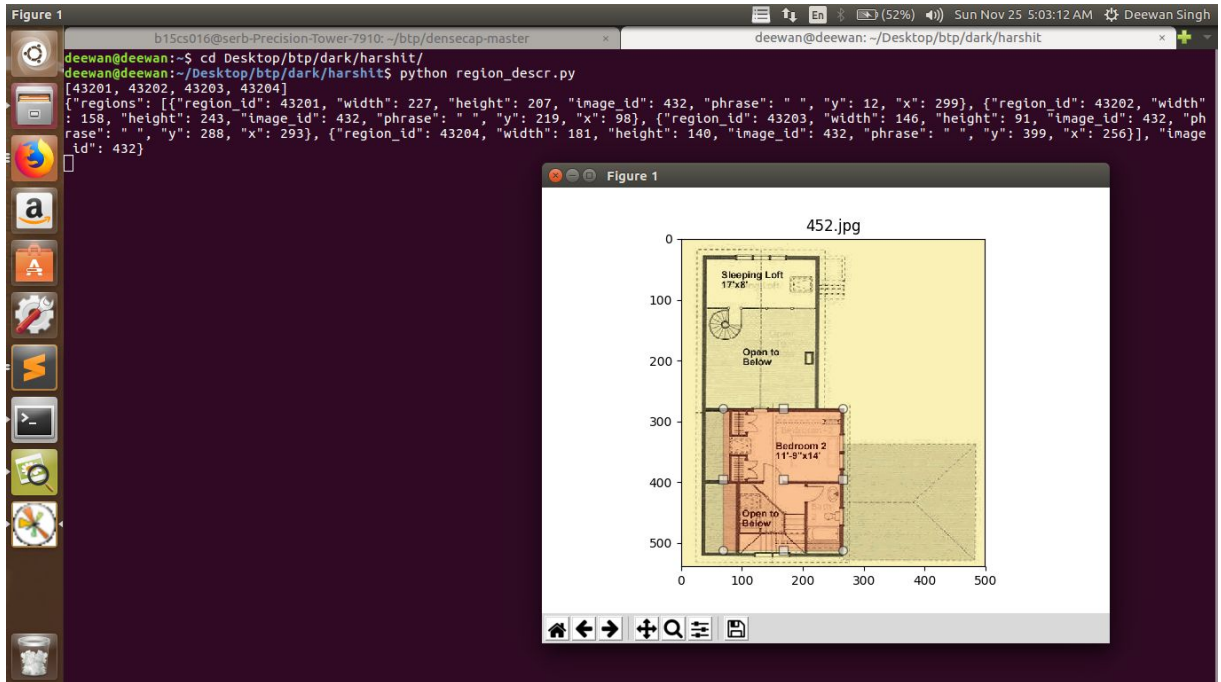


Fig 6: Self developed tool for region description annotation

Using this we can select regions, and it automatically generate the unique image id and region id along with bounding box coordinate, width and height and store them in a json file format. After getting this much details we add description to each region we generated. press the q key on keyboard and it will bring you to the next image.

Chapter 5

Training

Training of model is the most important and most time-consuming step of the project. We performed training on Yolo and Faster RCNN model by using 1600 images with corresponding annotations. We used 125 images for testing. For fine tuning of Yolo, we Changed number of classes to the number of objects. Changed the number of filters accordingly to produce the output in desired format. Specified names of objects in file. Yolo is implemented on natural images and we are working on documented images. Since natural images have high resolution than documented images, object detection is easier in natural images. So, to capture objects more accurately we increased the batch size. We also changed in threshold values to detect the objects having desired confidence score.

