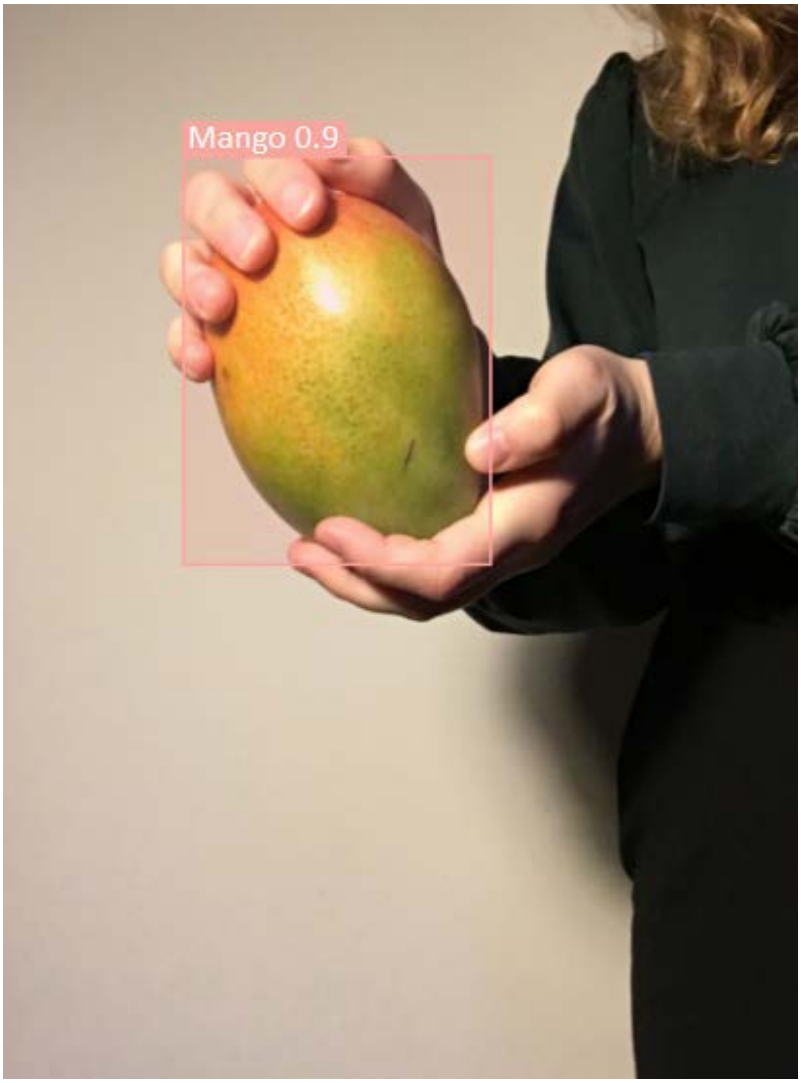
Automatic Detection of Hand-Occluded Fruits for Self-Checkout Registers



Vera Hessels

Layout: typeset by the author using LaTeX.
Cover illustration: taken and edited by the author.

Automatic Detection of Hand-Occluded Fruits for Self-Checkout Registers

Vera Hessels
13173006

Bachelor thesis
Credits: 18 EC

Bachelor *Kunstmatige Intelligentie*

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Supervisor
Dr. S. You and Ms. A. R. Raj

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 900
1098 XH Amsterdam

Semester 1, 2023

Abstract

Key terms— automatic fruit detection, hand-occlusions, self-checkout registers

Automatic fruit recognition has the potential to improve efficiency at self-checkout registers by eliminating the need to select fruits from a menu and instead directly detecting them in a customer's hand. In order to achieve this, the algorithm has to be able to deal with hand-occlusions in real-time. Additionally, the solution should be easily adaptable to new supermarket environments. Although there has been research done on occluded fruits, primarily in orchards, none of the studies have focused on hand-occlusions. For this study, two datasets have been created: *hfruits-base* and *hfruits-sm*, both of which contain pictures of eight different hand-held fruits. *hfruits-base* was collected against a white background, whereas *hfruits-sm* was collected in a supermarket. Four algorithms, namely YOLOV8, RT-DETR, SSD and Faster RCNN, were finetuned to *hfruits-base* and evaluated, which showed that YOLOV8 and RT-DETR were the most accurate. Without retraining, the two best models performed poorly on *hfruits-sm*. By retraining the models, including a small portion of *hfruits-sm*, the results improved significantly, with RT-DETR's best variant achieving a MAP@50 of 0.952 and a MAP@50:95 of 0.77, outperforming models trained on only *hfruits-sm*. Further research could focus on expanding the solution to a new supermarket environment, possibly experimenting with more advanced applications of domain adaptation.

Contents

Chapter 1

Introduction

Self-checkout registers provide supermarkets with an alternative means for customers to complete their grocery purchases. Despite the efficiencies they present, they can sometimes cause additional delays. Moreover, these delays create a greater dissatisfaction with customers than delays experienced when checking out with human personnel (Bulmer et al., 2018).

One key factor causing such inefficiencies is the difficulty scanning fruit at self-checkouts. While packaged products can be scanned using their barcode, fruits must either be preprocessed in-store by having the customer weigh and label them, or they must be selected from a menu at the self-checkout register. This process takes more time than scanning a barcode, therefore decreasing the overall efficiency.

A possible solution to this problem could be provided by automatic fruit recognition, a research area in the field of computer vision. By having the customer hold up the fruit to a camera, this process would be equally efficient as scanning a product's barcode. However, whereas the scanning of a barcode requires the entire barcode to be visible, the same principle cannot be applied to the automatic recognition of fruit, as the fruit will be held up by the client. Having the client put down the fruit and pick it up again would undermine the goal of ensuring the checkout process is as efficient as possible. Therefore, the fruit should be recognized even if it is partly occluded by the customer's hand.

Another concern is the fruit's background. Given that every supermarket has a distinct background, the model would have to relearn every new background, for which gathering data would be time-consuming and costly. Thus, the model should be able to learn new environments relatively easily.

These considerations formed the base for our research. This study's aim was to explore how new supermarket environments can be learned for detecting hand-occluded fruits. To answer this question, it was essential to first determine to what

extent hand-occluded fruits can be detected and what object-detection algorithms perform well at this task. This required an appropriate dataset, hence the creation of *hfruits-base* and *hfruits-sm*. The next question was whether the model would perform well on a new environment, only having been trained on the base dataset. Lastly, we wondered how the model's performance on a new environment could be improved.

Based on the literature reviewed, we suspected that hand-occlusions should not prevent fruits from being recognized, seeing that several studies have been conducted on fruits in orchards, where the fruits are occluded by leaves, branches or other fruits. Although some studies evaluated their model's performance against different backgrounds (Goyal et al., 2023), they did not focus specifically on improving upon this performance, making it an area for experimentation and speculation.

This solution would have multiple social benefits, the most notable being an increased efficiency. As mentioned before, automatic fruit recognition would reduce the time spent in supermarkets. This would have a positive effect on customers, as it would allow them to spend more time on other elements in their lives. Additionally, this solution would require less paper, considering that barcodes would no longer have to be printed at scales. Furthermore, automatic fruit recognition could be implemented at regular check-out services as well, lowering the workload for cashiers.

Not only are there social benefits, there are scientific advantages as well. Although there has been research on detecting occluded fruits, none of these studies have focused on hand-occlusions (Hou et al., 2023), hence this paper introduces a new problem and explores possible solutions. Moreover, this paper's solution approaches the problem from a domain adaptation perspective, which is a field focusing on aligning training and test data from different distributions (Farahani et al., 2021). The simplicity of the solution allows for easy generalisability, using the created base dataset, *hfruits-base*.

Throughout this paper, the terms base-trained, dual-trained and sm-trained will be used. Base-trained refers to a model trained on only *hfruits-base*, dual-trained refers to a model trained on both *hfruits-base* and *hfruits-sm*, and sm-trained refers to a model trained on only *hfruits-sm*.

This study used self-acquired data to investigate the ability of several models to detect hand-occluded fruit. Two datasets have been created: *hfruits-base* and *hfruits-sm*, which both contain eight categories of fruit being held up by someone in various ways. However, *hfruits-base* has been collected against a white background, whereas *hfruits-sm* has been collected at three different self-checkout registers at Jumbo.

The datasets have been labelled in LabelStudio (Tkachenko et al., 2020-2022)

and have been augmented using blurring, brightness adaptation and scaling. Four models were finetuned to *hfruits-base*, after which the two best performing models, evaluated using mean average precision, were chosen for further analysis. The base-trained models, YOLOV8 (Jocher et al., 2023) and RT-DETR (Lv et al., 2023), were tested on *hfruits-sm*, which showed that the results decreased drastically. The models then underwent extensive experimentation, including retraining the model on *hfruits-base* including a small portion of *hfruits-sm*. The results show that the dual-trained RT-DETR with augmented data performs well on *hfruits-sm* with a MAP@50 of 0.952 and a MAP@50:95 of 0.77. It outperforms the sm-trained models and achieves the same or higher accuracies on *hfruits-base* compared to the base-trained models.

This paper has been organised in the following way: chapter 2 provides an overview of previous work, chapter 3 explains how our datasets have been acquired and preprocessed, chapter 4 outlines the method employed, and chapter 5 presents and analyses the results. Lastly, chapter 6 discusses the study and chapter 7 summarizes the paper and makes suggestions for further research.

Chapter 2

Related Work

This section aims to provide an overview of common object detection algorithms and tactics for fruit detection, and of prior research in the field of fruit detection to establish a general understanding of previous findings.

The task of automatically detecting fruit has been investigated by several studies, ranging over several years. Before CNNs, feature extraction techniques, such as histograms of oriented gradients, local binary patterns, and speeded up robust features, as well as machine learning algorithms, such as support vector machines and artificial neural networks, were commonly used (F. Liu et al., 2017/07). At the end of 2019, the literature on automatic fruit detection started following the rising trend of CNN application (Naranjo-Torres et al., 2020).

Naranjo-Torres et al. (2020) identify two methods within the application of CNNs that are used for fruit image processing: designing a new model or finetuning a pretrained model. The latter is the primary method for fruit detection, with 96 percent of the analyzed articles using a pretrained network, the most common ones being VGG and ResNet.

This group consists of two subgroups: transfer learning and adapting the existing network to a specific objective. Adaptation is often carried out on Faster RCNN, which is made up of a feature extractor model and a region proposal model, in which the feature extractor model is changed (Ukwuoma et al., 2022).

Moreover, object detection algorithms can be divided into two categories: one-stage and two-stage algorithms (Yang et al., 2023). Whereas two-stage algorithms generate candidate boxes and classify objects within, resulting in high robustness, one-stage algorithms avoid generating candidate boxes, making them suitable for real-time detection.

Within the fruit detection field, there is a variety of studies that focus on the detection on occluded fruits. Hou et al. (2023) describe four methods for detecting occluded fruits: direct detection from images, differentiation between obstacles

and unoccluded fruits, restoration of occluded fruits, and detection through calculation and multiple sensors. However, the last two methods are not relevant to this research due to them taking up too much time or too many resources.

Deviating from the common approach of using a pretrained model, Saedi & Khosravi (2020) designed their own DCNN, which can recognize six different fruits in orchards. Their model's processing time compared to other existing models, such as ResNet152, is significantly lower with ResNet152 taking 351 ms per image and the model only taking 8 ms. However, this model is only used for classification, lacking the capability of proposing bounding boxes.

Sa et al. (2016) employed transfer learning using the VGG16 network, which was pretrained on the ImageNet data set, for Faster RCNN. Despite the network's ability to be finetuned to different fruits, as demonstrated in the article for six different fruits, the model operates for just one fruit at the time. Mirhaji et al. (2021) also used transfer learning for different versions of YOLO, including YOLOV2, YOLOV3 and YOLOV4, to detect oranges in orchards. The best performing model was YOLOV4, which is an older model at present.

Masked RCNN was trained by, Yu et al. (2019) to detect strawberries in an strawberry garden environment containing several occlusions. Despite its robustness, the model's processing speed is deficient, making it unsuitable for real-time detection. Additionally, Jia et al. (2020) trained a Masked R-CNN, but their focus was on detecting apples in orchards. Although the model is able to detect occluded apples effectively, the segmentation of occluded fruits, similarly noted in the work of Yu et al., may not be necessary for this research.

Adaptation was carried out by Bargoti & Underwood (2017), who experimented with using ZF and VGG16 as a feature extractor model for Faster RCNN, as done before in the original paper (Ren et al., 2017), for detecting apples, mangoes and almonds in orchards. Furthermore, Wang et al. (2021) adapted Faster RCNN, using a convolutional block attention module, a feature pyramid network, and a soft non-maximum suppression. While the authors state this network provides a solution to detecting fruits with a near color background, the network was only tested on young tomatoes. Mai et al. (2020) also adapted Faster RCNN by fusing three different classifiers in the region proposal network to improve the model's performance on small fruits. Although the model is faster than the original Faster RCNN, it is still significantly slower than other models, such as SSD.

YOLOV4 was adapted by Zheng et al. (2022) by integrating a residual network with a CSPDarknet53 backbone to increase to model's performance on small fruits. However, similar to various others models, the model is limited to detecting one fruit. An algorithm, based on the YOLOV5 network, was designed by Goyal et al. (2023), which can both detect fruits, as well as evaluate their quality. Nonetheless, as quality evaluation is not pertinent to this research, a big part of

the used data set proves to be irrelevant. Yang et al. (2023) adapted the existing YOLOV8 network by replacing the convolutions with depthwise separable convolutions, by designing a dual-path attention gate module, and by adding a feature enhancement module. Although the model achieved a mAP of 93.4 percent, it was exclusively trained and tested on tomatoes, therefore raising concerns about its generalisability to other fruits.

This section has outlined the previous findings in the field of fruit detection. The findings have been summarized in Table 2.1. F1 represents the harmonic mean of precision and recall. The other evaluation metrics have been elaborated upon in section 4.2. Despite the variety of studies, no study, to the best of our knowledge, has yet focused on detecting hand-occluded fruits. Therefore, this paper explores possible solutions to this problem, and assesses the models' performance in a real-life setting, specifically the self-checkout register environment in a supermarket.

