INFORMATION AND COMPUTER SCIENCE DEPARTMENT

Video-based Retail Store Shelf Monitoring

Project Report - R5

Team Members

Khaled Almutairy - 202204240

Mamoun Alghaslan - 200818720

Mentors

Prof. Elsayed Mohamed Elalfy

Dr. Abdul Jabbar Siddiqui

Submitted as part of the requirements for the ICS-619 for the Professional Master in Artificial Intelligence

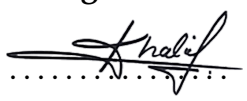
Term: 231, ICS-619: Project

December 13, 2023

Honor Pledge

We, the team members, affirm that all the submitted work is our original work, does not infringe any rights of others, does not include any cheated, plagiarized, fabricated, or falsified content, solely done for this course, and was not submitted and not to be submitted as a major part of any other course or activity. We also affirm that we have not received any unauthorized assistance in this work or disclosed any confidential information without a written consent (as attached). We also affirm that all team members have contributed as described in the work distribution. We also understand that we should not share or post the submitted work by any means without a written consent of the instructor. The submitted work is owned by KFUPM and can be published in any media as long as it gives credit to the contributing team members.

Team Members:

Name	KFUPM ID	Signature	Date
Khaled Almutairy	202204240		12/13/2023
Mamoun Alghaslan	200818720		12/13/2023

Contents

Summary

Seamless is Saudi Arabia's biggest event covering the latest innovations in payments, fintech, retail, e-commerce, home delivery, and digital marketing. While displaying the retail experience of tomorrow, cashier-less stores were demonstrated, which is one of the hottest topics for businesses nowadays with the availability of AI-powered shelf monitoring. Many leading companies are experimenting with unmanned shops with varying degrees of success, such as Amazon Go, Grabango, and Walmart's Intelligent Retail Lab. These companies used weight sensors, Bluetooth beacons, and video surveillance to solve the problem. Relying on video only is a promising topic, potentially dropping the need for other sensors.

The proposed project aims to build a robust Video-based Shelf Monitoring System (SMS) that is designed to revolutionize retail inventory management by tackling the challenges of monitoring On-shelf availability of products, specifically Out-Of-Stock (OOS) and Misplaced (MIS) products. This system will have a significant impact on the retail sector by enhancing efficiency, reducing operational costs, and increasing customer satisfaction.

The key components of SMS include a camera slider system mounted on each shelf. Each camera will move smoothly in a horizontal direction to capture images of the opposite shelf at regular intervals. These captured images will then be fed to a deep-learning model to identify OOS and MIS products. When an issue is detected, the system will trigger an alert that includes the location of the shelf. This proactive approach allows employees to quickly resolve the issue to prevent missed-sales opportunities. The adoption of automated shelf monitoring will save employees time by eliminating the need for manual checks. It will help staff to maintain a well-organized store, reducing customer frustration due to unavailability or misplacement of items.

In summary, the Shelf Monitoring System project has the potential to make a solid impact on the way retail stores monitor and manage their inventory by utilizing cutting-edge deep learning and computer vision technology. It's aligned with Saudi Arabia's initiatives to advance the Vision 2030 of innovation and automation.

Team Members and Contributions

- **Mamoun Alghaslan :** A KFUPM software engineering graduate with 10+ years of working experience in Saudi Aramco as an Exploration Systems Analyst, specializing in full-stack web applications development and support.
- **Khaled Almutairy:** An electrical engineer graduated from Indiana Tech (USA) with two years of experience as a project engineer.

Introduction

Currently, most retail stores address OOS/MIS items through manual inspections, which is laborious and time-consuming. Workers must count products on the shelves while correcting MIS items, prepare the OOS inventory, collect the items from the storage, and replenish the shelves.

Retailers spend a lot on inventory management. Customers will choose to buy their product elsewhere if the store often has OOS/MIS products. Enhancing the inventory management process will save time for everyone. The significance of the Shelf Monitoring System lies in its capacity to accelerate inventory management. Rather than checking all the shelves manually, the system will provide real-time detection and alerts to the store employees, enabling immediate corrective actions.

Several methods have been proposed to tackle the OOS problem in retail using a combination of image processing and machine learning techniques. The current approach for on-shelf availability detection encounters various limitations. Classical methods such as Machine Learning (ML) or image processing suffer from low accuracy even when trained with large data-sets, which is unsustainable in complex retail environments. Although deep learning shows promise for enhancing accuracy, it requires large, annotated data-sets and some methods are computationally expensive, which hinders real-time predictions. Shelf layout, object shapes, and occlusion of products on shelves can be challenging to achieve high precision. There is not a specific model that addresses OOS and MIS at the same time. These limitations emphasize the necessity for a more efficient and accurate solution.

We are aiming to fine-tune a pre-trained object detection model on a new data set from a supermarket, collecting product images while they are directly on the shelves, and then labeling items and empty spaces with bounding boxes. We will begin with one fixed storage unit, with all products placed correctly, and remove products for OOS situations, misplacing them for MIS, and then label them with the boundaries in a systematic manner to cover different possibilities.

Chapter 1 | Background and Related Works

This chapter covers an overview of the fundamentals of deep learning techniques, object detection algorithms, and the evaluation metrics used to assess their performance. In addition, a detailed comparison of related work of shelf monitoring systems is provided. By the end of this chapter, the reader will have a general understanding of deep learning, recent advances in object detection and recognition, and the evaluation metrics associated with each deep learning model that will be used in this project.

1.1 Object Detection and Recognition

The process of detecting an object within an image involves locating it and then finding the best bounding box. To detect the presence of an object is to localize it within a frame. Object recognition, which comes after detection, refers to the classification of the objects. There are two options for implementing these tasks together: traditional image processing techniques, and deep learning networks. The latter outperforms the former in every aspect except for training time. Luckily, fine-tuning a model on a new data-set has shown that transferring knowledge from a pre-trained model by modifying the last few layers is usually sufficient.

1.2 Deep Learning

Deep learning is a sub-field of ML, which is also known as representation learning. Specifically, it is based on artificial neural networks (ANN) with multiple processing layers that learn various levels of representations from raw data. Each layer learns to extract a higher level of abstract features from its input starting with the input data [1]. For example, in image recognition tasks, the image might be formulated as an array of pixel values; the early layers learn features that

represent the locations and orientations of edges in the image with different arrangements, while the middle layers assemble complex patterns of the edges that are part of a large object, such as eyes of a face. Higher layers capture the characteristics of objects to recognize that an image may contain a face. The essential aspect of this hierarchical representations learning process is that it is fully automated from the data without explicit human engineering.

There are many different types of deep learning techniques, with the most widely known being: supervised, semi-supervised, and unsupervised learning. Supervised learning is the most common type, which relies on a labeled data-set to train a mode. For example, when training a model to recognize products on shelves, we first provide the model with a data-set of images, each associated with the respective product names. At the beginning of training, the model starts with initial random guesses. We compute a loss function to measure the error between the output scores and the ground truth. During training, the model iteratively improves its predictions and adjusts its parameters to minimize this error. On the other hand, the unsupervised learning method does not require labeled data. The model can uncover hidden patterns and structures in the data. In the context of recognizing products on shelves, we feed the model with images without their labels, and the model may group similar products based on visual similarity. Finally, semi-supervised learning requires a small set of labeled data and a large amount of unlabeled data to train a model. It is a hybrid approach that combines the advantages of both supervised and unsupervised learning, which is useful when a large amount of labeled data does not exist.

1.2.1 Transfer Learning

There are many Machine Learning models that are efficient. While deep learning models are the most successful technique for computer vision problems, their training time can be computationally expensive and time-consuming. The model weights, or parameters, contain the learned features from the training data-set, which can take a lot of time even with a large amount of processing power. Luckily, there is not always a need to train a new model. A pre-trained model parameters can be used as a basis for another model and can be fine-tuned to adapt to the new task. The similarity between the original data-set and the new data-set dictates how much of the old model (how many layers) needs to be re-trained.

Moreover, shelf products have many intra-variations, and the conditions of shelf stocking

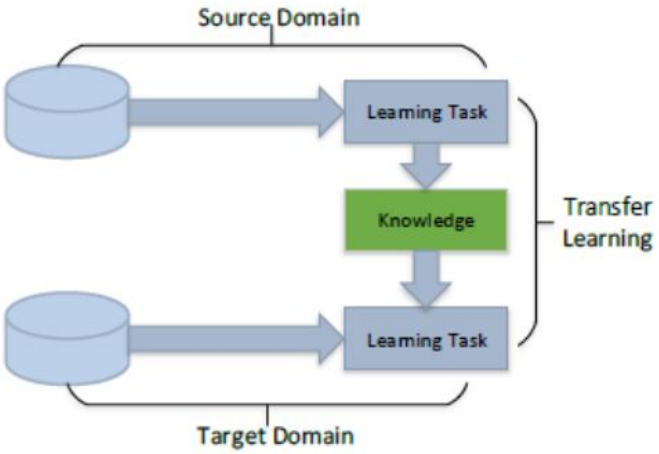


Figure 1.1: An illustration of diagram of transfer learning [2]

and lighting can make it challenging for a single model to generalize well enough to be used in differently conditioned shops. An important assumption when training any deep learning model is that the training and testing portions of the data-set come from the same distribution in the feature space. Using a different data-set for testing a model will yield bad results unless retrained from random initialization. Transfer learning applies previous knowledge to more easily solve new problems. Most established approaches rely on a pre-trained model such as ImageNet, COCO, and Google Open images. Transfer learning can assist in three key ways [3]: it can improve initial performance at the beginning of training the model when compared against the randomly initialized model, reduce the total training time required to learn a new task, and enhance the overall performance compared to the performance without using transfer learning.

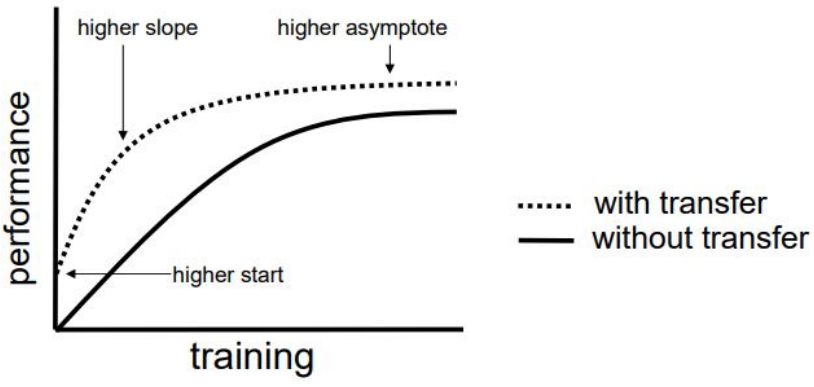


Figure 1.2: Possible improvements using transfer learning. [3]

1.3 Two-Stage Detectors

The two-stage detectors consist of two stages. In the first stage, it generates regions of interest (RoI) called region proposals. In the second stage, the extracted object proposals will be used to perform classification and localization. While two-stage detectors usually reach better localization and classification accuracy, they have a slower inference speed.

1.3.1 R-CNN

R-CNN is a Region-based Convolutional Neural Network (CNN), which was one of the most influential papers in object detection that was introduced in 2014 [4]. CNNs are a type of neural network that preserves the spatiality of data, with multiple stages of feature extraction where each consecutive stage computes more global, invariant, and abstract features. R-CNN is simple to understand. We start by feeding the algorithm an image, and then we extract region proposals from the image by running one of the region proposal methods, such as selective search. Because these region proposals could all be different sizes and different aspect ratios in the image, we warp them to a fixed size. Afterward, we perform a convolutional neural network for each of these warped image regions independently, which will output a classification score for each of these regions. Also, because these bounding boxes generated from the selective search method do not involve any learning mechanism, the CNN will have an additional regression layer, which will transform the region proposals boxes into the final correct bounding boxes for each object in the image.

1.3.2 Fast R-CNN

While R-CNN works well for detecting objects, it is very slow to run in real-time due to the need to run a separate CNN forward pass for each of our region proposals. This is why nowadays it is also called "slow" R-CNN. To overcome this problem, [5] proposed an alternative method called Fast R-CNN. It is basically the same as R-CNN except we swap the order of CNN and the region warping. This way we can process the whole image at a high resolution with a single CNN as a backbone network. The output of this network is a convolutional feature map of the entire image onto which we can project the region proposals. Finally, we run a lightweight pre-region

network that will produce our classification scores and bounding boxes regression transforms for each of the regions. This process is fast because most of the computation is done in the backbone network.

1.3.3 Faster R-CNN

Most of the run-time in Fast R-CNN is consumed by the region proposals that are being computed by the heuristic selective search algorithm on the CPU. Instead, we can eliminate the heuristic selective search algorithm and train a convolutional neural network to the region proposals. This method improved the run-time even further and is called Faster R-CNN [6].

1.4 One-stage Detectors

The one-stage detectors operate directly from the input image to predict bounding boxes and the classification scores in a single feed-forward fully convolutional network without using region proposals. One-stage detectors are much faster than two-stage detectors due to their uncomplicated design.

1.4.1 YOLO (You Only Look Once)

YOLO is one of the most powerful algorithms in object detection in terms of efficiency and speed. Unlike traditional approaches, YOLO treats object detection as a regression problem rather than a classification task. It uses a single convolutional network to predict bounding boxes with their associated class probabilities directly from an image in a single pass. End-to-end optimization is possible because of the fact that the detection pipeline is a single network.

The main concept of YOLO is that it divides the image into S x S grid. Each grid cell is responsible for generating K bounding boxes with confidence scores. These boxes are used to detect objects whose centers fall within that specific grid cell. Also, the model predicts the class probabilities for containing an object within each grid.

YOLO was released in 2016, and since then several improved versions have been proposed [7]. The first version of the YOLO series struggled with detecting small objects in an image.

Therefore, YOLOv2 was introduced with several improvements on top of YOLO [8]. It can handle higher-resolution images to mitigate the issue of detecting small objects. Other improvements include BatchNorm to enhance the model stability and anchor boxes to handle objects of different sizes.

In 2018, YOLOv3 was proposed with significant improvements over previous versions to improve the speed and accuracy [9]. One of the crucial improvements is the use of a new backbone called Darknet-53, which is based on ResNet architecture with 53 convolutional layers. Two years later, YOLOv4 was introduced with an enhanced version of Darknet-53 with CSPNet, which stands for "Cross Stage Partial Network" [10]. It was designed to optimize real-time inference speeds with high-quality performance.

