Stock Watch: Retail Shelf Stock-out Identification using Object Detection

Murali M K Dandu*, Srinivas Rao Daru*, Sri Harsha Pamidi*
University of California San Diego
mdandu, sdaru, spamidi @ucsd.edu

Abstract

*Out of stock is one of the major issues faced in many retail outlets and also a substantial loss concern in terms of the revenue. According to a **study** conducted by the IHL group, the total loss of sales due to out of stock amounts to over $1 trillion. In this project, we addressed this issue by developing methods for detecting empty/semi-empty spots in shelves in real-life settings. We collected a novel dataset to simulate real retail world that posses many challenges to be addressed and presenting scope for several approaches. Several variations of transfer learning and domain pre-training techniques are implemented using state-of-the-art object detection models. Ablation studies show extensive result comparison across different aspects of the model training and evaluation. Based on our experiments, retail domain pre-training and fine-tuning, YOLOv5 and RetinaNet architectures, and layer freezing gave better results compared to others.*

1. Introduction

According to reports from retail analysts, over 20% of Amazon's North American Retail sales can be attributable to customers who tried to buy the product from a local store but found it out-of-stock. This usually requires a person checking in on a frequent basis to keep track of the inventory on the racks, which is a time-consuming and labor-intensive task. Also, demand of different products at different seasons and times make the scheduling and replenishment a cumbersome task. In this project, we attempt to solve this issue by developing methods for detecting empty/semi-empty spots in shelves in real-life settings. These methods will aid the retail store agents in timely detecting out-of-stocks and replenishing them more efficiently.

1.1. Retail Shelf Object Detection

Recent works[4] in the last couple of years have tried to detect densely packed objects in retail setting using different datasets and models. These models require precise straight view angles and enormously tagged labels of thousands of similar images. This limits the algorithm's efficacy and utility because, a) it necessitates the installation of cameras at precise angles, b) significant amount of human engagement in gathering and label tagging repeatedly c) change of items and diverse object sizes in the shelf might require re-training of these models using large datasets. Also, we feel that specific viewpoints, such as straight front view, fail to provide us with depth perception and would be ineffective in predicting the extent of unavailability in semi-empty shelves. Limited work has been done on such real-world datasets and the efficacy of these object detection models under such settings. In this project, we plan to investigate various approaches to developing algorithms that can detect empty spaces in diverse image angles, noisy shelves, detecting precise targets (semi-depth empty and semi-height empty) and in small dataset settings.

Related Work section talks about literature on retail dense object detection and state of the art object detection algorithms. Methodology section breifs about the possible approaches, training pipeline and the evaluation metrics. The experiments section extensively discusses about various studies conducted based on the initial baseline results and error analysis. Finally, we conclude the work with providing the scope for future work.

The main contributions of our paper can be summarized as:

- Domain Pre-Training and Fine-Tuning approach to increase the performance in small data settings
- Novel dataset (collected and labelled) to experiment on real-life deployment scenarios
- Study the affect of various architectures, pre-training strategies, model sizes, loss functions, data augmentation techniques etc.

2. Data

The problem we have explored in this paper is to detect different levels of emptiness given an image of a retail shelf.

*The authors contributed equally

Because of the lack of such labelled data, we have collected a custom data set and labelled them to create Retail-2K dataset. But the amount of images required inorder to perfrom successful training would be very high. Hence, we also explored some existing data sets used in closely related tasks for pre-training purposes. One of such tasks thoroughly studied in the literature is dense object detection in retail settings. Though the tasks are different the information required to perform them is similar as emptiness detection is a opposite scenario of an object being present. In the next section **2.1**, we describe SKU-110K dataset introduced in the paper[4] for a dense object detection task. We also discuss the reasons why we can't directly use this data set for our task. Later, in the section **2.2** we describe the main Retail-2K dataset in terms of the task which we have collected ourselves.

2.1. SKU-110K Dataset

SKU-110K Dataset is a collection of 11,762 images with a total number of associated 1,733,711 bounding boxes. The images are split into a ratio of approximately 70:5:25 - train:validation:test ratio. The images are collected in diverse settings slike different lighting conditions and object patterns. Figure 1 shows a sample image from this dataset. The reason why we couldn't use this dataset is that the labels here are the bounding boxes surrounding different objects available in the images on the shelves. This dataset is useful as the data has been collected in a highly packed scenarios, which is evident from the average label count of 150 objects per image. The number of classes in this data is just one, which basically indicates the presence of an object.

