CSE 40536, Computer Vision II

Spring Semester, 2020
Final Report
Due Date: May 8th 2020

Sophia Abraham, Bhakti Sharma, Ying Qiu

1 Introduction

This project aims to visually identify objects within a dense clutter. Specifically, 10 individual objects (target objects) with different sizes, shapes, colors, textures, and material composites placed in a plastic tote, with some irrelevant items mixed in with the target objects. Various lighting conditions, data collection sensors (i.e., low quality webcam, high quality iPhone camera), and varied image capture angels Incorporated additional everyday challenges to consider for this detection task. Prior to developing proper methodologies to accomplish this task, there are a few important points to consider. Since the objects are all contained within a tote of fixed size, approximately 24" x 16" x 11" in dimension, the following should be noted:

- Occlusions will naturally occur, and fine-grained considerations should be set in place for small target objects which can be significantly occluded by the irrelevant items or larger target objects or even appear completely absent from the image frame.
- Target objects may appear in differed angles from what was originally captured due to the nature of a dense clutter. They may appear in a squeeze or tilted orientation due to the limited space in the tote. Models should be robust against these variations which differ from what was captured from the individual objects.
- The dimensions and shapes of the target objects may also vary across different image capture angles and models should be able to take this into consideration.
- The image size and resolution of the images will vary since two different camera sensors are utilized and performance should not degrade significantly across varied resolution.
- Every item may not be able to be captured for the input, given an unknown input the model should provide no detection or low confidence detections on unknown inputs.

While the idea of using traditional computer vision methods would be favorable, it was difficult to utilize the more traditional methods since the amount of available data was limited. Unique data augmentation techniques such as generating simulated clusters based off the individual items are potential possibilities that could be explored to accomplish the task without deep learning, however given the time constraints, this methodology only considers deep learning methods to rapidly prototype a solution which learns the features from the inputs to be able to provide detections.

However, different deep learning models utilize vastly different methodologies and techniques to perform detections. Potentially, there is pertinent information for performing the detection that one model is able to extract that another is not able to. Thus, we took inspiration from this idea and decided to experiment with 3 different deep learning models: MaskRCNN, YOLO and Single Shot Detection (SSD). Although given the time we were unable to perform a true fusion of the three models, an analysis of the performance and results indicates the variation among the models in the processing of detections and decision pipeline. This variation among the results may indicate the possibility of a combined fused model extracting varied pertinent features that can result in richer and more accurate detections. The specifications for each model and rationale for its selection are further illustrated in the sections below.

As with any deep learning model, we aspired to produce a model that could generalize and perform well independent of the input data. This proved to be an incredibly challenging goal and multiple data based experiments were developed regarding the limited data to extract key insights about the performance and robustness of these models to the varied illumination, scale, rotation and sensor properties. In addition, each model was fine tuned and specific discoveries for the individual models are included for analysis.

Data Collection

The input images collected for this project include a set of plastic tote images and a set of individual (target) object images. There are also multiple unrelated objects mixed with 10 target objects in the plastic tote, which makes this object detection task more difficult. For the object configuration (layout) in the tote, there are 6 conditions and 2 types of cameras as indicated in Table 1. After one configuration, the object configuration in the tote was changed and then took another set of tote images. Therefore, there are at least 120 tote images. At least 10 images were collected for each target object with 2 types of cameras, which yields 200 individual images.

Conditions	**Webcam**($Logitech - c615$)	**Mobile**($iphone$)
`Top view + Top light`	5	5
`Top view + Side light`	5	5
`Top view + Ambient light`	5	5
`side view + Top light`	5	5
`side view + side light`	5	5
`side view + Ambient light`	5	5

Table 1: Tote images collection ($minimum$)

Data Experiments:

1. *Experiment 1:* Train on individual item images and high resolution (captured from iPhone) tote and validate on low resolution tote images (captured with a webcam).
2. *Experiment 2:* Train on individual items and all tote images from one angle (top view) and validate on all images from a separate angle (side view).
3. *Experiment 3:* Train on individual item images and all tote images from one layout and validate on the second layout.

2 Mask-RCNN

While methods like YOLO and SSD detects objects and finds their positions based off coordinate bounding boxes, Mask-RCNN has the added benefit of semantic segmentation which pertains to pixel based classification. Since occlusion is a major thing to consider in this task, the semantic segmentation could potentially assist detecting objects even if it is heavily occluded. Depending on how well the model learns, even small items that can appear minimally visible in the field of view can be found with semantic segmentation.

Mask-RCNN[1] is an extension of Faster-RCNN which is a region-based convolutional neural network, which like SSD and YOLO, returns bounding boxes for objects with a class label and confidence score. There is an added branch for predicting an object mask in parallel with the bounding box recognition which predicts segmetation masks on each Region of Interest (ROI).

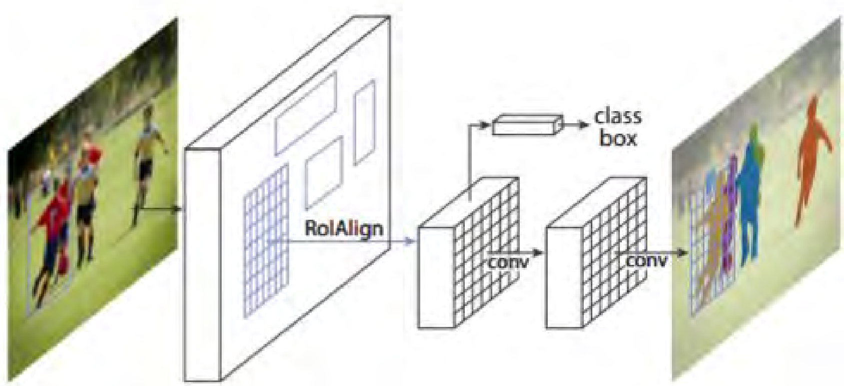


Figure 1: Mask R-CNN framework for instance segmentation. [1]

Mask-RCNN utilizes the same 2 stage procedure of Faster-RCNN which utilizes a region proposal network to propose regions of the feature map which contain the candidate object and predicts bounding boxes for the proposed regions. With Mask-RCNN, an RoIAlign layer is utilized to preserve the spatial information whereas RoIPool can cause misalignment. The output from this layer is fed into convolutional layers1 which generates a mask for each RoI and performs pixel based segmentation in addition to providing bounding boxes with labels and confidence scores.

Mask-RCNN was an easy choice for this task. Not only did it have the added benefit of instance segmentation which differed from the other models we were testing, but it was simple to train, generalized across tasks and produced state of the art results in multiple challenges including COCO 2016 Detection Challenge.

I attempted to analyze many aspects of Mask-RCNN. As mentioned in the introduction, the three experiments were included in order to see the robustness against variables such as illumination, scale, rotation and the sensors.

In addition to these three experiments, other methods were utilized in order to fine tune the model. There are numerous parameters that can be trained within Mask-RCNN. In addition to the usual ones such as learning rate, weight decay and momentum, additional parameters included the backbone, anchor size, backbone stride, gradient clipping just to name a few.

For the purpose of this project, the following parameters and aspects of Mask-RCNN were further explored:

- Backbone: ResNet50 vs. ResNet101
- Anchor size tuning, based off the methods in [4]
- Batch size 1 vs. 2 vs. 4 vs. 5
- Scaling individual items and data augmentation techniques
- Gradient clipping 5 vs 10
- Region proposal network tuning

After attempting to use different hyperparameter tuning methods such as hyperopt, the training methods consisted of 30 - 50 epochs, learning rate of .001, and momentum of 0.9. Other details are included in the configuration file for the source code. This was decided upon in order to limit the time for training and analyze results based off of minimal training. The results depicted below show the experimentation results for ResNet50 with a batch size of one to illustrate results based off a simple training schedule with limited data. We utilized average precision score to compare the models. The following were the computed AP's for each experiment using ResNet50 backbone:

- Experiment 1: AP @0.50-0.95: 0.409
- Experiment 2: AP @0.50-0.95: 0.486
- Experiment 3: AP @0.50-0.95: 0.050

Results for Experiment 1: Training on High Resolution (iPhone) and Validating on Low Resolution (webcam)

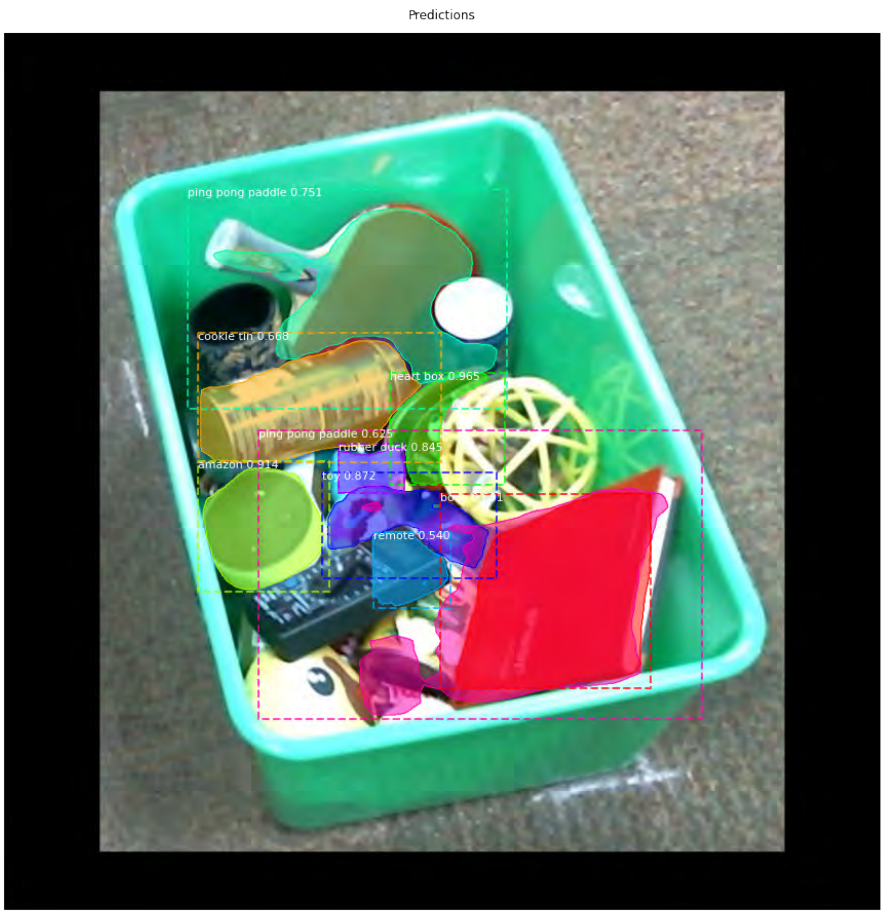


Figure 2: Detection masks for Experiment 1.