In a couple of months, YOLOv5 was introduced and developed with PyTorch [11]. It was built upon the success of the previous series with additional new features and improvements. The authors use different augmentation techniques to make the model less sensitive to the position of objects within the grid cells to make it more stable to exploding gradients. YOLOv5 comes with five scaled versions to balance the trade-off speed and accuracy, which makes it flexible to different applications and hardware requirements.

In 2022, YOLOv6 was published, which surpassed previous object detection models on accuracy and speed metrics [12]. Unlike previous YOLO versions, YOLOv6 adopts an anchor-free approach to simplify the training pipeline. The authors made additional changes to the architecture design including a penalizable backbone called EfficientRep and decoupled head. In addition, they used a quantization scheme to reduce the memory and computational requirements without affecting the performance as well as a self-distillation strategy for training that improved the accuracy of the model.

YOLOv7 was introduced in 2022 with the aim of increasing the accuracy of the model without the inference speed [13]. The model was trained from scratch without relying on pre-trained backbones. While YOLOv7 is an efficient object detection that can process images at a high rate of 155 FPS, it has some limitations. Like many other object detectors, YOLO v7 may not perform well in detecting small objects and may also struggle with object detection in crowded scenes. Additionally, the size of the objects can impact the accuracy of the model. Very large or small

objects are difficult to detect using YOLOv7. Nevertheless, it is sensitive to changes in lighting conditions, which may be inconvenient in real-world applications.

At the beginning of 2023, [14] YOLOv8 was published by Ultralytics, the same team who created YOLOv5. They made significant improvements over YOLOv5 to improve detection accuracy while maintaining high speed and efficiency. YOLOv8 enhanced the performance of object detection with respect to smaller objects by utilizing CloU [15] and DFL [16] loss functions for bounding box loss and binary cross-entropy for classification loss. The model achieved outstanding performance with a speed of 280 FPS. YOLOv8 comes with a user-friendly CLI and a well-structured Python package. Also, it supports not only object detection but also other vision tasks such as tracking, segmentation, pose estimation, and classification.

Recently in May 2023, [17] YOLO-NAS (Neural Architecture Search) the state-of-the-art object detection model was released. It is built to improve upon key limiting factors of current YOLO models, such as inadequate quantization support and insufficient accuracy latency tradeoffs. By doing so, the model is suitable for real-world edge-device applications. YOLO-NAS uses automatic architecture design using AutoNAC to identify the most suitable structure for a given task. This technology considers all the components of the inference stack, including compiler and quantization. In addition, AutoNAC uses a hybrid quantization method that selectively quantizes specific layers to optimize accuracy and latency tradeoffs while maintaining the overall performance.

1.4.2 SSD (Single Shot MultiBox Detector)

Another method of object detection is SSD, which uses a single Deep Neural Network (DNN), making it very efficient and suitable for real-time detection [19]. It uses mixed pre-defined anchor boxes with different sizes and aspect ratios to divide the bounding boxes. It scores and adjusts the predicted bounding boxes that best fit the detected object. The downside of SSDs is that they are not as accurate as other methods using multiple DNNs, which are able to derive more knowledge from the additional network depth.

Table 1.1: YOLO Architecture Summary [18]

Version	Date	Anchor	Framework	Backbone	AP (%)
YOLO	2015	No	Darknet	Darknet24	63.4
YOLOv2	2016	Yes	Darknet	Darknet24	63.4
YOLOv3	2018	Yes	Darknet	Darknet53	36.2
YOLOv4	2020	Yes	Darknet	CSPDarknet53	43.5
YOLOv5	2020	Yes	Pytorch	YOLOv5CSPDarknet	55.8
PP-YOLO	2020	Yes	PaddlePaddle	ResNet50-vd	45.9
Scaled-YOLOv4	2021	Yes	Pytorch	CSPDarknet	56.0
PP-YOLOv2	2021	Yes	PaddlePaddle	ResNet101-vd	50.3
YOLOR	2021	Yes	Pytorch	CSPDarknet	55.4
YOLOX	2021	No	Pytorch	YOLOXCSPDarknet	51.2
PP-YOLOE	2022	No	PaddlePaddle	CSPRepResNet	54.7
YOLOv6	2022	No	Pytorch	EfficientRep	52.5
YOLOv7	2022	No	Pytorch	YOLOv7Backbone	56.8
DAMO-YOLO	2022	No	Pytorch	MAE-NAS	50.0
YOLOv8	2023	No	Pytorch	YOLOv8CSPDarknet	53.9
YOLO-NAS	2023	No	Pytorch	NAS	52.2

1.5 Evaluation Metrics

Typically, Object detection performance is determined by several factors depending on the model's desired goal. Each prediction can be evaluated using the metrics in Table 1.2.

Table 1.2: Evaluation Metrics

Metric	**Equation**	**Explanation**
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	Measures overall correctness of predictions.
Precision	$\frac{TP}{TP+FP}$	Measures how many of the positive predictions are accurate.
Recall	$\frac{TP}{TP+FN}$	Measures how many of the actual positives are correctly predicted.
Mean Average Precision	$\frac{1}{N}\sum_{i=1}^{N} \text{AP}_i$	A comprehensive metric that combines precision and recall for all classes. AP is a precision-recall trade-off measurement.

Where:

- **True positive (TP):** Correctly predicted as positive.
- **True negative (TN):** Correctly predicted as negative.
- **False positive (FP):** Incorrectly predicted as positive.
- **False negative (FN):** Incorrectly predicted as negative.

1.5.1 Intersection Over Union (IoU)

For object localization, we need to measure how closely the model predicts the bounding box of an object. For that, we need to calculate the area of intersection and area of union (IoU). If $IoU \geq 0.5$, then we consider it a correct prediction.

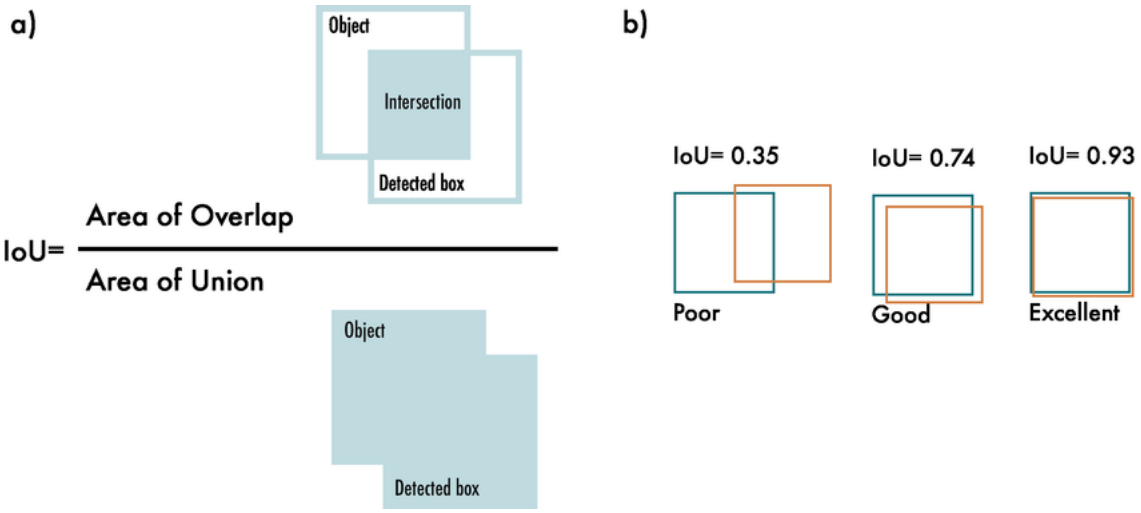


Figure 1.3: a) Intersection over Union b) Examples of different IoU [18]

1.6 Comparison of Related Work

Table 1.3: Comparison table of related work.

Description of work	**Method**	**Finding**
Developed image processing techniques to identify and count products and empty shelve locations. Can count even if two products are next to each other without voids [20].	Using MATLAB only, match the reference image with the target, then identify if the product is missing or misplaced.	Inference time is long, and voids of smaller sizes than pre-trained reference cannot be distinguished.
Data as panoramic images of the aisles were collected manually and then segmented and labeled [21].	Product detection using image processing to extract Geometry, Colour, and Texture, then supervised learning.	OOS accuracy at 84.5% and sensitivity 86.6% for labels.
Proposed an early detection method based on a low-cost embedded system. Uses both top and front-mounted RGB-D cameras in different situations [22].	3D point cloud reconstruction technique.	Great On-Shelf-Availability scores with errors less than 5%. Best of detecting product availability that OOS.
Utilized machine learning and computer vision techniques to detect product percentage, where 100% indicates the shelf is full, 30% low, and 10% sends a warning. Also predicts future sales [23].	Cameras used from two different angles: high-level and eye-level.	Accurate estimation of on-shelf availability and shelve availability percentage.
		Continued on next page

Table 1.3 – continued from previous page

Description of work	Method	Finding
Presented a robust deep learning method for monitoring items on shelves by categorizing detected changes as "product taken" or "product replenished/returned" and tracking all changes in a database [24].	Using ceiling surveillance cameras, detect background and foreground changes, then feed processed images to a CNN.	89.6% success rate for on-shelf availability detection.
Introduced the SKU-110k dataset that contains densely packed retail scenes with numerous items [25].	Benchmarking on baseline models, compared to Applying the EM-Merger unit to Deep-IoU scores.	Average precision of 49.2% by applying a modified RetinaNet with EM-Merger head and Soft-IoU. YOLO9000 is not suitable for 50+ objects.
Addressed the densely packed scene detection problem [26].	Modified approach involving random cropping and optimized Cascade RCNN.	Achieved a mean average precision of 58.7% on SKU-110k.
Introduced a low data benchmark baseline for dense object detection [27].	Used non-max suppression, multiscale testing, with same evaluation metric of COCO.	(mAP=0.56) at a low U of (0.5) using RetinaNet with only 312 images.
Provided a comprehensive review of deep learning research focused on retail product recognition [28].	Demonstrated multiple techniques including Deep Learning, CNNs, data augmentation, generative models, and one-shot learning.	Literature review for each method.

Continued on next page

Table 1.3 – continued from previous page

Description of work	Method	Finding
Proposed a new model combining semi-supervised learning and On-Shelf Availability [29].	Applied YOLOv4 to detect empty, almost empty, and three product categories using WebMarket dataset.	Outperformed RetinaNet and YOLOv3 in terms of accuracy.
Presents an end-to-end machine learning pipeline for real-time empty-shelf detection [30].	Focuses on data quality improvement through data collection, cleaning, and correct data annotation before model training. Utilizes runtime optimizations.	Achieved a mean average F1-score of 68.5% on the test set, with the ability to process up to 67 images per second on Intel Xeon Gold and up to around 860 images per second on an A100 GPU

Chapter 2 | Technical Details of Existing Solutions

2.1 Introduction

The chapter looks into the technical details of existing solutions for detecting out-of-stock (OOS) products on shelves. Various image processing methods using MATLAB have been explored with limited success. Some solutions use image stitching, label detection, and depth information from RGBD point clouds. Deep learning approaches involve Soft-IoU scores, EM-Merger units for bounding box clustering, and innovative methods for handling densely packed objects. Dataset challenges are addressed through semi-supervised learning, with fine-tuned models like RetinaNet, YOLOv3, and YOLOv4. The importance of quality data is emphasized in designing real-time, computationally efficient pipelines, considering accuracy and runtime trade-offs. Figures illustrate system diagrams, network architectures, and the machine learning pipeline for empty shelf detection.

2.2 Classical Method

Basic image processing techniques implemented in MATLAB are employed for low cost and reasonable accuracy. These algorithms include, but are not limited to, Binary Robust Invariant Scalable Key points (BRISK), Features from Accelerated Segment Test (FAST), HARRIS-Stephens, Maximally Stable External Regions (MSER), and MinEigen. [20] proposed a solution that uses image processing by storing a list of product cropped images to serve as the reference, capturing target shelf images using video or camera device, matching reference images with target images both in grayscale and then identifying if the product is misplaced or missing. The input and target image feature descriptors are extracted using the SURF algorithm, then matched and returning the resulting indices of the matched features, passed to the Estimate Geometric

Transform function to determine the presence of the reference in the target, and lastly drawing a bounding box over it. This method has a limitation when dealing with occlusion.

OOS products can be easily detected by looking for empty spots on shelves. However, empty shelves do not always indicate that a product is OOS. The presence of labels is also an important factor for distinguishing whether an empty space is for an OOS product or simply just an empty space. The proposed method by [21] consists of three parts: Stitching panoramic images using homography estimation and Fast Explicit Diffusion (FED), detecting labels using a cascade of weak classifiers measuring Local Binary Pattern (LBP), and detecting OOS by doing aisle segmentation, OOS segmentation, vertical separation of OOS candidates, filtering candidates by L*C*h* color space, and finally feature extraction and classification.

A proposed solution [24] detects the changes in products as decreasing (product taken) or increasing (replenished or returned) with a given initial shelf layout. The proposed process detects changed regions using background subtraction while removing foreground changes between consecutive frames (such as a customer walking through the store) based on moving speed. It then classifies the actual changes using CIFAR-10-based and CaffeNet-based (Figure 2.1) networks in four classes: Product Taken (decrease), Product Replenished (increased), Small position change (No change), and an unrelated false positive (No change) (Figure 2.2). OSA is calculated by predefined monitoring areas as boundary boxes, and then calculating the ratio between the product area and monitoring area in a binary threshold space (Figure 2.3).