2.2. Retail-2K Dataset

The major reason for a new dataset requirement is that this specific task hasn't been explored much and so we weren't able to find emptiness class labelled datasets. So we have captured images in the same retail settings and finally made a set of 300 images with approximately 2000 associated bounding boxes. We randomly split the dataset into 70:15:15 for training, validation and test set respectively. We have limited our number of images as we wanted to explore the idea of transfer learning from. Some of the key features of our dataset is that we have collected them in diverse angles, occlusions, objects in the images and the distance from which the picture has been taken. We did this inorder to make our model robust it would require minimal fine tuning during deployment. Some of the sample images from our dataset can be seen in the figure 2

After filtering out and finalising the images, we have labeled into three classes using data labelling tools like Roboflow. We made rectangular bounding boxes across areas which we believe are empty. We have decided the labels based on the type of the emptiness involved

- Full-Empty(**FE**) means that there is no object present at all in the space
- Semi-Height-Empty(**SHE**) implies that some object are present on the rack but more of them can be stacked onto them
- Semi-Depth-Empty(**SDE**) implies that the space on the rack immediately available is empty but some objects are there in the back of the shelf

The corresponding label distributions are 30:30:40 for FE, SHE and SDE respectively. A sample labelled image can be seen in the figure 3. We have also performed data augmentation techniques like image rotation (0%-15%), brightness (-15% to +15%), horizontal flipping during training. A box level brightness addition has also been done in one of the experiments which is discussed in the later sections. These augmentations have been used to make the model more robust to real world settings and also to increase the size of our dataset.

Figure 1. SKU-110K Dataset

Figure 2. Retail-2K Dataset

Figure 3. Retail-2K sample image with labels

3. Related Work

3.1. Retail Dense Object Detection

Images in the retail domain can be densely packed, with multiple objects, often identical, positioned close together. On existing standards, the performance of modern

object/no-object detectors is remarkable but still restricted. This can be tackled using a layer for estimating the Jaccard index as a detection quality score, and an EM merging unit that utilizes our quality scores to resolve detection overlap ambiguities are presented in [4].

3.2. Out-of-Stock Detection

Out-of-stock (OOS) detection here[1] is solved by combination of deep learning and image processing techniques like getting the object location information using Faster R-CNN algorithm and then followed by three out-of-stock detection methods: canny operator, gray level co-occurrence matrix, and color features. This paper consists of lot of image post processing techniques that may not scale well for other datasets.

3.3. Object Detection

We are modelling the task as a Object Detection problem, since every type of emptiness(full-empty, semi-depth empty, semi-height empty) has its own special characteristics and features that helps in classifying the class. Object detection methods are often classified as neural network-based or non-neural. Non-neural approaches involve first defining features like Haar features, Scale-invariant feature transform (SIFT), Histogram of oriented gradients (HOG) features, followed by classification. On the other hand, neural techniques based on convolutional neural networks(CNN), accomplish the end-to-end object detection without explicitly defining features. In this work we focused on Region Proposals (Fast R-CNN, Faster R-CNN) and You Only Look Once (YOLO) approaches.

3.3.1 Region Proposal Networks

The Region-based Convolutional Neural Network (R-CNN)[3] is comprised of a region selector which identifies regions of pixels in an image that may represent objects known as "regions of interest." These RoIs are sent to a convolutional neural network which processes each region independently and encodes the feature maps into a single-dimensional vector using fully connected layers and then mapped to the output classes. Fast R-CNN[2] combines feature extraction and region selection into one network and employs a RoI pooling layer, which takes feature maps and regions of interest and provides the corresponding features for each region. Faster R-CNN[8] incorporates the region extraction mechanism into the object detection network. It employs a region proposal network, which takes the feature maps produced by the convolutional neural network and proposes a set of bounding boxes where objects might be located. This incorporation of region detection into the main neural network architecture aided Faster R-CNN in achieving near-real-time object detection speed.

