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 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).

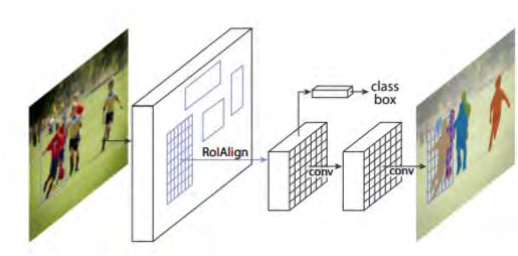


Figure 1: Mask R-CNN framework for instance segmentation. [1]

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.

Results Training on High Resolution (iPhone) and Validating on Low Resolution (webcam)

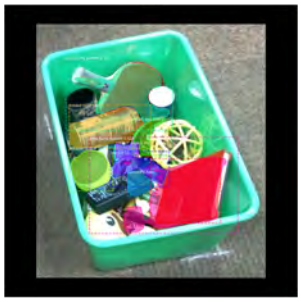


Figure 2: Detection masks for resolution experiment.

Figure 3: IOU detection for resolution experiment.

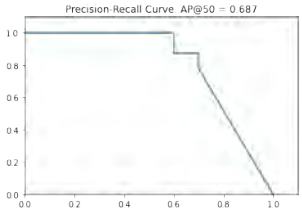


Figure 4: Precision plot for resolution experiment.

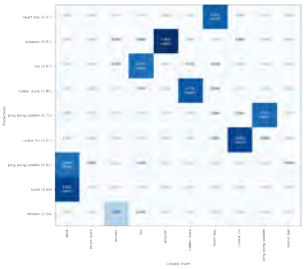


Figure 5: Confusion matrix for resolution experiment.

Results Training on One Layout and Validating on Different Layout

Figure 6: Detection masks for layout experiment.

Figure 7: IOU detection for layout experiment.

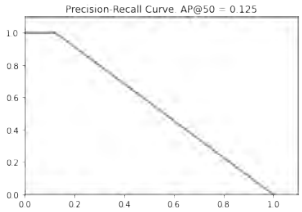


Figure 8: Precision plot for layout experiment.

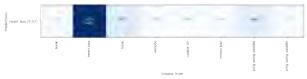


Figure 9: Confusion matrix for layout experiment.

Results Training on One Angle (Top) and Validating on Different Angle (Side)

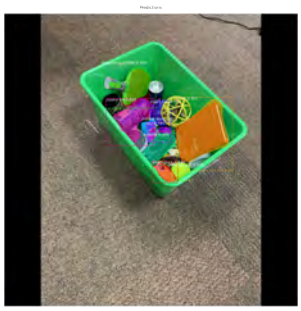


Figure 10: Detection masks for angle experiment.

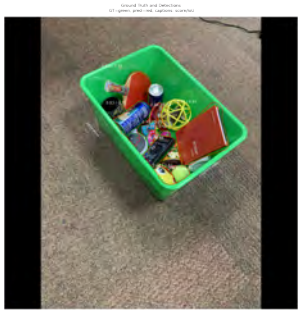


Figure 11: IOU detection for angle experiment.

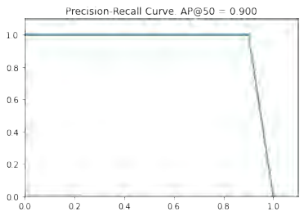


Figure 12: Precision plot for angle experiment.

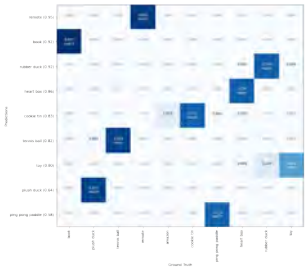


Figure 13: Confusion matrix for angle experiment.

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.