Chapter 6

Results and Discussions

The results obtained for the Yolo and Faster RCNN models are as follows:

6.1 Results and Accuracy for Yolo: -

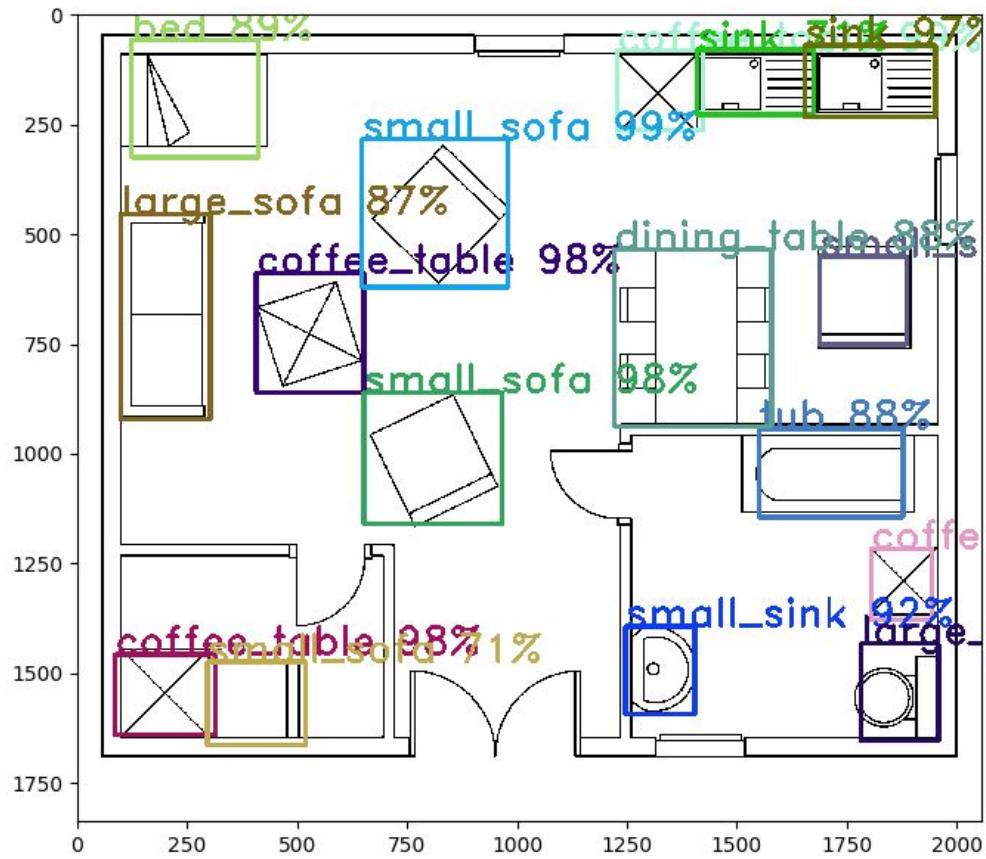


Fig 7: Objects recognized by Yolo model with their confidence percentage

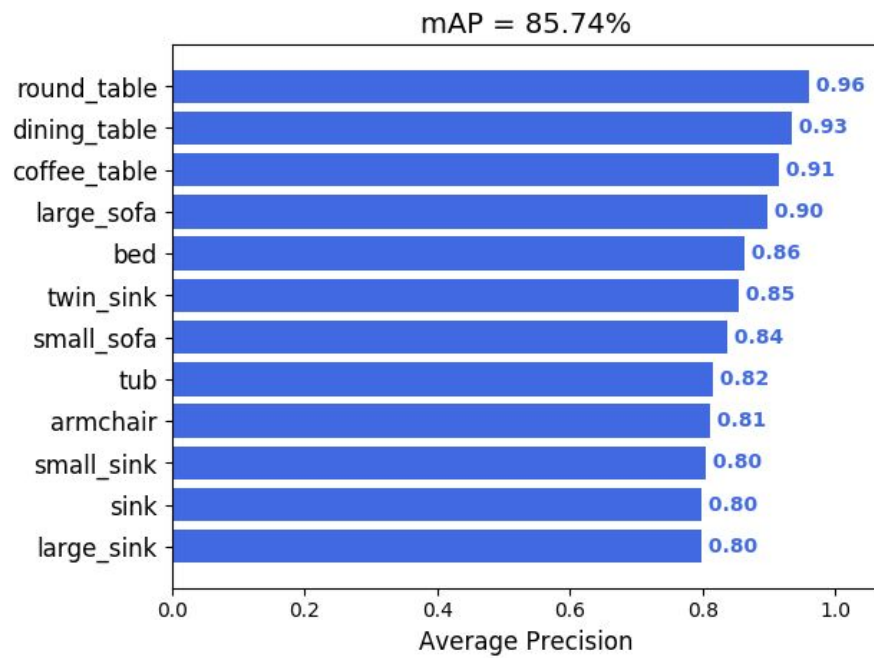


Fig 8: Graph for measuring accuracy of Yolo

6.2 Results and Accuracy for Faster RCNN

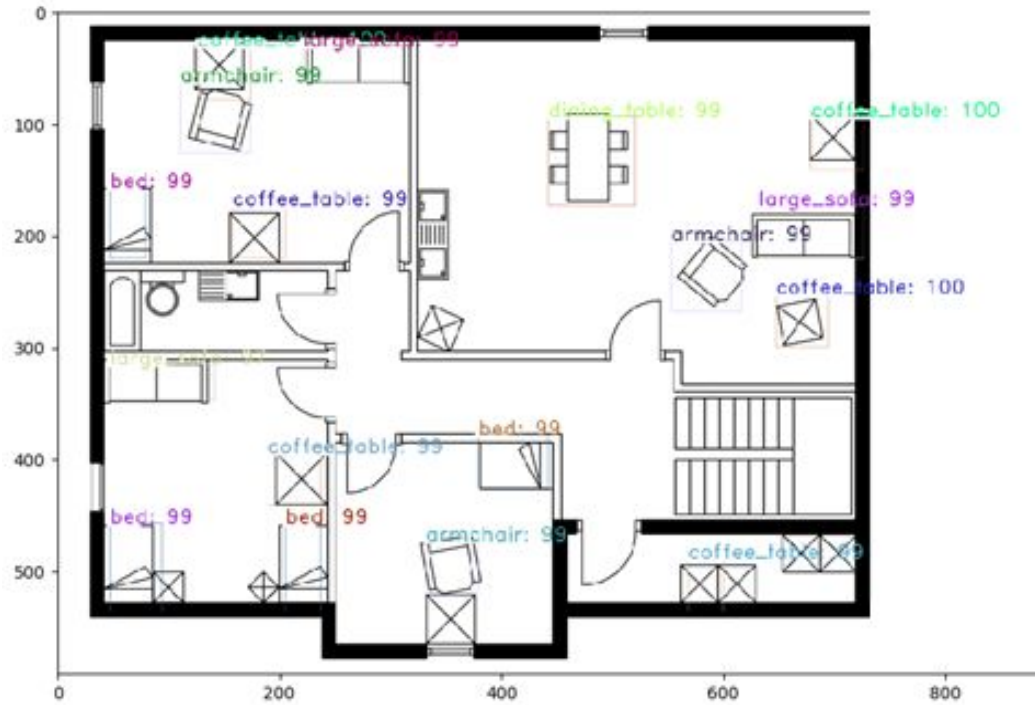


Fig 9: Objects recognized by Faster RCNN model with their confidence percentage

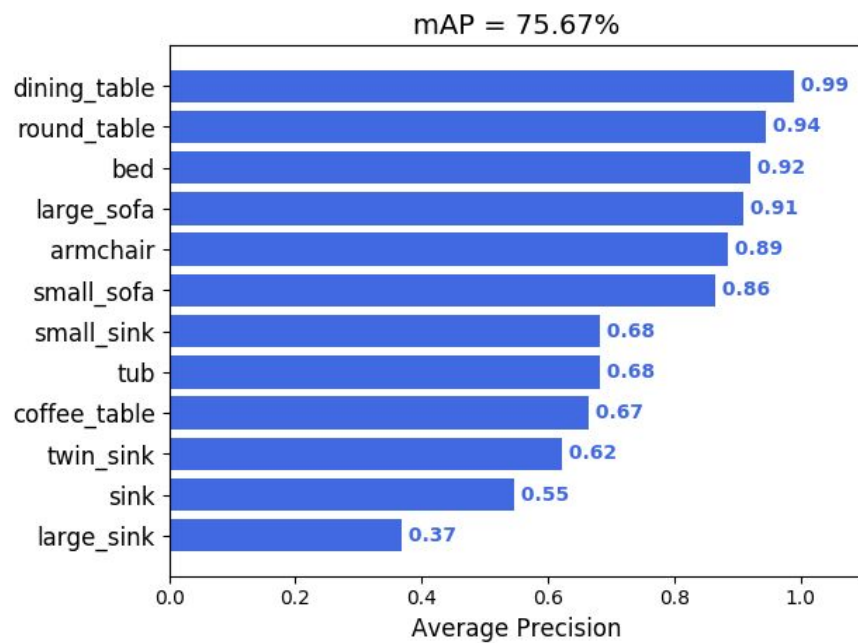


Fig 10: Graph for measuring accuracy of Faster RCNN

Calculation of Accuracy

To predict accuracy, we are considering the two factors one is predicted class object and another is bounding box predicted by object. To consider accuracy for bounding boxes we are using Intersection over Union method (IoU) which is defined as area of overlap divided by area of union of actual bounding box and predicted bounding box.

we are looking for 3 conditions.

1. 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 some threshold value, according to the following equation.



$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

Fig 10: Formula for calculation of IoU

2. The class label of the detection bounding box and the ground truth bounding box must be the same.
3. The probability of the object's recognition must be greater than some specific thresholds

