AI-based machine vision for retail self-checkout system

by Anton Rigner

Master Thesis in Mechanical Engineering

Faculty of Engineering at Lund University

LU Supervisor: Dr. Carl Olsson

ETH Supervisor: Prof. Konrad Schindler

AIRS Supervisor: Alejandro Garcia

Abstract

In recent years advances in computing power, availability of large annotated datasets and AI algorithms have enabled the rise of reliable object identification and tracking. This thesis describes the development and implementation considerations of a system for object detection in the retail industry. This project have been conducted as a collaboration between ETH Zürich and start-up company AI Retailing Systems, who wants to automate parts of the retailing experience, namely the checkout procedure in a retail store. A data set of images with corresponding bounding boxes and pixel-segmentations has been gathered, consisting of ten Swiss retail products. Relevant theory is discussed and three state of the art neural network architectures are reviewed and evaluated for the specific application and dataset. The thesis concludes with a discussion of the main challenges for this type of solution, a recommendation for the object detection model to be used and pointers for future work.

Keywords: Object Detection, Retail, Automated Checkout

Contents

1 Introduction

In recent years advances in computing power, research in AI algorithms and the availability of large datasets have enabled the rise of reliable object identification and tracking. This thesis describes the development and prediction-model considerations of a object detection system for retail items. This project has been conducted as a collaboration between ETH Zürich and the start-up company AI Retailer Systems, who wants to automate parts of the retailing experience, namely the checkout procedure in a retail store.

This is to be achieved by tracking and identifying objects that the customer puts in their physical shopping cart, and then using this information to build a "virtual shopping cart", which would potentially remove the need for a cashier to scan the items and handle payments in a convenience store. Development steps and considerations, as well as model choice and data dependencies are discussed, concluding with a final recommendation for which object detection model should be deployed for the first prototype.

1.1 Background

1.1.1 Background on AI Retailer Systems

AI Retailer Systems is a start-up company active in the retail space, with the long-term mission to automate the retail experience. Automating the checkout procedure of the retail market is not a new thing, it has been around since the 80's [1] and has worked in various ways, with the self-scanning solution having been the most prominent one. Making the customer able to scan the products themselves removes the need for a cashier at every checkout-point. However, this requires the retail outlet to put trust into their customers, hoping that they actually scan all the products they are bringing out of the store.

Routine checks are performed to see if all the items were scanned, but this is not done for every customer as it would defeat the purpose of the increased convenience and reduced need of cashiers. This means that there is a risk of revenue loss from customers not scanning their items correctly. To combat this, solutions such as the Amazon Go project [2] have utilized various sensors to fully automate the check-out procedure without the need for scanning barcodes, instead utilizing cameras, scales and depth-sensors to understand what is happening in the store and detect when a customer is taking an item from the shelves.

Utilizing several types of sensors improves the accuracy of such a system, since more information about the environment is known. However, it also introduces a large cost, and such investments are a high barrier to already existing retailers operating on small margins and without the means of huge investments. Therefore this project together with AI Retailer Systems aims to develop a prototype utilizing only computer vision to identify the retail products that each customer is picking.

1.1.2 Computer Vision and Object Detection

In order to automate the checkout process the developed system needs to keep track on what products the customer is picking, without the use of human input (a cashier). Computer vision is the technology and research area where a computer is able to use images and other inputs to gain a higher level of understanding of the environment it is acting in, in a similar way as the human vision sense. Recent advances in computer vision algorithms based on statistical learning have made it possible to detect and classify objects in a given scene (image) and this thesis will discuss a few such algorithms and apply them to the given problem formulation for the retail space.

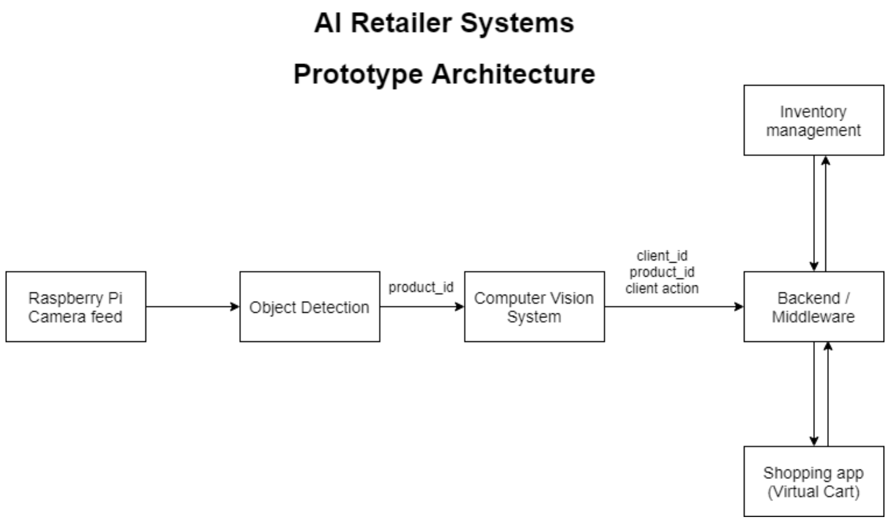

Figure 1: Diagram over the components of the prototype system

1.2 Aims and objectives

The primary deliverable for this thesis project is a working object detection model that is sufficiently accurate to showcase the prototype of an automated checkout solution similar to a vending machine. The model needs to be running in real-time in order to always be able to detect any products picked by the customer.

1.2.1 Model evaluation and comparison

To determine the best architecture and parameters for the object detection model, three well-known state of the art models will be trained with the gathered data set and evaluated on their performance. The performance indicators used are the mean average precision (mAP) discussed in Section 3.2.1, in the same format as the established COCO benchmark [3] for object detection, and runtime speed.