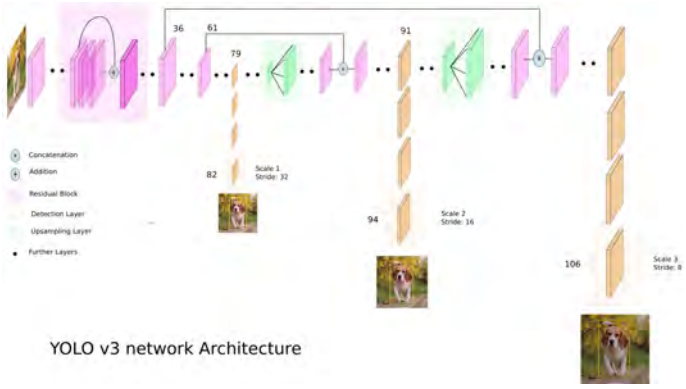


Figure 14: Yolov3 architecture

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 1. 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.

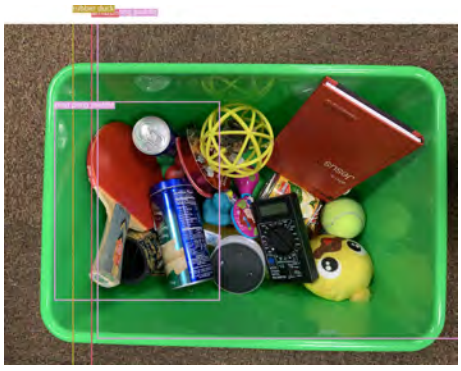


Figure 15: Training using individual images

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 1: Experimental Design

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

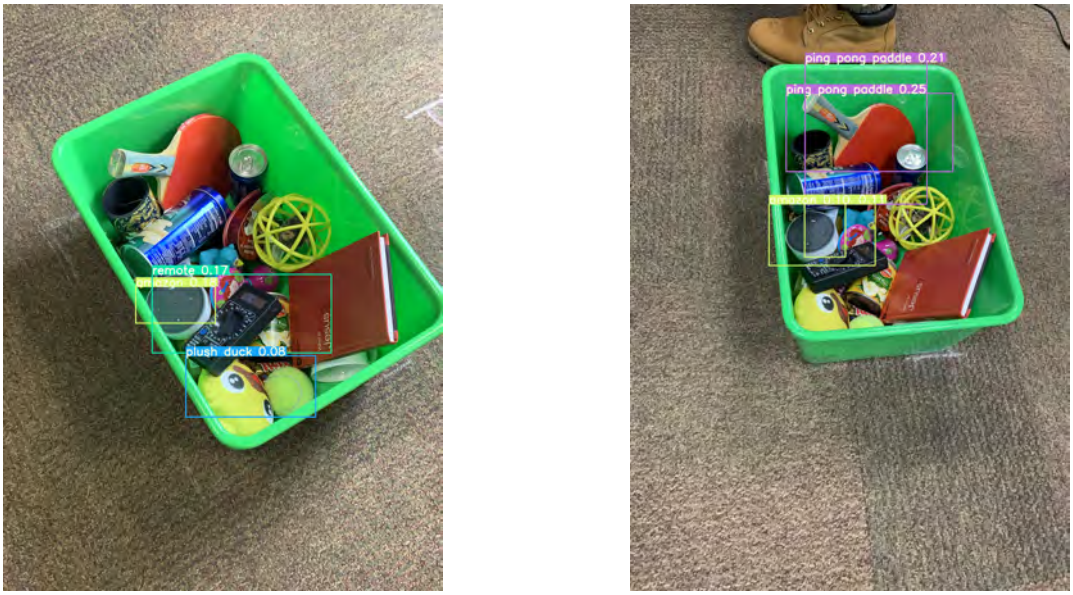
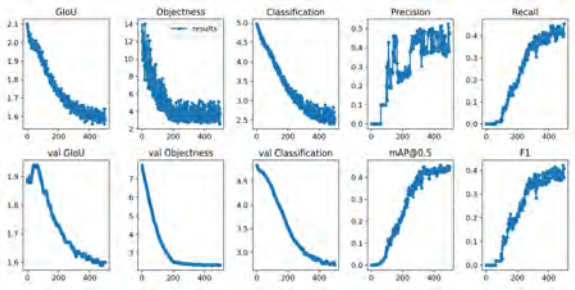


