Department of Electronic & Telecommunication Engineering

University of Moratuwa

EN3160 - Image Processing and Machine Vision

Product Detection in Densely Packed Scenes

(Project Report)

200087A Chandrasiri Y.U.K.K.

200094R Clarance L.G.S.

Abstract

This report presents a solution to the detection challenge held within the CVPR 2020 Retail-Vision workshop. Our approach involves training a YOLO NAS model for the detection of retail product items from shelf images, employing the SKU110K dataset. The source code is available at [GitHub]

Introduction

The primary objective of this challenge is to detect products within densely populated store displays. This task is centered around the comprehensive SKU110K dataset, which encompasses 11,762 shelf images collected from various supermarkets worldwide. The dataset focuses on a single-class object detection task.

SKU-110K images are partitioned into train, test, and validate splits. However due to the limitations of resources, our Training split consists of 80% of the images (1000 images) and their associated bounding boxes; 10% of the images (100), are used for validation and 100 images are used for testing. Images were selected at random, ensuring that the same shelf display from the same shop does not appear in more than one of these subsets.
The rest of the report is structured as follows: First in **Related work** we mention the existing alternatives for addressing this problem, in **Method** we briefly explain our solution and additionally highlight the features which we found important. **Results** contain all the experiment results and the explanations. In the final subsection **Discussion** we discuss how we can improve our solution.

Related Work

For the purpose of setting up a fair baseline two main popular object detection models were chosen: Faster-RCNN and RetinaNet. Both models are used with default configurations utilizing ResNet-50 backbone and FPN.

ResNet-50 is a popular variant of the Residual Network (ResNet) architecture, known for its deep structure and skip connections that enable the training of very deep neural networks. ResNet-50 consists of 50 layers and is widely used as a backbone network.

FPN (Feature Pyramid Network) leverages a top-down architecture with lateral connections to build a feature pyramid from a single-scale input image.

The following image shows the performance of the default models,

Table 1: Revisiting default models

Model	mAP	$AP@^{0.5}$	$AP@^{0.75}$	AR^{300}
RetinaNet-r50-fpn	0.463	0.751	0.532	0.512
Faster-RCNN-r50-fpn	0.523	0.850	0.592	0.582

Method

In our project, we employed Python as our programming language and leveraged Google Colab as our coding environment. Initially, we undertook the essential task of installing the required packages and dependencies.

To facilitate the training of our dataset, we employed the "**supergradients**" package. Additionally, we utilized the "**Supervision**" and "**OpenCV**" packages to perform annotation and generate output for our results. Rendering was achieved with the "**libglu1**" and "**libglu2**" libraries.

Our workflow commenced by utilizing the "**dataloader**" component within the "**supergradients**" package to load our data into the coding environment. Specifically, we curated a dataset comprising 1000 training images, 100 validation images, and 100 test images, all sourced from the SKU 110K dataset. To standardize the format of these images for training, validation, and testing purposes, we employed the Coco detection YOLO format.

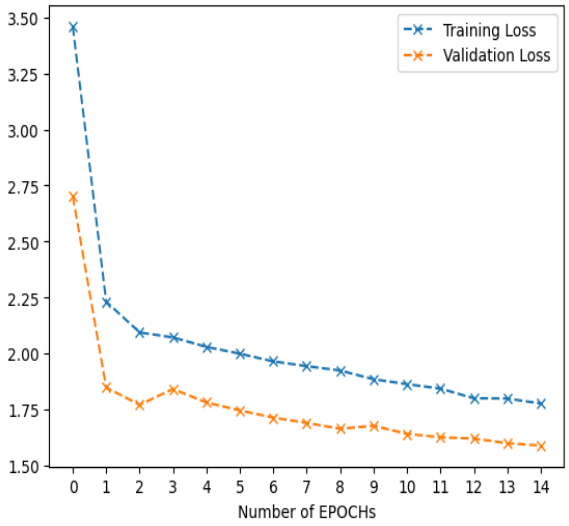
To optimize the efficiency of our project, we employed **PyTorch** to determine the availability of **CUDA**, and when it was accessible, we executed our code on a GPU T4 in Google Colab. Subsequently, we imported a pre-trained **YOLO** model specified for these tasks "**YOLO-NAS-S**" (with 'S' indicating the small version), along with pre-trained weights from the **Coco dataset**, which had been trained for similar tasks. For our loss function, we utilized the "**Ppyoloeloss**".