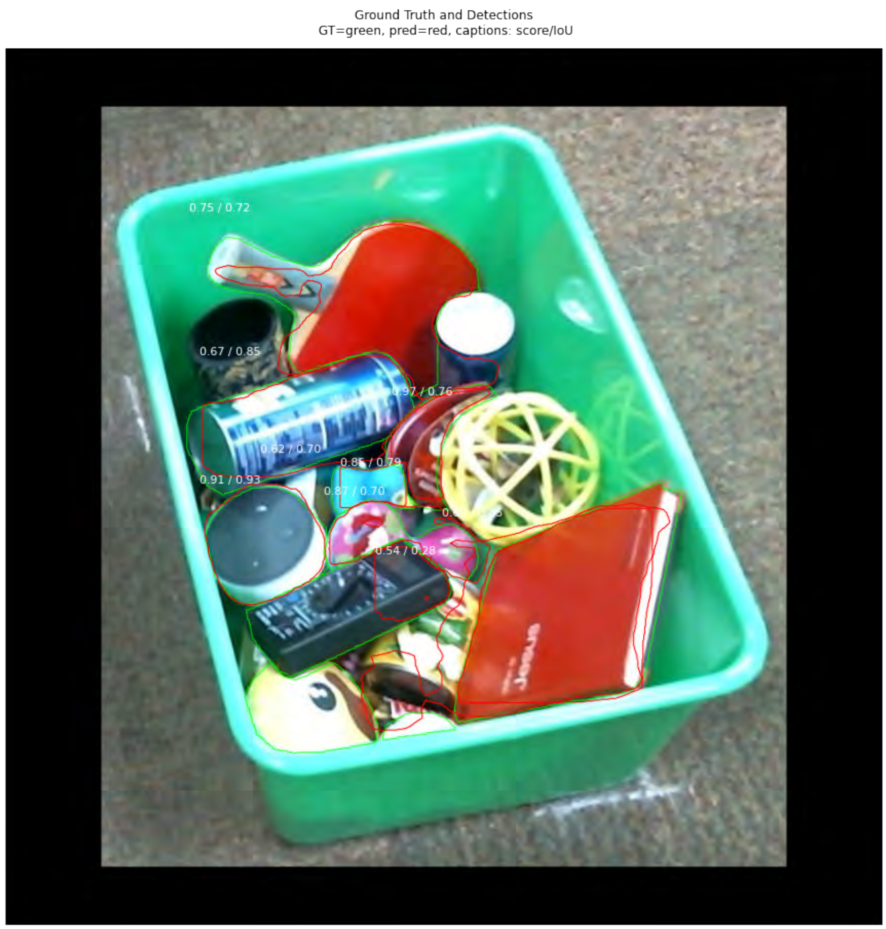


Figure 3: IOU detection for Experiment 1.

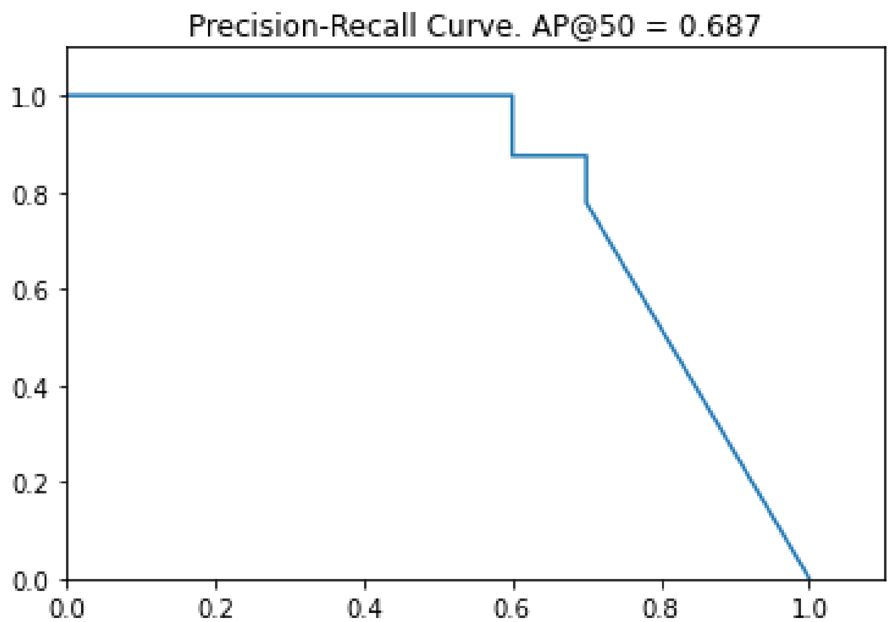


Figure 4: Precision plot for Experiment 1.

Figure 5: Confusion matrix for Experiment 1.

Results for Experiment 2: Training on One Angle (Top) and Validating on Different Angle

(Side)

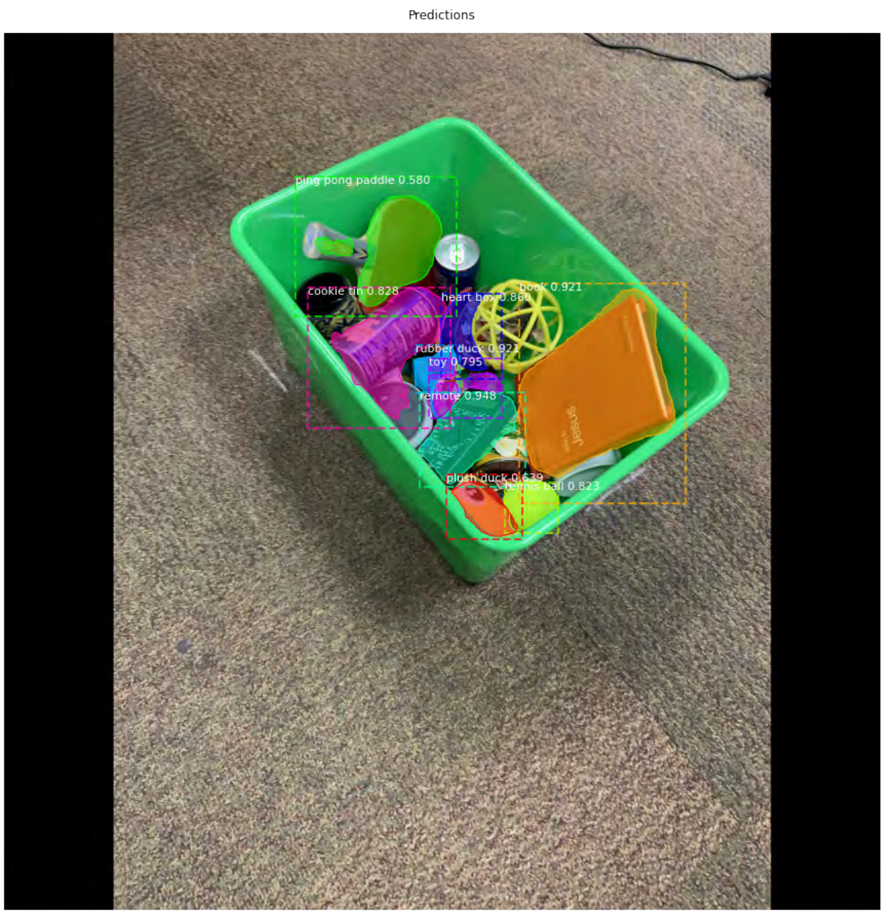


Figure 6: Detection masks for Experiment 2.

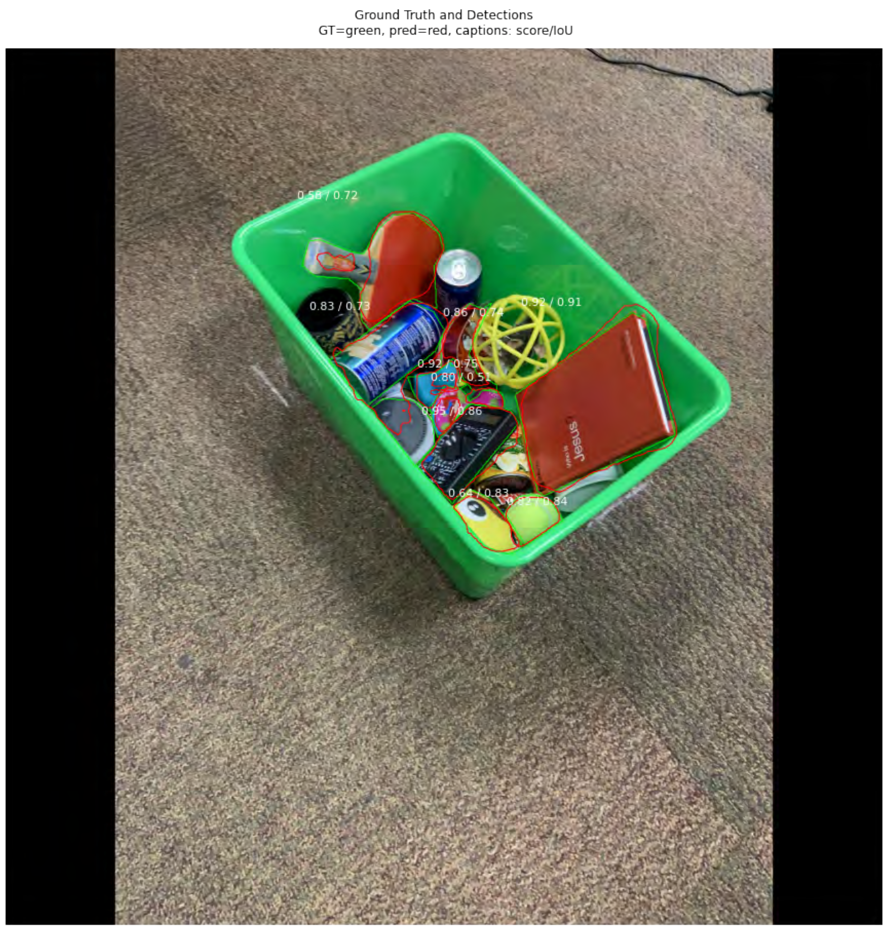


Figure 7: IOU detection for Experiment 2.