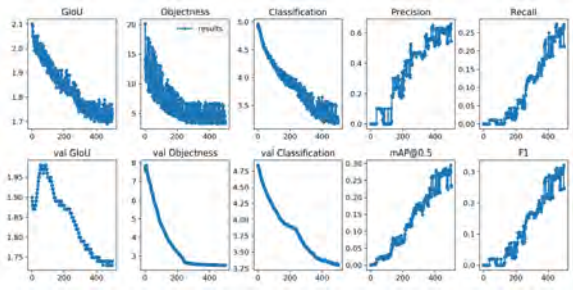
Figure 17: Detection results for training on one angle (top) and validating on different angle (side)

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

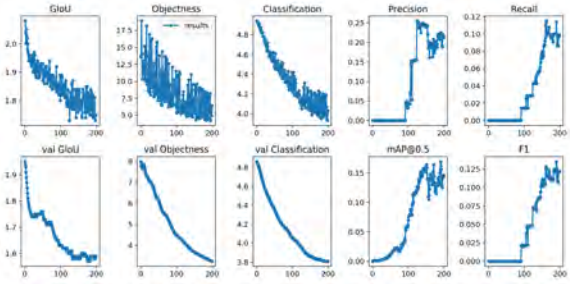
All 3 models were trained using 6000 max batches, 0.7 conf-threshold in yolo layers, 500 epochs except for the experiment 3, which was trained on 200 epochs. The image size was kept [320, 640] and the network resolution was increased by using height=832 and width=832, this increases the precision and 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, training is necessay, 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 could use 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. 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.

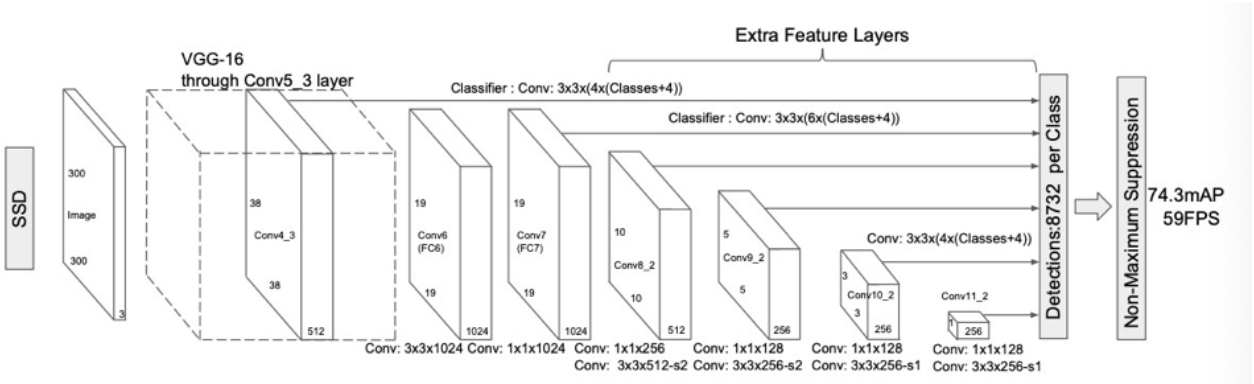


Figure 21: The extra feature layers at the end of a base network predict the offsets to bounding boxes of different scales and aspect ratios and their confidences [2].

To find answer to the questions "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

Table 2: 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.