Figure 2.1: CaffeNet-based network with five convolutional layers, three pooling layers, and three fully-connected layers. [24]

Figure 2.2: Examples of images from each class of product change classification. [24]

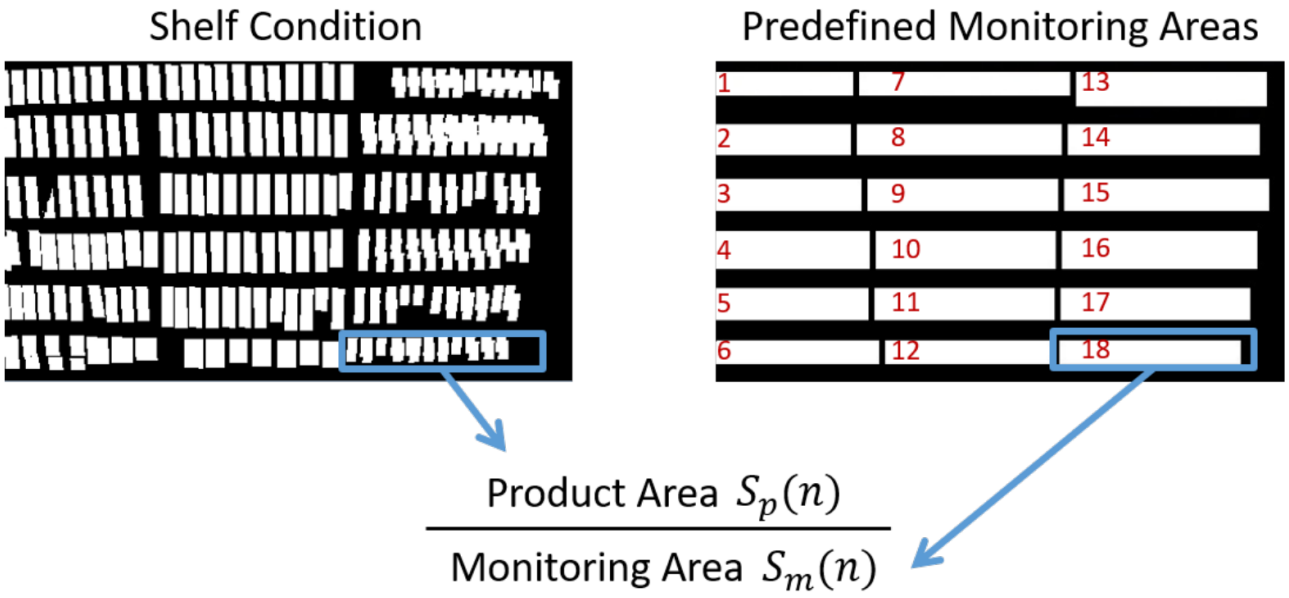


Figure 2.3: On-Shelf-Availability estimate using binary threshold of products (Left) and monitoring area boundary boxes (Right). [24]

2.3 Depth Estimation Method

RGBD (RGB images with Depth information) point cloud data captured using consumer-grade sensors have been implemented to estimate the availability of products and OOS, with no prior knowledge [22]. The proposed system was made to monitor perishable fresh products stored in countertops, baskets, or crates, although it can monitor any type of layout.

Top-mounted and front-facing camera setups were experimented with a reference model that is first calibrated on empty shelves, regardless of the camera orientation or floor layout; meaning it could be flat, or inclined in any direction. Multiple images are taken of the reference shelf floor and averaged together to reduce the Maximum Likelihood Estimation (MLE) error. Once

the products have been placed, another calibration is done and depth points sticking up of the shelf floor are used to estimate the On-Shelf-Availability (OSA). Segmentation is applied to distinguish points belonging to products from the original reference plane using a threshold that is automatically set from the sensor's Root Mean Square Error (RMSE).

A square 2D grid with a lower resolution of the point cloud image is applied, and all points fitting inside the grid cell are averaged and used as the height. The volume of the product is estimated as the multiplication of the grid cell size and average height. When the products are stocked, the original volume is calculated as V_{max}. As new images (Multiple aggregated images to average out noise) are acquired in fixed periods and the height is reduced when a customer picks an item, the OSA is calculated as $100\% \times V_{new}/V_{max}$. If the product has a relatively fixed size, and only a fixed number of items can be fitted in the area of interest, then the product count estimation can be calculated as $OSA\% \times N_{max}$.

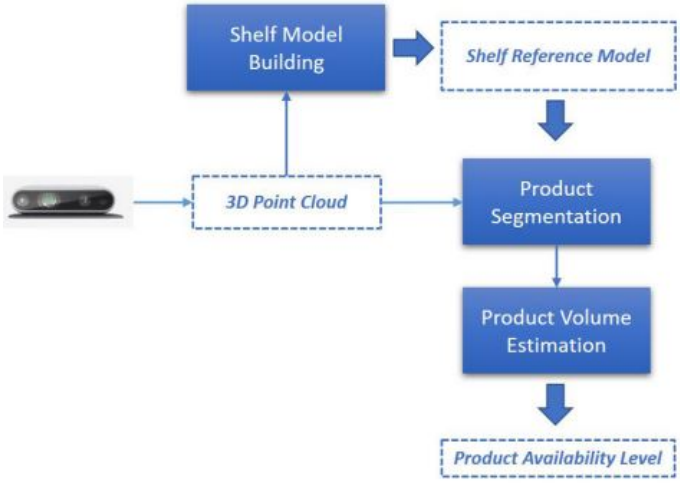


Figure 2.4: Workflow of the depth estimation system in a visual diagram [22].

2.4 Deep Learning Method

Recent advances in object detection techniques have attained remarkable success. However, detecting densely packed objects, especially in scenes containing many identical objects closely packed together, such as products on shelves, remains a challenging task. To tackle this issue, [25] proposed an innovative deep learning-based method with two key components.

The first component is the Soft-IoU score, which estimates the similarity between the predicted bounding box and the ground truth box. It is computed by incorporating a convectional

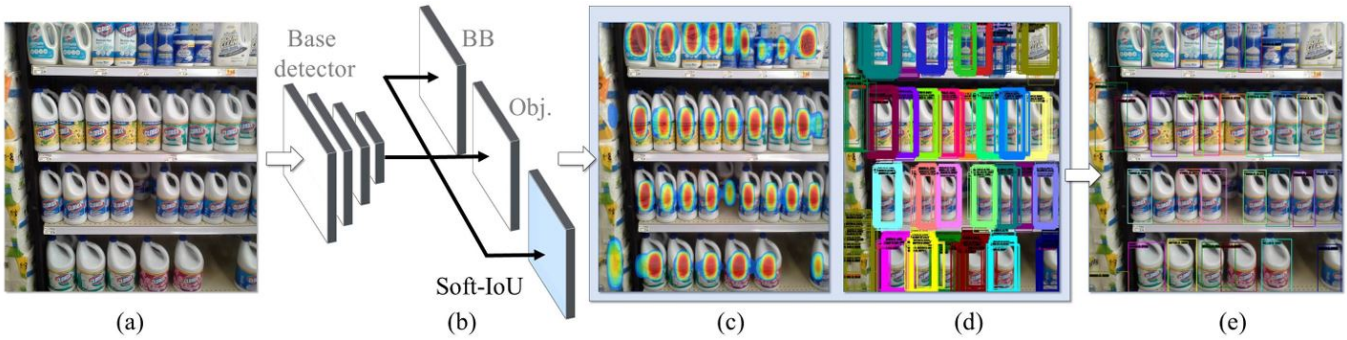


Figure 2.5: System Diagram [25]. (a) Input image. (b) The base network incorporating bounding box (BB) and objectness (Obj.) components, along with Soft-IoU layer. (c) EM-Merger transforms Soft-IoU into a Gaussian heat-map, representing (d) objects detected by multiple overlapping bounding boxes. (e) It subsequently assesses these box clusters to generate a single detection per object (e).

layer as a head on top of the Region proposal network (RPN). This Soft-IoU value helps to resolve the issue of distinguishing overlapping objects that are closely positioned and may have similar appearances.

The second component is the EM-Merger unit, which transforms the predicted bounding boxes along with their corresponding Soft-IoU scores into a Mixture of Gaussians representation. The Expectation-Maximization (EM) algorithm is then employed to cluster these representations into groups, effectively separating overlapping or adjacent detected bounding boxes that were initially identified as separate objects. Figure 2.5 provides a visual representation of the proposed solution

The most difficult part of any machine learning solution is collecting the dataset. Often, real-world datasets are unlabeled, which means that significant human effort is required for annotation. This is particularly the case in retail stores, where collecting and labeling the dataset is an expensive process. There are various techniques to tackle this problem with satisfactory performance, one of which is semi-supervised learning, where a large amount of unlabeled data is utilized in combination with a smaller set of labeled data to train a model. [29] adapt semi-supervised learning and to fine-tune three pre-trained models, RetinaNet, YOLOv3, and YOLOv4 model on the labeled set based on mAP, F1-score, and recall evaluation metrics. Subsequently, the unlabeled set is assigned pseudo-labels generated from the best-performing model's predictions. These pseudo-labels are treated as ground truth during the training process for the unlabeled data. The best model is then retrained using both the labeled data and the unlabeled

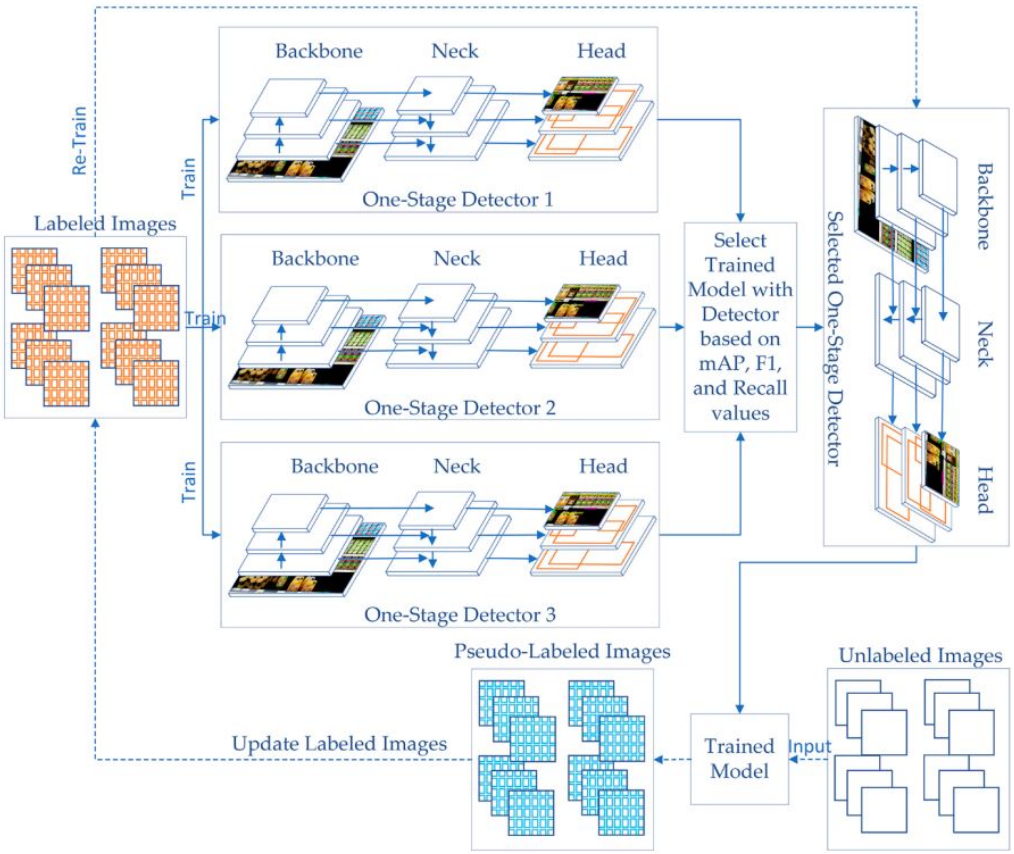


Figure 2.6: The overall framework of the suggested SOSA method [29].

data with the assigned pseudo-labels. The final trained model is employed to detect "Product", "Empty Shelf", and "Almost Empty Shelf" areas on the shelves. Figure 2.6 depicts the architecture diagram.

[30] The authors of this paper focus on the quality of the data, starting from collection, cleaning, and annotation before modeling. They collected a dataset of 1000 images following well-defined guidelines and annotated the images. The main goal of this work is to propose an end-to-end, real-time, computationally efficient pipeline solution for empty-shelf detection. They compared two different versions of both EfficientDet and YOLOv5 models to balance accuracy and inference run-time trade-offs. Additionally, they conducted extensive latency and throughput analysis of the models, utilizing several quantization and inference run-time optimization techniques. The observations are as follows. First, the time spent on image decoding and preprocessing must be optimized using parallelization. Second, the choice of batch size affects the model's maximum throughput. Third, memory requirements for model deployment should be determined carefully as they significantly impact both latency and throughput. Lastly, runtime optimizations like OpenVINO can boost the model's performance.

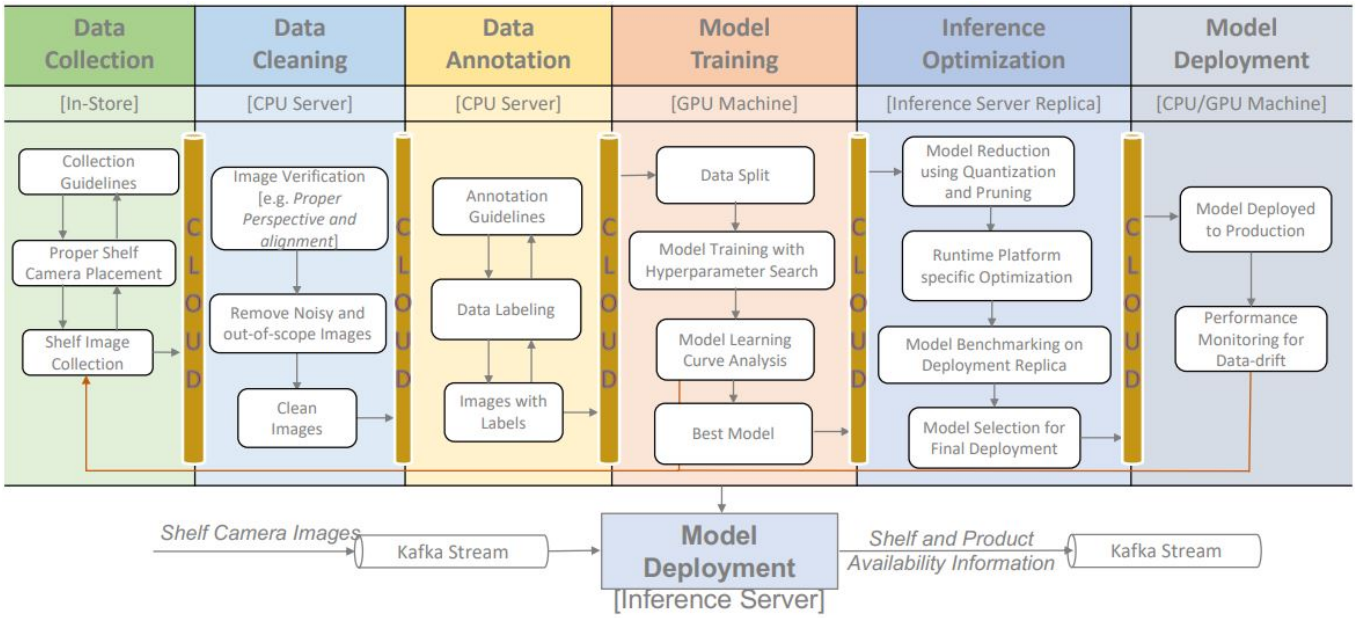


Figure 2.7: illustrates a complete machine learning pipeline designed for the real-time detection of empty shelves [30].