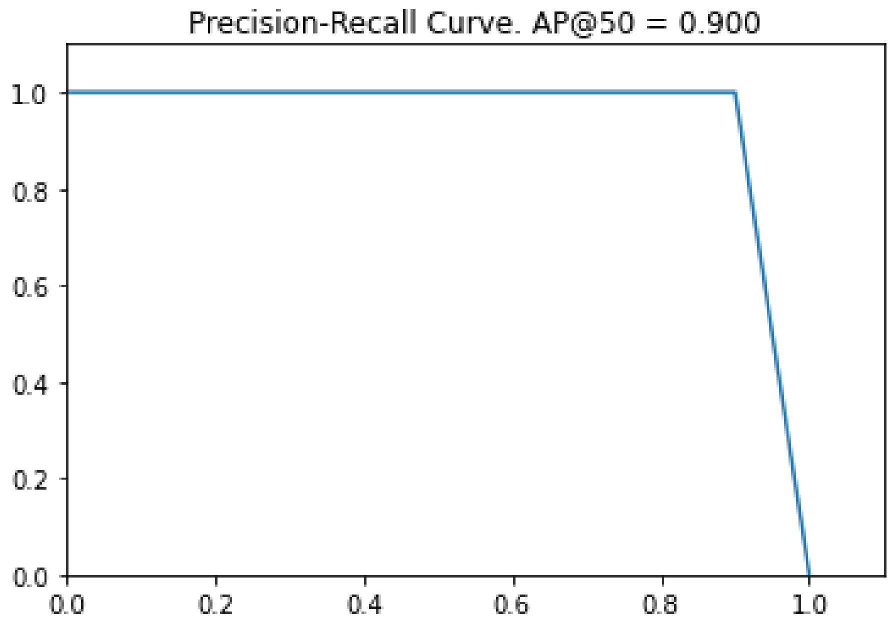


Figure 8: Precision plot for Experiment 2.

Figure 9: Confusion matrix for Experiment 2.

Results for Experiment 3: Training on One Layout and Validating on Different Layout

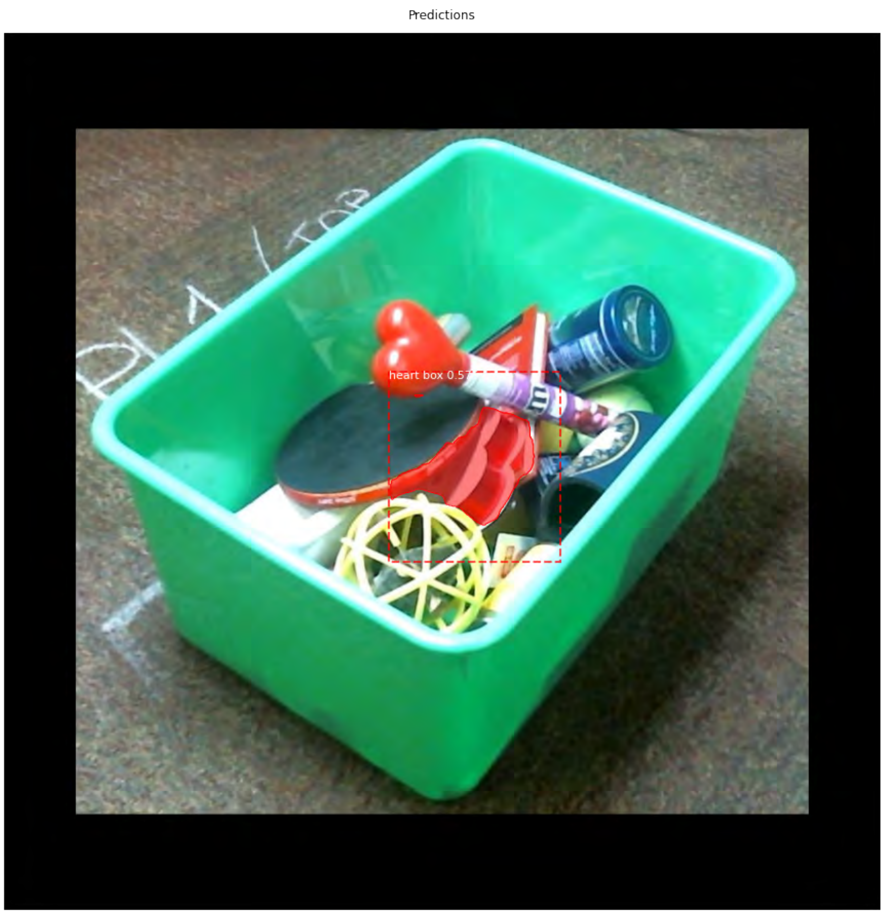


Figure 10: Detection masks for Experiment 3.

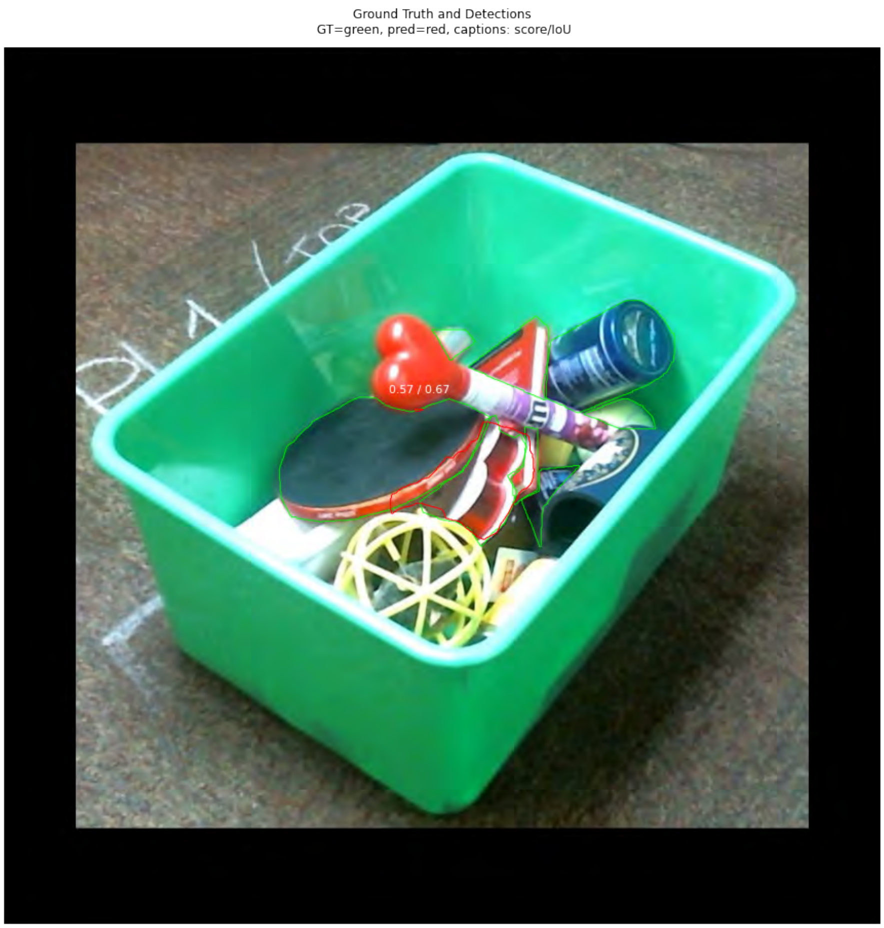


Figure 11: IOU detection for Experiment 3.

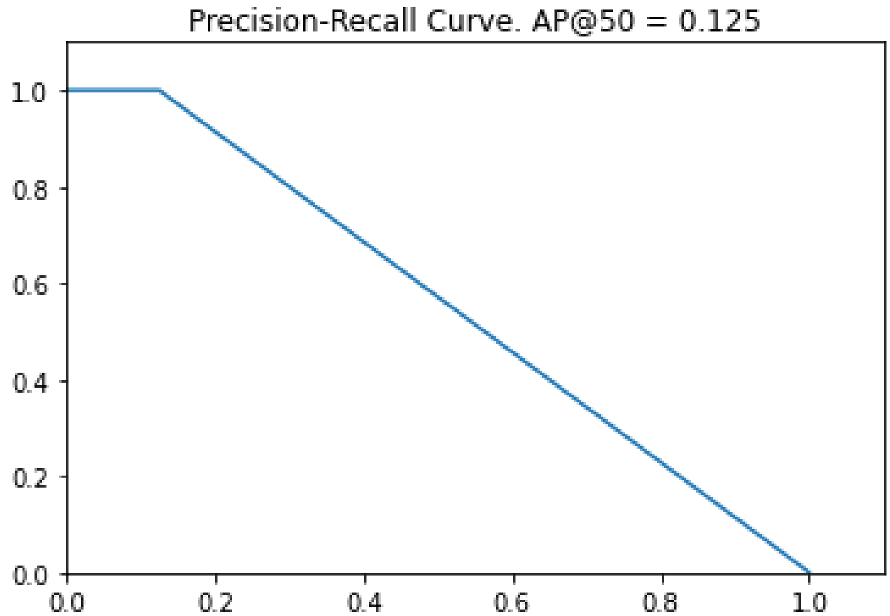


Figure 12: Precision plot for Experiment 3.

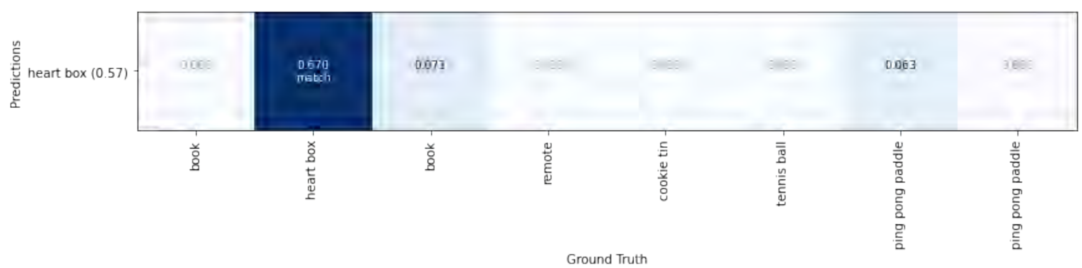


Figure 13: Confusion matrix for Experiment 3.