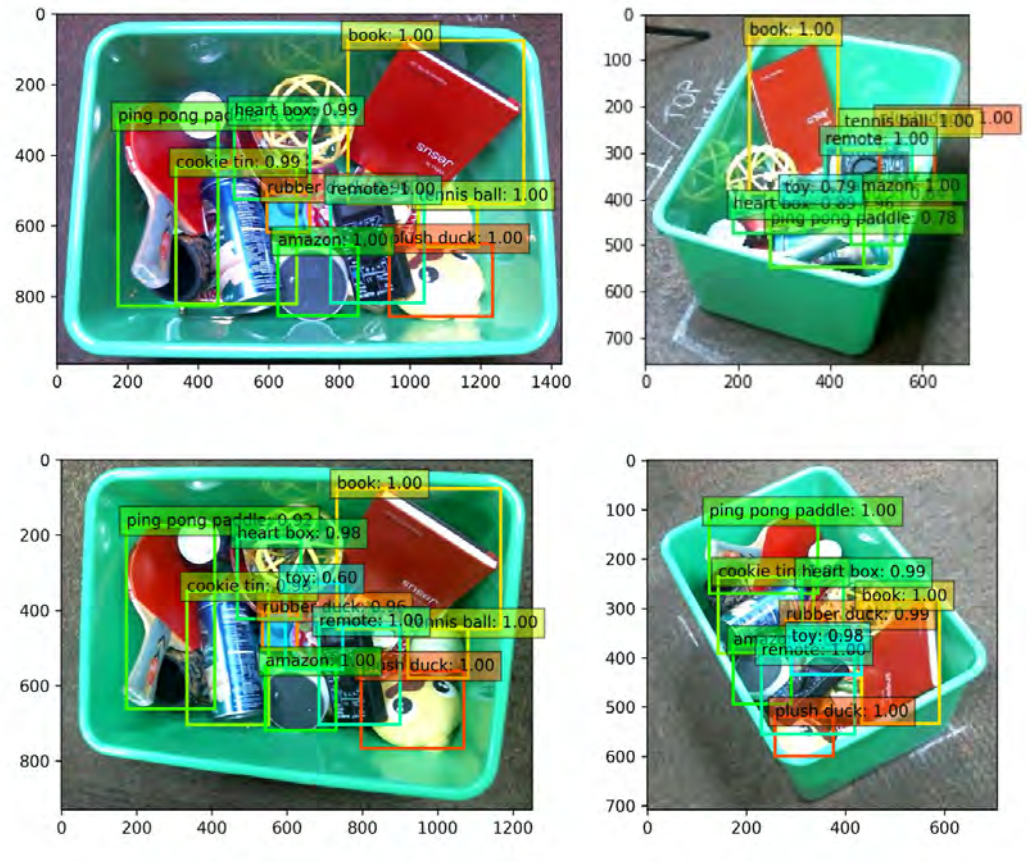


Figure 22: training on images collected by webcam, the left column presents top images and the right column presents side images

- 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. However, because of connection issues and GPU constraints,

the training was actually performed only 1249 epochs and the loss value is around 2.7. The corresponding detected tote images are presented in Figure 23. The same as the results from webcam images, large objects have higher scores while smaller objects or objects with occlusions shows lower score. It would be very interesting to train the model with optimized epoch number and other hyper parameters to see the benefits from the high resolution sensors.