Authors	**Model Used**	**Environment**	**Fruits**	**Accuracy**
Saedi & Khosravi (2020)	Deep CNN	Orchard	Sour cherries, plums, peaches, nectarines, apples, apricots	Accuracy=0.9976 (classification only)
Sa et al. (2016)	Faster RCNN	Farm	Rockmelons, strawberries, apples, avocados, mangoes, oranges, sweetpeppers (one at a time)	F1 = 0.83
Mirhaji et al. (2021)	YOLOV4	Orchard	Oranges	MAP = 0.9088
Yu et al. (2019)	Mask RCNN	Garden	Strawberries	Precision = 0.9578, recall = 0.9541
Jia et al. (2020)	Mask RCNN	Orchard	Apple	Precision = 0.9731, recall = 0.9570
Bargoti & Underwood (2017)	Faster RCNN	Orchard	Mangoes, almonds, apples	F1 = 0.8623
Wang et al. (2021)	Faster RCNN	Greenhouse	Young tomatoes	MAP = 0.9846
Mai et al. (2020)	Faster RCNN	Orchard	Oranges	MAP = 0.77985
Zheng et al. (2022)	YOLOV4	Garden	Tomatoes	MAP = 0.9444
Goyal et al. (2023)	YOLOV5-based model	Simple white background, fruit market	Apples, bananas oranges, tomatoes	MAP = 0.9280
Yang et al. (2023)	YOLOV8	Garden	Tomatoes	MAP = 0.934

Table 2.1: Previous Studies in Fruit Detection

Chapter 3

hfruits-Dataset

For this study we have curated our own datasets, as there were no existing datasets that met the requirement for tackling the problem of detecting hand-occluded fruits. The requirement entailed having the fruit being held up by a person. Two datasets have been created: a dataset with a simple background, *hfruits-base*, and a dataset with a complex background, *hfruits-sm*.

3.1 *hfruits-base*

3.1.1 Data Acquisition

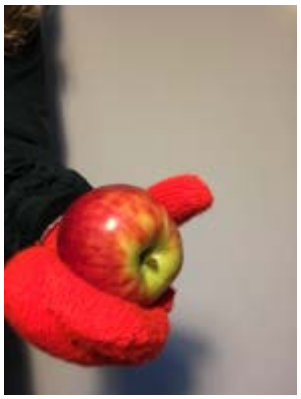
The dataset *hfruits-base*, in which the *h* represents the word *hands*, consists of pictures of hand-held fruit that were collected using an IPhone 6s. The benefit of using an older phone model is that the photographs are not of the highest standard, meaning that the integration of the trained model will be feasible for supermarkets because of the moderate cost. The pictures were taken against a white background with a light source facing the fruit. The simple background allowed for the focus to be on detecting hand-occluded fruits without the background distracting the model. The light source emulated the supermarket light conditions, seeing that these stores often feature bright lighting. Because the focus was not on detecting fruit under different light conditions, the light source remained fixed.

The dataset contains a total of 2176 images divided over eight categories, including apple, banana, kiwi, lemon, mango, orange, pear and persimmon. All of the categories contain at least 248 images and at most 312 images, as shown in Table 3.1. All images have a height of 2576 pixels and a width of 1932 pixels.

Label	**Size**
Apple	278
Banana	253
Kiwi	263
Lemon	248
Mango	282
Orange	261
Pear	312
Persimmon	270
Total	2167

Table 3.1: *hfruits-base* Category Sizes

The eight categories are all varied, thereby ensuring variability within the dataset, which enables the chosen model to learn from a spectrum of scenarios. One method introducing this variability is the way, in which the fruits are held. For instance, a fruit can be held at the bottom or at the top. Additionally, the fruits can be found in the individual's left hand, right hand and both hands at the same time. Not only are both hands used, gloves are also incorporated into the pictures to further augment the data set's diversity. This ranges from images without gloves to images with dark blue gloves or with bright orange mittens, as can be seen in Figure 3.1.



a) Apple in left hand in bright orange mitten

b) Banana in right hand in dark blue glove

c) Kiwi in both hands

d) Lemon in right hand

e) Mango in left hand in bright orange mitten

f) Orange in left hand in dark blue glove

g) Pear in right hand in dark blue glove

h) Persimmon in both hands

Figure 3.1: Part of the Images in *hfruits-base*

3.1.2 Data Preprocessing

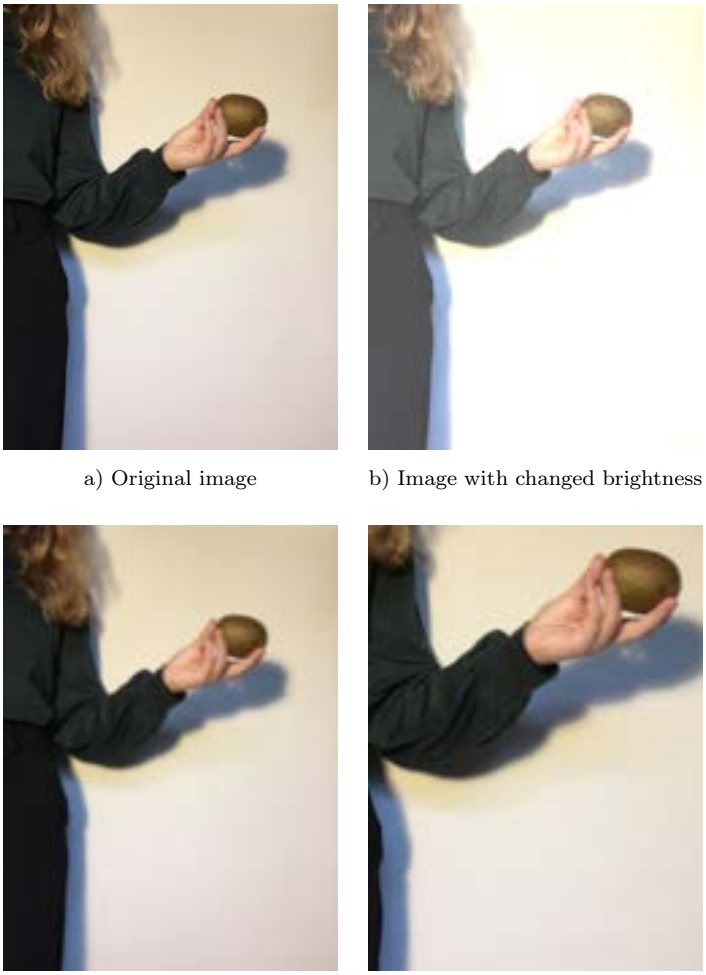
The data preprocessing comprised data labelling and data augmentation. The images were labelled using Label Studio, which is an open source data labelling platform (Tkachenko et al., 2020-2022). The fruits were manually labelled by drawing bounding boxes, after which the labels were exported in the required format. For the models YOLOV8 and RT-DETR, each image is associated with a similarly-named label stored in a text file, containing the object's class, as an integer, the object's normalized x- and y-coordinates, and the object's normalized width and height. For Faster RCNN and SSD, the dataset was put in a PASCAL-VOC format, in which the labels are stored in XML-files. A display of the two different methods of storing the datasets can be found in section 10.1.

The dataset was shuffled and divided into a training, validation and test set with a ratio of 50, 20 and 30 percent respectively. This partitioning was done per category to guarantee an equal distribution of the fruits over the sets. Afterwards, the training and validation sets underwent a process of data augmentation, resulting in an expansion of their size by a factor of approximately four. Applying data augmentation to the training set ensures that the model will be more robust, as there will be more varied samples. Anticipating the presence of diverse samples in real-world situations, we augmented the validation set as well, ensuring that the augmentations accurately emulate real-world conditions.

Figure 3.2 shows three augmentations, designed to emulate real-world situations, that were applied to the images: blurring the image with a varying blur, adapting the image's brightness with a differing factor and scaling the image. The augmentations were done using the library OpenCV in Python, see Table 3.2. The labels remained unchanged for the blurring and brightness changes, while the labels for scaling had to be adjusted accordingly. Some scaled images were discarded, as the fruit bounding box was no longer within bounds. The final training, validation and test sets have a size of 4291, 1725 and 650 images respectively.

Augmentation	**Function (from OpenCV)**	**Factor**
Blurring	GaussianBlur(img, (kernelsize, kernelsize), blur)	kernelsize: 101, blur: between 3 and 10
Brightness Adaptation	convertScaleAbs(img, α, β)	α: 1, β: between 50 and 100
Scaling	resize(img, (width, height))	width and height multiplied by 1.5

Table 3.2: Image Augmentation Parameters



a) Original image

b) Image with changed brightness

c) Blurred image

d) Scaled image

Figure 3.2: Data Augmentation Techniques

3.2 *hfruits-sm*

3.2.1 Data Acquisition

The dataset *hfruits-sm*, in which *h* represents the word *hands* and *sm* represents the word *supermarktet*, was collected in a Jumbo supermarket, with permission of the staff. The dataset contains the same eight categories of fruits being held up by a person as *hfruits-base*, namely apple, banana, kiwi, lemon, mango, orange, pear and persimmon. The pictures were taken with the same IPhone 6s to allow for a cost-friendly supermarket integration. Moreover, they were taken at three different self-checkout registers to ensure diversity within the dataset with one angle facing the entrance of the supermarket and the other two facing the supermarket itself at two different distances. The size of the dataset is 772 images with each category containing at least 61 images, which can be seen in Table 3.3. Similarly to *hfruits-base*, the images have a height of 2576 pixels and a width of 1932 pixels. Part of the dataset is shown in Figure 3.3.

Label	**Size**
Apple	118
Banana	61
Kiwi	109
Lemon	113
Mango	81
Orange	89
Pear	118
Persimmon	83
Total	772

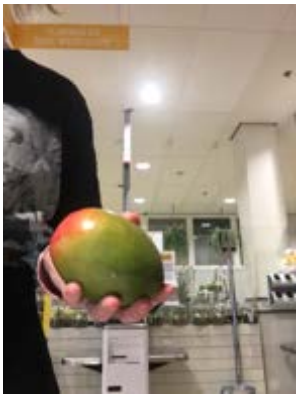
Table 3.3: *hfruits-sm* Category Sizes

a) Apple in left hand at location 3

b) Banana in left hand at location 1

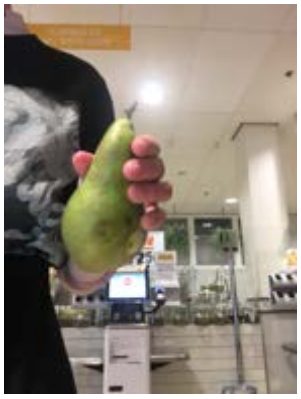
c) Kiwi in right hand at location 3

d) Lemon in right hand at location 2



e) Mango in left hand at location 1

f) Orange in right hand at location 2



g) Pear in right hand at location 1

h) Persimmon in right hand at location 2

Figure 3.3: Part of the Images in *hfruits-sm*

3.2.2 Data Preprocessing

hfruits-sm was labelled using Label Studio (Tkachenko et al., 2020-2022) in a similar fashion to *hfruits-base*, as outlined in subsection 3.1.2. Seeing that the only models used on *hfruits-sm* were YOLOV8 and RT-DETR, the dataset was only structured according to the YOLO-format, which can be seen in section 10.1.

Depending on the task, several parts of the data set were used, for which different preprocessing techniques were required. As outlined in section 4.3, *hfruits-sm* was both used for testing and training purposes. For testing, the dataset did not require any more preprocessing techniques. On the other hand, training required the dataset to be separated into a training and testing set, with a proportion of 20:80 respectively. This proportion was specifically chosen to allow for thorough testing while simultaneously evaluating the model's performance with a small portion of specialized data. The training set was both used in its original format, as well as in an augmented format. This augmented format contained the same augmentations to the pictures that were used for *hfruits-base*, with the only difference being that these augmentations were also applied concurrently, adding more difficulty to the pictures. The parameters, shown in Table 3.2, stayed the same.

Chapter 4

Method

4.1 Model Testing

Four different models were chosen to finetune to *hfruits-base*: Faster RCNN (Ren et al., 2017), SSD (W. Liu et al., 2016), YOLOV8 (Jocher et al., 2023) and RT-DETR (Lv et al., 2023). Additional details regarding the models can be found in section 10.2, in which their functionality is explained. Transfer learning was applied to all of these models to achieve the best results. PyTorch was used to implement Faster RCNN and SSD (Paszke et al., 2019), while Ultralytics was used to implement YOLOV8 and RT-DETR (Jocher et al., 2023)(Lv et al., 2023). All models underwent training for 50 epochs to ensure a fair learning opportunity. The remainder of the parameters are outlined in Table 4.1.

Model Name	**Model**	**Number of Parameters**	**Weights**	**Optimizer**	**Learning Rate**
Faster RCNN	fasterrcnn_ resnet50_fpn	43,712,278	FasterRCNN_ ResNet50_FPN_V2 (Paszke et al., 2019)	Adam	0.0001
SSD	ssd300_vgg16	35,641,826	SSD300_VGG16_ Weights (Paszke et al., 2019)	Adam	0.0001
YOLOV8	YOLOv8x	68,131,272	yolov8x.pt (Jocher et al., 2023)	-	-
RT-DETR	RT-DETR-HGNetv2-L	32,000,180	rtdetr-l.pt (Lv et al., 2023)	-	-

Table 4.1: Model Parameters

4.2 Evaluation

The results were evaluated using mean average precision (MAP), which is a popular metric used for object detection (Padilla et al., 2020). MAP (Equation 4.1), is a metric for object detection that calculates the mean of the average precision across all classes.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} APi. \tag{4.1}$$

Average precision is calculated by determining the area under the precision-recall curve, which is obtained by plotting the precision and recall values against varying confidence score thresholds of the model. To calculate the area under this curve, it is necessary to obtain the precision and recall values. Precision represents the proportion of correctly predicted positive samples, whereas recall signifies the proportion of positive samples that are qualified as positive. These values are calculated using the number of true positives (TP), false positives (FP) and false negatives (FN).

$$Precision = \frac{TP}{TP + FP}, \tag{4.2}$$

$$Recall = \frac{TP}{TP + FN}. \tag{4.3}$$

The category, to which a prediction belongs, is determined by the intersection over union, or IOU (Equation 4.4). The IOU returns the proportion of overlap between the ground-truth bounding box, which is the box that has been manually added to the image, and the predicted bounding box. If the IOU is equal to or larger than the set threshold, the sample is said to belong to the class in question.

$$IOU = \frac{area\,of\,overlap}{area\,of\,union}. \tag{4.4}$$

The average precision can be calculated using a range of IOUs, for which common values are ten IOUs in a range of 50% to 95% with steps of five percent, or a single IOU value, which is usually 50% or 75% (Padilla et al., 2020). These variations can then be referred to as AP@50:5:95, AP50 and AP75 respectively. When averaging over these variations to compute the mean average precision, the same format can be used.