Discussion: I abstained from including the images from all the trials and tests that were run and will provide a brief overview and description here as well as a discussion of the results obtained just on the ResNet50 backbone. Multiple data augmentation techniques were tried, however, based on experimentation the simpler augmentations were more favorable than the heavier ones. This is perhaps due to the issue of when the augmentations became excessive, the image became too distorted for the model to recognize and did not simulate the occluded variations in the validation data. However, the model significantly improved when simpler augmentations were applied such as rotation, translation and scale. These were more true to what the model would witness in the validation data. However this also indicates the critical value of data augmentation in the model's learning capability. Without data augmentation, the model performed quite poorly in all trials because of the little amount of data available. However, the limited amount of available data was not an issue when simple augmentation techniques are applied. The model would significantly benefit from more data though.

Furthermore, the backbone was realized to not play such a drastic role. Thus it is more favorable to use a lighter architecture like ResNet50 over ResNet101 since the results were not drastically different. Future experimentation could include adding drop out and analyzing the complexity further to see whether the model could be further simplified based off the results. Even with very little training the results were quite good for the resolution and angle experiments.

The issue of scale was also something to consider regarding the individual object images in the training. I attempted to utilize multiple scaling techniques in order to address the issue of scale which was prominent and affected the results significantly. Methods include rescaling and registering the individual objects based off of a reference in the tote image and then padding with a mirror of the border as well as substituting every pixel with a single value. However, both of these methods failed. When including only the high quality phone images the model's performance appeared to improve significantly. Other techniques that could be explored include using an alpha channel in order to separate the foreground and background in order to be able to properly scale the images and aid the model in detecting them in the validation set.

Although the images have not been included, with the detection threshold of 0.9, Mask-RCNN appeared to have learned the features of the individual objects quite well because it proved to have no detections in any of the test images. Thus, regarding unknown data inputs, it remains quite robust even given very limited training.

It is evident that I had not trained the model enough. While 50 epochs show okay results for the angle and resolution experiments, they could be significantly improved with a longer training schedule as well as properly scaled individual images. In future experiments, I would want to train this model for more epochs and then see if the results would improve on the layout experimentation which was expected to perform poorly.

It was also clear that the anchor size played a big role in whether the small objects could be detected. The denser the boxes, the better the detections appeared on smaller items. However, a general one size could not be applied to the entire model to produce great results. If this project was continued, it would be interesting to determine adaptive anchor sizes based off of ratios for the object so that the smaller the object relatively in the set could have a denser anchor applied to improve the results.

There were many other things that I would have delighted in attempting to do but however faced difficulty due to time constraints. Data pre processing could have potentially significantly improved the results. This includes perhaps developing image mosaics in which the individual items are merged together in various formats which could potentially improve what results are obtained regarding the dense clutter. While the number of anchors used proved to make a difference, a small change in the anchor size for the region proposal network also showed improvement. This would be another interesting parameter to explore and further experiment with. While the SGD optimizer was used for this experiment, it would also be interesting to attempt and incorporate different optimizers like Adam and witness their performance results. Post processing could also be done in which false positives are removed and the model is retrained to improve accuracy.

The layout experiment highlights the big difficulty regarding unseen configurations of known data inputs. While Mask-RCNN works well for known configs and unknown objects it fails for unknown layouts and exploring more novel methods for generalizing the model would also be a another thing to explore. This could include experimenting with open set recognition methodologies and even more advanced data augmentation techniques. This could include variations of dense clutter layouts by aggregating individual item images. Furthermore, more instances of each item could be included by batch downloading images of the items from google.

Although I played with many parameters some of them just failed like the one with the gradient clipping of 10 caused exploding weights. It would have been interesting to further explore other confines that could prevent the explosion and see the impact of these parameters on the model's accuracy.

3 YOLO

The most salient feature of yolov3 is that it makes detections at three different scales. YOLO is a fully convolutional network and its eventual output is generated by applying a 1 x 1 kernel on a feature map Figure 14 [3]. In yolov3, the detection is done by applying 1 x 1 detection kernels on feature maps of three different sizes at three different places in the network. Detections at different layers helps address the issue of detecting small objects. The upsampled layers concatenated with the previous layers help preserve the fine grained features which help in detecting small objects. These features also successfully detect occluded objects.

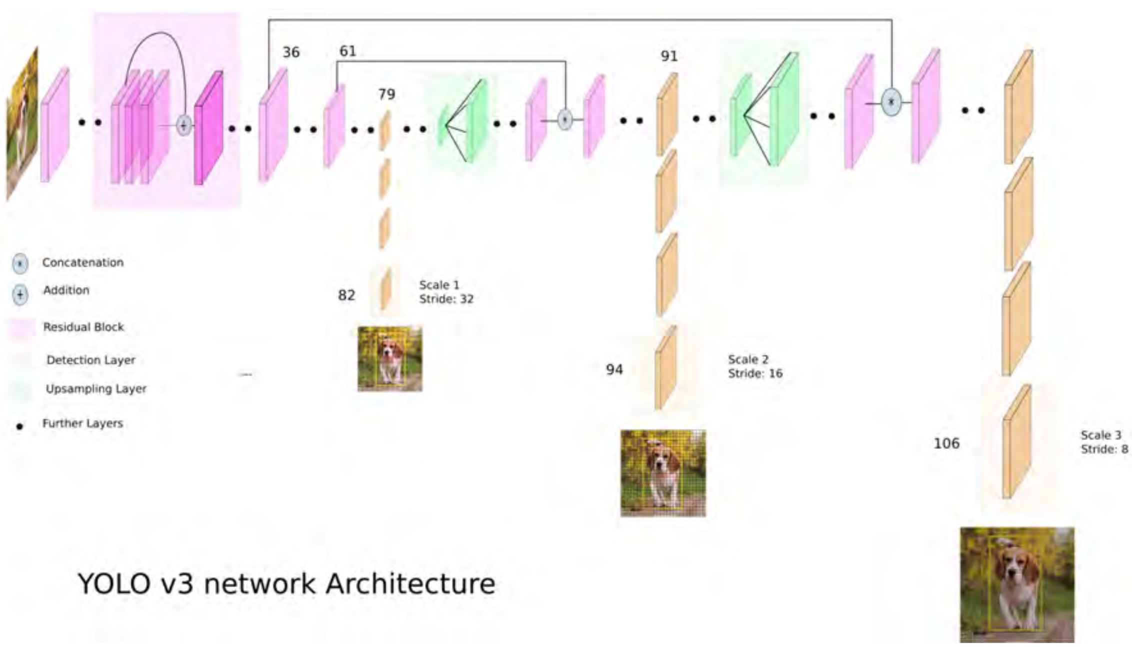


Figure 14: Yolov3 architecture

Based on the factors listed, yolov3 was selected to detect the small objects in the project. For yolov3 to robustly identify everyday objects within a tote, it needs at least 1 similar object in the Training dataset with about the same: shape, side of object, relative size, angle of rotation, tilt, illumination as the object in tote. With a proper training dataset, it successfully detects occluded small objects. With the correct maximum batch number (2000 iterations for each class(object)) yolov3 performs really well. To check the robustness of yolov3 to varying illumination, scale, rotation and selected properties of sensors, I performed the 3 experiments described in Table 2. When using individual images in the training set, I faced issues with scaling, Figure 15 shows the detection result in this case. Yolov3 requires objects of similar sizes in training dataset as in testing dataset. So for the 3 experiments I excluded individual images from the training set.

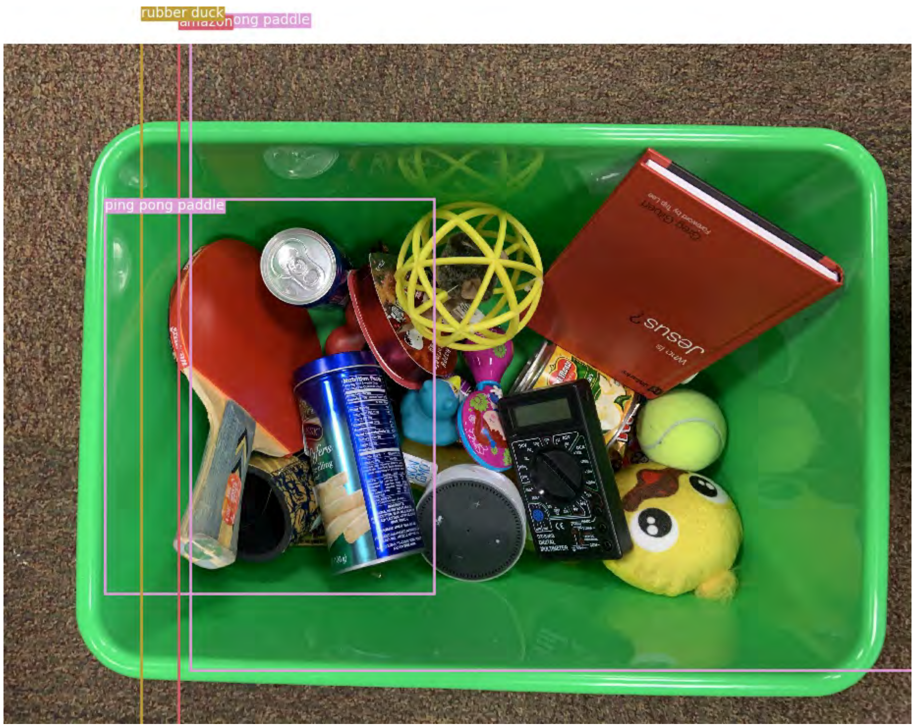


Figure 15: Training using individual images

Experiments	Conditions	Training	Detection
1	`Sensor`	Mobile tote images	Webcam tote images
2	`Angle`	Top view tote images	Side view tote images
3	`Layout`	Layout 1 tote images	Layout 2 tote images