3.3.2 You Only Look Once (YOLO)

YOLO[7] incorporates the entire object detection and classification process into a single network and does everything in a single pass, rather than extracting features and regions sequentially. In one assessment, a single neural network predicts bounding boxes and class probabilities straight from entire images. When compared to state-of-the-art detection algorithms, YOLO makes more localization errors but is less likely to predict false positives on background. YOLO learns highly generic representations of items. Unlike classifier-based techniques, YOLO is trained on a loss function that directly corresponds to detection performance, and the entire model is trained concurrently. YOLOv5[5] is the latest engineering optimised version of YOLO that is currently available.

3.3.3 RetinaNet

In retail dense detector training, we would have high foreground-background class imbalance. To address this imbalance between classes, the Focal Loss[6] reshapes the standard cross entropy loss in such a way that it down-weights the loss assigned to well-classified cases and focuses training on a limited set of hard examples, preventing the detector from being overloaded with a large number of easy negatives during training.

RetinaNet is one of the best one-stage object detection models for dense and small scale objects. Feature Pyramid Networks (FPN) and Focal Loss were used to create this, which improves on conventional single-stage object detection models. Total loss can be adaptively adjusted between easy and hard samples by employing focal loss.

3.3.4 YOLOR

YOLOR (You Only Learn One Representation)[9] is an unified network that encodes implicit and explicit knowledge simultaneously, just as the human brain can learn knowledge through both normal and subconscious learning. When implicit information is put into a convolutional neural network with kernel space alignment, prediction refinement, and multi-task learning, we can see that it improves the performance of all tasks. The unified network's implicit representation has exceptional capabilities in capturing the physical meaning of various actions.

4. Methodology

In the scenarios where there is limited amount of data few techniques can be used to make sure models work better. Some of these techniques are transfer learning, fine tuning and few shot learning. Transfer learning is a technique in which a model is pretrained with similar datasets to learn the different information after which generally the

output layers are removed and some new layers are added to make the model usable to the current data. In finetuning we retrain our model to our dataset inorder for the model to world. Both these techniques can be used in combination as well to first pretrain on different dataset and finetune to our dataset. Few shot learning is a different approach where only very few samples are used to train the model.

In our experiments we performed different combinations of transfer learning and fine tuning to understand the effects of these techniques on the model's performance. For pretraining we have used a combination of pretrained models trained on COCO, which are then fine tuned onto dense objects by being trainig the model on the SKU dataset. We have explored techniques like freezing layers while retraining our model. By performing a combination of these techniques we create the final pretrained model. The reason for following such kind of a transfer learning approach is because of the availability of huge datasets which would help the model understand relevant information about the object detection in densely packed scenarios. The general idea of our approach is explained in the figure 4. We repeated the same approach for different architectures to understand the effect of change of models . In all the experiments, after we get the final pretrained model we finetune it on the Retail-2K datasets.

A different kind of approach using the idea that the empty spaces are the negation of locations of the objects is shown in the figure 5. We want to explore this approach as future work to our current project.

While training our models we have used some existing loss functions, one each for the Localization, Classification and the Confidence sub tasks involved in the problem. We need to use different losses for each of them as they try to optimize different part of the output. We try to understand the effects of the mentioned techniques by comparing different metrics discussed in the next section which tells us about the performance of different models.

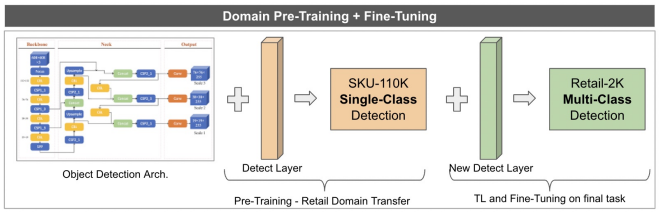

Figure 4. Flow chart of the our approach

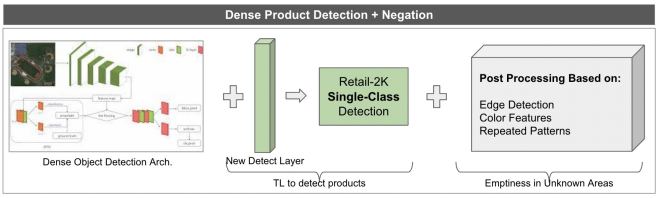

Figure 5. Flow chart of an another possible tehcnique

4.1. Evaluation Metrics

Evaluation metrics used and reported in object detection research are Mean Average Precision at different IoU thresholds along with Precision and Recall. IoU is the metric used to evaluate the overlap between ground truth bounding boxes and predicted ones.