As MAP@50 only requires at least 50 percent of the predicted bounding box to correspond with the true bounding box, this score should be relatively high to be considered good. MAP@50:95 has stricter requirements, which means a slightly lower score will be acceptable. Considering that we want our model to increase the efficiency at self-checkout registers, it is imperative that the MAP@50 and MAP@50:95 are at least 0.90 and 0.75 respectively.

4.3 Generalisability and Improvements

As *hfruits-base* is a relative simple dataset, the base-trained models were evaluated on *hfruits-sm* as well to determine the models' ability to generalise to different backgrounds.

Moreover, to further enhance the performance on *hfruits-sm*, we have conducted extensive experiments to determine what adaptations would lead to the highest accuracy on *hfruits-sm*. These experiments included retraining the best performing models on *hfruits-base* while including a small portion of *hfruits-sm*, forming our so-called dual-trained models. Additionally, the models have been run on *hfruits-base* with a small, augmented portion of *hfruits-sm*. Furthermore, the models have also been trained while having the validation set be the same as the training set, as the *hfruits-base*'s validation set does not contain samples from *hfruits-sm*. We specifically decided against this, because we wanted the required specialized data to be as minimal as possible.

Lastly, to validate how well the dual-trained models performed on *hfruits-sm*, the best performing models were also finetuned using just *hfruits-sm*. To accomplish this, the same number of images were used as were added to the dual-trained model. Additionally, the sm-trained model was tested on *hfruits-base* to determine the generalisability of a model trained on a more complex dataset.

Chapter 5

Results and Analysis

5.1 Model Testing

After testing the models on *hfruits-base*, it became evident that two models, YOLOV8 and RT-DETR, significantly outperformed the other two, as seen in Table 5.1. Not only were their MAP-scores higher, they also demonstrated faster prediction times. YOLOV8 and RT-DETR met the set threshold to be considered well-performing models, hence they were chosen for further analysis.

Model Name	**MAP@50**	**MAP@50:95**	**Prediction Speed per Image (s)**
YOLOV8	0.994	0.774	0.0054
RT-DETR	0.99	0.77	0.0037
SSD	0.854	0.520	0.9985
Faster RCNN	0.850	0.517	1.0119

Table 5.1: Model Testing on *hfruits-base*

5.2 Model Evaluation

Without retraining, the performance of YOLOv8 and RT-DETR on *hfruits-sm* decreased drastically, as shown in Table 5.2. While YOLOV8's prediction speed did not seem to decrease significantly, RT-DETR became nearly twice as slow.

Model Name	**MAP@50**	**MAP@50:95**	**Prediction Speed per Image (s)**
YOLOV8	0.447	0.292	0.0055
RT-DETR	0.52	0.393	0.0069

Table 5.2: Base-trained Model Testing on *hfruits-sm*

Although RT-DETR seemed to perform better at first glance, disregarding its prediction speed, the APs per category told a different story.

Table 5.3 shows the performance of YOLOV8 on every category. Although some fruit categories performed slightly worse, namely banana, pear and persimmon, the differences were not extreme. The best performing category was orange, with an AP@50 of 0.698 and AP@50:95 of 0.568, which is still below the set requirement.

Category	**AP@50**	**AP@50:95**
Apple	0.467	0.249
Banana	0.268	0.104
Kiwi	0.466	0.294
Lemon	0.616	0.404
Mango	0.556	0.396
Orange	0.698	0.568
Pear	0.259	0.139
Persimmon	0.25	0.178

Table 5.3: Base-trained YOLOV8 Testing on *hfruits-sm*, per category

Table 5.4, on the other hand, does show extreme differences between the categories, namely between apple and pear. Whereas the category apple performed as well on *hfruits-sm* as on *hfruits-base*, with an AP@50 of 0.986 and AP@50:95 of 0.759, the category pear was barely recognized, with an AP@50 of 0.0589 and AP@50:95 of 0.0356.

Category	**AP@50**	**AP@50:95**
Apple	0.986	0.759
Banana	0.263	0.156
Kiwi	0.767	0.558
Lemon	0.308	0.243
Mango	0.712	0.554
Orange	0.326	0.273
Pear	0.0589	0.0356
Persimmon	0.737	0.564

Table 5.4: Base-trained RT-DETR Testing on *hfruits-sm*, per category

The confusion matrix of YOLOV8 can be seen in Figure 5.1. The most notable element in the matrix, is the tendency of the model to recognize persimmons when there are none present. Moreover, it shows that a lot of fruits were not recognized at all, meaning the model classified the image as background. Additionally, some fruits were often misclassified. For instance, mangoes were more often recognized

as pears instead of as themselves. Section 10.3.1 presents some visual examples of YOLOV8's detections.

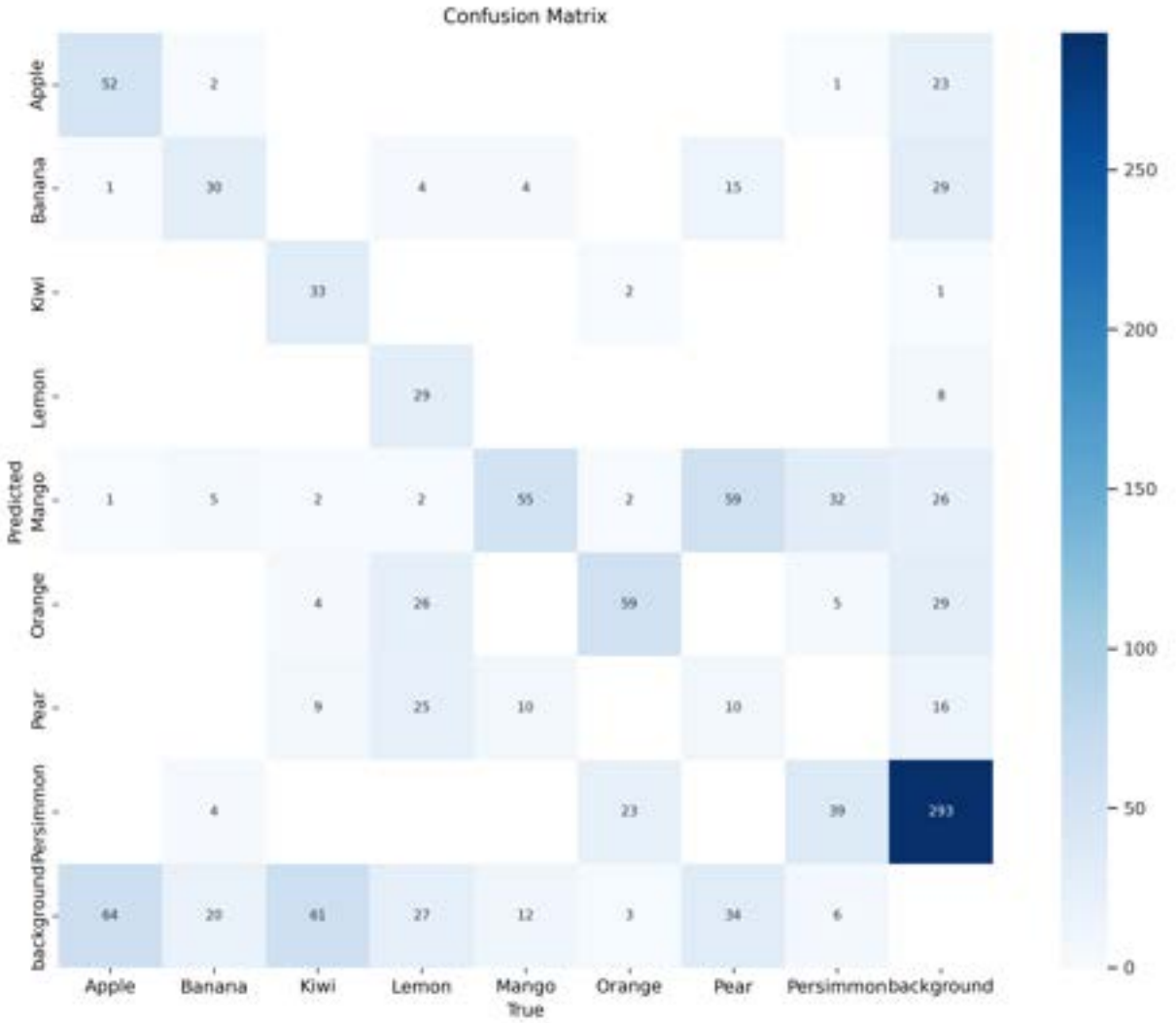


Figure 5.1: Confusion Matrix of Base-trained YOLOV8 on *hfruits-sm*

Figure 5.2 shows the confusion matrix of RT-DETR. This model also had a tendency to recognize persimmons, and oranges, when there were none. There were also still fruits that seemed to be not detected at all. There were some misclassifications that caused the AP per category to go down. Similar to YOLOV8, a lot of mangoes were classified as pears. Some visual examples are shown in section 10.3.2.

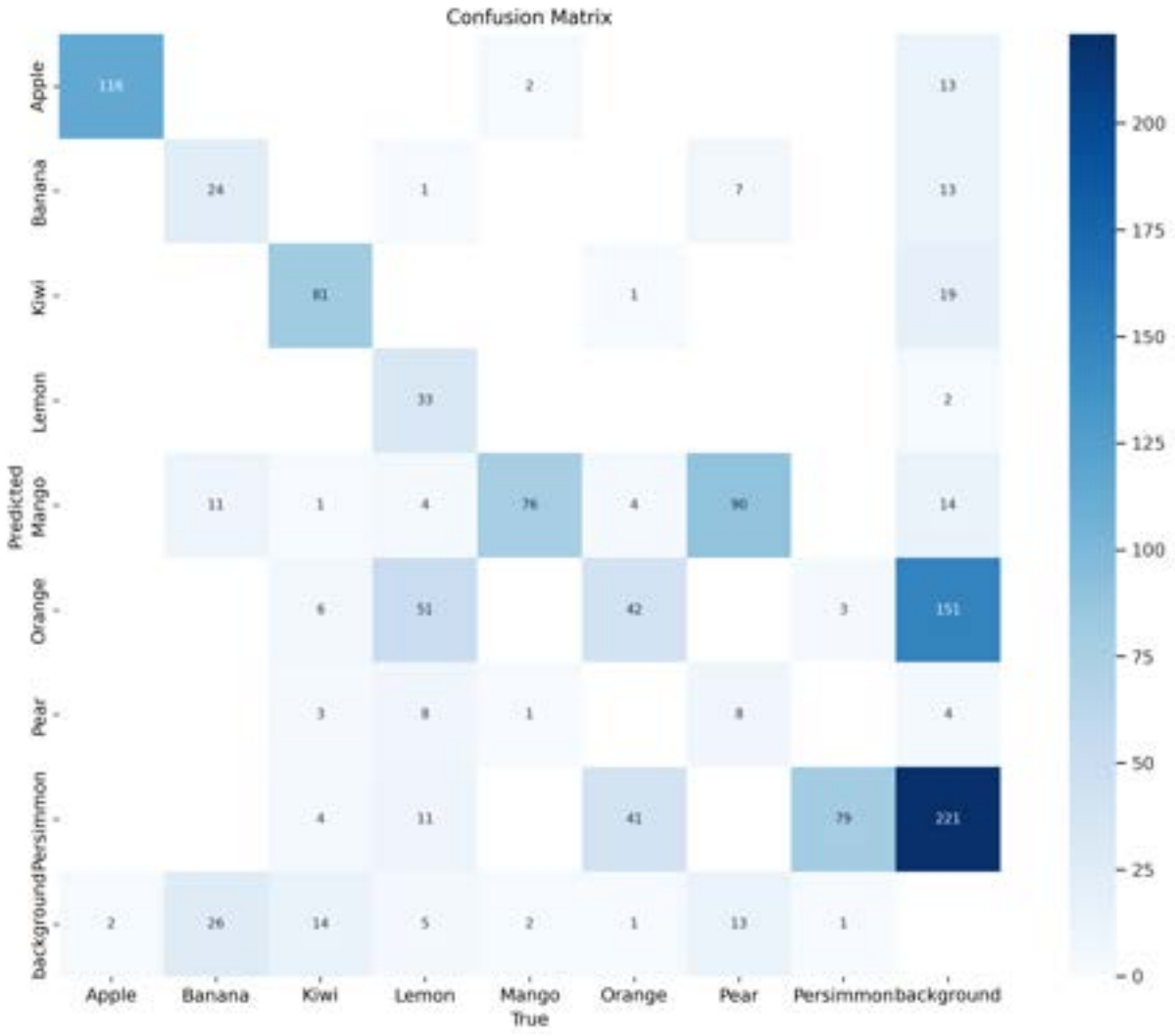


Figure 5.2: Confusion Matrix of Base-trained RT-DETR on *hfruits-sm*

5.3 Model Improvement

The results of retraining YOLOV8 and RT-DETR are shown in Figure 5.3 and Figure 5.4. The former shows the models' results on *hfruits-base*, whereas the latter shows the results on *hfruits-sm*. When comparing the dual-trained models' performance on *hfruits-base* to the base-trained models' performance, shown in Table 5.1, it is visible most dual-trained models performed as well as or better than the base-trained model. The only exceptions to this are the MAP@50-95-values of RT-DETR (no val), RT-DETR (augmented data) and RT-DETR (no val, augmented data), although the difference was minuscule.

In general, the dual-trained YOLOV8 performed slightly better on *hfruits-base* than the dual-trained RT-DETR. It is noteworthy that having a validation set did not have an influence on YOLOV8's weight, hence there are only two distinct sets of weights.