2.5 Conclusion

Various algorithms and methodologies have been proposed to detect out-of-stock (OOS) items, but less attention has been given to identifying misplaced items. Classical methods have their limitations, particularly in detecting occluded objects. Moreover, these methods often struggle to generalize effectively to new and diverse shelf layouts, and product arrangements and are also sensitive to changes in lighting conditions, which can affect their performance. While 3D point cloud methods have shown promising results in estimating OOS by using depth cameras, they do not recognize objects on shelves or detect misplaced items. Deep learning methods offer the flexibility to address these limitations, but most existing solutions primarily focus on OOS detection and ignore the problem of products being in the wrong location. Notably, recent object detection and recognition models like YOLOv8 and YOLO-NAS have not been utilized in solving the on-shelf availability problem. Therefore, our key contribution to this project is to experiment with the recent state-of-the-art models and provide an end-to-end solution for both OOS and product misplacement.

Chapter 3 | Experimentation and Evaluation of Exiting Solutions

3.1 Introduction

This chapter outlines our approach to building a dataset for the classifying of Out-of-Stock (OOS) and misplaced products. We present a detailed procedure of data collection, preprocessing, and augmentations. The primary objective is to establish a baseline model by critically evaluating existing solutions. Section 3.2 discusses our unique dataset creation process, Section 3.3 outlines data preprocessing, and Section 3.4 presents the experiment execution and results.

3.2 Data Collection

To address the lack of publicly available datasets classifying Out-of-Stock and misplaced products, we built one from the Danube supermarket in Saudi Arabia with three classes: Product, OOS, and MIS. Product misplacement occurs when a product is shifted by some distance from its intended position, either vertically or horizontally. However, the distance is not clearly defined. Thus, we consider a product as misplaced if it is located in a position where another product occupies the space between it and its designed placement within a group. For the OOS scenario, we simply remove every item of that product.

The images were captured using a mobile camera mounted on a tripod stand, positioned parallel in front of a shelf for a flat perspective. The dataset focuses only on five shelves, with a total of about 150 products in them. Each product is removed once as if OOS, then an image is captured, and then placed back in the original position, resulting in 150 images, each with a single OOS product. Next, 32 products were randomly selected, and each product was intentionally misplaced in five different places, each with a captured image, resulting in 160 images. A total

of 310 images were collected for quick experimentation and evaluation, with plans to scale up the data in the future to at least 1000 images. The dataset was fully labeled using Roboflow, and the dataset was split into a 90/5/5% ratio for Training, Validation, and Test sets. Unfortunately, Roboflow does not inherently balance the datasets; therefore a manual balancing process was made to ensure precise representation. Following this process, both the Validation and Test sets exhibit a balanced distribution, comprising 15 samples each of OOS and MIS images.

Figure 3.1: Example of a MIS sample (top row) with a bounding box in yellow

3.3 Data Preprocessing and Augmentations

In the preprocessing phase, the images were resized to a resolution of 640x640 pixels. Additionally, we applied auto-adjust contrast through Contrast Stretching. For data augmentations, various transformations were applied to enrich the dataset using Roboflow. Hue, saturation, brightness, and exposure were randomly adjusted within a range of -25% to +25%. We also in-

Figure 3.2: Example of an OOS sample (bottom row) with augmentations

tegrated blurring and noise effects into the data. Furthermore, the Cutout augmentation, which involves removing specific image regions, was performed with three boxes, each covering 10% of the image size. Finally, bounding box transformations are implemented including horizontal shear within $\pm 5°$ and vertical shear within $\pm 1°$.

3.4 Experiment results and conclusion

Using Paperspace online Jupyter Notebook with a single Quadro-P6000 GPU, YOLOV5m and YOLOV7 models were fine-tuned on the dataset for 200 epochs. The model training performance is collected using a Tensorboard. In conclusion, the dataset is extremely imbalanced. The total count of instances of objects in the test images is 6020, of which 25 is MIS and 15 OOS, with the same ratio in the training and validation sets. That is due to the fact that most items on the selves are classified as Product. YOLOv7 has showed great learning capability out-of-shelf with

only 200 epochs of fine-tuning when compared to YOLOv5 which performed badly especially with OOS. Figures 3.1 and 3.2 show the precision, recall, mean average precision mAP, mAP0.5, and mAP0.5:0.95. A more detailed visualization of the model training (fine-tuning) is in the Appendices chapter A1.1.

Class	**Images**	**Instances**	**P**	**R**	**mAP50**	**mAP**
all	40	6020	0.362	0.408	0.55	0.416
MIS	40	25	0.356	0.16	0.246	0.139
OOS	40	15	0.0891	0.0667	0.409	0.255
Product	40	5980	0.64	0.999	0.994	0.853

Table 3.1: YOLOv5 performance metrics on test set

Class	**Images**	**Instances**	**P**	**R**	**mAP50**	**mAP**
all	40	6020	0.916	0.893	0.918	0.672
MIS	40	25	1	0.68	0.759	0.42
OOS	40	15	0.753	1	0.995	0.752
Product	40	5980	0.996	0.999	0.999	0.844

Table 3.2: YOLOv7 performance metrics on test set

Chapter 4 | System Design

4.1 Model Fine-tuning

Using a similar setup explained in 3.4, the YOLOv8s model was fine-tuned on the dataset. The following table summarize the performance metrics. Also, two predicted test samples are shown in 4.1. More examples can be found in the appendix A1.2

Class	Images	Instances	P	R	mAP50	mAP
all	30	4523	0.987	0.909	0.957	0.767
MIS	30	15	1	0.727	0.882	0.648
OOS	30	15	0.962	1	0.995	0.78
Product	30	4493	0.998	1	0.995	0.873

Table 4.1: YOLOv8 performance metrics on test set

4.1.1 Discussion

The YOLOv8 model exhibits strong overall performance on the test set, as detailed in Table 4.1. It achieved high precision (0.987) and recall (0.909). The mean average precision at 50% (mAP50) and overall mean average precision (mAP) are 0.957 and 0.767, respectively. For Misplaced products (MIS), the model achieves perfect precision (1) but with a lower recall (0.727), suggesting potential improvements in localization (mAP = 0.648). Regarding the Out-of-stock (OOS) items, the model demonstrates a good performance with a precision of 0.962, perfect recall (1), and high mAP values (0.995 and 0.78 for mAP50 and mAP, respectively). The 'Product' class is not our primary focus, as we are specifically interested in detecting OOS and MIS. However, including this class is necessary to enable the model to learn the correct location of each product. In conclusion, YOLOv8 shows reliable performance, especially in detecting Out-of-stock items. While

(a) Example of predicted misplaced product.

(b) Example of predicted out-of-stock product.

Figure 4.1: Examples of predicted product issues. (a) shows an example of a predicted misplaced product, while (b) illustrates an example of a predicted out-of-stock product.

achieving high precision for misplaced products, there is room for improvement in recall.

4.1.2 Error Analysis

In the process of evaluating the model's performance, a detailed error analysis was conducted to identify instances where the model failed to correctly detect the classes. Notably, our investigation revealed a specific challenge related to the detection of misplaced products, particularly those situated in front of items with similar shapes or colors.

The model struggled in scenarios where the misplaced products shared visual similarities with neighboring items. A visual representation of these challenges is illustrated in 4.2. We believe that it's necessary to augment the dataset with a variety of samples, particularly images with different product arrangements. Although the current model demonstrates a good performance for this task, a larger and more diverse dataset would provide it with a richer set of cases to learn from.

(a) Example illustrating the model's difficulty in detecting a misplaced product in a visually complex scene.

(b) Example showcasing challenges in detecting a misplaced product when surrounded by visually similar items.

Figure 4.2: Examples illustrating instances where the model missed the correct detection of misplaced products.

4.2 System components

The system has the following four components:

1. **Detection model**: The detection model is a fine-tuned YOLOv8s (You Only Look Once version 8 small) architecture used for shelf-monitoring. YOLOv8s is chosen for its efficiency and real-time processing speed.

2. **Model API**: The Model API serves as the interface between the detection model and the rest of the system. Implemented as a RESTful API, it allows seamless communication between components. This API receives image data from the Web Server, processes it through the detection model, and returns the detected objects along with their bounding boxes and confidence scores. It ensures scalability and flexibility by abstracting the underlying model implementation details.

3. **Web Server**: The Web Server acts as the central hub for system operations, housing the core application logic. It comprises three sub-components:

   (a) **Website server**: The website server provides a user-friendly interface for adminis-

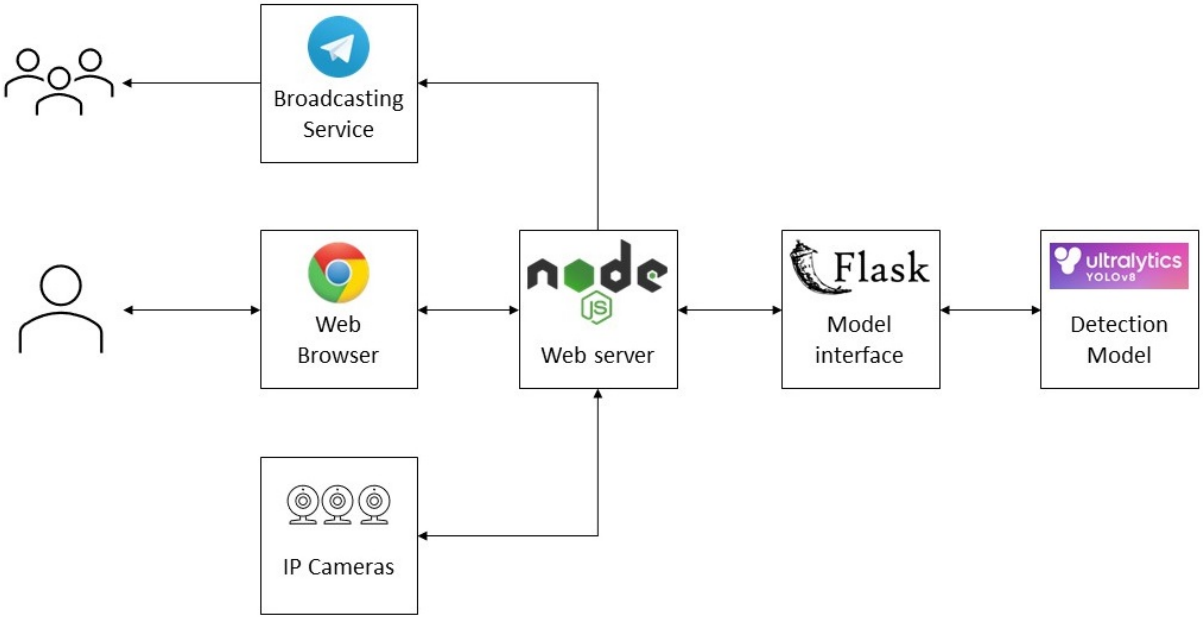


Figure 4.3: System high-level design

trators to manage IP cameras and configure notification settings. Admins can add, remove, or modify registered IP cameras and associated employees. Additionally, this component facilitates the customization of notification thresholds and rules, enhancing the system's adaptability to different supermarket layouts and operational requirements.

(b) **Web worker**: The Web Worker is responsible for periodically querying all registered IP cameras for new images. It efficiently manages the scheduling and distribution of image retrieval tasks, ensuring a timely and systematic collection of data.

(c) **Broadcasting service**: The Broadcasting Service is tasked with sending notifications to registered users based on the object detection results. It integrates with various communication channels, such as telegram service or push notifications, and applies configurable notification policies set by administrators. This component guarantees timely and relevant alerts to employees.

4. **IP Cameras**: IP Cameras are distributed over supermarket shelves to capture real-time images. Each shelf has a camera slider system to move smoothly in a horizontal direction at regular intervals, capturing images of the opposite shelf. By utilizing this slider system, supermarkets may reduces the number of required cameras, thereby achieving a cost-efficient solution.

Chapter 5 | Preliminary Prototype

5.1 Implemented Components

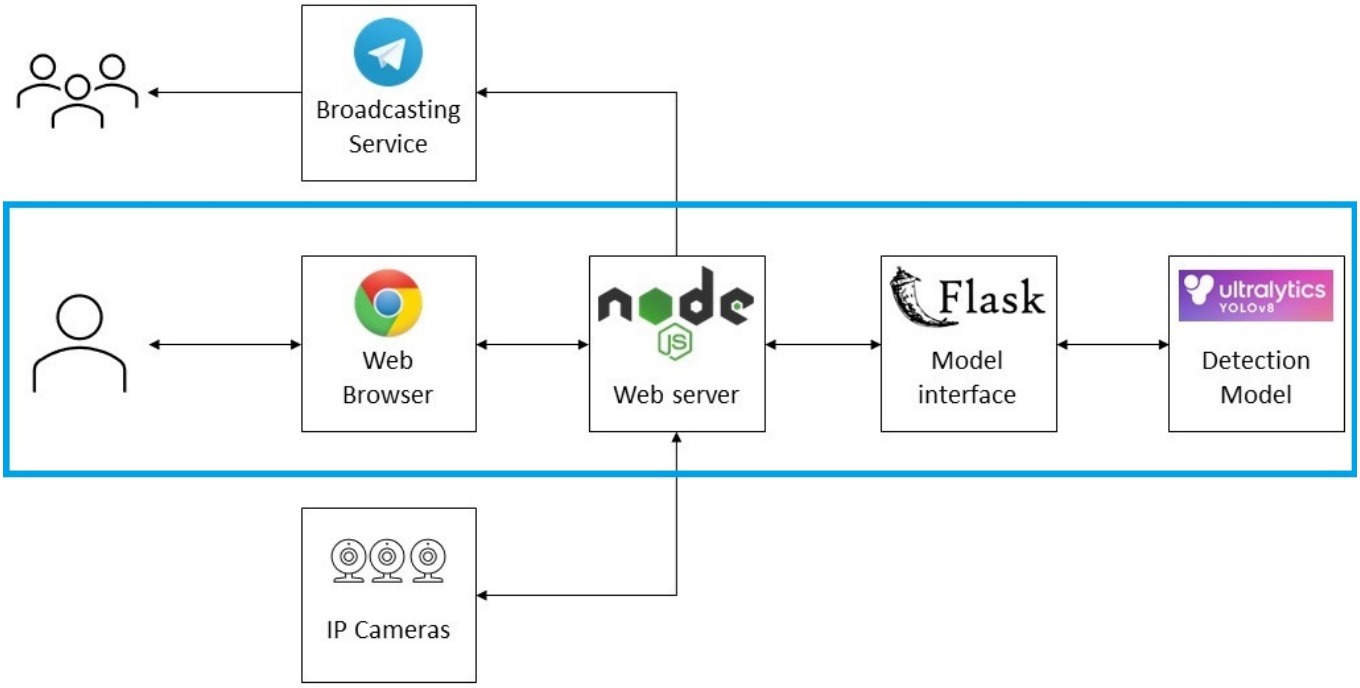


Figure 5.1: System high-level design with the preliminary implemented components boxed

1. **Application server**: The Application Server is implemented as a *Node.js* server using the *Express* framework. It runs locally, listening for incoming requests. The server serves a single HTML page with embedded JavaScript functions. This user interface allows users to either upload a photo or capture one using the device's camera. Upon user interaction, the Application Server sends the captured or uploaded photo to the Model Server for processing. The resulting photos, both the original and those annotated with object detection information, are displayed side-by-side on the HTML page. The use of *Node.js* and *Express* ensures a lightweight and responsive server environment.