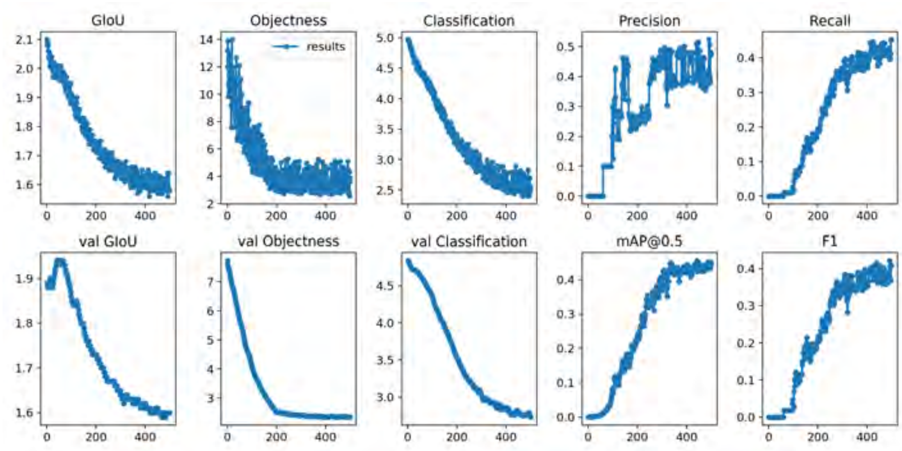
Table 2: Experimental Design

Figure 16: Detection results for training on high resolution (iPhone) and validating on low resolution (webcam)

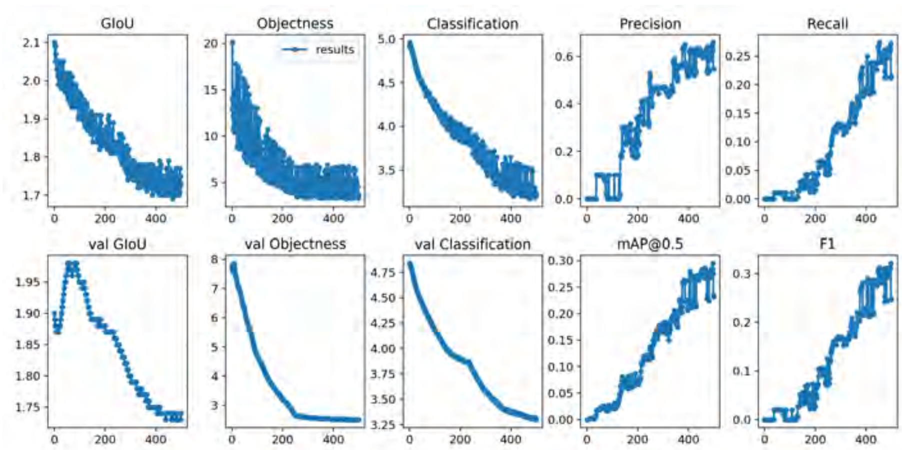
Figure 17: Detection results for training on one angle (top) and validating on different angle (side)

Figure 18: Detection results for training on one layout and validating on different layout

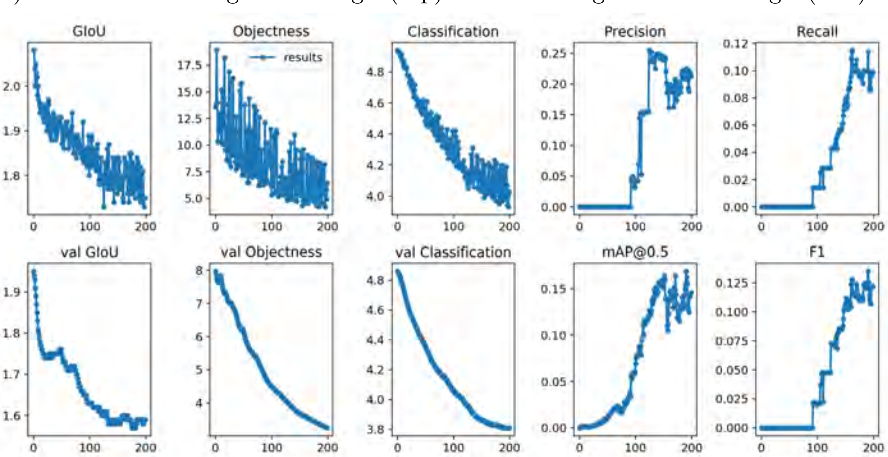
All 3 models were trained using 6000 max batches, 0.7 conf-threshold in yolo layers, 500 epochs except for the experiment 3, which was trained on 200 epochs. The image size was kept [320, 640] and the network resolution was increased by using height=832 and width=832, this increases the precision and makes it possible to detect small objects. To make comparisons and analyze these results I used the GIoU, mAP accuracy metrics. GIoU measures how much our predicted boundary overlaps with the ground truth, the GIoU on the validation set is a good measure to check if the model is being trained well. Figure 19 shows the plots for all three cases. The best case was experiment-1, training on mobile images and testing on webcam images. All three cases were trained using the darknet weights.



(a) Results for training on high resolution (iPhone) and validating on low resolution (webcam)



(b) Results for training on one angle (top) and validating on different angle (side)



(c) Results for training on one layout and validating on different layout

Figure 19: Results for the 3 experiment cases

As the model in experiment-1 performed the best, I used that model for detection on unseen objects. The results are shown in Figure 20. The false detections are only on one object with low confidence threshold and can be easily avoided by increasing the threshold to 0.2-0.3. The model performs well with unseen data.

Figure 20: Testing on unseen data

For the yolov3 model, I fine-tuned the darknet weight files for 500 epochs to get the results but even at 250 epochs, the results start looking good. Because the image dataset I used was very small, I trained the model using batch size 16. These models would perform better with large datasets, if the model has seen enough images of the test objects in training dataset, the accuracy and performance would increase. This approach can be extended to novel objects with the condition - for each object which we want to detect, there must be at least 1 similar object in the training dataset with about the same: shape, side of object, relative size, angle of rotation, tilt, illumination. It is desirable that the training dataset include images with objects at different: scales, rotations, lightings, from different sides, on different backgrounds - we should preferably have 2000 different images for each class or more, and then train for 2000*classes iterations or more.

Based on the results I deduced that the experiment-1 gave the best results. This is because the training was done on tote images where the model was trained on the occluded images and was unable to generalize to unseen orientations. In experiment-1 the illumination, rotation of object varies and the model behaves robustly under these conditions. In other experiment cases, the model is trained on occluded parts of the objects and is unable to give good results for unseen orientations. But based on experiment-1 the model performs well in detecting small, occluded objects given the proper training dataset. Scaling was the main issue I was unable to resolve given the time constraints. For future work I would definitely like to try scaling the individual images and include them in training dataset. That should improve the performance of all 3 experiment cases significantly.

4 SSD

SSD (Single Shot MultiBox Detector) [2] is one of the most popular object detection models with high accuracy (0.743 mAP on VOC2007 test) at real-time speed. SSD is robust to different object sizes for the following reasons: (1) default box (anchor box) has different scales and aspect ratios; (2) The convolutional layers after the base network decrease in size progressively (as shown in Figure Figure 21) so that the detections at multiple scales is possible. Furthermore, SSD can take images with different size as inputs. Based on the reasons listed above, SSD was chosen to complete this object detection task.

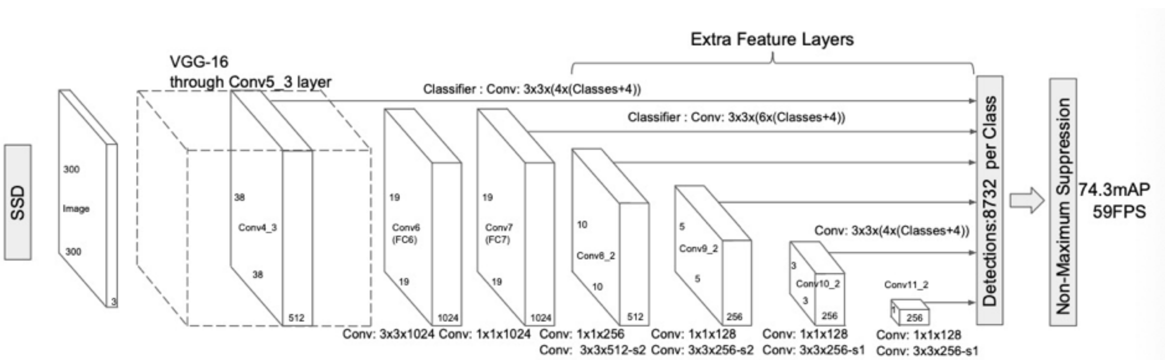


Figure 21: The extra feature layers at the end of a base network predict the offsets to bounding boxes of different scales and aspect ratios and their confidences [2].

To find answers to the questions, such as “what is the robustness of the proposed methods to varying illumination, scale, rotation and selected properties of sensors?” and “Is training necessary? If so, what is the least amount of training required to robustly match occluded objects?”, the following experimental design as listed in table 1 was proposed:

Conditions	Training	Detection
Sensor	All webcam images except those tote images for detection	Randomly chose one webcam tote image from each combination
	All iphone images except those tote images for detection	Randomly chose one iphone tote image from each combination
The amount of Training	All individual object images	All tote images
	All individual object images and one tote image from webcam and one tote image from iphone	All tote images except the tote images for training
Layout	All individual object images and tote images of object layout one	All tote images of object layout two
	All individual object images and tote images of object layout two	All tote images of object layout one

Table 3: Experiment Desgin

In order to do comparisons with different models, the metric was used to evaluate the accuracy of SSD on this object detection task is mAP (mean average precision), which is a widely accepted metric in measuring the accuracy of object detectors. mAP calculates the average of AP over all categories, in our case 10 categories.

- **Training on images collected by webcam**

  The first set of experiments is the training on tote images collected by webcam and all the individual object images and the images for validation were excluded. Detection was on tote images collected by webcam. To avoid over-fitting, the epoch number was incremented from 1000 to 12000 gradually to monitor the loss values. Finally, epoch number was set to 8000 and the loss is around 0.5. Parts of the detected results are shown in Figure 22. The mAP calculated from this training model is around 0.7733. It is observed that most of the objects were successfully detected because similar tote images have been seen in the training process. However, the toy and amazon device were detected in some tote images but not detected in other ones. When we lowered the threshold for the confidence score, these two objects started to show up with fairly low score because of large occlusions.