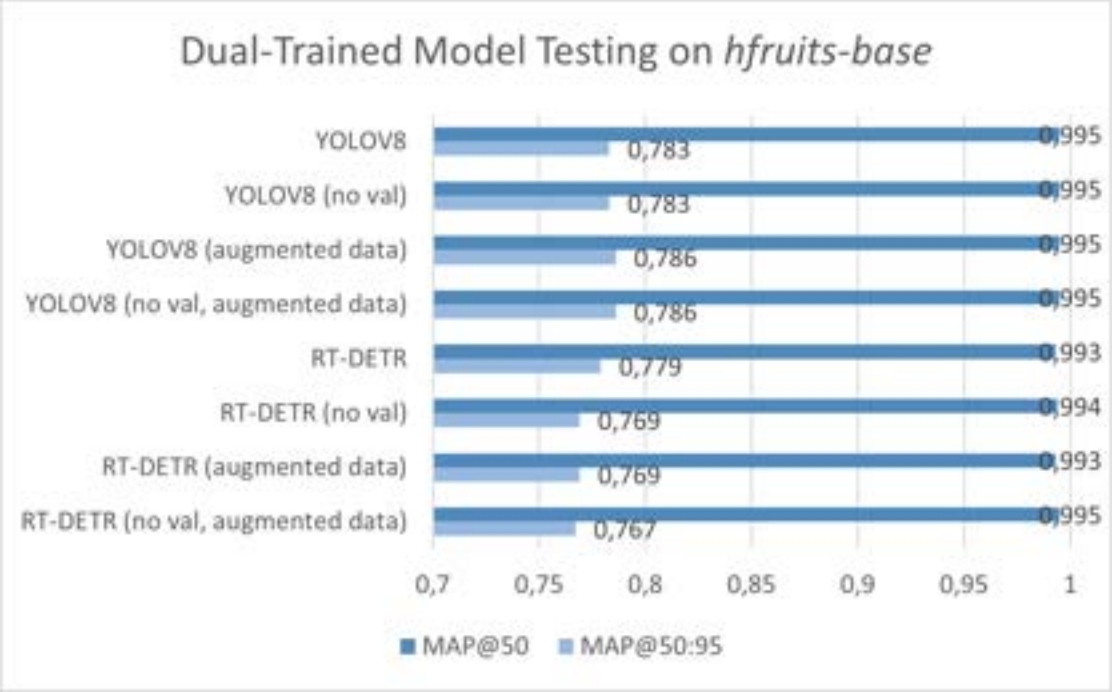


Figure 5.3: Dual-Trained Model Testing on *hfruits-base*

The dual-trained models' performance on *hfruits-sm* is significantly better than the base-trained models' performance. The worst performing dual-trained model is RT-DETR without a validation set. The best performing model is RT-DETR with augmented data with a MAP@50 of 0.952 and a MAP@50:95 of 0.77. However, considering that YOLOV8 performs better on *hfruits-base*, YOLOV8, with

or without a validation set and with augmented data, is a serious contender as well. Averaging over the MAP-values of these two models results in a MAP@50 of 0.9735 and a MAP@50:95 of 0.7665 for YOLOV8 and a MAP@50 of 0.9725 and a MAP@50:95 of 0.7695 for RT-DETR. While YOLOv8 has a better MAP@50-score, RT-DETR has a better MAP@50:95-score, which means the models are on par.

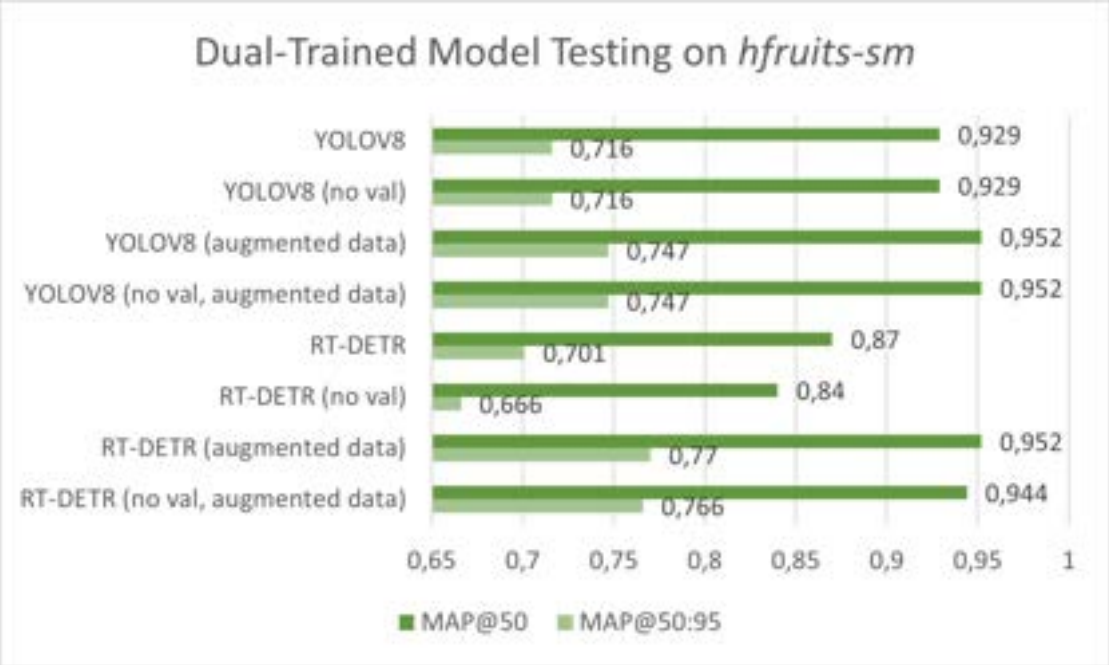


Figure 5.4: Dual-Trained Model Testing on *hfruits-sm*

All variants of YOLOV8 achieved higher prediction speeds on *hfruits-sm* than the variants of RT-DETR, as seen in Table 5.5.

Model Name	**Prediction Speed per Image (s)**
YOLOV8	0.0054
YOLOV8 (no val)	0.0054
YOLOV8 (augmented data)	0.0055
YOLOV8 (no val, augmented data)	0.0055
RT-DETR	0.0070
RT-DETR (no val)	0.0076
RT-DETR (augmented data)	0.0070
RT-DETR (no val, augmented data)	0.0064

Table 5.5: Dual-Trained Model Testing Speed on *hfruits-sm*

The results of the best-performing YOLOV8 variant are shown in Table 5.6. YOLOV8 with data augmentation achieved high accuracies for both AP@50 as well as AP@50:95 for five out of eight categories: apple, lemon, mango, orange and persimmon. The category kiwi fell short with a AP@50:95 of 0.731. The category banana performed well on AP@50 with a score of 0.936, but less well on AP@50:95 with a score of 0.671. The category pear showed a similar pattern, achieving an AP@50 of 0.928 and an AP@50:95 of 0.631.

Category	**AP@50**	**AP@50:95**
Apple	0.995	0.828
Banana	0.936	0.671
Kiwi	0.938	0.731
Lemon	0.975	0.772
Mango	0.991	0.811
Orange	0.902	0.763
Pear	0.928	0.631
Persimmon	0.949	0.772

Table 5.6: Dual-trained YOLOV8 (augmented data) Testing on *hfruits-sm*, per category

Figure 5.5 shows the corresponding confusion matrix, in which it is clear that the model often detected fruits when there were none, especially bananas, decreasing the overall performance. It also shows some misclassifications, such oranges being classified as lemons and pears as mangoes. Figure 5.6 shows some visual examples of the model's predictions.

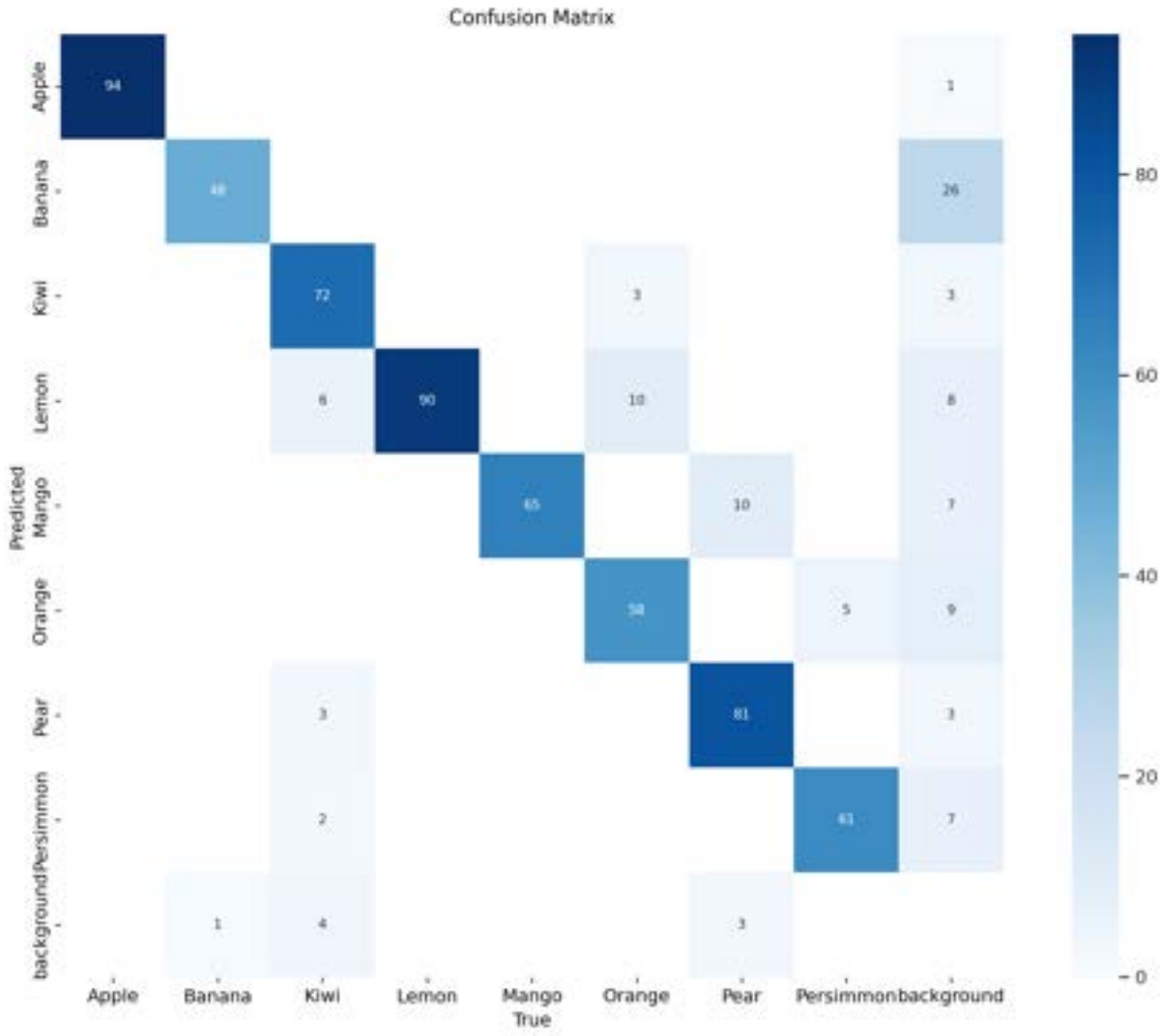


Figure 5.5: Confusion Matrix of Dual-Trained YOLOV8 (augmented data) on *hfruits-sm*