2. **Model server**: The Model Server, implemented as a *Flask* application, is responsible for

handling the fine-tuned model. It exposes a single POST method that accepts image data for processing. The uploaded image is resized to 640×640 resolution, a format compatible with the model's input requirements. The image is then fed through the fine-tuned model, and the server returns the predictions, including detected object classes, bounding boxes, and confidence scores. The *Flask* application is designed for simplicity and efficiency, focusing on the specific task of object detection. This modular approach allows for easy integration into the broader system architecture while maintaining a streamlined and dedicated service for model processing.

Conclusion

In this project, we proposed and developed a Video-based Retail Store Shelf Monitoring System (SMS) using deep learning models. The model with the best performance is implemented and deployed in the application. The SMS aims to address the challenges faced in retail inventory management, specifically focusing on Out-Of-Stock (OOS) and product misplacement (MIS).

To tackle the shortage of publicly available datasets specifically designed for classifying OOS and MIS, we created a dataset from the Danube supermarket. The dataset contains high-resolution images of a subsection of a shelf, which were carefully labeled for the presence and location of individual products, as well as identifying instances of OOS and product misplacement. We plan to scale up the dataset in the future, covering a wider range of sections of a shelf as we think it would be valuable for the community to benchmark the performance of retail shelf monitoring system models.

Based on the evaluation results presented in Table 4.1, our developed SMS system utilizing YOLOv8 demonstrates excellent performance in accurately detecting instances of OOS and MIS. The system achieves high precision (P) and recall (R) values across the different classes, resulting in mean Average Precision (mAP) scores of 0.882 for MIS and 0.995 for OOS.

Furthermore, we developed a preliminary prototype of the SMS, showcasing the implemented components and their functionalities. The prototype serves as a proof-of-concept for the feasibility and potential of our system. In the next term, we will focus on testing the system in real-world retail environments with the slider camera system.

Our future work can focus on refining the system's accuracy, scalability, and integration with existing retail management systems. Additionally, we will explore adding additional functionalities such as shelf restocking to further enhance the system's capabilities.

In conclusion, the proposed Video-based Shelf Monitoring System offers a promising solution to enhance retail inventory management. As we continue to iterate on the system, our ultimate goal is to provide a robust and scalable tool that not only addresses current challenges but also anticipates and adapts to the evolving needs of the retail industry.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] J. M. Eyu, "Application development for product recognition on-shelf with deep learning," Ph.D. dissertation, UTAR, 2022.

[3] L. Torrey, J. Shavlik, T. Walker, and R. Maclin, "Transfer learning via advice taking," *Advances in Machine Learning I: Dedicated to the Memory of Professor Ryszard S. Michalski*, pp. 147–170, 2010.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[5] R. Girshick, "Fast r-cnn in proceedings of the ieee international conference on computer vision (pp. 1440–1448)," *Piscataway, NJ: IEEE.[Google Scholar]*, vol. 2, 2015.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[8] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[9] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[10] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[11] G. Jocher. (2020) Yolov5 by ultralytics. Accessed: February 30, 2023. [Online]. Available: https://github.com/ultralytics/yolov5

[12] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie *et al.*, "Yolov6: A single-stage object detection framework for industrial applications," *arXiv preprint arXiv:2209.02976*, 2022.

[13] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 7464–7475.

[14] G. Jocher, A. Chaurasia, and J. Qiu. (2023) Yolo by ultralytics. Accessed: February 30, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[15] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-iou loss: Faster and better learning for bounding box regression," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020, pp. 12 993–13 000.

[16] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang, and J. Yang, "Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 002–21 012, 2020.

[17] R. team. (2023) Yolo-nas by deci achieves state-of-the-art performance on object detection using neural architecture search. Accessed: May 12, 2023. [Online]. Available: https://deci.ai/blog/yolo-nas-object-detection-foundation-model/

[18] J. Terven and D. Cordova-Esparza, "A comprehensive review of yolo: From yolov1 and beyond. arxiv 2023," *arXiv preprint arXiv:2304.00501*.

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, pp. 21–37.

[20] R. Moorthy, S. Behera, S. Verma, S. Bhargave, and P. Ramanathan, "Applying image process-

ing for detecting on-shelf availability and product positioning in retail stores," in *Proceedings of the Third International Symposium on Women in Computing and Informatics*, 2015, pp. 451–457.

[21] L. Rosado, J. Gonçalves, J. Costa, D. Ribeiro, and F. Soares, "Supervised learning for out-of-stock detection in panoramas of retail shelves," in *2016 IEEE International Conference on Imaging Systems and Techniques (IST)*. IEEE, 2016, pp. 406–411.

[22] A. Milella, A. Petitti, R. Marani, G. Cicirelli, and T. D'orazio, "Towards intelligent retail: Automated on-shelf availability estimation using a depth camera," *IEEE Access*, vol. 8, pp. 19 353–19 363, 2020.

[23] H. Priyanwada, K. D. Madhushan, C. Liyanapathirana, and L. Rupasinghe, "Vision based intelligent shelf-management system," in *2021 6th International Conference on Information Technology Research (ICITR)*. IEEE, 2021, pp. 1–6.

[24] K. Higa and K. Iwamoto, "Robust shelf monitoring using supervised learning for improving on-shelf availability in retail stores," *Sensors*, vol. 19, no. 12, p. 2722, 2019.

[25] E. Goldman, R. Herzig, A. Eisenschtat, J. Goldberger, and T. Hassner, "Precise detection in densely packed scenes," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5227–5236.

[26] T. Rong, Y. Zhu, H. Cai, and Y. Xiong, "A solution to product detection in densely packed scenes," *arXiv preprint arXiv:2007.11946*, 2020.

[27] S. Varadarajan, S. Kant, and M. M. Srivastava, "Benchmark for generic product detection: a low data baseline for dense object detection," in *Image Analysis and Recognition: 17th International Conference, ICIAR 2020, Póvoa de Varzim, Portugal, June 24–26, 2020, Proceedings, Part I 17*. Springer, 2020, pp. 30–41.

[28] Y. Wei, S. Tran, S. Xu, B. Kang, M. Springer *et al.*, "Deep learning for retail product recognition: Challenges and techniques," *Computational intelligence and neuroscience*, vol. 2020, 2020.

[29] R. Yilmazer and D. Birant, "Shelf auditing based on image classification using semi-supervised deep learning to increase on-shelf availability in grocery stores," *Sensors*, vol. 21, no. 2, p. 327, 2021.

[30] D. Jha, A. Mahjoubfar, and A. Joshi, "Designing an efficient end-to-end machine learning pipeline for real-time empty-shelf detection," *arXiv preprint arXiv:2205.13060*, 2022.

Chapter A1 | Appendices

A1.1 Appendix I: Training experiments detailed results

A1.1.1 Training Overview - YOLOv5

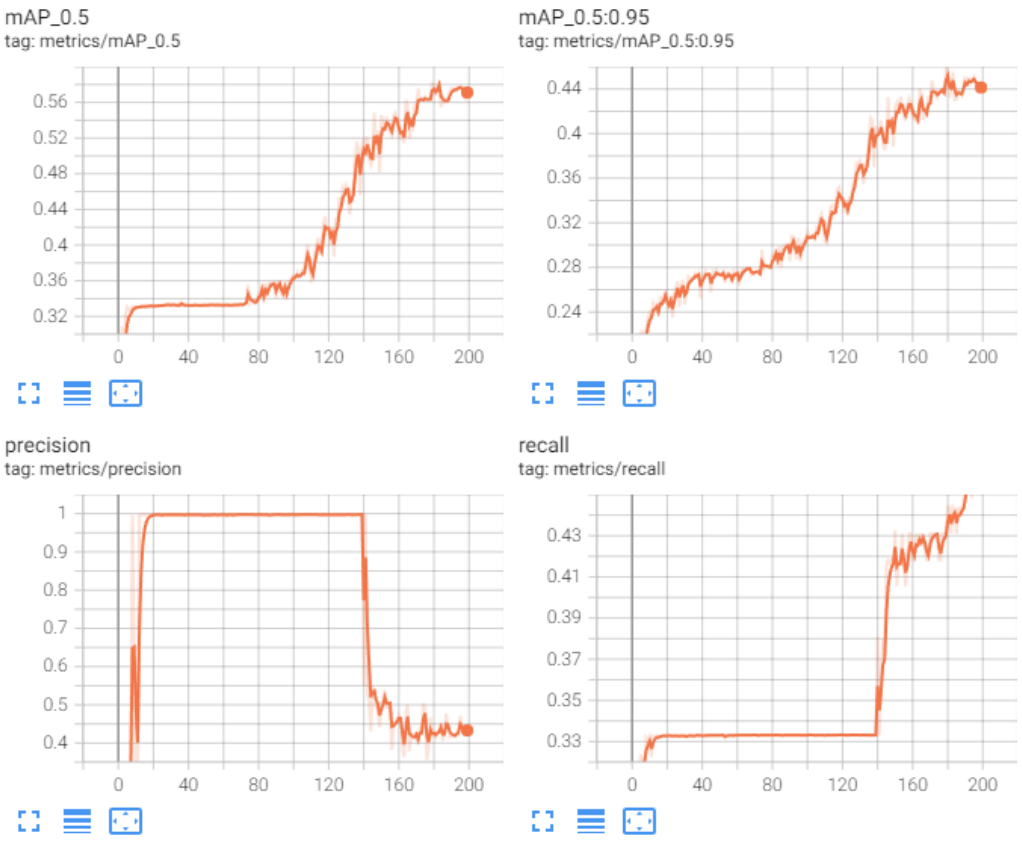
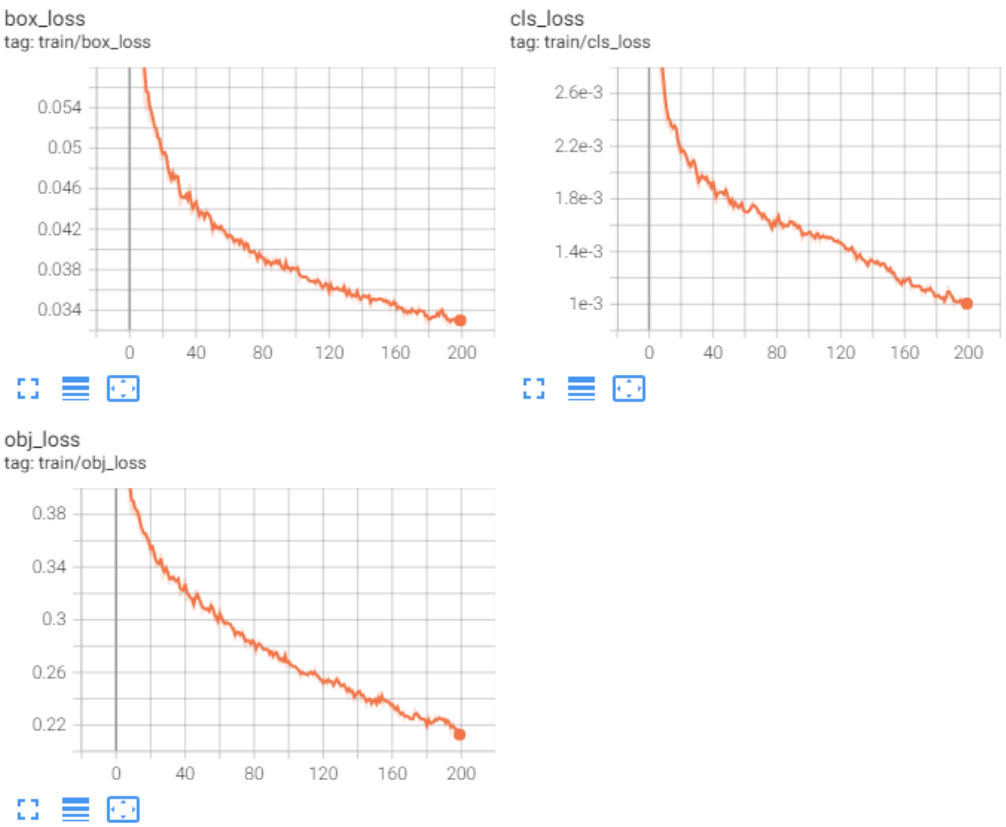
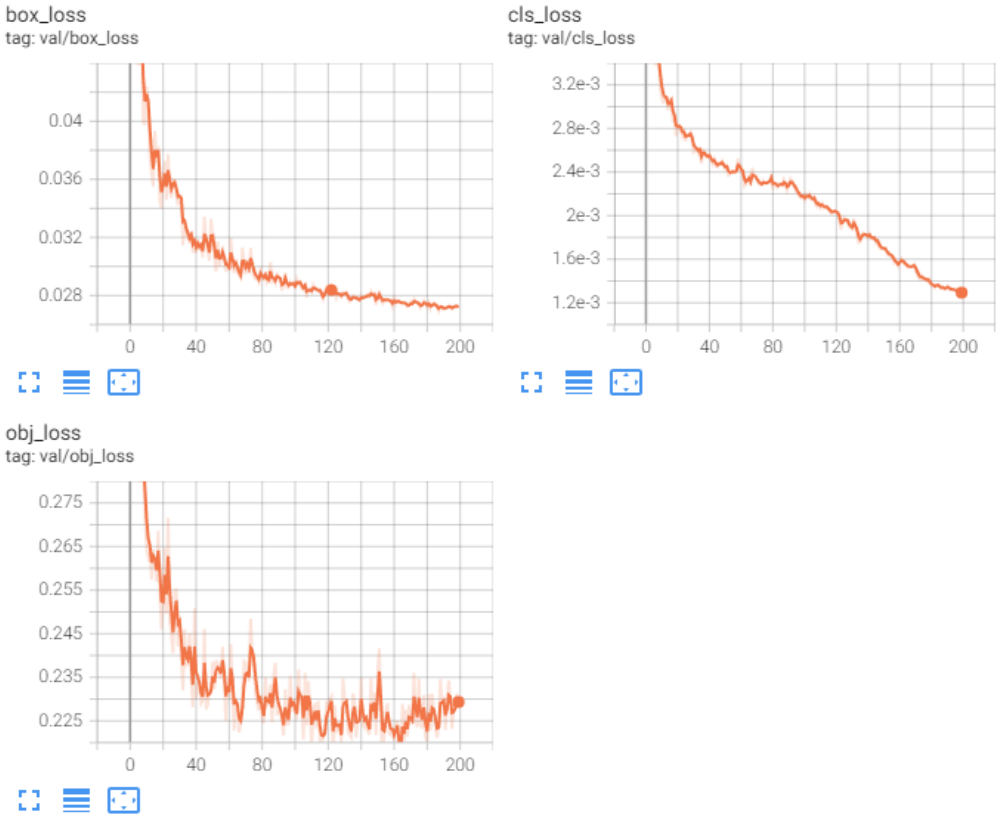