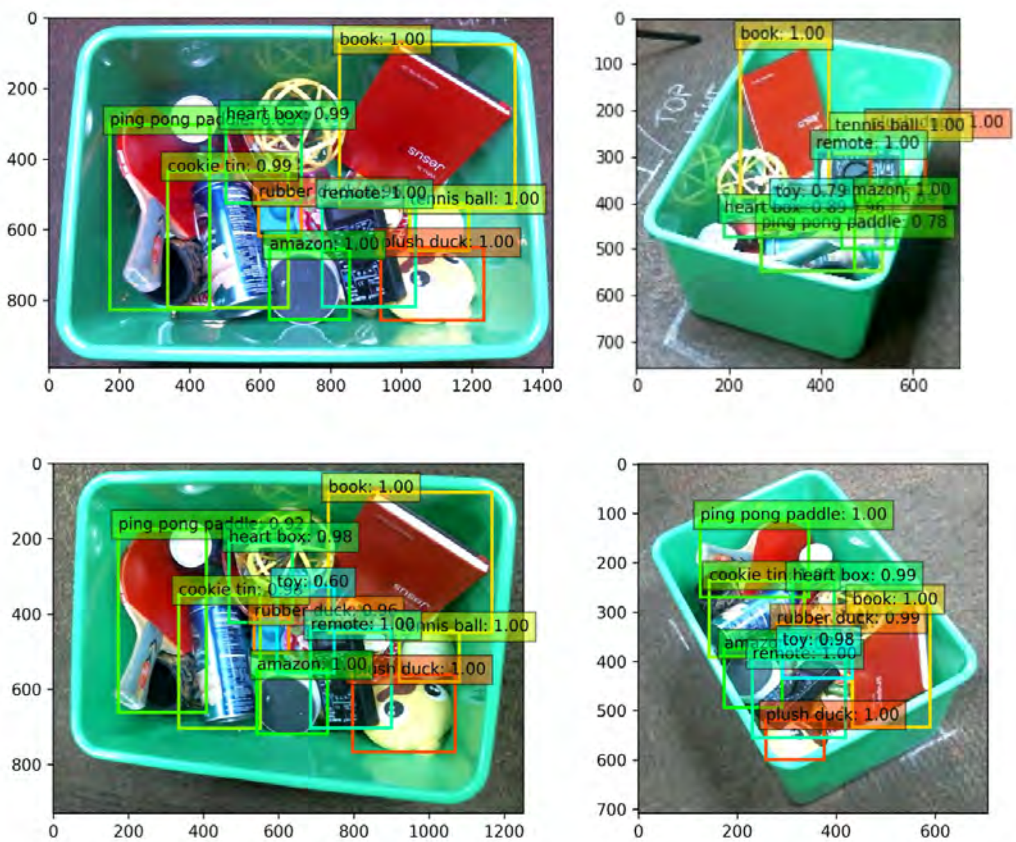


Figure 22: training on images collected by webcam, the left column presents top images and the right column presents side images

- **Training on images collected by phone**

The second set of experiments is the training on tote images collected by iPhone and all the individual object images and images for validation were excluded. Detection was on tote images collected by iPhone. The training process on phone images was supposed to be the same as that on webcam images. The training was done by 8000 epochs and the corresponding detected tote images are presented in Figure 23. Similar to the results from webcam images, large objects have higher scores while smaller objects or objects with occlusions shows lower score. Because the images collected by Iphone have much higher resolution, the training time on Iphone images is almost three times longer than that on webcam images, however, obvious detection accuracy improvement was not observed.

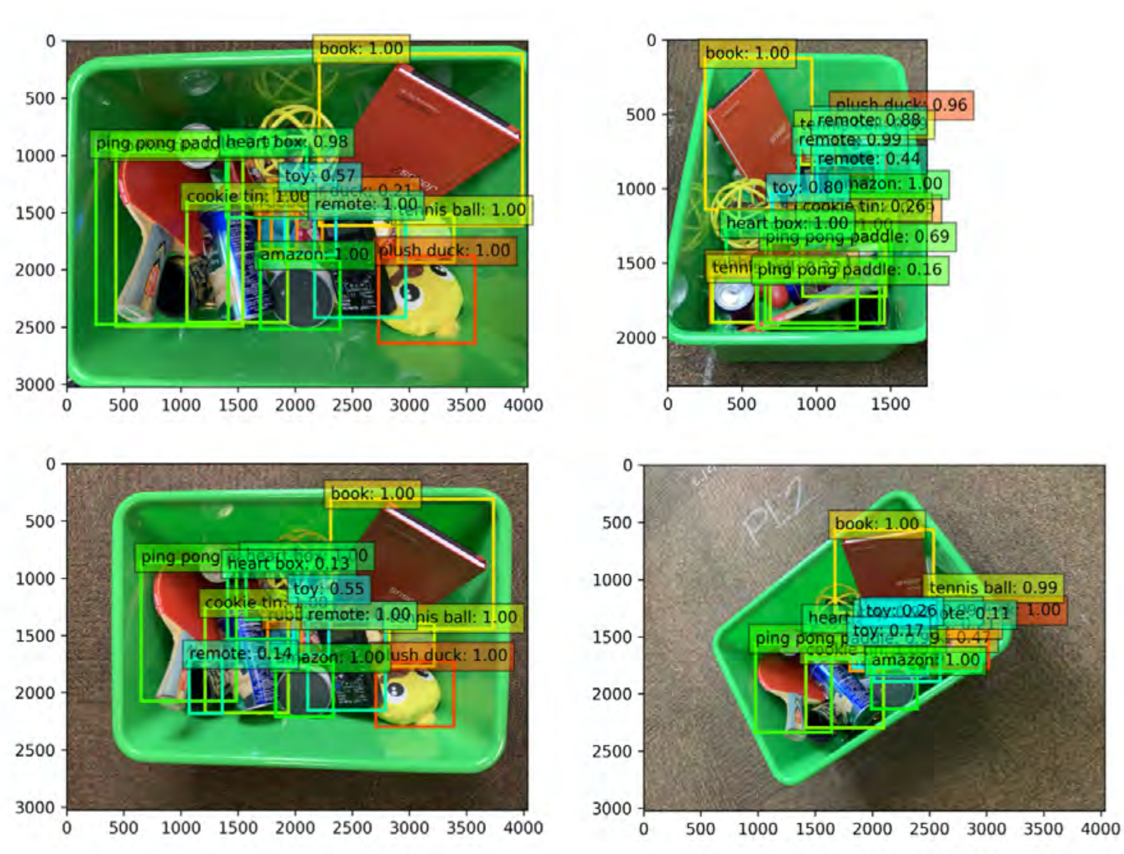


Figure 23: training on images collected by phone, the left column presents top images and the right column presents side images

- **Training on individual object images only**

  The third set of experiments is training on all the individual object images only and detection on tote images. Only individual object images were used in the training made the detection task even harder. The representative results are shown in Figure 24. To show the objects can be detected as many as possible, the confidence score threshold was set to 0.1. It is shown that about 4 objects were recognized per tote image, mainly, toy, heart box, rubber duck, and cookie tin, etc. However, the location detection were extended to the whole image. It is likely that the same object has different dimensions in training and detection images. Adding few number of tote images might be helpful to locate target objects.

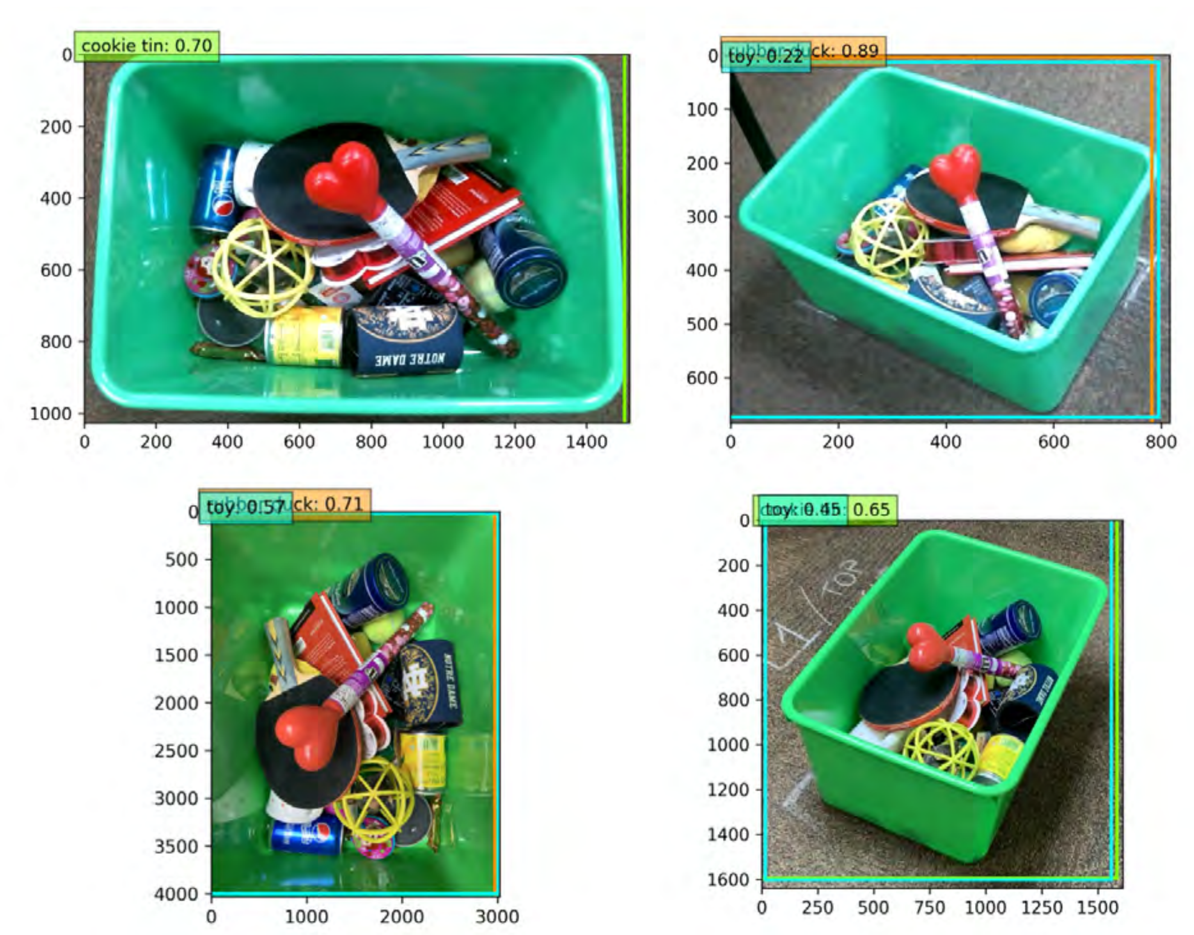


Figure 24: training on individual object images only, the left column presents top images and the right column presents side images, top row presents tote images from webcam and bottom row presents tote images from iPhone

- **Training on individual object images and two tote images**

  To improve the result of the model trained only on individual object images, two tote images were added to the training image set including one tote image from webcam and one tote image from iphone. When tote images are used for training, the tested images can be different from (Figure 25) or same as (Figure 26)the tote images for training. It is interesting that the target object bounding boxes are more reasonable as compared with the results in the last experiment however the positions of these boxes are off the right object positions. Therefore, two tote images are useful to select the correct bounding box size but have limited effects on the right position prediction. Furthermore, the detection on the same layout tote images presents higher confidence scores and more accurate bounding box prediction, which is consistent with the abovementioned experiment results.