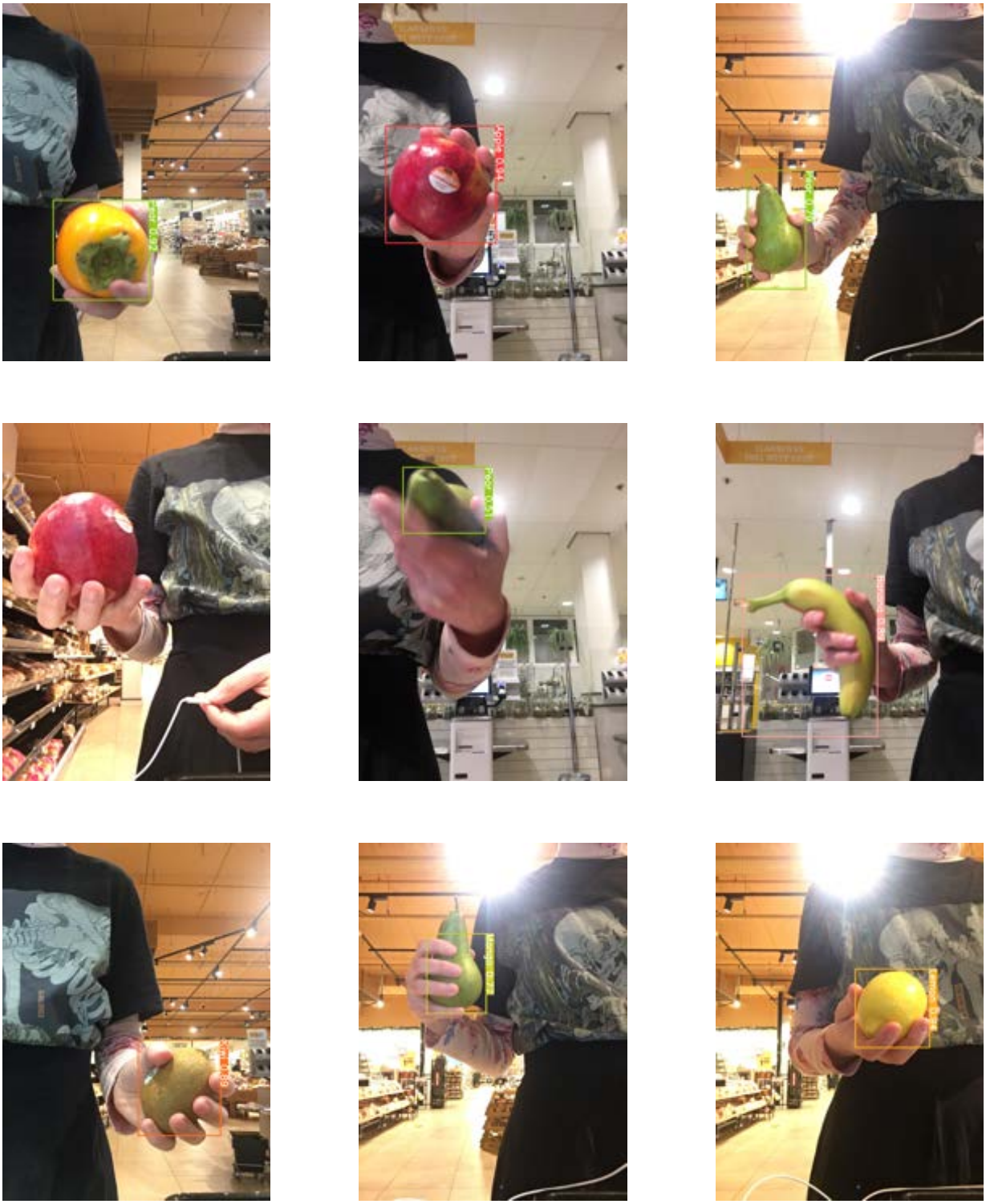


Figure 5.6: Dual-Trained YOLOV8 (augmented data) predictions on *hfruits-sm*

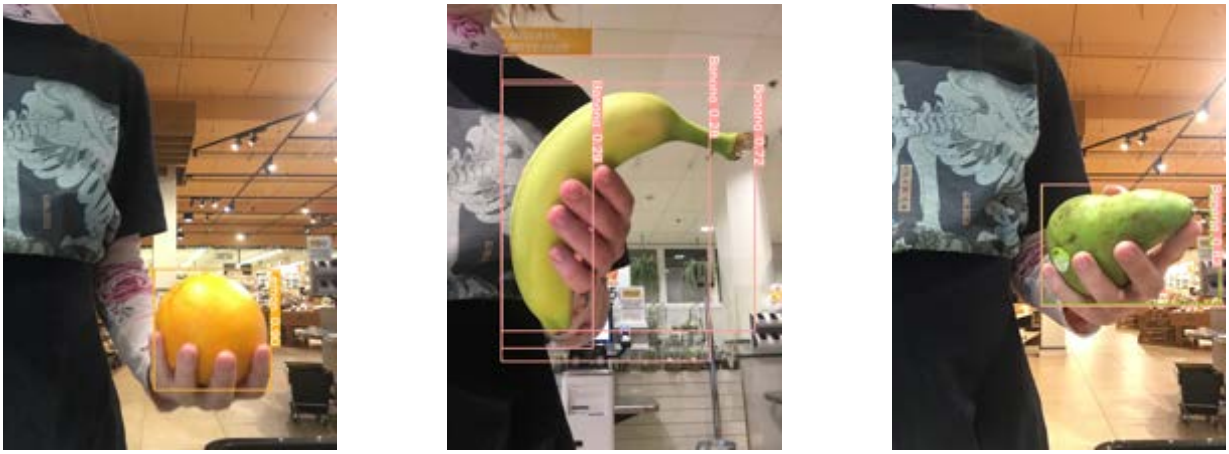


Figure 5.6: Dual-Trained YOLOV8 (augmented data) predictions on *hfruits-sm* (continued)

The scores per category of RT-DETR's best variant have been set out in Table 5.7. RT-DETR with augmented data also achieved high scores in five out of eight categories: apple, banana, lemon, kiwi and mango. The category persimmon came very close with an AP@50:95 of 0.746. The categories orange and pear were further off with AP@50:95s of 0.729 and 0.658 respectively.

Category	**AP@50**	**AP@50:95**
Apple	0.995	0.813
Banana	0.974	0.859
Kiwi	0.968	0.763
Lemon	0.942	0.754
Mango	0.993	0.805
Orange	0.859	0.729
Pear	0.917	0.658
Persimmon	0.904	0.746

Table 5.7: Dual-Trained RT-DETR (augmented data) Testing on *hfruits-sm*, per category

RT-DETR's confusion matrix, as seen in Figure 5.7, shows that the model detected less non-existing fruit than YOLOV8. It did have some misclassifications. Some of RT-DETR's predictions are shown in Figure 5.8.

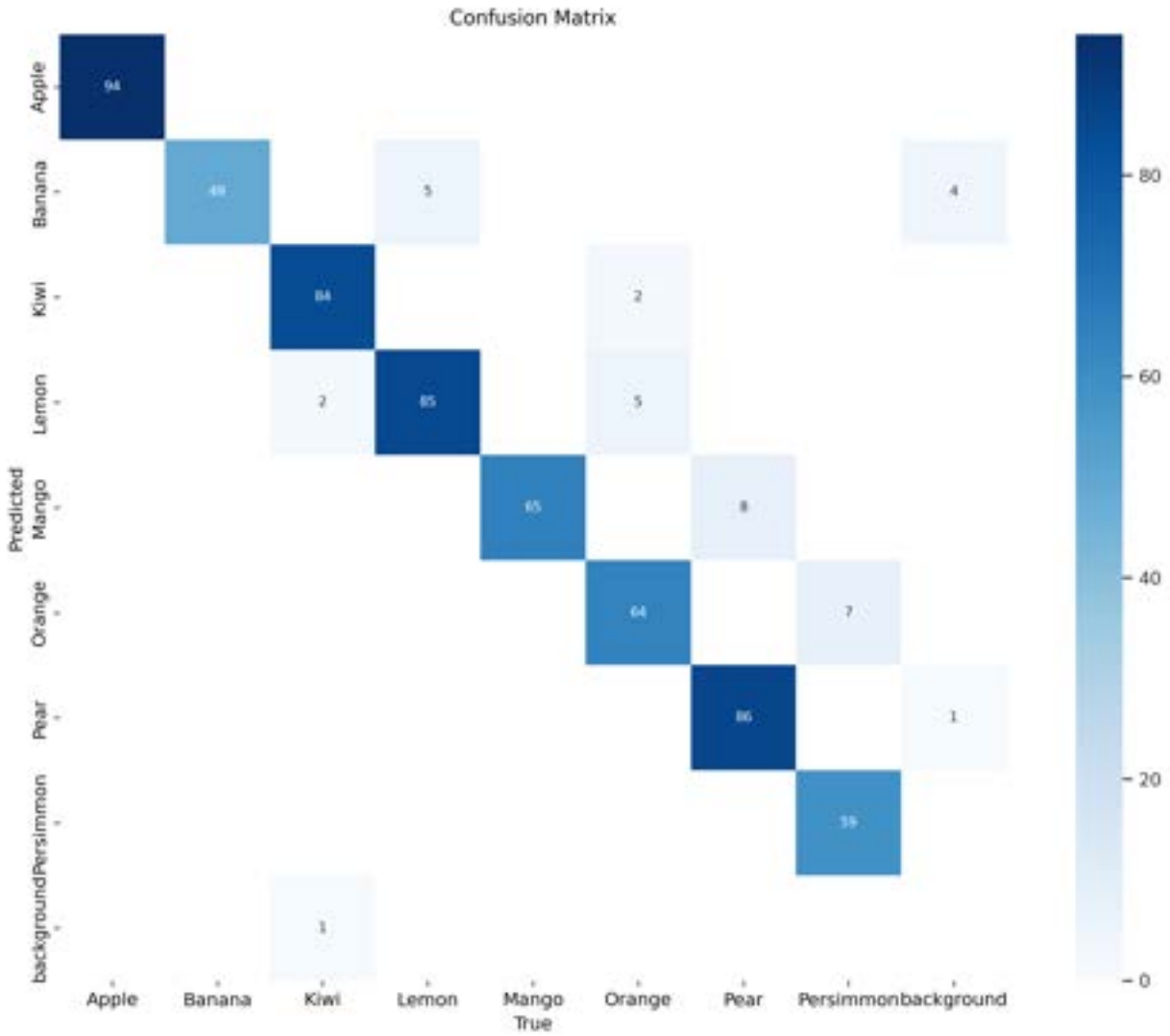


Figure 5.7: Confusion Matrix of Dual-Trained RT-DETR (augmented data) on *hfruits-sm*

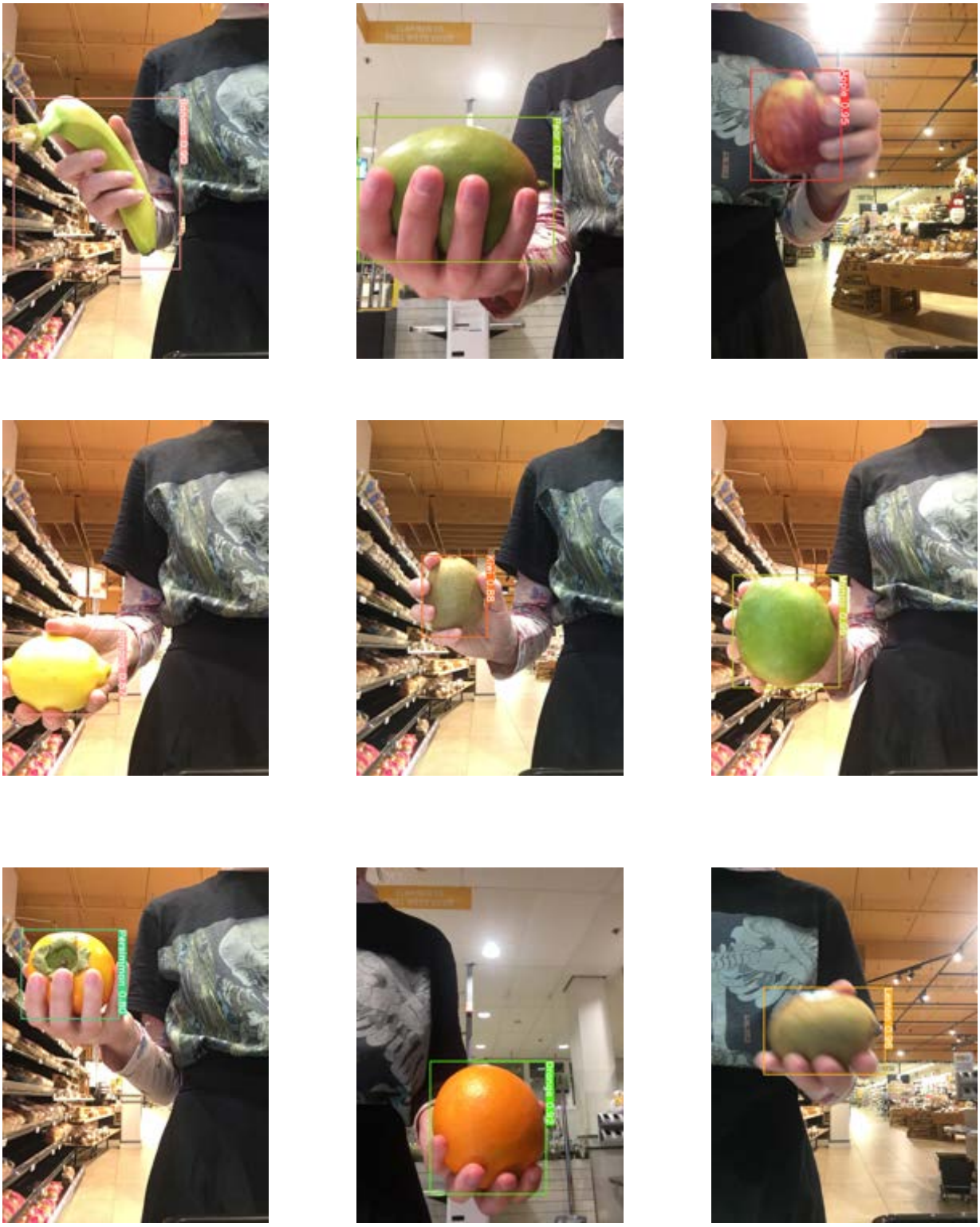


Figure 5.8: Dual-Trained RT-DETR (augmented data) predictions on *hfruits-sm*

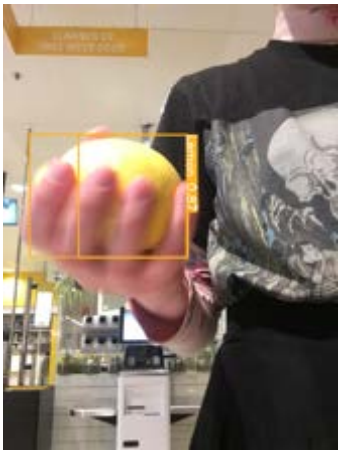
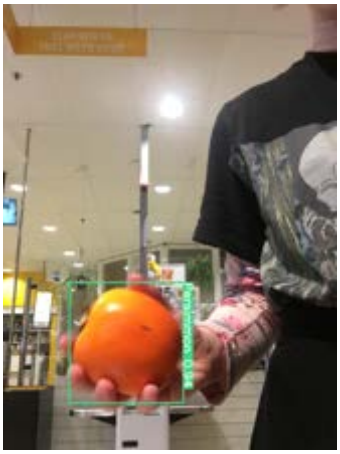




Figure 5.8: Dual-Trained RT-DETR (augmented data) predictions on *hfruits-sm* (continued)

5.4 Model Comparison

The results of finetuning YOLOV8 and RT-DETR to *hfruits-sm* are set out in Table 5.8. Surprisingly, even with the small amount of data, RT-DETR achieved quite high accuracies. However, the dual-trained RT-DETR (augmented data) did still perform better.

Model	**MAP@50**	**MAP@50:95**	**Prediction Speed per Image (s)**
YOLOV8	0.743	0.555	0.0056
RT-DETR	0.903	0.722	0.0066

Table 5.8: Sm-Trained Model Testing on *hfruits-sm*

To determine how the sm-trained models are able to generalise to a new background, they have been tested on *hfruits-base*, as shown in Table 5.9. The table demonstrates that sm-trained models were also not able to easily adapt to a new environment.

Model	**MAP@50**	**MAP@50:95**	**Prediction Speed per Image (s)**
YOLOV8	0.359	0.219	0.0056
RT-DETR	0.475	0.326	0.0067

Table 5.9: Sm-Trained Model Testing on *hfruits-base*

Chapter 6

Discussion

This study looked at a model's ability to detect hand-occluded fruits in new supermarket environments. Simply training a model on a *hfruits-base* for detection on *hfruits-sm* did not prove to be effective, given the prevalence of non-existent fruit detections. Another significant concern was the observation that some of the predicted bounding boxes were not tight-fitting. One could argue that this is not essential for self-checkout fruit detection, but detections that include a lot of background, might increase the likelihood of fruit detection where none exist.