Figure A1.1: YOLOv5 Metrics: mAP-0.5, mAP-0.5:0.95, precision, recall



(a) YOLOv5 Training Loss: box, class, object



(b) YOLOv5 Validation Loss: box, class, object

Figure A1.2: Combined YOLOv5 Training and Validation Loss

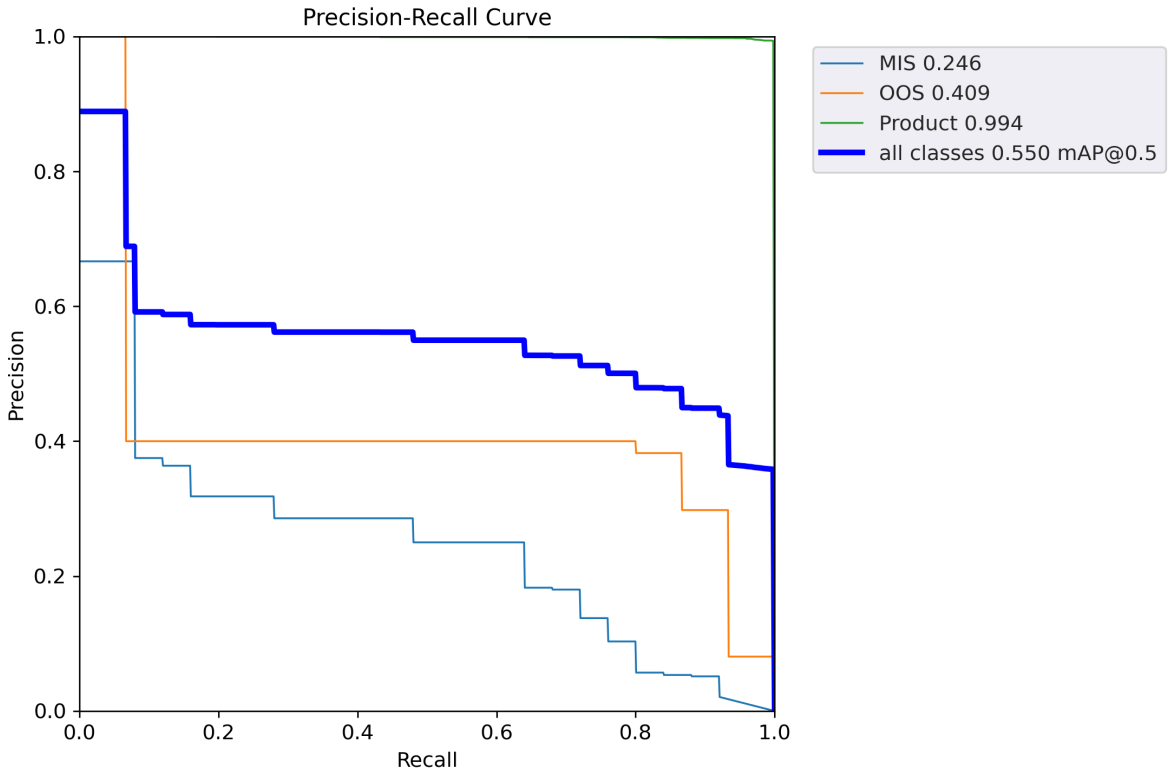


Figure A1.3: YOLOv5 PR-Curve

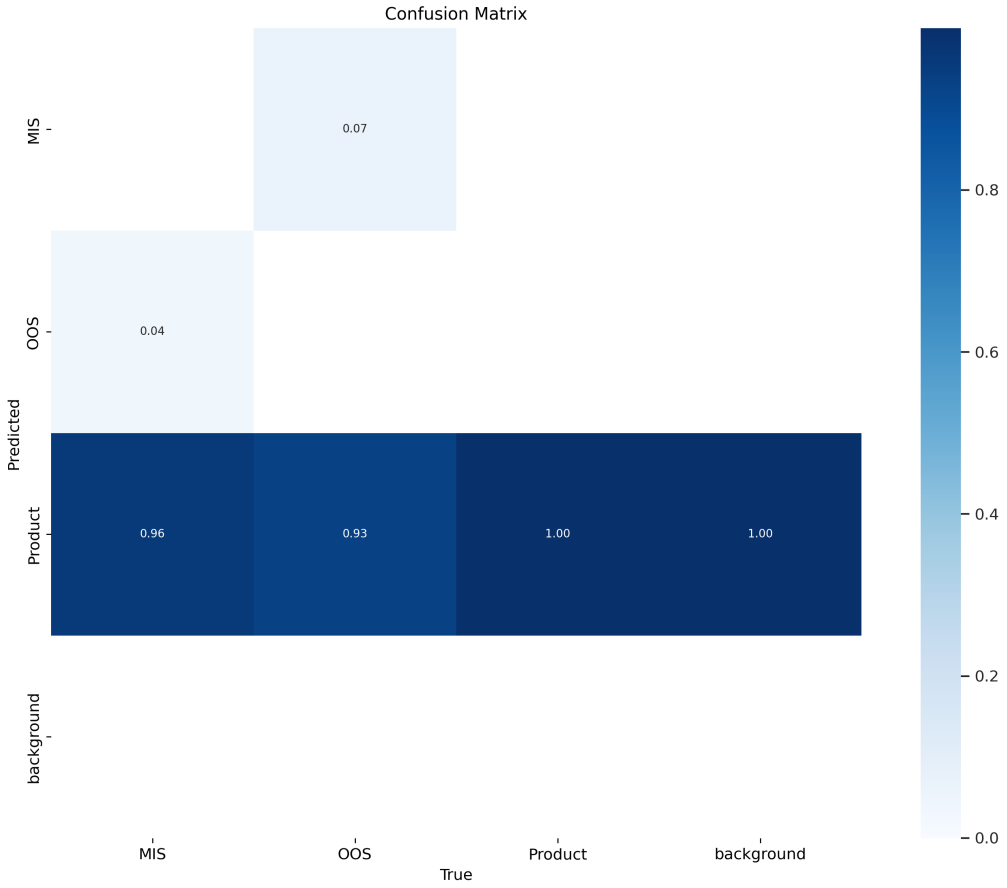


Figure A1.4: YOLOv5 Confusion Matrix

A1.1.2 Training Overview - YOLOv7

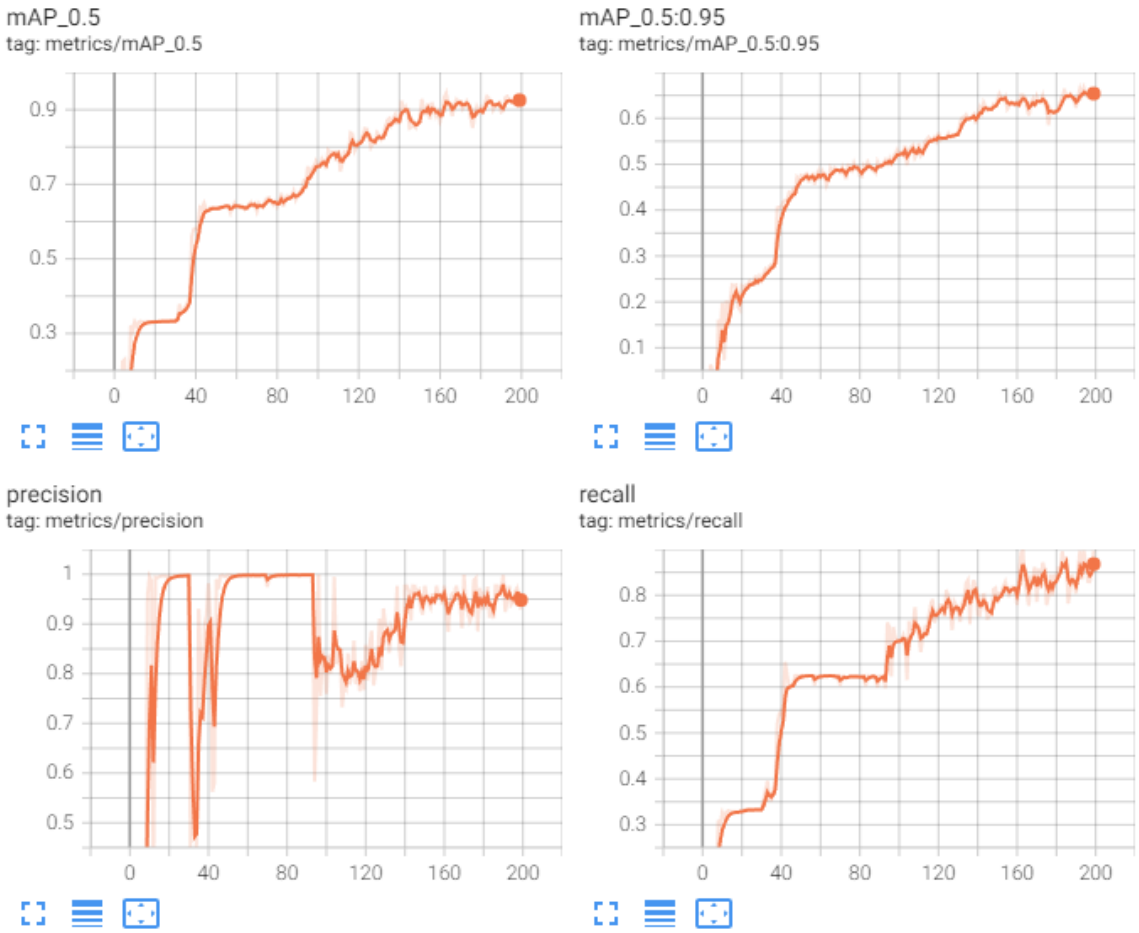
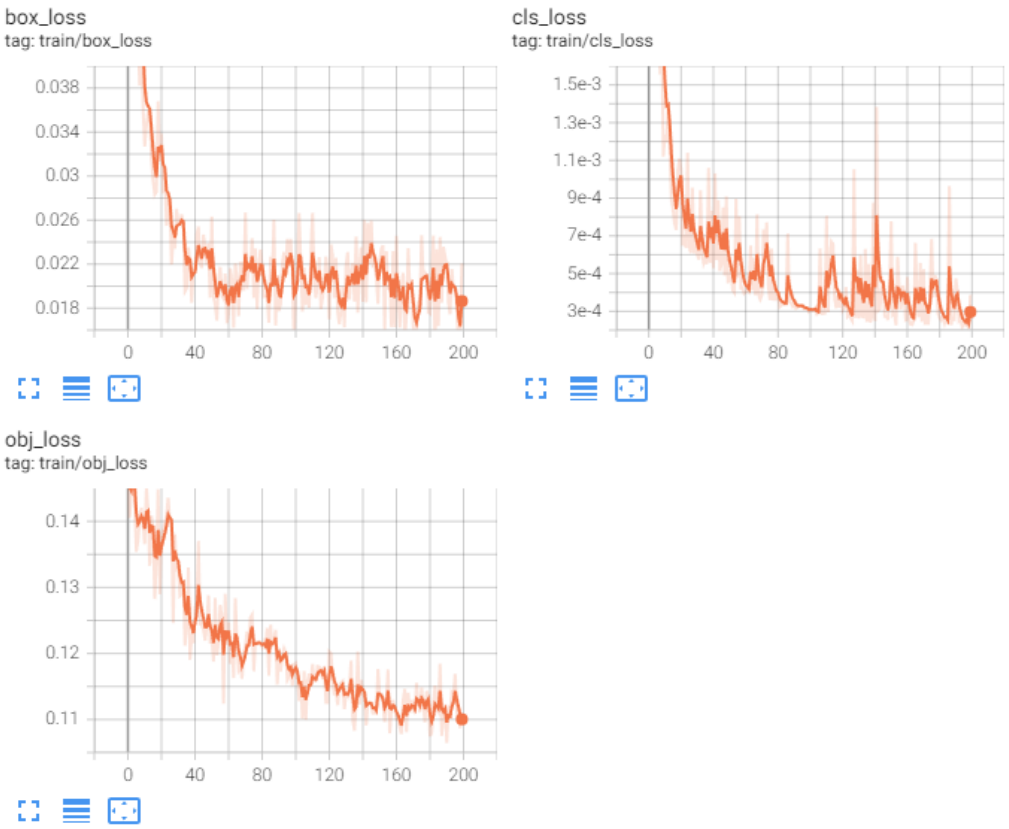
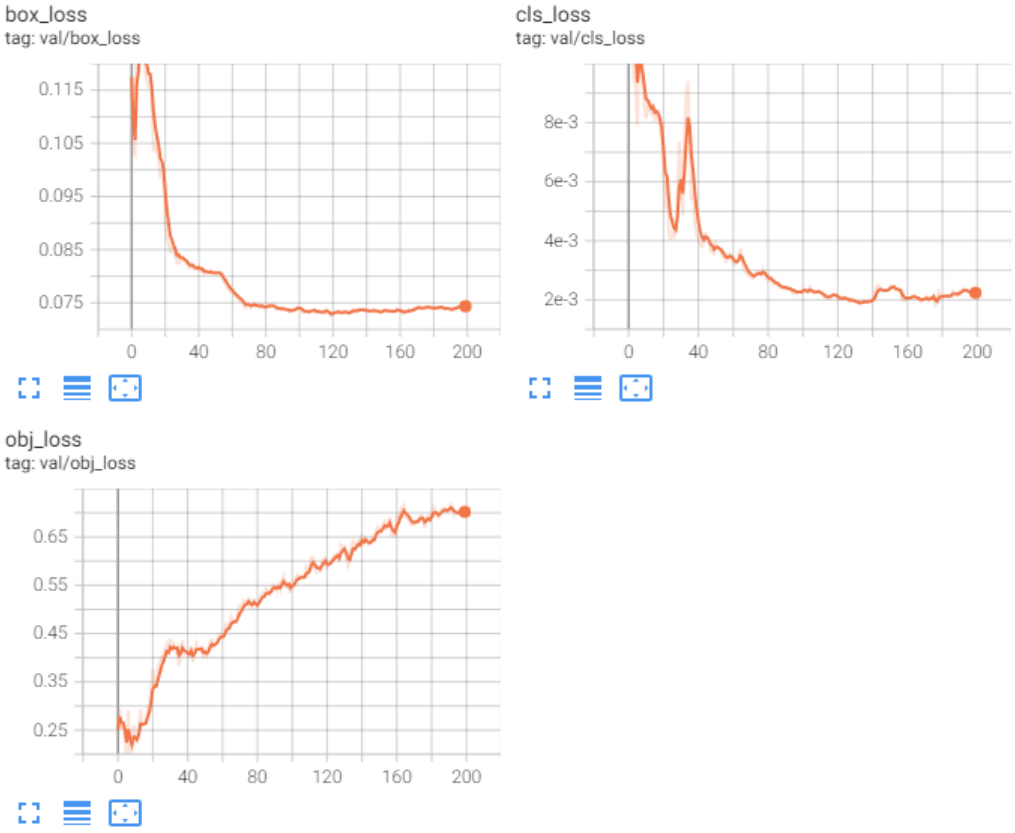


