CS230

Object Detection in Cluttered Environments for Robotic Rummaging

Dane Brouwer
Department of Mechanical Engineering
Stanford University
daneb@stanford.edu

1 Problem Statement & Motivation

The problem of robotic rummaging and manipulation in cluttered bins and shelves, such as that seen in figure 1, is an active area of research. Generally speaking, state of the art robotic systems do not embrace contacts that occur outside of the primary manipulating surfaces, i.e. non-prehensile contacts, but rather avoid them at all costs. The predominant approach is to methodically clear a path via repeated pick and place operations. In contrast, humans display immense skill when manipulating with the forearm or back of the hand in highly cluttered environments, enabling swift path clearing, and can even perceive object properties (such as mass, sliding vs. toppling, or shape) with non-prehensile tactile sensations. This ability is especially useful in constrained and occluded scenes, where vision is not reliable.

Figure 1: Spice cabinets in my on campus apartment (left) and childhood home (right) displaying cluttered scenes which are exemplary of the goal deployment environment.

Large area tactile sensors are being developed to help perform these tasks, but a key component in the perception pipeline to deploy these sensors is to identify and distinguish objects in these cluttered scenes. Once objects are detected, a low risk path toward a target object may be determined and continual detection may provide insights regarding object mobility as a task progresses. This perception goal consists of an input image and a neural network which outputs a mask that details bounding boxes and/or segmentations to distinguish items in these cluttered environments. Many highly capable object detection models have been trained and deployed to perceive images in this way, but a high accuracy in cluttered scenes is difficult to achieve without specifically training a network with these goals in mind.

,

I present a neural network specifically trained on the task of detecting bottles on cluttered shelves, such as in spice cabinets and refrigerators. This model has many potential uses, including the determination of safe entry locations in cluttered scenes by investigating the height-to-width ratio (γ) of each object. Under the assumption of a uniform mass, an object's likelihood of toppling when pushed by a robot arm can be approximated by the ratio of its height and width. Tall, thin objects are more likely to topple than are short, wide objects.

2 Dataset

A first step in effectively detecting bottles in cluttered environments is to gather an appropriate dataset. These datasets are gathered with the goal in mind of performing transfer learning on a pre-trained model ("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x") in the detectron2[1] environment, developed by Facebook Artificial Intelligence Research (FAIR). As such, it is necessary to operate in the native formatting of detectron2 which corresponds to Microsoft's Common Objects in Context (COCO) format. A transfer learning approach is advantageous in this case, because the data type and task being performed are very similar to that of existing models and the relative size of data that I can train is much smaller than what was used in the pre-training of these models.

Two such datasets were gathered for the purposes of this project, the first being one from Google's Open Images Dataset V6 (OI)[2] that contains the classes "bottle", "salt and pepper shakers", "spice rack", and "cupboard". This set contains ∼20,000 images and was obtained with the use of $downloadOI.py$[3], courtesy of Sunita Mallick. Conversion of the annotations into COCO format for this dataset was performed with the use of $convert\_annotations.py$[4], courtesy of Claudio Michaelis. Unfortunately, though this dataset was successfully registered into detectron2, a mismatch in the annotations and downloaded images proved fatal when attempting to train and a simpler approach became necessary.

Though the pre-trained model was trained using the COCO datset, the possibility of improving detection performance by simply retraining with a subset of the COCO dataset[5] was conducted. This second dataset derived from COCO contains the singular class of "bottle". The COCO "bottle" dataset was manually downloaded by referring to an annotation json file which was obtained by filtering the full annotation file with $filter.py$[6], courtesy of Tae Young Kim. This dataset includes ∼9,000 images, each with a resolution of 640x480 pixels. Examples can be seen in figure 2.

Figure 2: Sample images from the COCO "bottle" dataset with annotation masks shown.

Since this dataset is relatively small for a deep learning task, a 95-5 split was used for training and development, respectively (8,501 train, 379 dev). An unbiased estimate of error was not necessary and therefore a formal validation split was not used. However, a simulated deployment in two highly cluttered spice cabinets (fig. 1) was performed for a qualitative analysis of the performance in the final desired environment.

3 Hyperparameter & Architecture Choices

The predominant architecture of the proposed neural network is obtained through a call to detectron2's built in model zoo. The model I chose is called "COCO-InstanceSegment-

ation/mask_rcnn_R_50_FPN_3x" and consists of a 50-layer residual network to combat vanishing and exploding gradients as often seen in very deep networks. This model also uses Mask R-CNN to generate segmentation masks and a Feature Pyramid Network to extract features. This model is pre-trained to perform object detection and instance segmentation on the full COCO dataset. The primary alteration to this network is conducted by compressing the output layer into a single class and retraining on the COCO "bottle" dataset. In this way, the prior architecture and learned parameters can be leveraged to generate fantastic results using relatively few training examples and a short amount of training.