Although YOLOV8 and RT-DETR performed considerable better on *hfruits-sm* after being dual-trained, they still contained some errors. One crucial problem is the three detections of the same banana, as can be seen in subsection 10.3.1. It is essential it is only detected once, as a customer should not pay thrice the price for one banana. Moreover, especially for YOLOV8, the detection of non-existent fruits remained a problem. This is extremely undesirable for self-checkout registers, as customers should not be charged for fruits that do not exist.

While the dual-trained models did perform better than the sm-trained models, the performance of RT-DETR is noteworthy to highlight. With merely 154 images divided over eight categories, RT-DETR was able to achieve a MAP@50 of 0.903 and a MAP@50:95 of 0.722 on *hfruits-sm*. Despite these accuries not reaching our standard for a good performance, they come close, prompting the question of how many images would be sufficient.

This research has a couple of limitations. Firstly, the datasets only contain eight different fruits. It is unclear whether *hfruits-base* will allow for the straightforward learning of fruits that are not in the dataset. However, there have been studies focusing on the integration of new fruits. For instance, Zhang et al. (2021) explored how unlabelled fruits could be easily learned, using the CycleGAN network, a pseudo-labelling process, and a pseudo-label self-learning process. Nonetheless, the authors acknowledge that there is still room for further research in this area. Ad-

ditionally, fruits are not the only items without barcodes at supermarkets. Other items include vegetables and certain types of bread, such as croissants. It remains uncertain if these items could be learned using domain adaptation as well. Furthermore, this solution does not take weight into account, which might be necessary for accurate pricing. Not to mention, the solution would need to be adapted to real-time video instead of to static pictures.

It can be argued that mean average precision is not sufficient for real-time fruit detection. Mao et al. (2019) suggest that video detection algorithms should contain a delay metric as well to accurately evaluate an algorithm's performance. Considering that we want fruits to be detected as efficiently as possible, this is something worth exploring.

Chapter 7

Conclusion

This paper investigated to what extent object detection algorithms are able to detect hand-occluded fruits in new supermarket environments. It explored the detection of hand-occluded fruits and the appropriate algorithms, assessed the effectiveness of base-training for detection on a new environment, and experimented with various methods to improve the performance on the new environment.

Two data sets were created, containing eight categories of hand-held fruit: *hfruits-base* and *hfruits-sm*, the latter of which was collected in a supermarket environment. After finetuning four models, YOLOV8, RT-DETR, SSD and Faster RCNN, to *hfruits-base* and testing them on the same dataset, it became clear that YOLOV8 and RT-DETR achieved high accuracies, using mean average precision as a metric. Therefore, we can conclude that the detection of hand-occluded fruits is feasible. Simply applying the base-trained models to *hfruits-sm*, without retraining, did not work well. However, retraining the model, using a small part of *hfruits-sm*, proved to be very effective. RT-DETR with augmented data achieved a MAP@50 of 0.952 and a MAP@50:95 of 0.77, outperforming the sm-trained models. Based on these results, we can say that hand-occluded fruits can be detected in this specific new supermarket environment. The question at hand is whether it will work for other supermarket environments.

Further research could focus on several aspects, most notably: determining whether dual-training using *hfruits-base* will work for other supermarket environments. It could also focus on the expansion of fruit variety. As mentioned in chapter 6, the collected datasets only contain a select number of fruits. A further study could investigate how new fruits could be integrated efficiently into these datasets. It could also explore whether it is possible to use even less specialized data for the existing fruits while preserving the performance. Another focus could be on simultaneous detection of multiple fruits, as this could further improve efficiency at self-checkout registers. Moreover, it is worth investigating why certain

algorithms detect fruits where there are none. This is detrimental to self-checkout efficiency, hence it should be avoided. Lastly, more research could be done in more advanced applications of domain adaptation using these datasets.

Chapter 8

Acknowledgements

I am thankful to my supervisor, Dr. You, for his guidance and feedback on my work. I would also like to thank my supervisor, Ms. Raj, for her input and support. Lastly, I acknowledge the Jumbo management and employees for allowing me to collect *hfruits-sm* in their supermarket.

References

Bargoti, S., & Underwood, J. (2017). Deep fruit detection in orchards. In *2017 ieee international conference on robotics and automation (icra)* (p. 3626-3633). doi: 10.1109/ICRA.2017.7989417

Bulmer, S., Elms, J., & Moore, S. (2018). Exploring the adoption of self-service checkouts and the associated social obligations of shopping practices. *Journal of Retailing and Consumer Services*, *42*, 107-116. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0969698917307348` doi: https://doi.org/10.1016/j.jretconser.2018.01.016

Deng, Z., Sun, H., Zhou, S., Zhao, J., Lei, L., & Zou, H. (2018). Multi-scale object detection in remote sensing imagery with convolutional neural networks. *ISPRS Journal of Photogrammetry and Remote Sensing*, *145*, 3-22. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0924271618301096` (Deep Learning RS Data) doi: https://doi.org/10.1016/j.isprsjprs.2018.04.003

Farahani, A., Voghoei, S., Rasheed, K., & Arabnia, H. R. (2021). A brief review of domain adaptation. In R. Stahlbock, G. M. Weiss, M. Abou-Nasr, C.-Y. Yang, H. R. Arabnia, & L. Deligiannidis (Eds.), *Advances in data science and information engineering* (pp. 877–894). Cham: Springer International Publishing.

Goyal, K., Kumar, P., & Verma, K. (2023). Ai-based fruit identification and quality detection system. *Multimedia Tools and Applications, 82, 24573–24604* . doi: https://doi.org/10.1007/s11042-022-14188-x

Hou, G., Chen, H., Jiang, M., & Niu, R. (2023). An overview of the application of machine vision in recognition and localization of fruit and vegetable harvesting robots. *Agriculture*, *13*(9). Retrieved from `https://www.mdpi.com/2077-0472/13/9/1814` doi: 10.3390/agriculture13091814

Jia, W., Tian, Y., Luo, R., Zhang, Z., Lian, J., & Zheng, Y. (2020). Detection and segmentation of overlapped fruits based on optimized mask r-cnn applica-

tion in apple harvesting robot. *Computers and Electronics in Agriculture*, *172*, 105380. Retrieved from https://www.sciencedirect.com/science/article/pii/S0168169919326274 doi: https://doi.org/10.1016/j.compag.2020.105380

Jocher, G., Chaurasia, A., & Qiu, J. (2023). *Ultralytics yolov8* [software]. Retrieved from https://github.com/ultralytics/ultralytics

Liu, F., Snetkov, L., & Lima, D. (2017/07). Summary on fruit identification methods: A literature review. In *Proceedings of the 2017 3rd international conference on economics, social science, arts, education and management engineering (essaeme 2017).* Atlantis Press. Retrieved from https://doi.org/10.2991/essaeme-17.2017.338 doi: 10.2991/essaeme-17.2017.338

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Computer vision – eccv 2016* (pp. 21–37). Cham: Springer International Publishing.

Lv, W., Xu, S., Zhao, Y., Wang, G., Wei, J., Cui, C., ... Liu, Y. (2023). *Detrs beat yolos on real-time object detection.* (Software available at: https://github.com/lyuwenyu/RT-DETR)

Mai, X., Zhang, H., Jia, X., & Meng, M. Q.-H. (2020). Faster r-cnn with classifier fusion for automatic detection of small fruits. *IEEE Transactions on Automation Science and Engineering*, *17*(3), 1555-1569. doi: 10.1109/TASE.2020.2964289

Mao, H., Yang, X., & Dally, W. J. (2019, October). A delay metric for video object detection: What average precision fails to tell. In *Proceedings of the ieee/cvf international conference on computer vision (iccv).*

Mirhaji, H., Soleymani, M., Asakereh, A., & Abdanan Mehdizadeh, S. (2021). Fruit detection and load estimation of an orange orchard using the yolo models through simple approaches in different imaging and illumination conditions. *Computers and Electronics in Agriculture*, *191*, 106533. Retrieved from https://www.sciencedirect.com/science/article/pii/S0168169921005500 doi: https://doi.org/10.1016/j.compag.2021.106533

Naranjo-Torres, J., Mora, M., Hernández-García, R., Barrientos, R. J., Fredes, C., & Valenzuela, A. (2020). A review of convolutional neural network applied to fruit image processing. *Applied Sciences*, *10*(10). Retrieved from https://www.mdpi.com/2076-3417/10/10/3443 doi: 10.3390/app10103443

Padilla, R., Netto, S. L., & da Silva, E. A. B. (2020). A survey on performance metrics for object-detection algorithms. In *2020 international conference on systems, signals and image processing (iwssip)* (p. 237-242). doi: 10.1109/IWSSIP48289.2020.9145130

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You only look once: Unified, real-time object detection.*

Ren, S., He, K., Girshick, R., & Sun, J. (2017, jun). Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis; Machine Intelligence*, *39*(06), 1137-1149. doi: 10.1109/TPAMI.2016.2577031

Sa, I., Ge, Z., Dayoub, F., Upcroft, B., Perez, T., & McCool, C. (2016). Deepfruits: A fruit detection system using deep neural networks. *Sensors*, *16*(8). Retrieved from https://www.mdpi.com/1424-8220/16/8/1222 doi: 10.3390/s16081222

Saedi, S. I., & Khosravi, H. (2020). A deep neural network approach towards real-time on-branch fruit recognition for precision horticulture. *Expert Systems with Applications*, *159*, 113594. Retrieved from https://www.sciencedirect.com/science/article/pii/S0957417420304188 doi: https://doi.org/10.1016/j.eswa.2020.113594

Terven, J., Córdova-Esparza, D.-M., & Romero-González, J.-A. (2023, November). A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. *Machine Learning and Knowledge Extraction*, *5*(4), 1680–1716. Retrieved from http://dx.doi.org/10.3390/make5040083 doi: 10.3390/make5040083

Tkachenko, M., Malyuk, M., Holmanyuk, A., & Liubimov, N. (2020-2022). *Label Studio: Data labeling software.* Retrieved from https://github.com/heartexlabs/label-studio (Open source software available from https://github.com/heartexlabs/label-studio)

Ukwuoma, C. C., Zhiguang, Q., Heyat, M. B. B., Ali, L., Almaspoor, Z., & Monday, H. N. (2022). Recent advancements in fruit detection and classi-

fication using deep learning techniques. *Mathematical Problems in Engineering*, *2022*. Retrieved from https://doi.org/10.1155/2022/9210947 doi: 10.1155/2022/9210947

Wang, P., Niu, T., & He, D. (2021). Tomato young fruits detection method under near color background based on improved faster r-cnn with attention mechanism. *Agriculture*, *11*(11). Retrieved from https://www.mdpi.com/2077-0472/11/11/1059 doi: 10.3390/agriculture11111059

Yang, G., Wang, J., Nie, Z., Yang, H., & Yu, S. (2023). A lightweight yolov8 tomato detection algorithm combining feature enhancement and attention. *Agronomy*, *13*(7). Retrieved from https://www.mdpi.com/2073-4395/13/7/1824 doi: 10.3390/agronomy13071824

Yu, Y., Zhang, K., Yang, L., & Zhang, D. (2019). Fruit detection for strawberry harvesting robot in non-structural environment based on mask-rcnn. *Computers and Electronics in Agriculture*, *163*, 104846. Retrieved from https://www.sciencedirect.com/science/article/pii/S0168169919301103 doi: https://doi.org/10.1016/j.compag.2019.06.001

Zhang, W., Chen, K., Wang, J., Shi, Y., & Guo, W. (2021, 06). Easy domain adaptation method for filling the species gap in deep learning-based fruit detection. *Horticulture Research*, *8*, 119. Retrieved from https://doi.org/10.1038/s41438-021-00553-8 doi: 10.1038/s41438-021-00553-8

Zheng, T., Jiang, M., Li, Y., & Feng, M. (2022). Research on tomato detection in natural environment based on rc-yolov4. *Computers and Electronics in Agriculture*, *198*, 107029. Retrieved from https://www.sciencedirect.com/science/article/pii/S0168169922003465 doi: https://doi.org/10.1016/j.compag.2022.107029