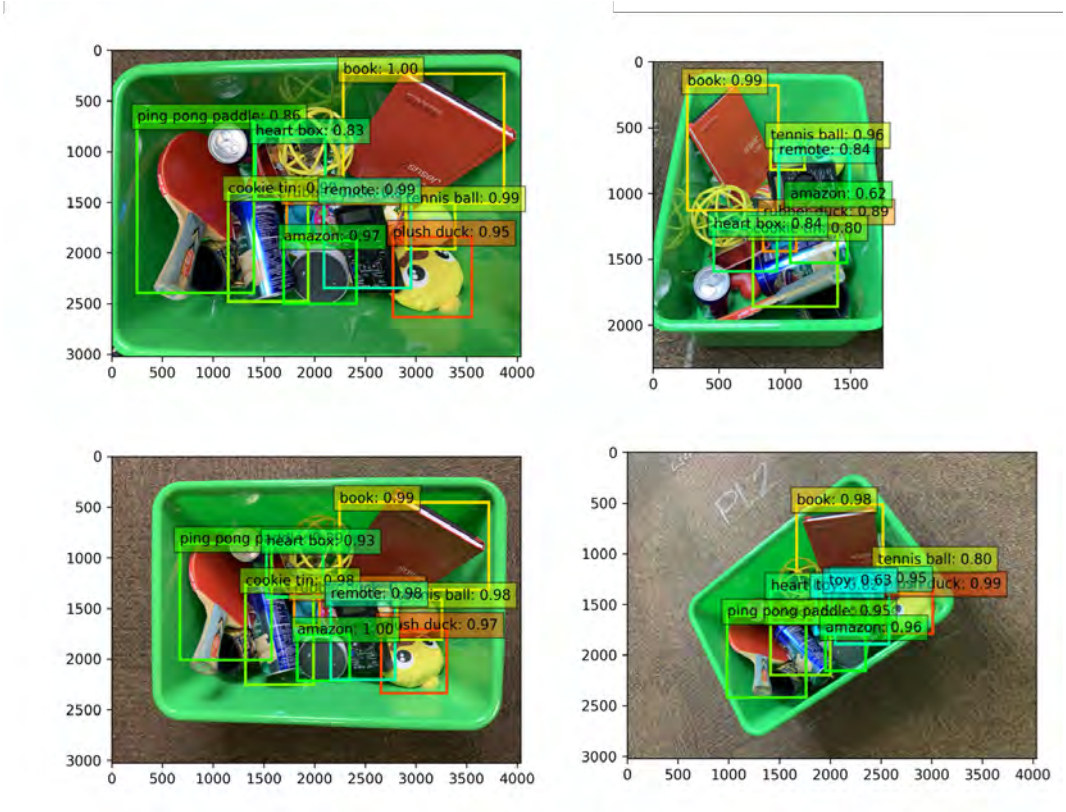


Figure 23: training on images collected by phone, the left column presents top images and the right column presents side images

- 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 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.

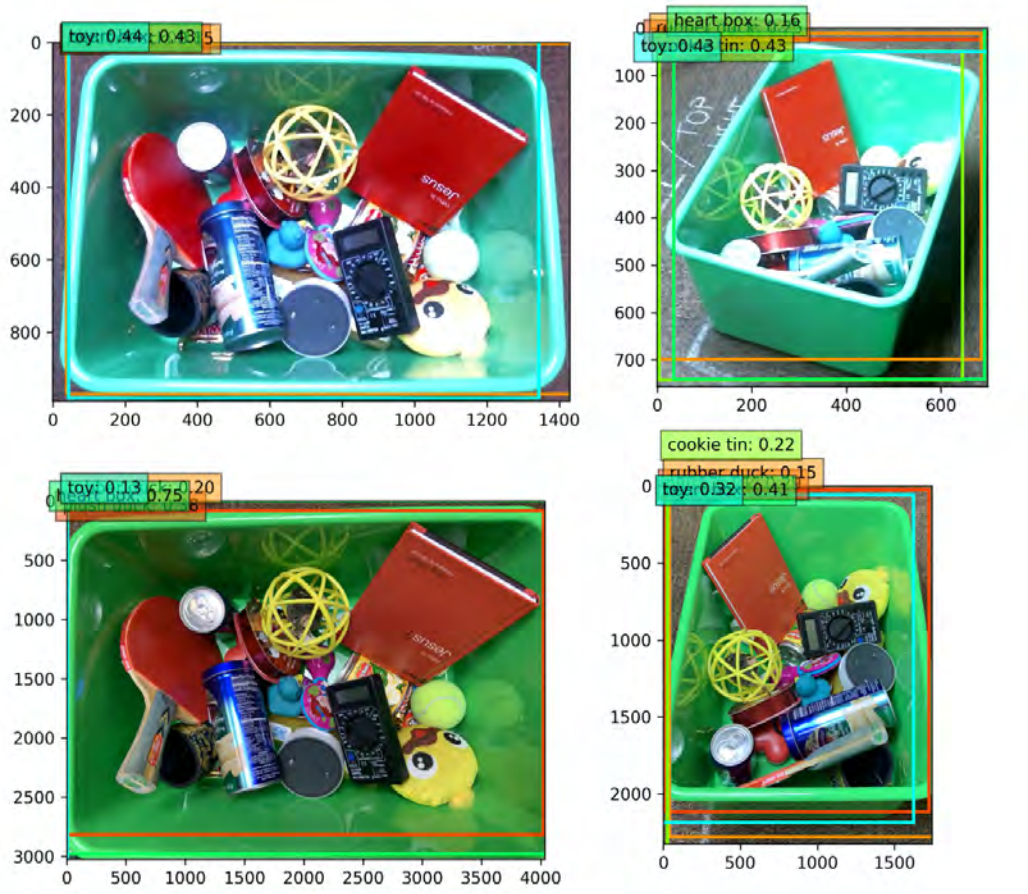


Figure 24: training on individual object images only, the left column presents top images and the right column presents side images, top row presents tote images from webcam and bottom row presents tote images from iPhone

- 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 25. 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.