$$IoU = \frac{AreaofIntersection}{AreaofUnion}$$

Using a certain IoU threshold, Average Precision is defined as the area under the Precision-Recall curve

$$\text{AP} = \int_0^1 p(r)\, dr$$

Hence, the final metrics we concentrate are mAP@0.5 ($AP^{0.5}$) and mAP@0.5:0.95 (AP)

5. Experiments

Following the above approach described we have carried out several experiments to understand the impact of various components in training and evaluation pipeline.

5.1. Baselines

We present the baseline results in Table 1 using the Transfer Learning from state of the art YOLOv5 architecture. Using the COCO pre-trained weights, we fine-tuned the model using Retail-2K dataset for around 200 epochs. The best epoch in terms of validation mAP is taken to present the results.

	Validation		Test	
Model	**$AP^{0.5}$**	**AP**	**$AP^{0.5}$**	**AP**
YOLO v5 COCO	43.5	20.5	32.4	12.9

Table 1. Baseline metrics

The training and validation curves (in green) in Figure 6 show that the model reaches near-stabilization after 80 epochs. Validation box loss reaches complete stability while the classification loss fluctuates. The metrics plot (in green) in Figure 9 show that the MAP metrics stabilize around 80 epochs but fluctuates due to small dataset size.

The class wise metrics in Table 2 show that the model is performing well for full-empty class relative to the other two classes. We can also see sample detections in Figure

8 from the test set for all the three classes. Combining the class wise metrics and initial sample detections - The lower performance can be attributed to the toughness and ambiguity in identifying those classes. For example, the stacking probability in a shelf should be rightly assessed based on the product height and shape for semi-height empty detection. In case the floor of rack is not visible, depth perception should be rightly identified for semi-depth empty detection.

Class	Validation		Test	
	$AP^{0.5}$	AP	$AP^{0.5}$	AP
All	43.5	20.5	32.4	12.9
Full Empty	65.0	33.3	53.0	21.2
Semi Height Empty	37.3	16.0	12.6	4.5
Semi Depth Empty	28.2	12.3	31.7	13.1

Table 2. Class-wise metrics for baseline model

Given the initial results and error analysis, we have pursued several techniques and ablation studies to improve the performance. The below subsections talk about the ideas, analysis and relevant comparisons.

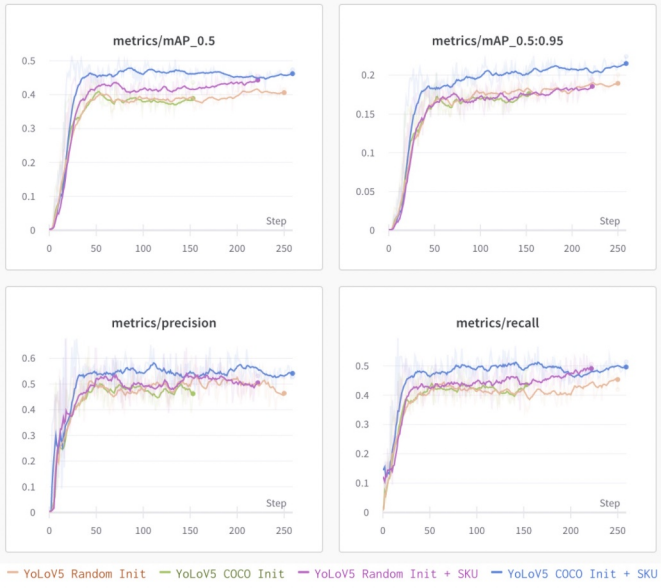

Figure 9. Validation metric curves for different initialization and SKU pre-training settings

5.2. Different Architectures

There are various architectures that are recently proposed that achieve state of the art in different object detection settings. These include YOLOvx, Faster-RCNN, Fast-RCNN, RetinaNet, YOLOR etc with various sizes and backbone variants. Each architecture achieves SOTA performance in specific areas like dense object packing, inference latency etc. Table 3 shows experiments with difference architecures and variants to understand the impact of Transfer Learning on our Retail-2K dataset.