Figure A1.5: YOLOv7 Metrics: mAP-0.5, mAP-0.5:0.95, precision, recall



(a) YOLOv7 Training Loss: box, class, object



(b) YOLOv7 Validation Loss: box, class, object

Figure A1.6: Combined YOLOv7 Training and Validation Loss

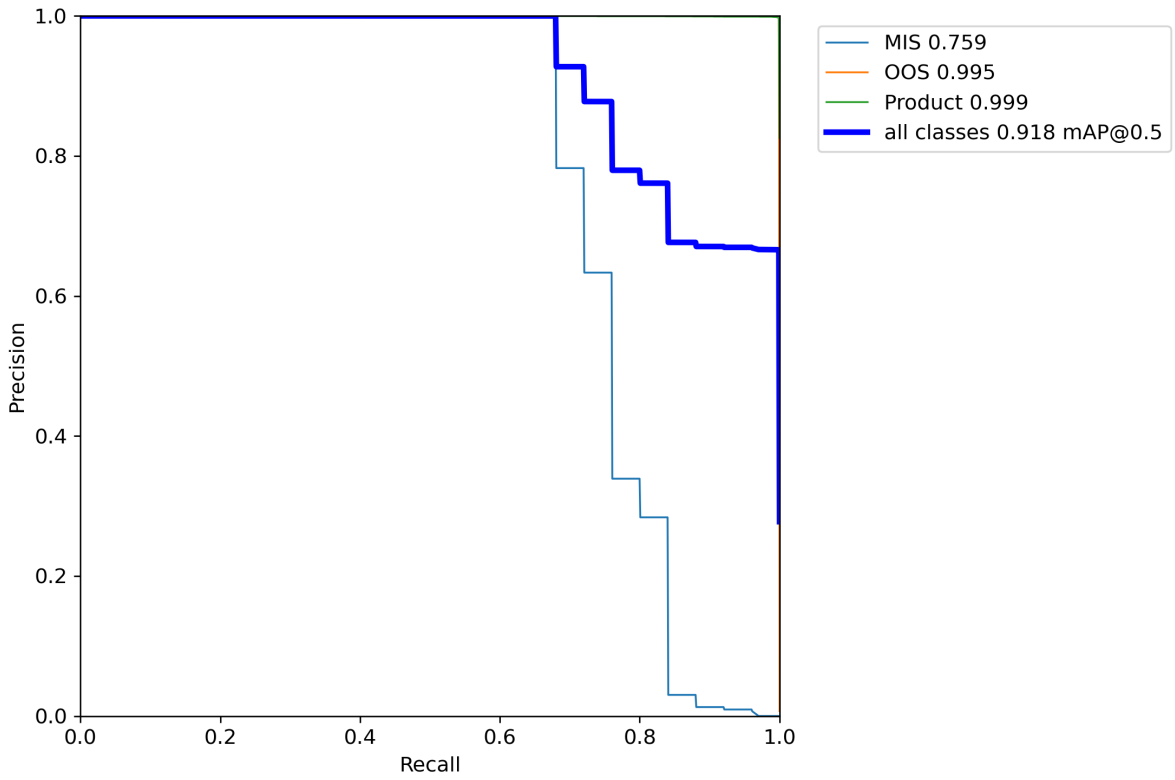


Figure A1.7: YOLOv7 PR-Curve

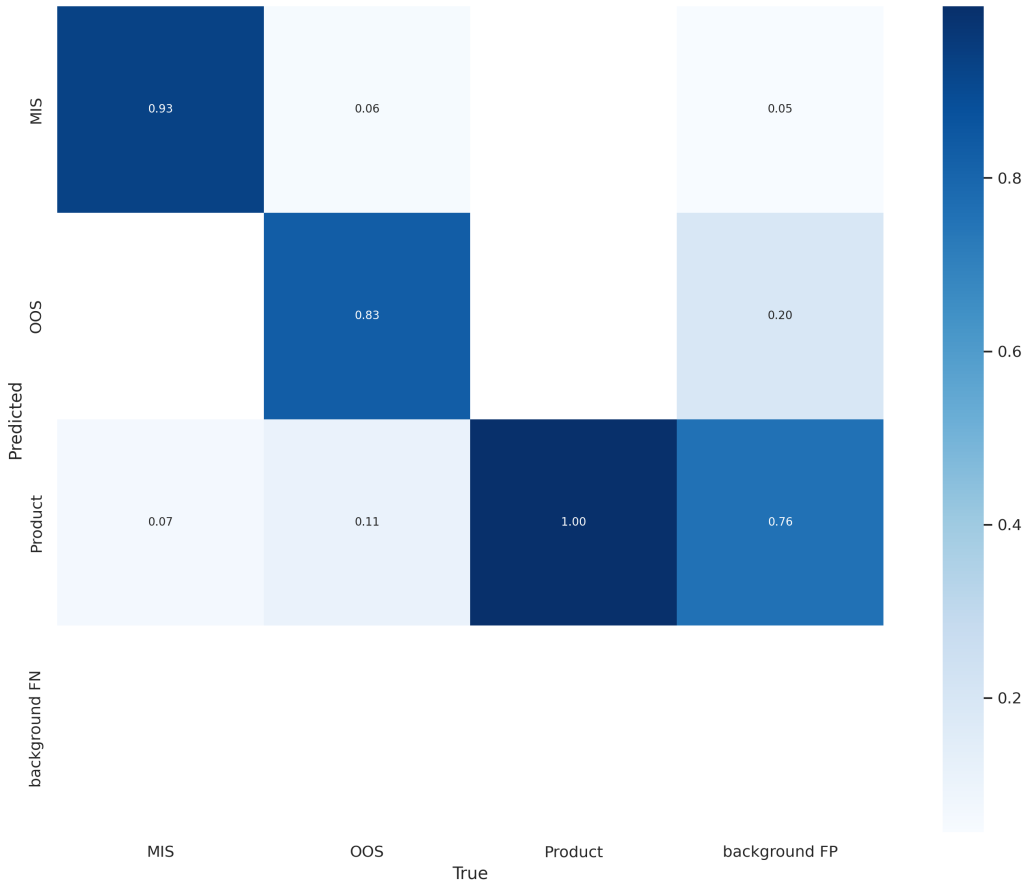


Figure A1.8: YOLOv7 Confusion matrix

A1.2 Appendix II: Training and testing model detailed results

A1.2.1 Training

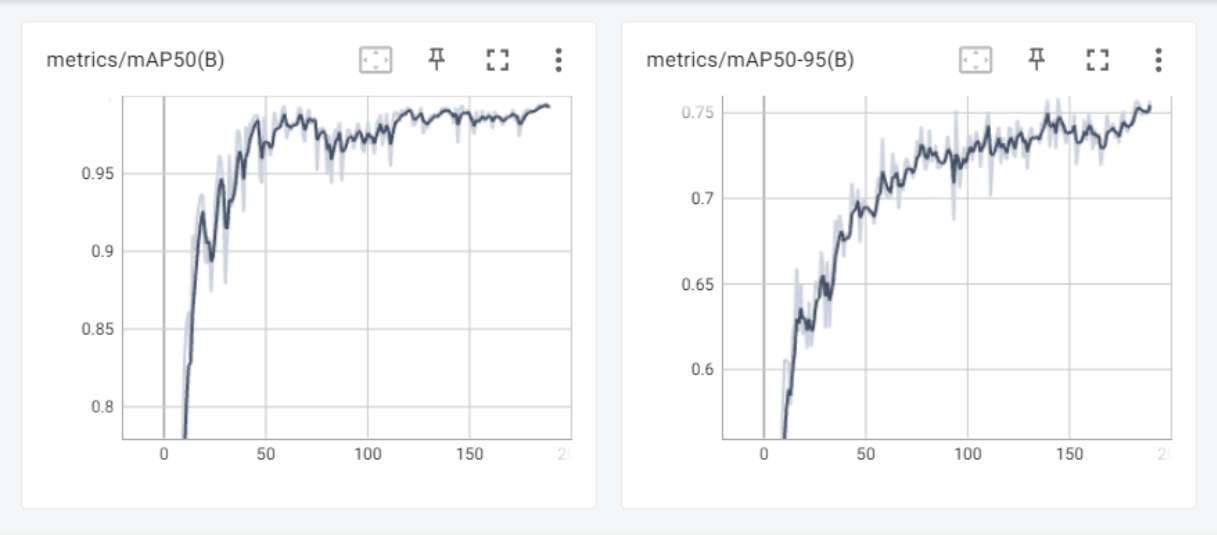


Figure A1.9: YOLOv8s Metrics: mAP-0.5, mAP-0.5:0.95

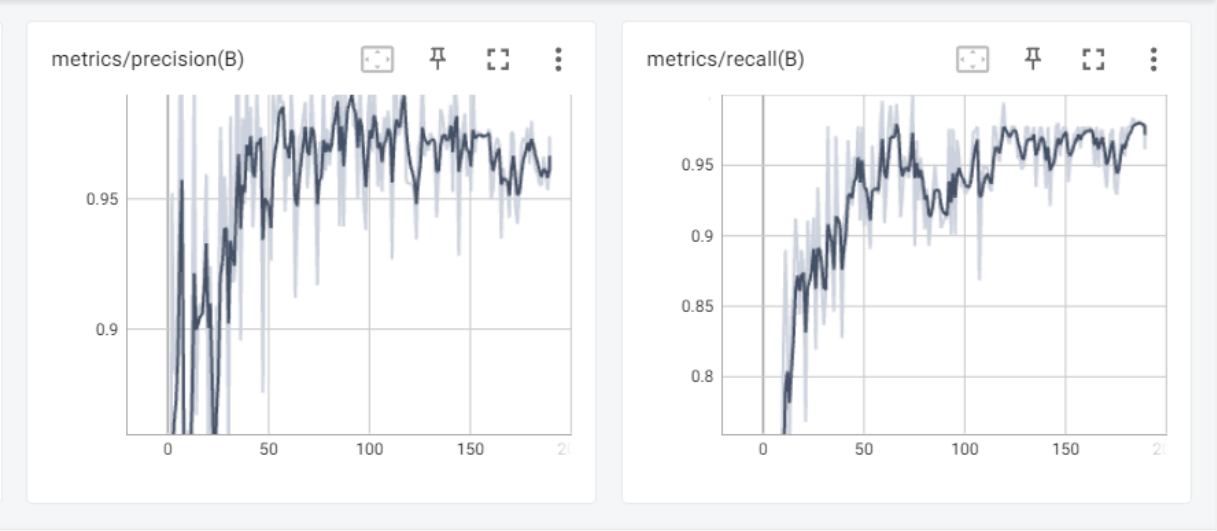
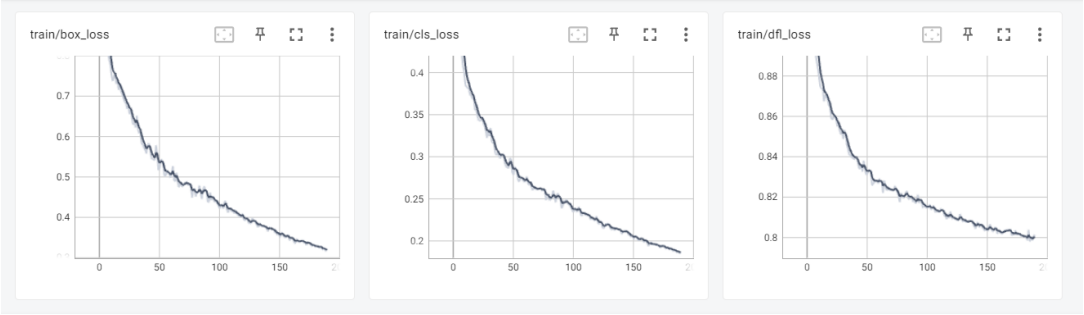
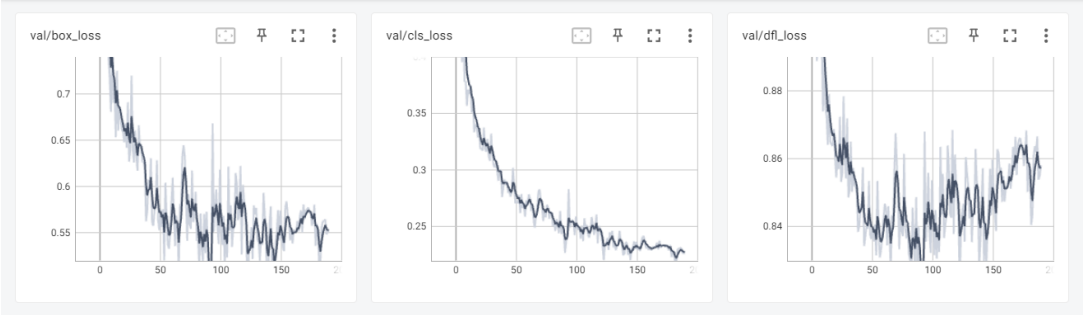