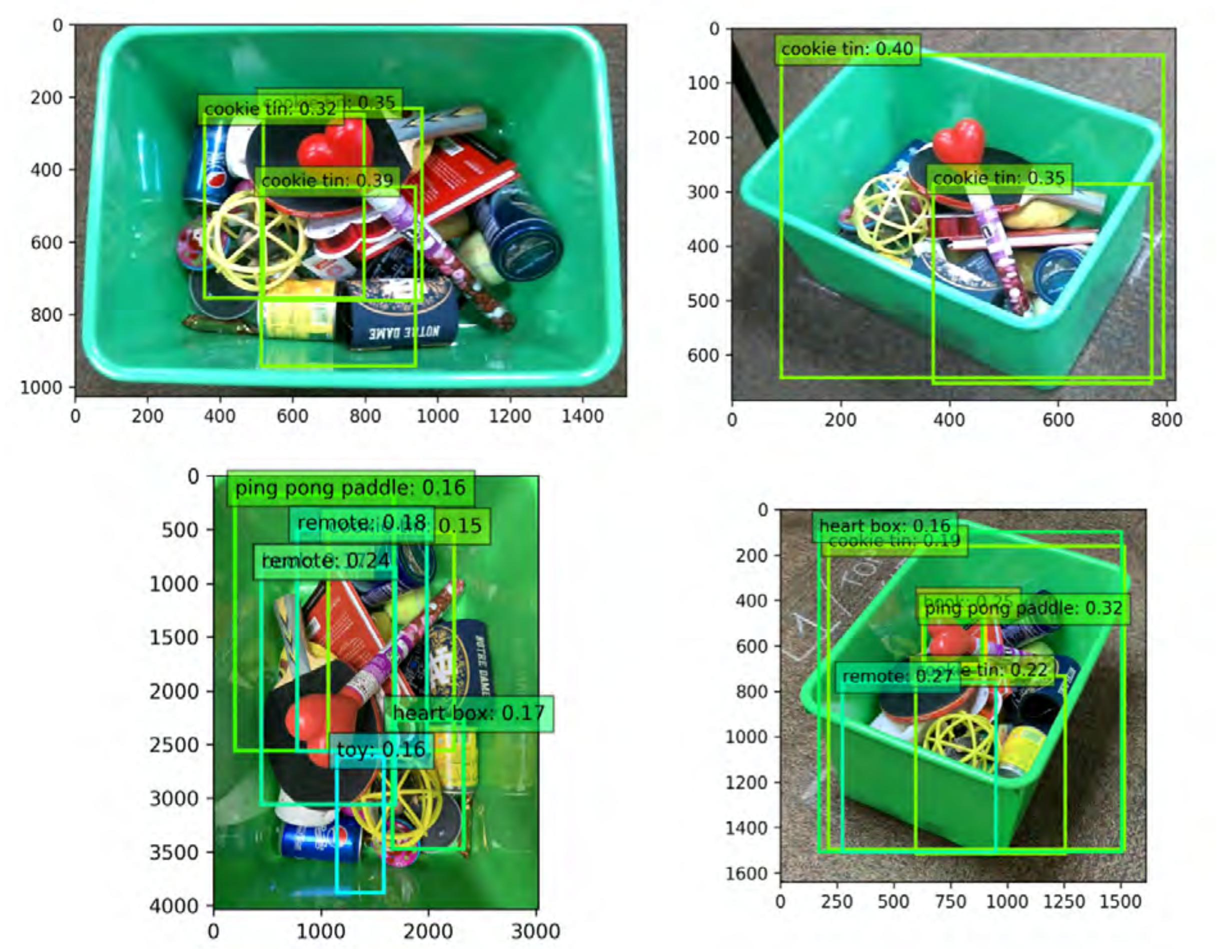


Figure 25: Training on individual object images and two tote images and testing on tote images with different layout from the two tote images for training. The left column presents top images and the right column presents side images, top row presents tote images from webcam and bottom row presents tote images from iPhone

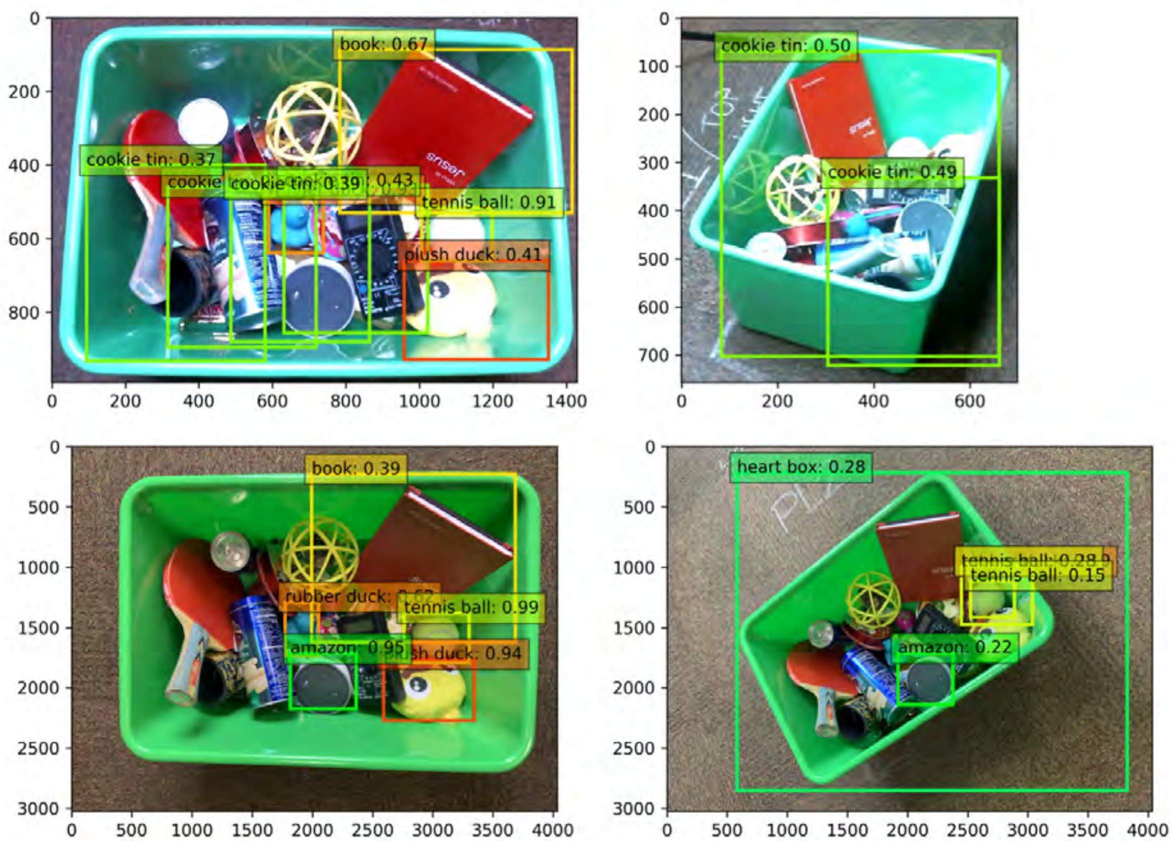


Figure 26: Training on individual object images and two tote images and testing on tote images with the same layout as the two tote images for training. The left column presents top images and the right column presents side images, top row presents tote images from webcam and bottom row presents tote images from iPhone

- **Training on individual object images and tote images of one layout**

  To evaluate the robustness of SSD on different layouts, two layout tote images were used in training and testing separately. All the images used in this experiment were collected by webcam to eliminate effect from the camera type. The representative results are shown in Figure 27 and Figure 28. It seems that large items were detected when SSD was trained with the first layout while more small items were found when it was trained with the second layout. It proves that the layout of objects in the tote have an impact on the detection results. Furthermore, for a successful detection it is very critical to have similar layout or at least similar object position and angle in the training. It would be very interesting to check the detection results if we could take individual object images in the same green plastic tote with as many as object positions and angles and then feed them to SSD for training.