Model	Validation		Test	
	$AP^{0.5}$	AP	$AP^{0.5}$	AP
YOLOv5	43.5	20.5	32.4	12.9
Faster RCNN - FPN*	39.2	15.7	27.2	11.7
Faster RCNN - R101+FPN	40.2	16.5	30.2	11.7
Faster RCNN - R101+DC5	41.8	18.6	32.2	12.6
Faster RCNN - R101+C4	41.2	17.6	30.8	11.6
Fast RCNN - R50+FPN	40.7	17.0	32.5	12.6
RetinaNet - R101+FPN	43.6	19.4	37.1	14.1
RetinaNet - R50+FPN	41.2	17.6	34.8	13.4
YOLOR	46.0	22.0	32.9	13.6

Table 3. Performance using difference architectures and variants

Figure 10 shows that RetinaNet has better performance compared to YOLOv5 and YOLOR, which in turn performed better than Faster and Fast RCNN. These results reciprocate with the claims that RetinaNet (which includes Focal Loss) works better in densely packed scenes. However, it is surprising that RCNN variants performance is lower than YOLOv5 given that the literature claims that the former works well for densely packed scenarios. Based on our experiments, we found that YOLOv5 is faster, robust and relatively highly performant and hence the later experiments are conducted using that architecture.

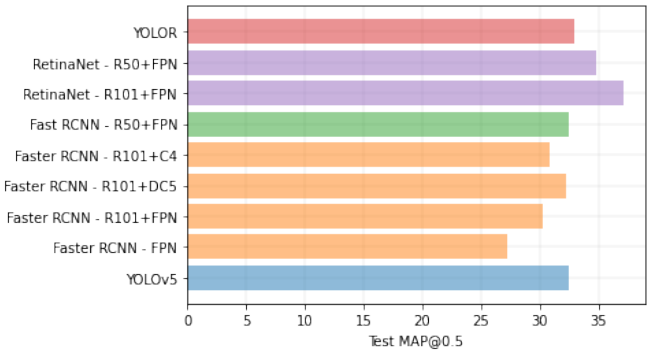

Figure 10. Test mAP@0.5 for different architectures

5.3. SKU Pre-Training and Transfer Learning

As we have mentioned in our approach before, there is a significant domain shift from COCO dataset to the Retail-2K dataset. And also, as our dataset is relatively small, it may not be sufficient to significantly shift the domain. Hence, we utilized the SKU-110K dataset to first transfer the weights from standard COCO to densely packed retail shelf settings and then trained on our final task. We first initialize the architecture either randomly or with COCO trained weights. Then we train on SKU-110K dataset as a

*R101+FPN(32x8d)