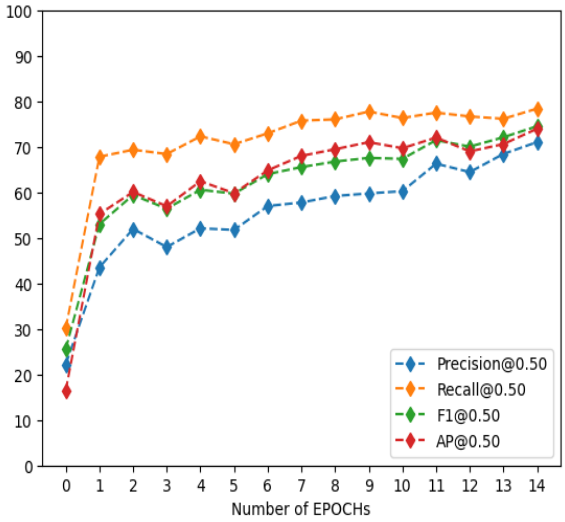
We conducted training for 15 epochs and achieved promising results, including an Average Precision score exceeding 0.7.

Results

The final performance evaluation of our model showcases its effectiveness in addressing the task at hand. Our model achieved a commendable precision score of 0.7109, indicating its ability to accurately classify positive instances. Additionally, a recall of 0.7845 demonstrates its proficiency in capturing a substantial portion of the relevant data. The F1 score, which balances precision and recall, stands at 0.7458, signifying a well-rounded performance. Furthermore, the **Mean Average Precision (MAP) score of 0.7402** highlights the model's capability to rank and retrieve relevant information effectively. In terms of training and validation loss, our model displayed a minimal training loss of 1.7765 and a validation loss of 1.5888, which suggests that it is well-regularized and generalizes effectively to unseen data. These results collectively indicate that our model has achieved a strong level of performance and holds promise for the intended application.

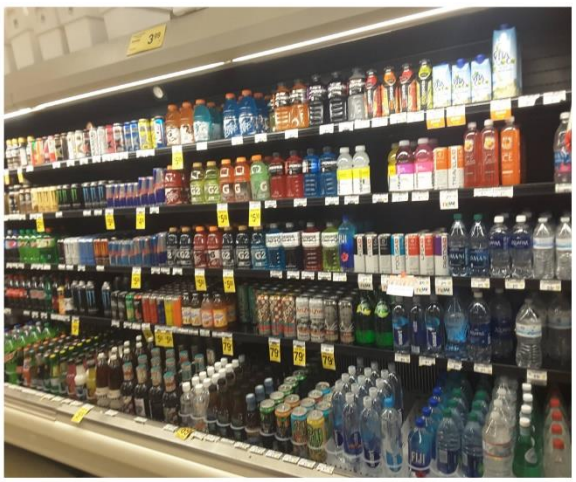
The following graphs illustrate how our model's performance evolves as the number of training epochs progresses.





Upon completing the model training phase, we proceeded to assess its performance using the test dataset provided by the original dataset. The results for a set of chosen sample images are presented below.

Testimage1.jpg



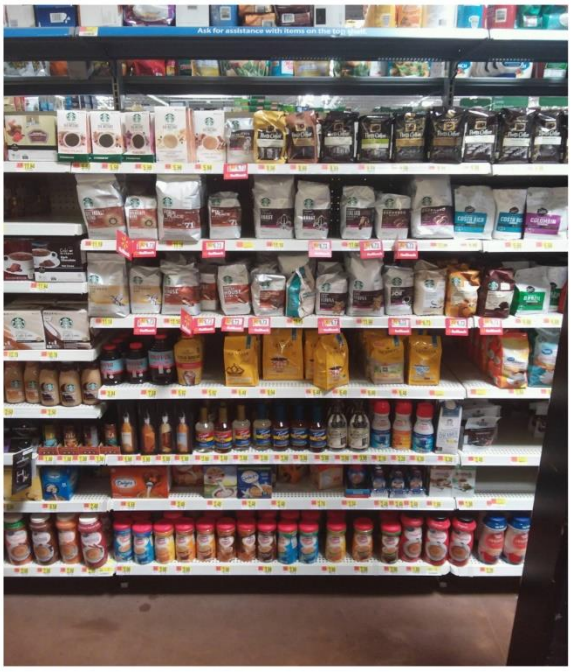
Original Image
.