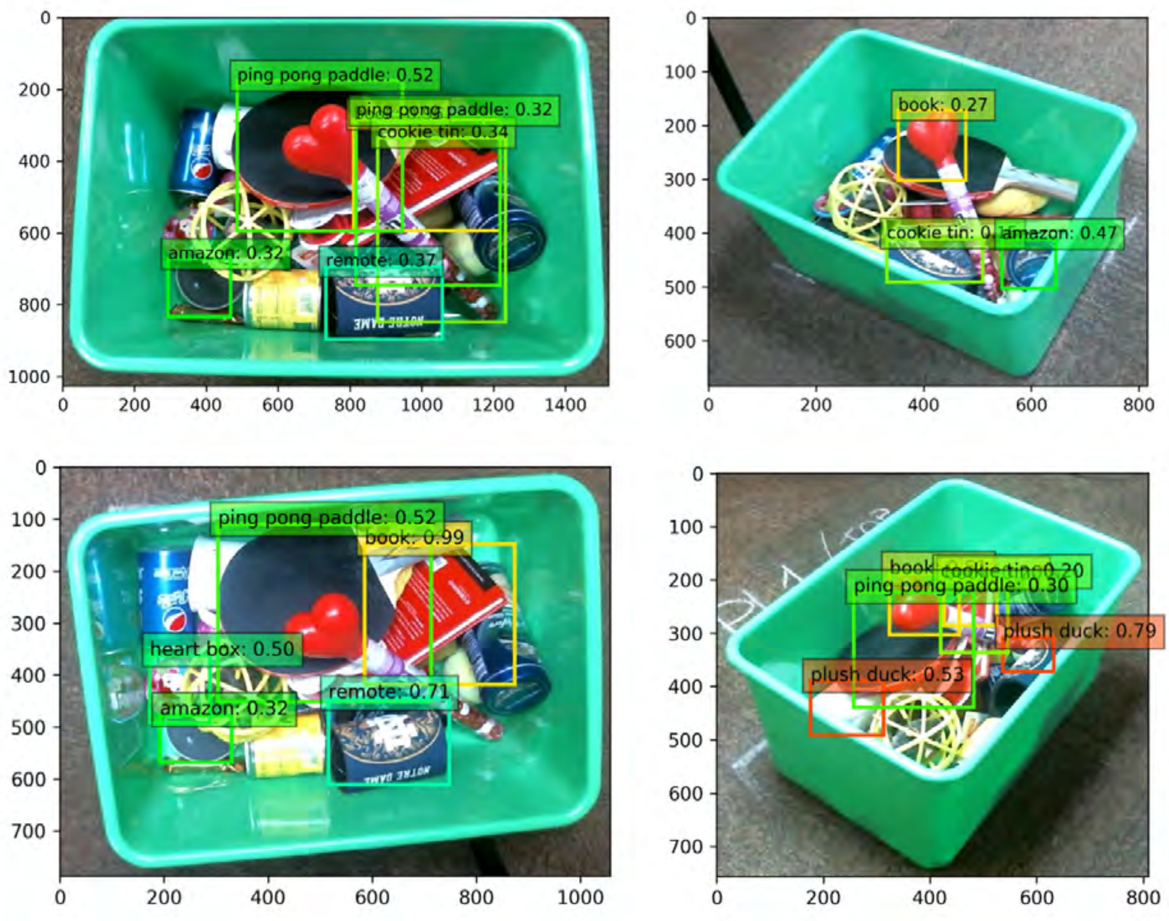


Figure 27: Training on individual object images and the first layout tote images and tested on the second layout tote images. The left column presents top images and the right column presents side images.

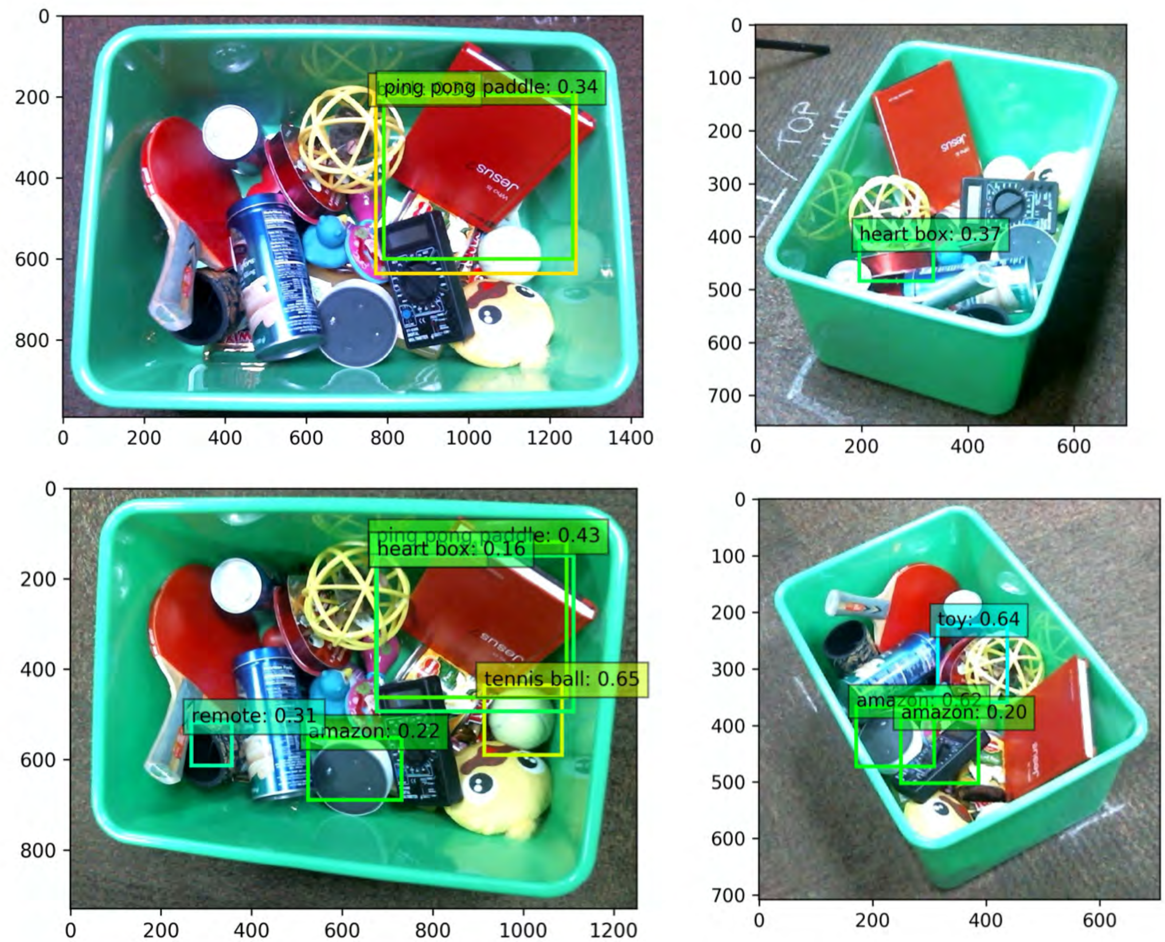


Figure 28: Training on individual object images and the second layout tote images and tested on the first layout tote images. The left column presents top images and the right column presents side images.

- **Detection on unseen objects**

The model trained in the first set of experiments were used to detect unseen objects and the representative results are shown in Figure 29. To be more stringent to this training model, the confidence score was set as low as 0.1, which is 0.6 for detection on tote images. It is observed that: (1) unseen objects were easily recognized with high score as the large objects in the training; (2) Unseen objects with similar colors as the training objects are prone to be false positive cases; (3)Green tote and small objects were not detected even the threshold is as low as 0.1. It is likely that the green tote has been seen in the training and remembered as the background.

For SSD, the most important factor is that in training images and testing images there are some similarities on the detected objects, such as position, angle, the surrounding object layout etc. If this kind of similarity exists, then a positive high confidence match can be achieved with a few input information and very large occlusion. Training is necessary for the successful detection. Better sensor, which can provide high resolution images, would bring some improvements in the detection results however, this benefit is not as large as expected.

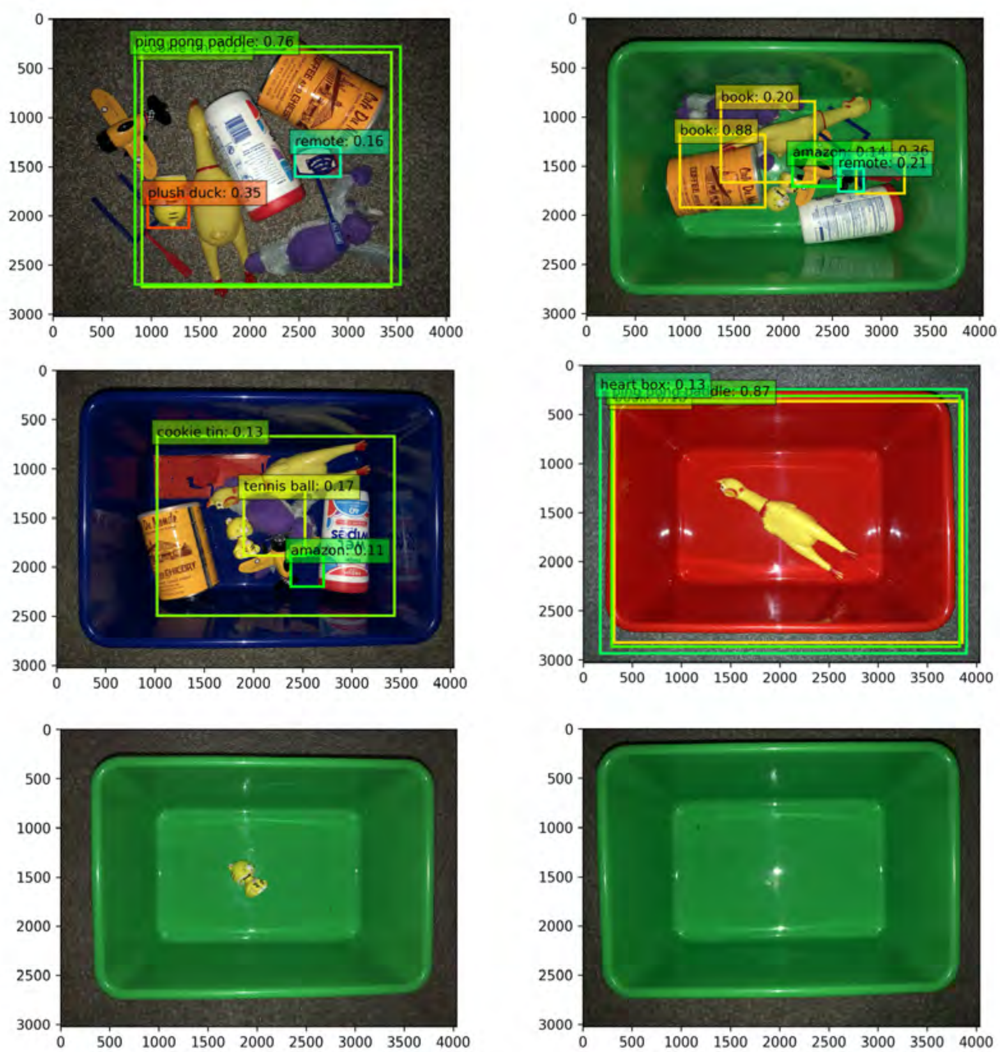


Figure 29: training on images collected by phone

References

[1] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[2] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[3] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[4] Xue Yang, Jirui Yang, Junchi Yan, Yue Zhang, Tengfei Zhang, Zhi Guo, Xian Sun, and Kun Fu. Scrdet: Towards more robust detection for small, cluttered and rotated objects. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8232–8241, 2019.