Though the pre-trained network performed relatively well when deployed in the highly cluttered spice cabinets (see fig. 3 (a)), upon close inspection, there are several bottles that were not successfully detected, leaving room for improvement for this specific task and deployment environment.

A brief comparison of several hyperparameters was conducted to determine the relative success of the transfer learning. Though an ideal approach would consist of randomly sampling parameters from a logarithmically distributed range, time limitations constrained the implementation to a simple exploration. As shown in table 1, two variations each of the minibatch size, momentum parameter, β, and learning rate, α, were investigated. Each configuration was trained for 300 iterations to detect a single class ("bottle"). The maximum number of detections per image was set to 100 and a confidence threshold of 70% was used to determine if a detection would register and display in the output.

4 Results & Insights

Though it is difficult to be certain, it appears from the brief hyperparameter exploration that higher stochasticity, a momentum parameter which exponentially weights over fewer samples (thus generating noisier gradient descent steps), and a smaller learning rate seem to provide an ideal training configuration. With that said, the pre-trained network is reported to have achieved an AP of 40.2 for generating bounding boxes on all classes, while the most successful of the training configurations used to implement transfer learning was only able to maintain this level of precision for the single class. Therefore, one might assume that the additional training was unsuccessful.

However, under close inspection of figure 3, where each model's inference of a simulated deployment is shown, it appears that configurations 1 & 2 generate more desirable masks than the pre-trained model. Configuration 3 seems to generate comparably accurate masks as the pre-trained ones while configuration 4 falls short.

Configuration No.	Minibatch Size	Momentum Par., β	Learning Rate, α	AP (bbox)
Pre-trained	—	—	—	40.2
1	2	0.9	0.001	40.28
2	4	0.9	0.001	37.4
3	2	0.9	0.002	30.78
4	2	0.95	0.001	34.87

Table 1: List of training configurations explored and their associated Average Precision (AP) on generating bounding boxes.

Interestingly, the successful configurations appear to outperform the baseline especially in the exceptionally cluttered region in the bottom shelf of the right example. Whereas the pre-trained model only correctly identified $\sim$50% of the bottles, configurations 1 & 2 recognized more than 85% of instances. It is important to note that a drawback of configurations 1 & 2 relative to the pre-trained is that the bags seen in the bottom shelf of the left image are incorrectly identified as bottles. Though this is a serious issue in more general applications of object detection, if this model is solely used for obstacle detection on shelves for the purpose of reaching into clutter, this mislabelling may actually be more of an advantage than a detriment.

Figure 3: Inference performed from the pre-trained detectron2 model (a) and transfer learning as trained by configurations 1 (b), 2 (c), 3 (d), and 4 (e) on the exemplary deployment scenarios.

5 Conclusion & Future Work

In summary, a dataset containing $\sim$9,000 images of bottles and the corresponding annotations was gathered and utilized in tandem with detectron2 to load a pre-trained network and conduct transfer learning with an altered output layer. Though insufficiently characterized quantitatively, the resulting models appear to improve object detection in highly cluttered scenes, an environment in which robots still struggle to perform satisfactorily. These achievements come as a result of simpler, more single-minded algorithms which are less discriminatory: though they mitigate the false negatives often seen in current algorithms, they suffer from increased false positives.

With this in mind, future work should focus on generalizing performance across a wider range of objects to reduce false positives while maintaining improved perception in clutter and occlusion. Progress toward this end goal could consist of conducting a more thorough hyperparameter exploration, exploring performance differences as the pre-trained model architecture varies, and — as is so often the case — fine tuning the dataset according to training insights to ensure a successful deployment. With continual development of these models, robots will be enabled to perceive and manipulate objects in the home like never before.

6 References

[1] Facebook Artificial Intelligence Research (FAIR), detectron2 github repository, https://github.com/facebookresearch/detectron2.

[2] Google, OpenImages Dataset V6, https://storage.googleapis.com/openimages/web/download.html.

[3] Mallick, S.P., $downloadOI.py$ from learnopencv github repository, https://github.com/spmallick/learnopencv/blob/master/downloadOpenImages/downloadOI.py.

[4] Michaelis, C., $convert\_annotations.py$ from openimages2coco github repository, https://github.com/bethgelab/openimages2coco.

[5] Microsoft, Common Objects in Context (COCO) Dataset, https://cocodataset.org/#download.

[6] Kim, T.Y., $filter.py$ from coco-manager github repository, https://github.com/immersive-limit/coco-manager.