The image with annotated boxes corresponding to the provided labels

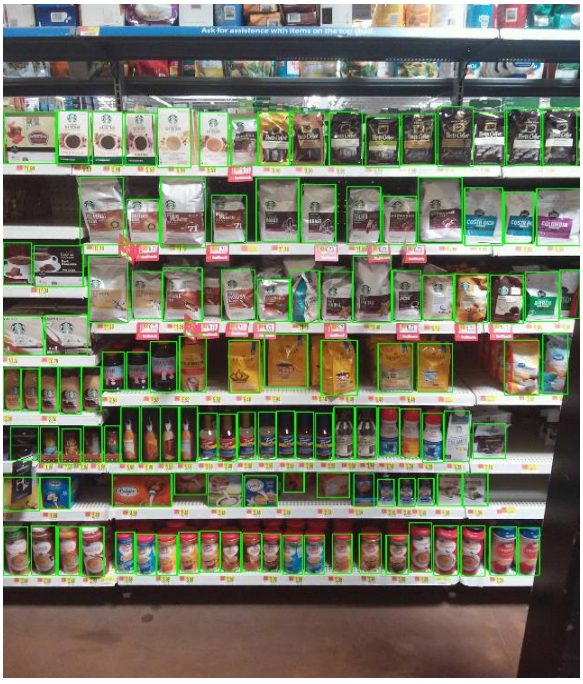
The image with annotated boxes predicted by our model

Comparison of our model and provided labels

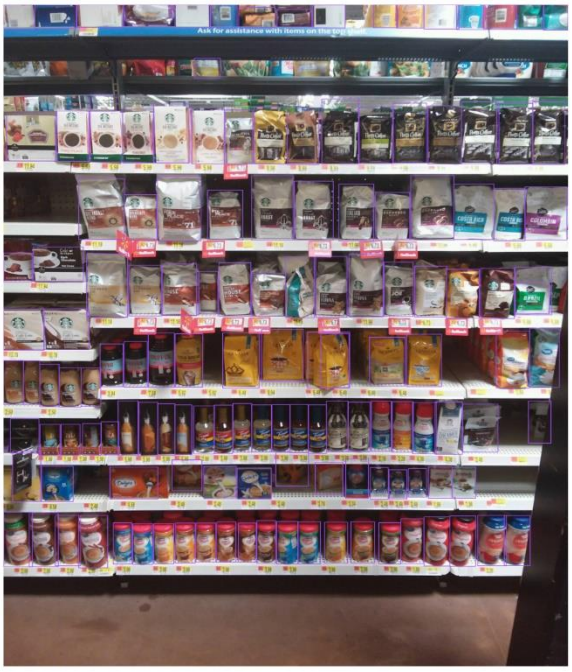
Testimage2.jpg



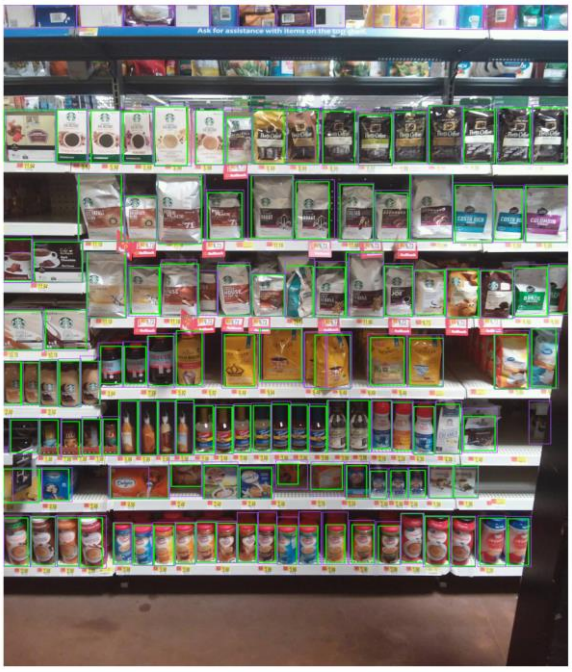
Original Image
.



The image with annotated boxes corresponding to the provided labels



The image with annotated boxes predicted by our model



Comparison of our model and provided labels

Testimage3.jpg

Original Image

.

The image with annotated boxes predicted by our model

Discussion

As previously mentioned, our model was trained using a limited dataset of 1000 images for the training set and 100 images for the validation set, over a period of 15 epochs. Due to the constraints of available GPU resources and the session time limitations within the Colab notebook environment, our training process was constrained. However, with access to high-end GPUs and the utilization of Colab Pro, it would be possible to train the model on the entire dataset for a high number of epochs.

By leveraging enhanced computational capabilities, we anticipate that the current mean Average Precision (mAP) value of 0.7402 can be further improved, leading to the development of a more refined and accurate model. The ability to train the model on a larger dataset could potentially yield significant enhancements in both the model's performance and its ability to accurately detect products within crowded store displays.

References

[1] Challenge Overview

[2] Data set and Codebase

[3] Technical report

[4] Technical report