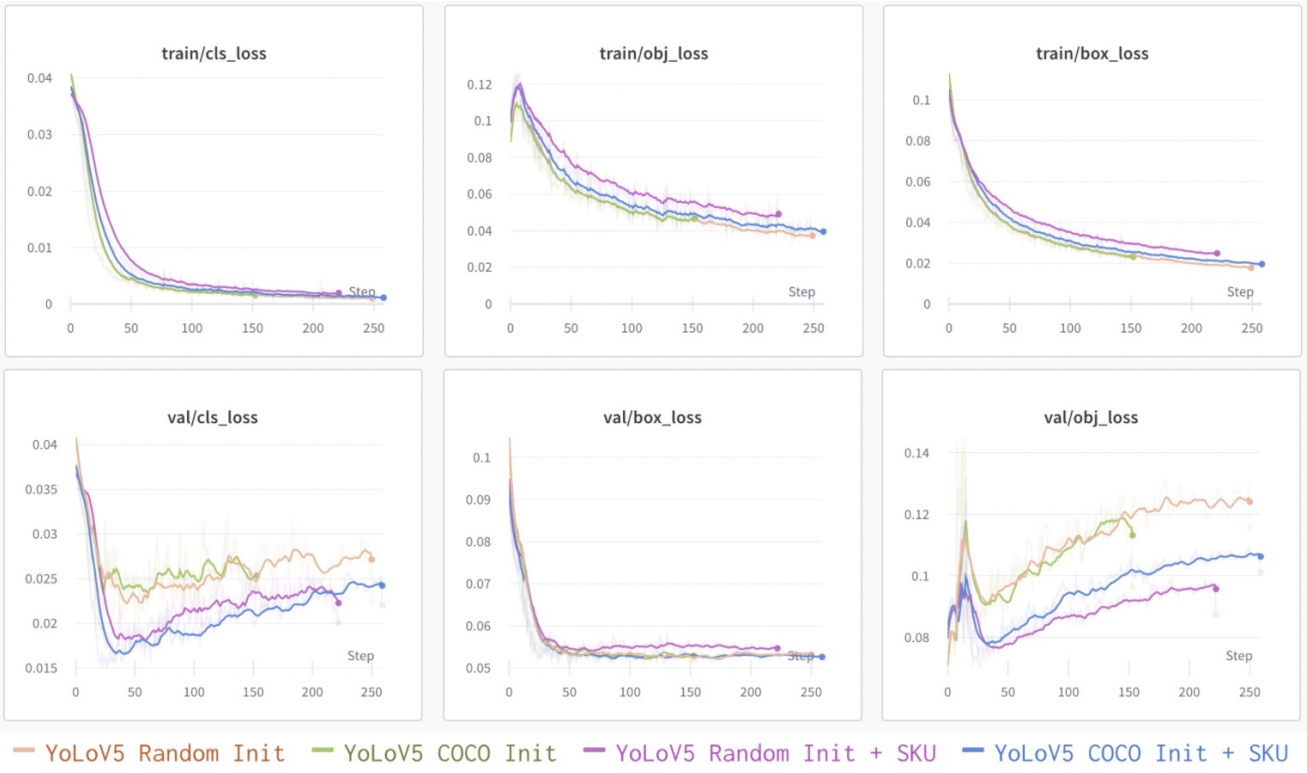

Figure 6. Training and validation loss curves for different initialization and SKU pre-training settings

Figure 7. Testset Ground Truth

Figure 8. Testset Detections (using YOLOv5m COCO + SKU)

single-class product detection. Finally, train on Retail-2K as a multi-class empty detection.

	Validation		Test	
Initialization + Dual Training	**$AP^{0.5}$**	**AP**	**$AP^{0.5}$**	**AP**
YOLOv5 Rand Init.	41.3	18.2	29.2	9.2
YOLOv5 Rand Init. + SKU	45.5	19.6	30.1	10.6
YOLOv5 COCO Init.	43.5	20.5	32.4	12.9
YOLov5 COCO Init. + SKU	**47.7**	**22.8**	**36.5**	**14.6**

Table 4. Performance using SKU dual training and different initializations

From Table 4, we can see that using SKU-110K in pre-training (dual transfer learning) strategy increased the performance by 4% compared to just COCO initialization. Even while using random initialization with SKU pre-training, the results surpass the COCO initialization which shows the importance of SKU dataset in domain transfer especially in cases where the fine-tuned dataset size is small.

The training and validation losses from Figure 6 show that SKU pre-trained models show faster convergence and higher performance compared to other settings. Overall, this additional pre-training improved the MAP results by 3-4%.

5.4. Changing and Freezing Backbone Layers

Motivated by how domain transfer (SKU pre-training) improved our results, we wanted to understand the impact of freezing the backbone layers after SKU pre-training. Table 5 shows that there is an improvement in performance when we freeze the final few layers in the backbone compared to full fine-tuning. However, as we freeze more layers, the learning doesn't happen well for the fine-tuned dataset and the performance drops. This is due to the fact that the high-level SKU product features are embedded in the initial layers and the task specific fine-tuned features are learned by the last layers.

	Validation		Test	
Backbone Freeze	**$AP^{0.5}$**	**AP**	**$AP^{0.5}$**	**AP**
No	47.7	22.8	36.5	14.6
Last-5	**49.0**	**19.7**	**38.1**	**14.3**
Last-10	42.8	17.2	32.6	12.3
Last-15	43.3	18.1	31.2	11.1
Last-20	41.4	15.8	26.6	09.2

Table 5. Performance with freezing different number of layers in backbone of YOLOv5 COCO with SKU pre-trained model

5.5. Architecture Size

Till now, we have used the small version of YOLOv5 (Yolov5s) as the dataset we have is relatively small. However, we experimented with YOLOV5m (medium) to understand the gains per size on our limited dataset along with SKU pre-training. The medium architectures differ in terms of number of blocks in the backbone, head and neck of the standard YOLOv5 architecture. Table 6 shows that the medium size models indeed gave better performance compared to small model architecture. Also, the previous results of SKU pre-training holds true here as well where we can see the medium model with SKU pre-trainning gave highest validation $mAP^{0.5}$ of 51.8%. Comparing the middle rows, we can deduce that the use of SKU pre-traininng with small model is equivalent to using medium model directly. Hence, we can gain in model size if a similar domain dataset is available.