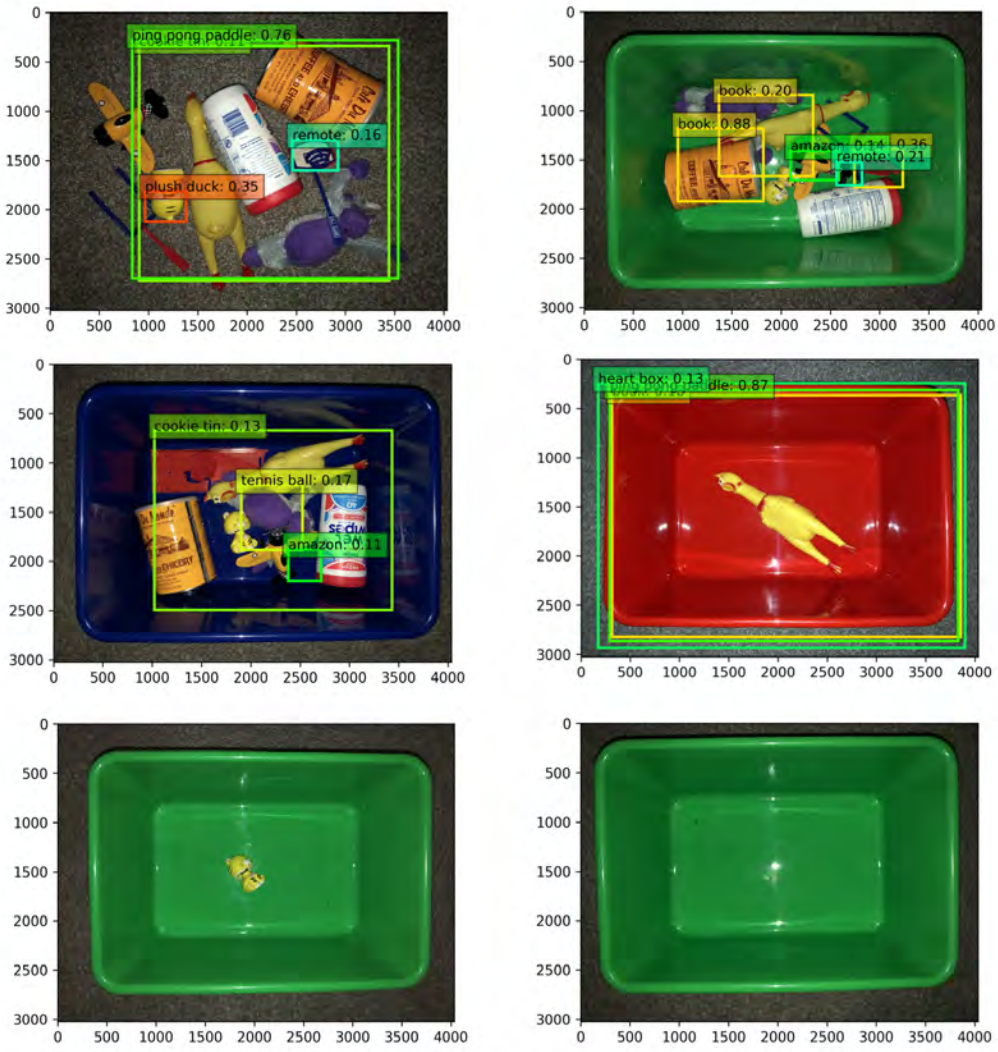


Figure 25: training on images collected by phone

- To answer the two questions: (1) it seem that better sensor would bring some improvements in the detection results however, this benefit is not as large as expected. (2) Some objects can be detected with training only on individual object images, but more modifications about the default box should be performed to yield better detection results.

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.