ground truth objects that have not been detected are determined as false negatives. then computing the true positives and false negatives number for each category. average precision is calculated by $TP / (TP + FN)$.

Chapter 7

Summary and Conclusions

we created floor plan dataset to cover different architectural and drawing conventions of floor plans from many sites. different floor plans may have different representation for the same object. this is an issue and also affect the accuracy of models. we calculated average accuracy for each object type and then calculate the mean accuracy of all objects. we analysis the results

for Yolo and Faster RCNN models. The Faster RCNN model scans the input dataset multiple times and therefore is much slower in comparison to Yolo which scans the data only once. faster RCNN require more training then yolo and yolo has more accuracy then Faster RCNN. we can improve the accuracy by increasing size of dataset.

FUTURE WORK-

increase the dataset to improve the accuracy of object detection models.

after detecting models our focus is on generating the description of regions in the given floor plan image and represent the different regions in the image in graphically and identifying the relations between different objects in the floor plan.

and then by merging all these generate a graphical representation of the entire image.

References

- [1] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi.: “You Only Look Once: Unified, Real-Time Object Detection”.
- [2] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, Li Fei- Fei.: “Connecting Language and Vision Using Crowdsourced Dense Image Annotations”.
- [3] Zahra Ziran, Simone Marinai.: “Object Detection in Floor Plan Images”, Dipartimento di Ingegneria dell’Informazione (DINFO), Universit`a degli Studi di Firenze - Florence, Italy.
- [4] Joseph Redmon, Ali Farhadi.: “YOLO9000: Better, Faster, Stronger”
- [5] Fei-Fei Li, Justin Johnson, Andrej Karpathy,
“<https://cs.stanford.edu/people/karpathy/densecap/>”
- [6] Pulkit Sharma, Analytics Vidhya, “A Step-by-Step Introduction to the Basic Object Detection Algorithms”,
“<https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/>”
- [7] Kbardool, “<https://github.com/kbardool/keras-frcnn/blob/master/README.md>”
- [8] Thtrieu, darkflow, “<https://github.com/thtrieu/darkflow>”
- [9] Arthur Ouaknine, “Review of Deep Learning Algorithms for Object Detection”,
“<https://medium.com/comet-app/review-of-deep-learning-algorithms-for-object-detection-clf3d437b852>”
- [10] Jonathan Hui, “Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3”,
“https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088”
- [11] SESYD, “<http://mathieu.delalandre.free.fr/projects/sesyd/symbols/floorplans.html>”
- [12] ROBIN dataset, [online] Available: “<https://github.com/gesstalt/ROBIN.git>”

Acknowledgements

[1] Dr. Chiranjoy Chattopadhyay, Assistant Professor, Department of Computer Science and Engineering, Indian Institute of Technology Jodhpur.

[2] Shreya Goyal, PhD student, Department of Computer Science and Engineering, Indian Institute of Technology Jodhpur.