	Validation		Test	
Model + Init. + Pre-Training	**$AP^{0.5}$**	**AP**	**$AP^{0.5}$**	**AP**
YOLOv5s COCO Init.	43.5	20.5	32.4	12.9
YOLov5s COCO Init. + SKU	47.7	22.8	36.5	14.6
YOLOv5m COCO Init.	47.3	23.8	31.2	14.2
YOLov5m COCO Init. + SKU	**51.8**	**25.4**	**38.0**	**15.8**

Table 6. Performance of YOLOv5 small vs medium architectures

5.6. NMS IoU and Confidence Thresholds

Non Maximum Suppression (NMS) is an inference strategy that selects one entity (bounding box) from a set of several overlapping bounding boxes. The selection criteria can be chosen to produce specific results. We utilized IoU to calculate the overlap. If the IoU exceeds the threshold, the corresponding bounding box is removed from the inference. As seen in the Figure 11, as the IoU threshold increases, recall would be increasing as the number of bounding boxes prediction would increase and the precision would decrease since the number of incorrect prediction of bounding boxes increases. The values are similar if the threshold value is less than 0.65 and post that the recall increases gradually but the drop in precision is drastic.

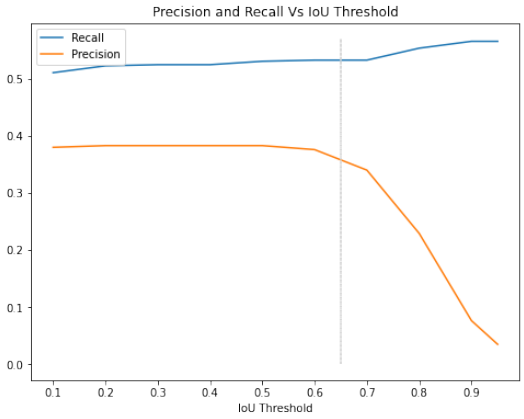

Figure 11. Validation PR trends for different NMS IoU thresholds

Confidence threshold is defined as the minimum confidence that the model has on a detected object (box confidence score). The box confidence score P(object) is 1 if a box is detected else 0. If the confidence threshold is greater than the box confidence score then the corresponding bounding box omitted from the inference. As seen in the Figure 12, as the confidence threshold increases, recall would decrease as less number of bounding boxes will be listed whereas the precision would increase since we are showing only the high confidence bounding boxes. We can select the confidence threshold based on our use-case i.e, bigger thresholds if higher precision is expected and lower thresholds if recall is the priority.

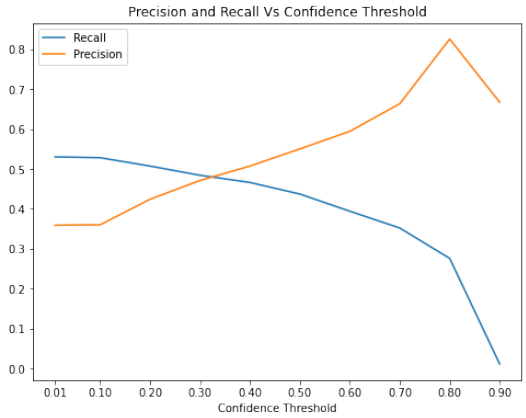

Figure 12. Validation PR trends for different confidence thresholds

5.7. Model Size vs. MAP

Model size is a significant factor in influencing model complexity and speed. It is further important when we want to deploy such models in edge devices in cameras. Figure 13 shows the relationship between the validation MAP values and the magnitude of the model weights for various models. This reveals interesting information on the model's complexity and the MAP relationship. All of the YOLO models are in the lower weight categories, whilst the Faster-RCNN models are in the higher weight sizes. As we can see, the YOLOv5m and YOLOv5s are good examples of MAP and model size trade-offs. On the other hand, even if the model sizes are large, we were unable to attain acceptable accuracies with RCNN models.