Chapter 10

Appendices

10.1 Dataset Structures

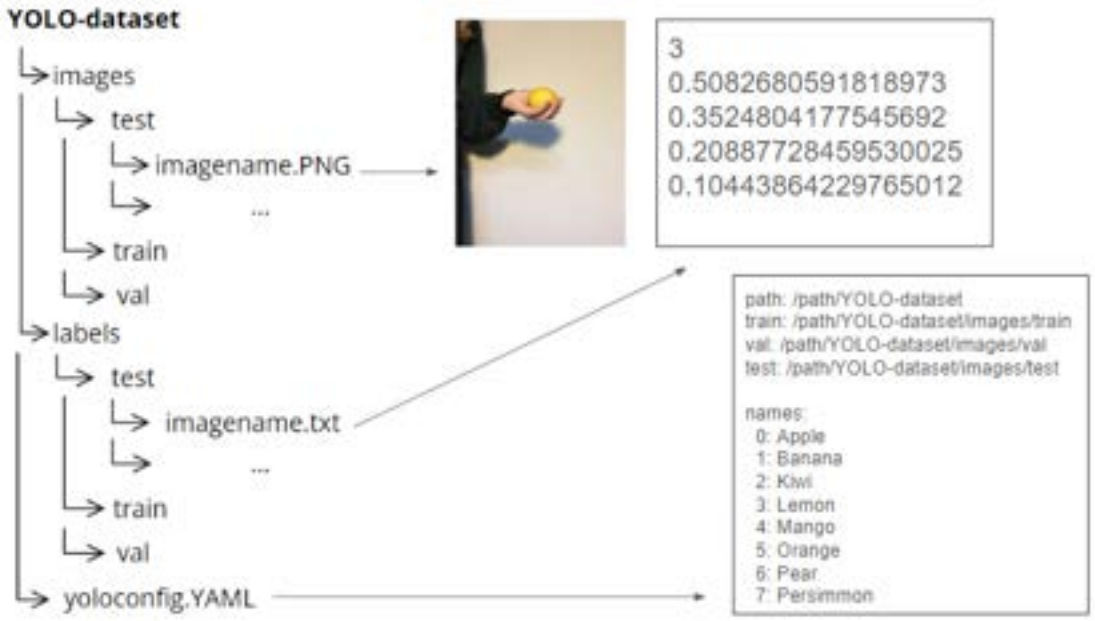


Figure 10.1: *hfruits-base*: YOLO structure

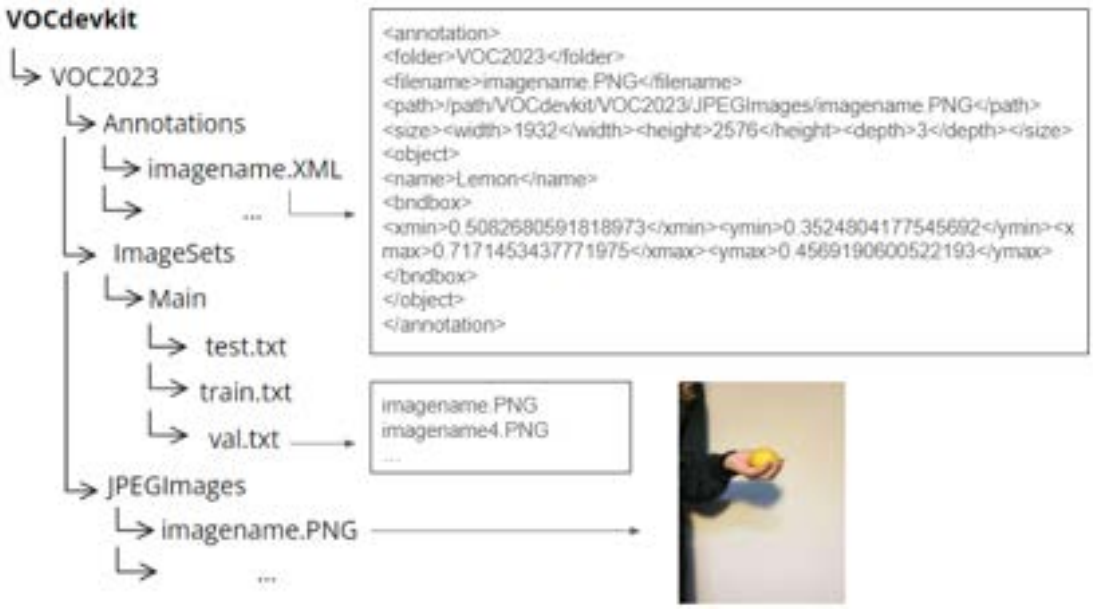


Figure 10.2: *hfruits-base*: PASCAL-VOC structure)

10.2 Chosen Models

10.2.1 Faster RCNN

Faster Region-based Convolutional Network, abbreviated as Faster RCNN, is a two-stage object detection algorithm (Ren et al., 2017). It was obtained by merging Fast RCNN and RPN, or Region Proposal Network. Figure 10.3 shows the model's structure. The model uses a convolution neural network as a backbone, VGG in this example, which extracts the images' features. These features are used in both stages. The first stage entails proposing regions, which is done by the RPN, a deep fully connected convolutional network. Afterwards, the regions are used by the Fast RCNN detector to detect objects. Whereas Fast RCNN and RCNN use external region proposal methods, Faster RCNN unifies this process by integrating the RPN, making the detection process more time-efficient and accurate.

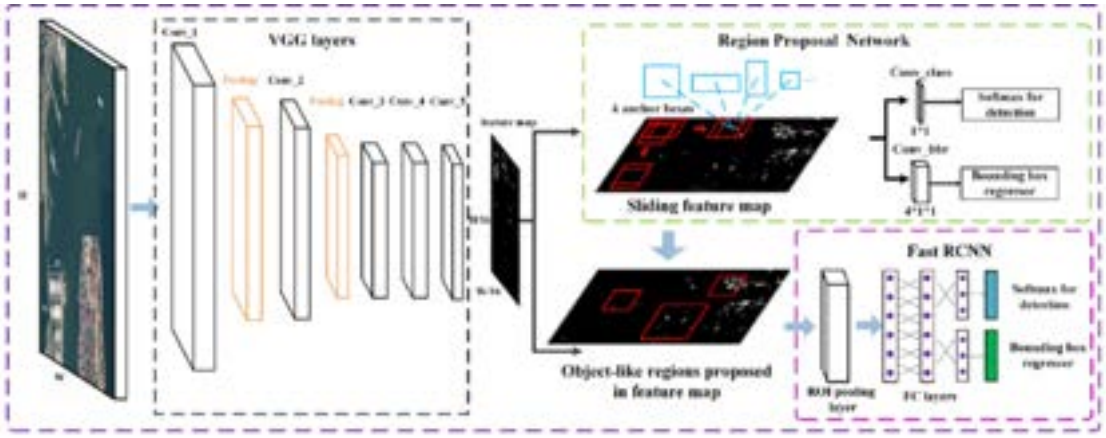


Figure 10.3: Faster RCNN structure (Deng et al., 2018)

10.2.2 SSD

SSD, or Single Shot MultiBox Detector, is a one-stage object detection algorithm, which is made up of a single deep neural network (W. Liu et al., 2016). The network uses default boxes, which are obtained by discretizing the output space, and feature maps to accurately detect objects. The model's structure is shown in Figure 10.4. The model builds upon the VGG-16 network, but instead of fully connected layers it adds convolutional layers at the end. On the VOC2007 test set with two different input sizes, the model outperformed Faster R-CNN.

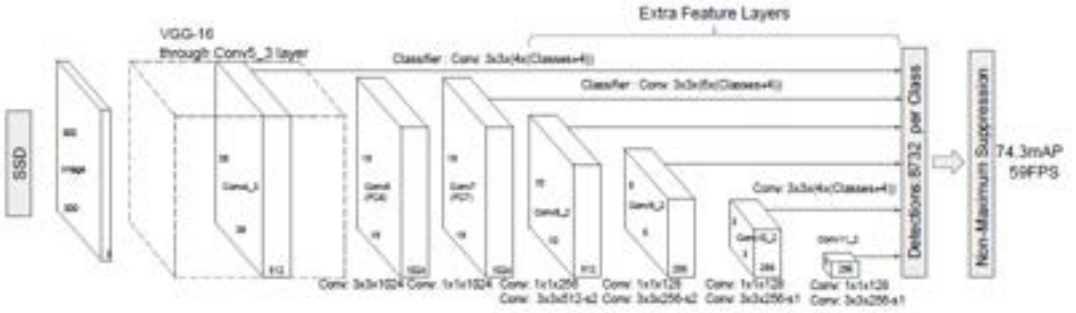


Figure 10.4: SSD structure (W. Liu et al., 2016)

10.2.3 YOLOV8

You Only Look Once, or YOLO, is an algorithm that was introduced in 2015 (Redmon et al., 2016), after which multiple versions of the model improved upon the original's model accuracy, including YOLOv8 in 2023 (Terven et al., 2023)(Jocher et al., 2023). The YOLO algorithm detects all objects simultaneously by dividing

the input image into a grid. As the network only requires a single forward pass, it is classified as a one-stage algorithm. As shown in Figure 10.5, the model consists of a backbone, neck and head. In the backbone, feature maps are extracted, after which the neck merges them together. Finally, the head generates predictions.

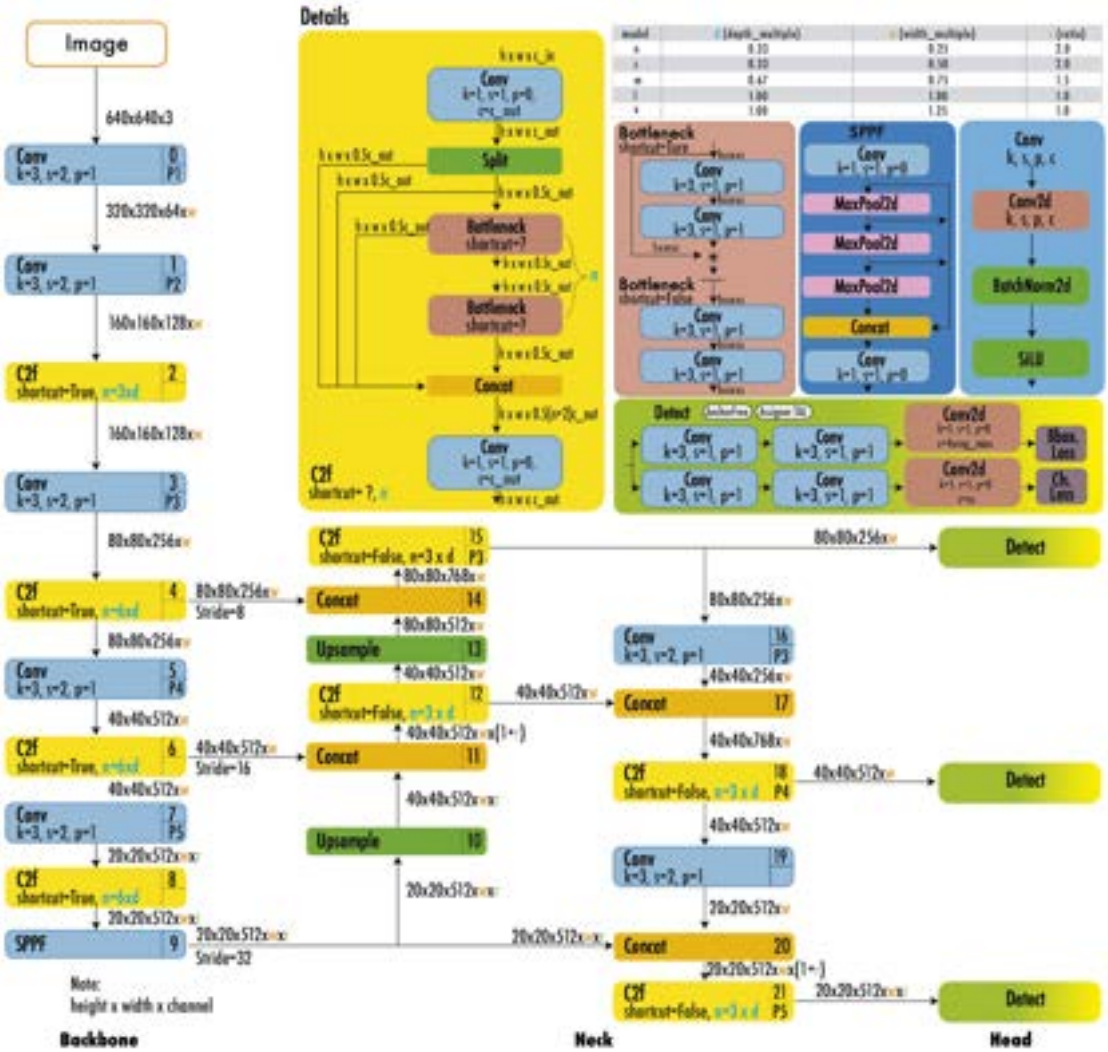


Figure 10.5: YOLOV8 structure (Terven et al., 2023)

10.2.4 RT-DETR

RT-DETR, Real Time DEtection TRansformer, is a two-stage algorithm based on the transformer encoder-decoder architecture, consisting of three components (Lv et al., 2023). These include a backbone, a hybrid encoder and a transformer decoder with auxiliary prediction heads, as can be seen in Figure 10.6. On the

COCO val2017 data set, RT-DETR outperformed all other versions of YOLO of the same scale in both accuracy and speed.

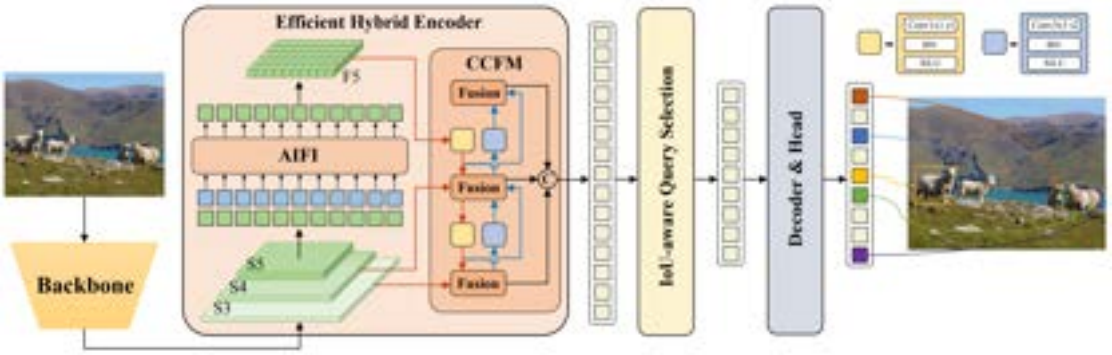


Figure 10.6: RT-DETR structure (Lv et al., 2023)