Figure A1.10: YOLOv8s Metrics: precision, recall



(a) YOLOv8s Training Loss: box, class, object



(b) YOLOv8s Validation Loss: box, class, object

Figure A1.11: Combined YOLOv8s Training and Validation Loss

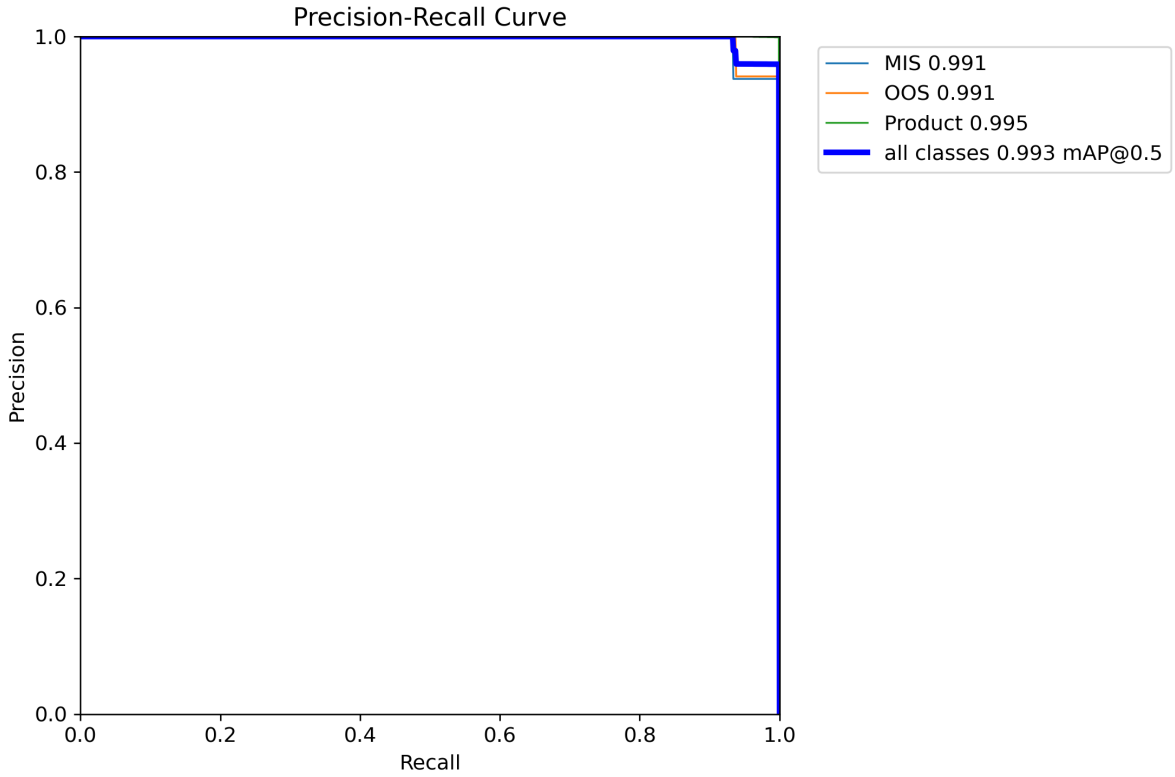


Figure A1.12: YOLOv8 PR-Curve

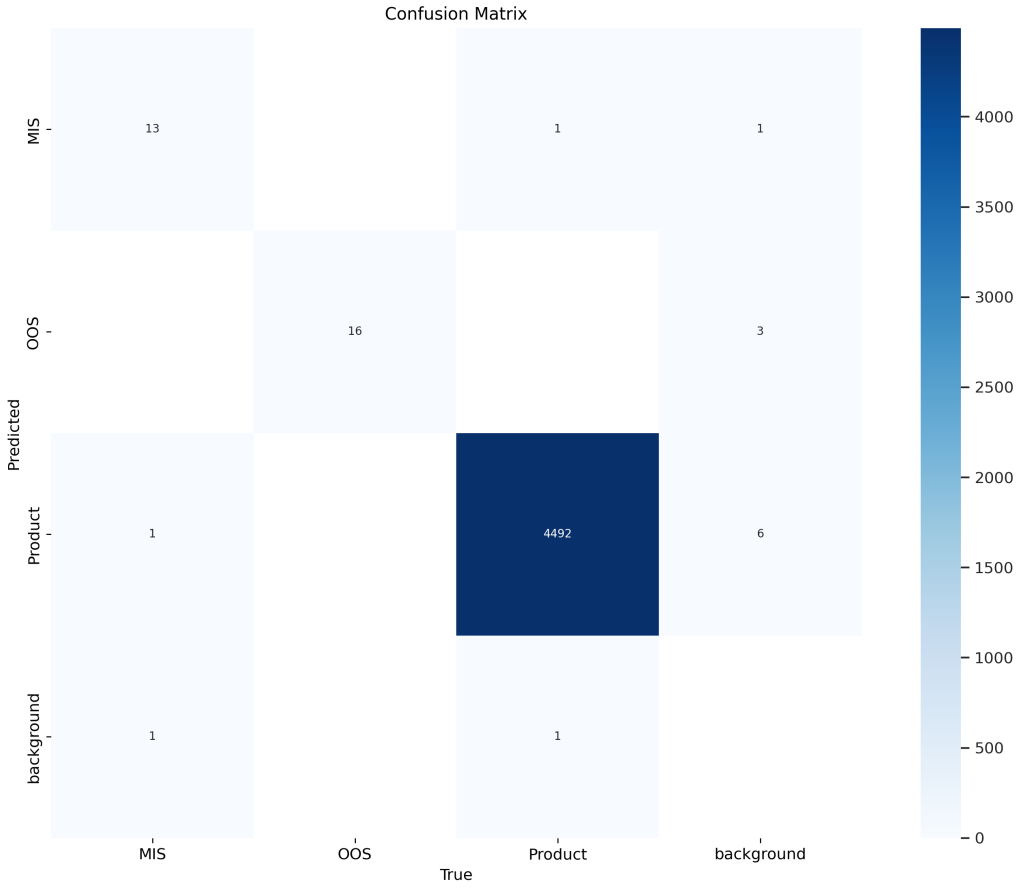


Figure A1.13: YOLOv8 Confusion matrix

A1.2.2 Testing

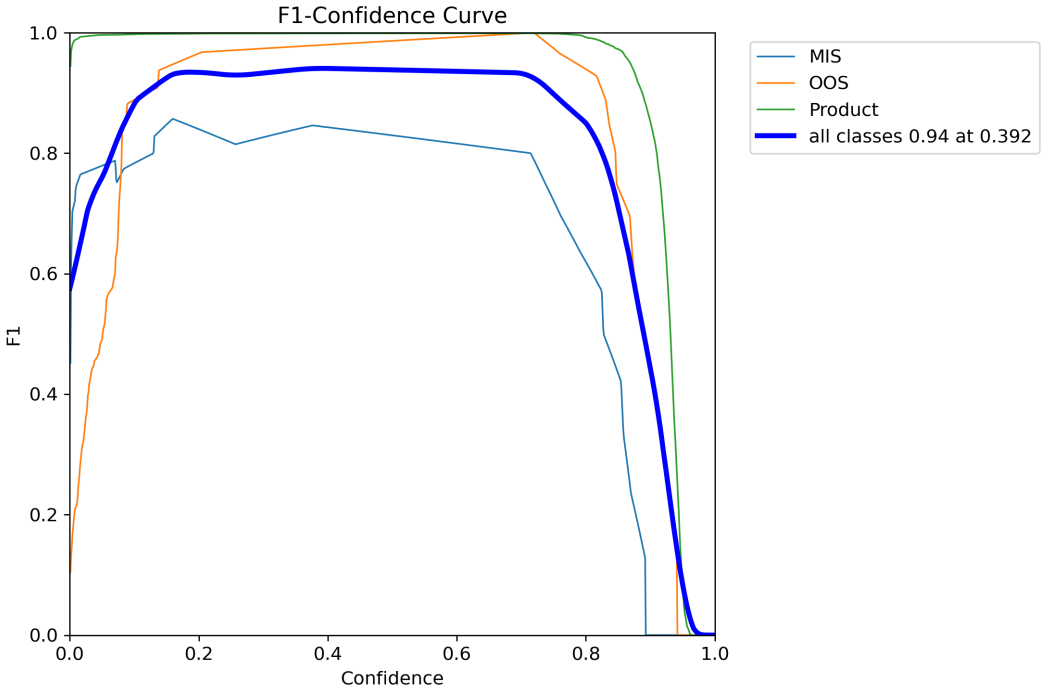


Figure A1.14: YOLOv8 Testing F1-Curve

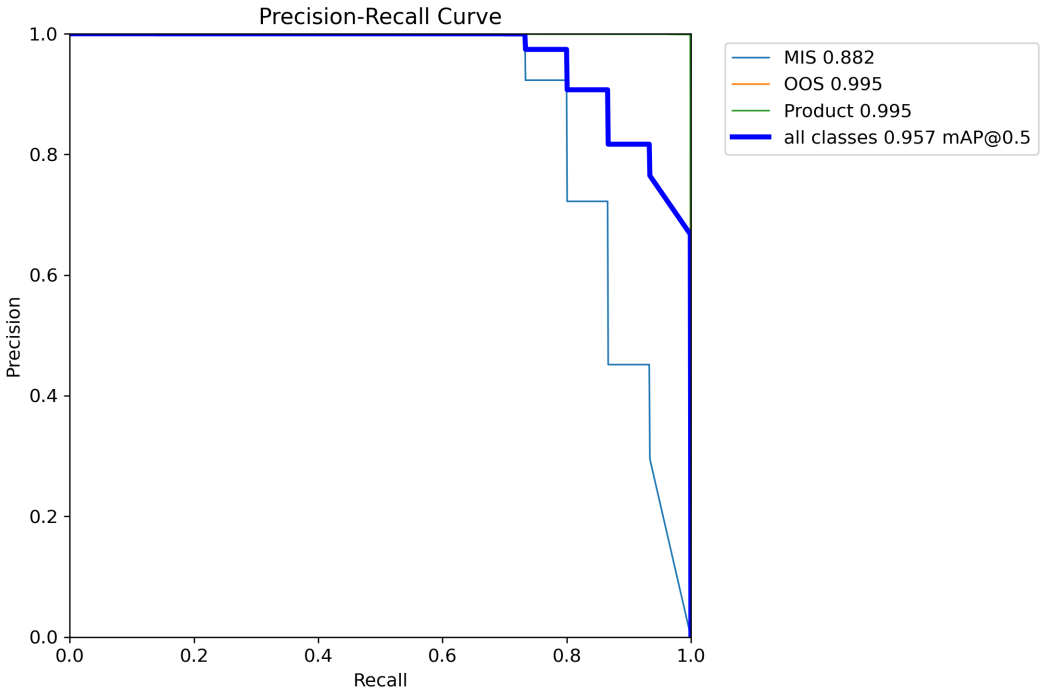


Figure A1.15: YOLOv8 Testing PR-Curve

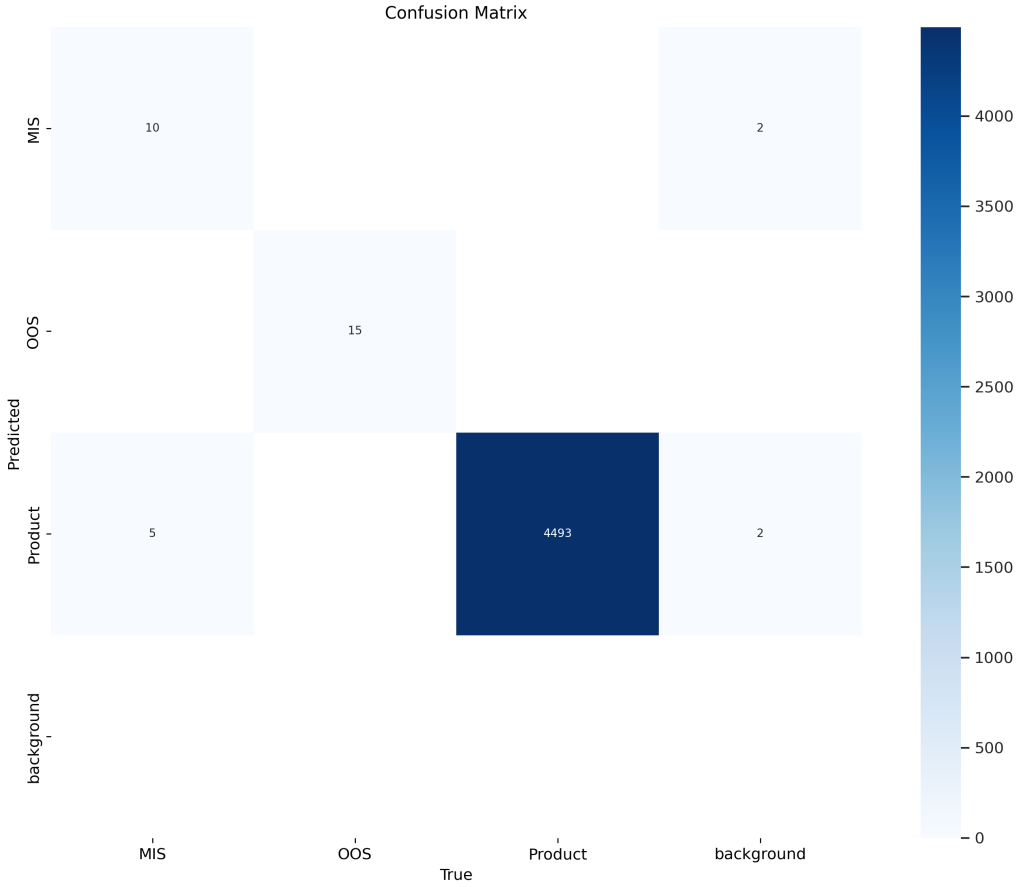


Figure A1.16: YOLOv8 Testing Confusion matrix

(a) Example of predicted misplaced product.

(b) Example of predicted out-of-stock product.

Figure A1.17: Examples of predicted product issues. (a) shows an example of a predicted misplaced product, while (b) illustrates an example of a predicted out-of-stock product.

(a) Example of a predicted misplaced product.

(b) Illustration of a mispredicted misplaced product (second row, blue product).

Figure A1.18: Examples of predicted product issues. Subfigure (a) shows an example of a correctly predicted misplaced product, while Subfigure (b) illustrates a case where the misplaced product was mispredicted.