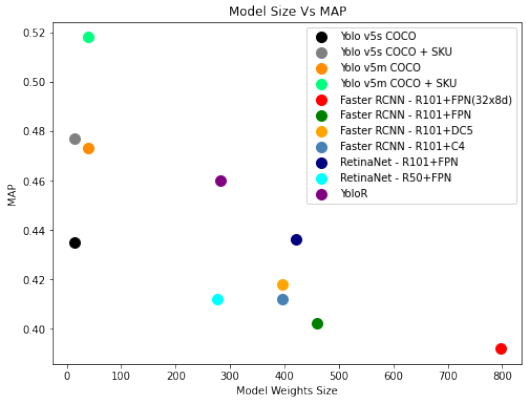

Figure 13. Model weights size(MB) in comparison with MAP

5.8. Single Class Empty Detection

Another sub-problem in emptiness detection is to know how well any class of empty is being detected. Also, based on our sample detections, we realized that the detections are working fine but the class labels aren't being detected well enough. To understand this better, we ran experiments on treating this problem as a single-class object detection. Table 7 shows that when we use single-class inference the performance increased by around 8% showing that bounding box learning in our models is more successful than classification learning. If we perform both training and inference as single-class, the values are additional 10% higher which strengthens the previous argument. Overall, our models are performing very well in detecting the bounding boxes and relatively less successful in discriminating the classes.

	Validation		Test	
Multi vs Single Class	**$AP^{0.5}$**	**AP**	**$AP^{0.5}$**	**AP**
MC Training + MC Inference	47.7	22.8	36.5	14.6
MC Training + SC Inference	53.5	24.4	43.9	17.5
SC Training + SC Inference	63.9	31.9	53.4	27.1

Table 7. Performance metrics of single-class emptiness detection

5.9. Focal Loss and Box level Augmentation

We have carried out experiments by adjusting focal loss weightages, but found that our model is taking significantly longer time to converge. Also, we tried a novel technique of using box-level data augmentation which increases the brightness of bounding box region. We explored this option since lot of our boxes are very dark and we don't want the model to learn that. However, the results are very similar and didn't show any improvement.

6. Conclusion

In this project, we have developed techniques to detect different categories of emptiness in retail shelves in real-world settings. The collected small novel dataset presented lot of modelling challenges and opportunities to be explored. SKU pre-training and transfer learning gave better results compared to other settings. YOLOv5 models have better performance as well as trade-off in terms in terms of model size. We also saw that careful considerations layer freezing, NMS IoU, and confidence thresholds can further improve the performance.

In future work, we plan to focus on semi-height and semi-depth classes by introducing special loss metrics and layers that tackle product stalking. Also, the feasibility of second idea of negating the product detections can be explored. For deployment settings, knowledge distillation techniques can be explored to further reduce the size of the model.

References

[1] Jun Chen, Shu-Lin Wang, and Hong-Li Lin. Out-of-stock detection based on deep learning. In De-Shuang Huang, Vitoantonio Bevilacqua, and Prashan Premaratne, editors, *Intelligent Computing Theories and Application*, pages 228–237, Cham, 2019. Springer International Publishing.

[2] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[3] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[4] Eran Goldman, Roei Herzig, Aviv Eisenschtat, Jacob Goldberger, and Tal Hassner. Precise detection in densely packed scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[5] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, TaoXie, Jiacong Fang, imyhxy, Kalen Michael, Lorna, Abhiram V, Diego Montes, Jebastin Nadar, Laughing, tkianai, yxNONG, Piotr Skalski, Zhiqiang Wang, Adam Hogan, Cristi Fati, Lorenzo Mammana, AlexWang1900, Deep Patel, Ding Yiwei, Felix You, Jan Hajek, Laurentiu Diaconu, and Mai Thanh Minh. ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference, Feb. 2022.

[6] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017.

[7] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[8] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[9] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. You only learn one representation: Unified network for multiple tasks. *CoRR*, abs/2105.04206, 2021.