10.3 Trained Model Predictions on *hfruits-sm*

10.3.1 YOLOV8

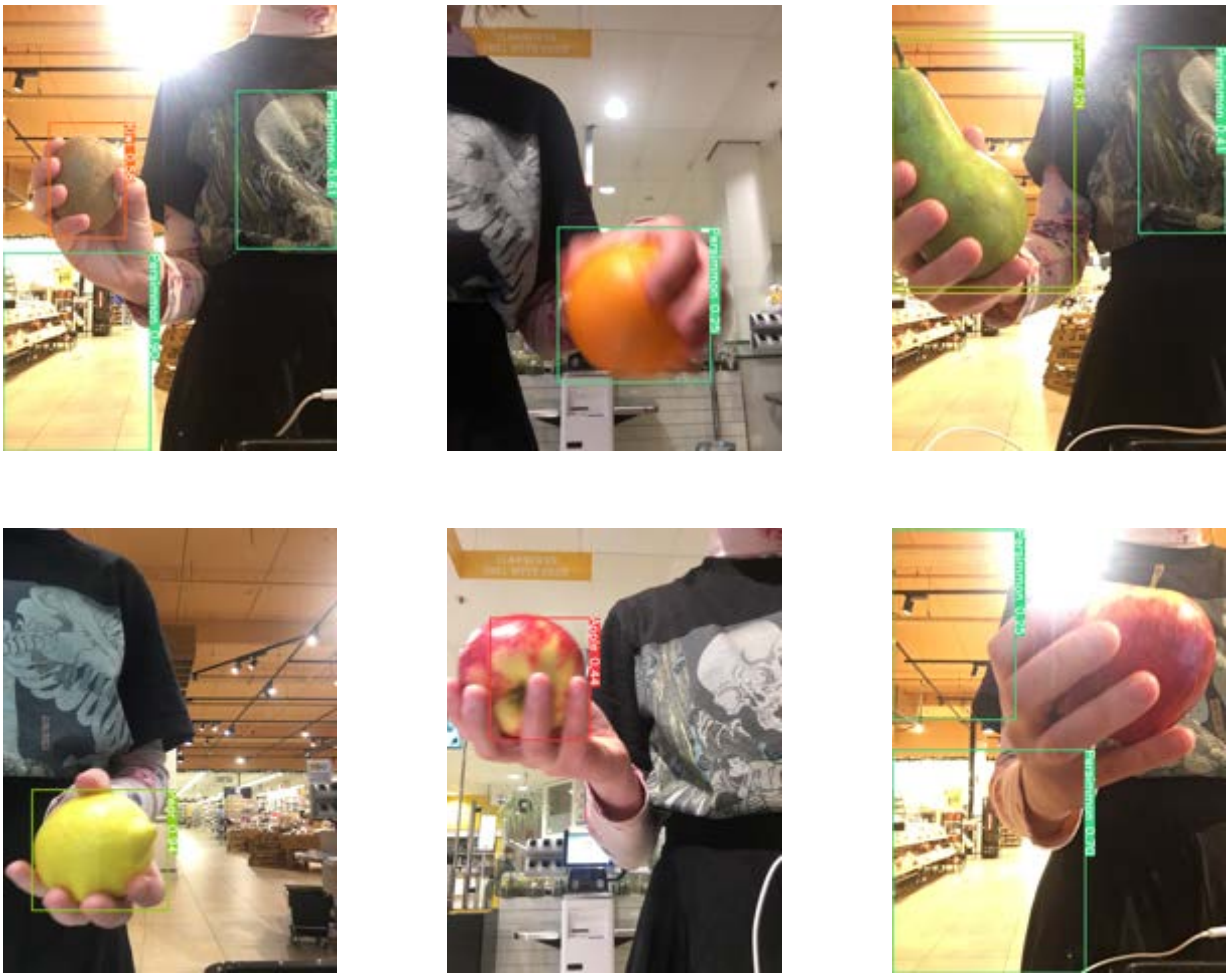


Figure 10.7: Trained YOLOV8 predictions on *hfruits-sm*

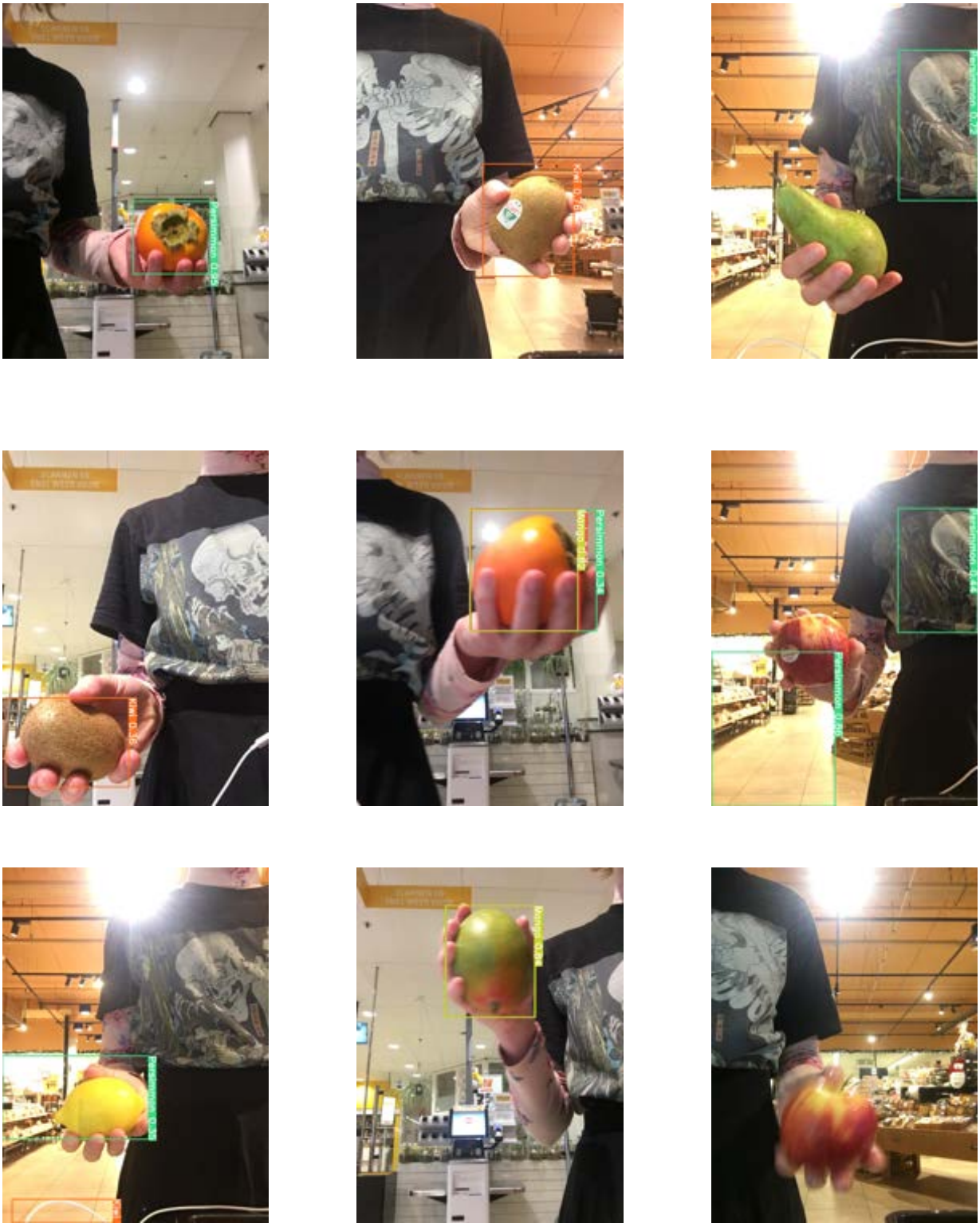


Figure 10.7: Trained YOLOV8 predictions on *hfruits-sm* (continued)

10.3.2 RT-DETR

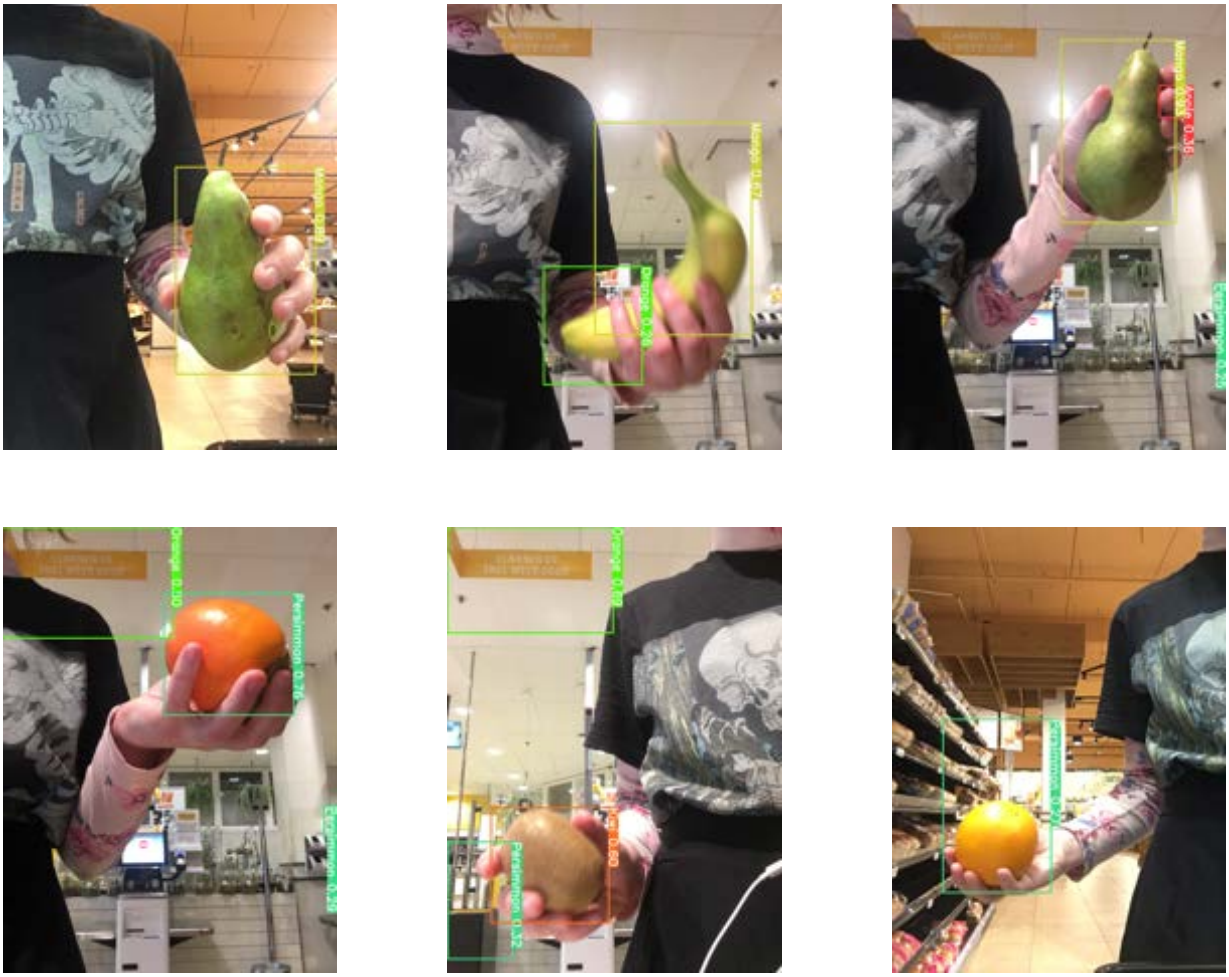


Figure 10.8: Trained RT-DETR predictions on *hfruits-sm*

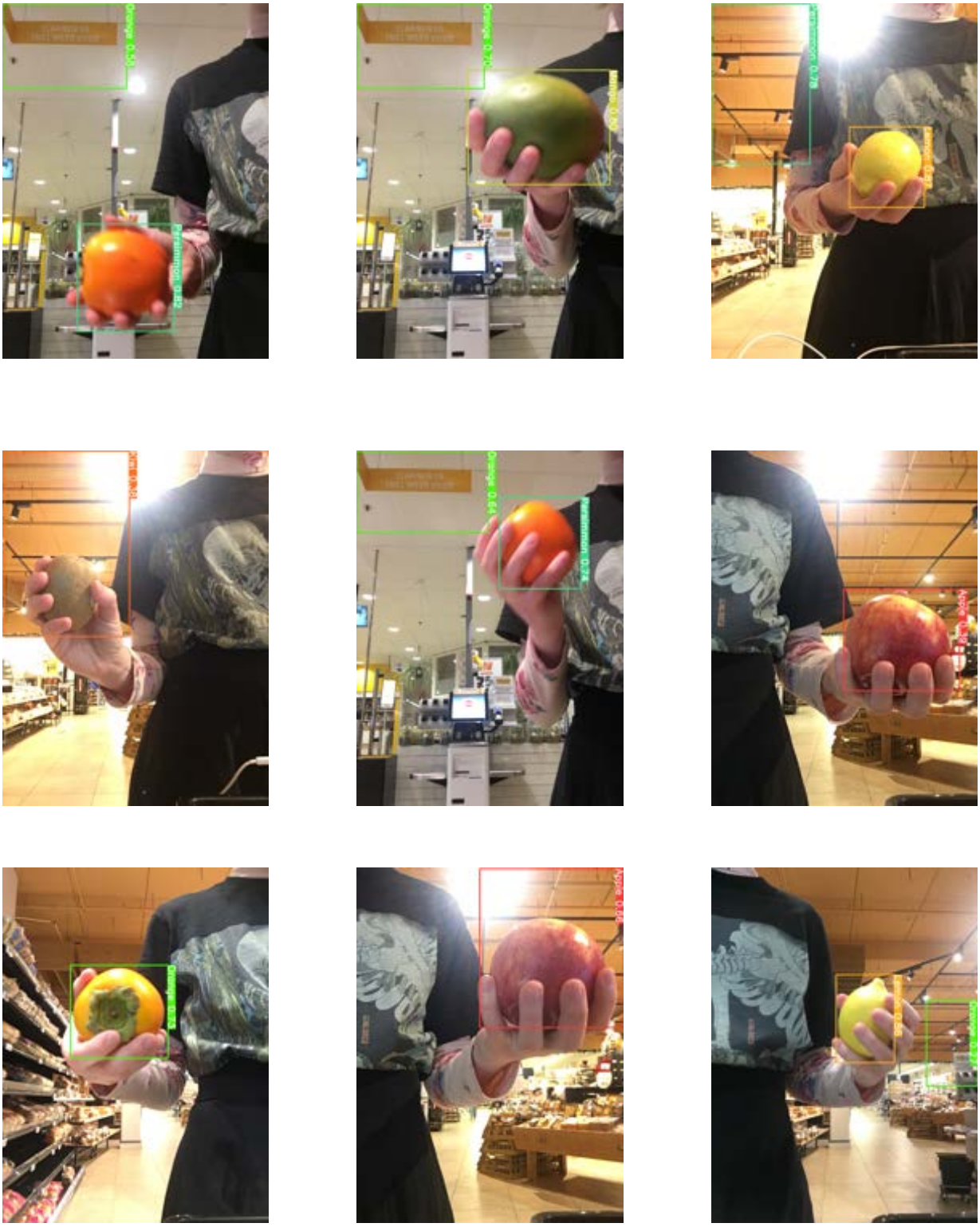


Figure 10.8: Trained RT-DETR predictions on *hfruits-sm* (continued)