An empirical evaluation of conditions affecting the accuracy of the system has also been performed. The differing conditions include FPS, number of objects in the frame and brightness conditions.

1.2.2 Data gathering

Data in the form of images labelled with the correct descriptive information is required in order to utilize the computer vision models based on statistical learning. A data set consisting of 10 different retail products have been gathered and annotated.

1.3 Constraints and limitations

Developing a prototype for a fully autonomous shop is a very complex task, and is thus outside the scope of this project. Instead, a simplified version utilizing the same technology is to be developed as a proof of concept. The prototype can be thought as a more sophisticated vending machine, utilizing computer vision to detect the product the customer is picking. This brings added convenience to the customer, since she can freely retrieve and put back products from the shelf.

1.3.1 Availability of literature and intellectual property

The field of computer vision is an incredibly wide area expanding in an explosive manner, pushed forward both by industry research centres as well as academia. The reason for this is the rich range of applicability, everywhere from autonomous vehicles, smartphone applications, photo and video analysis, automation to military defence systems. Even if the complexity of these systems and time taken to develop them are very high the community has still remained relatively open, with lots of openly available state of the art research on the topic, as well as open source code available for many of the best performing methods.

The evaluated models in this thesis are licensed through the MIT-license. The MIT license permits reuse within proprietary software provided that all copies of the licensed software include a copy of the MIT License terms and the copyright notice [4].

1.3.2 Hardware

The computational unit consists of a desktop computer with a Nvidia RTX 2080 GPU, which is handling the heavy computations for the deep-learning models in real time. In the future this computation could be done in a distributed manner for each vending machine unit, or done in the cloud.

2 Theory

The contents of the thesis assumes that the reader has a decent understanding of deep learning-based computer vision systems and how they work, with some prior experience within object detection. As such, the most basic concepts within these fields will not be fully explained and left to the reader to realize. However, the thesis will discuss architectural considerations and related theory that distinguishes one model from another.

2.1 Object Detection

Object detection is the task to automatically predict the presence of certain objects in a given input image. The prediction should output the localization and object class of the detected objects. This information is usually embedded in either a bounding box or pixel-precise mask with coordinates in the image, together with the corresponding object class.

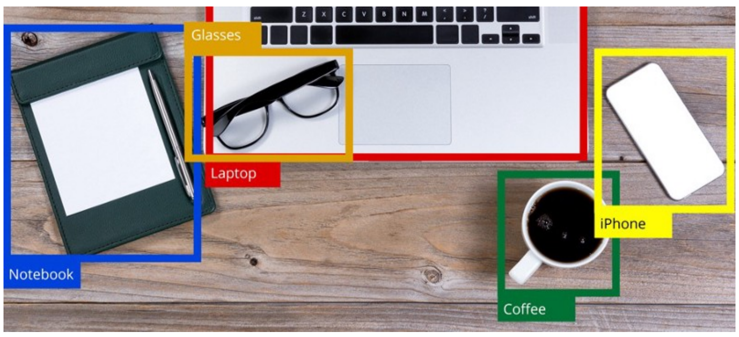

Figure 2: Example of object detection, outputting bounding boxes and classification for the objects found in the image. [Online]

Classic object detection algorithms rely on detecting key-points, edges and corners, known as features in an image, and combining a certain set of features to represent an object, for example a human face. The feature detectors are often constructed as convolutional filters which are applied to the image one wants to do object detection on. The convolutional kernels were often hand-built in order to extract the desired feature, which was a time-consuming task and was largely reliant on human intuition on which features would best represent a certain object.

2.2 Deep Learning for Computer Vision

Neural networks are not a new thing, and have been around since the 1970s. In recent years, very fast progress has been made in several computer vision tasks, stemming from an increase in computing power, the availability of large public annotated datasetes, and the development in deep neural networks, so called deep learning. AlexNet [5] sparked a revolution in computer vision, showing significantly more accurate results in the ImageNet [6] classification task, utilizing deep neural networks.

2.2.1 Supervised Learning

Deep neural networks are a type of supervised learning algorithm and need labeled data to train the model. Because of their big parameter (weight) space they require a large amount of training data to generalize well.

2.2.2 Transfer Learning

To decrease the size of the data set to be gathered, transfer learning can be used. Transfer learning is a technique where a model is pre-trained with data from other sources than the target environment you want to run the model on. By training the model with data that is readily available in large amounts, such as the COCO dataset [3], the model is tuned to detect low-level features and objects. Of course such a trained model is not able to detect the specific products we need to identify, but this can be solved by re-training the last few layers, the so called 'heads' of the network, responsible for high-level semantics and object classification. This way the weights trained on the original large COCO dataset are preserved, and are very good at detecting the low-level features such as edges, corners and colors that make up the higher-level understanding of objects in the later layers, which are now fine-tuned for our specific object classes.

2.3 State of the Art Models

As previously mentioned this is a field which moves very fast, but a few different architectures have continuously been used and improved on over several iterations. The object detection model in this project will be based on such a state of the art neural network architecture. All of the tested models use the ResNet50 backbone for feature extraction.

2.3.1 Faster R-CNN / Mask R-CNN

The Faster R-CNN architecture [7] for object detection have been developed over several years and iterations by UC Berkeley and Facebook AI Research (FAIR), led by Ross Girshick. R-CNN [8] stands for region-based convolutional network. The model takes an image as input and outputs detected objects in the form of bounding boxes (segmentation masks in the case of Mask R-CNN [9]), together with the class label for each detected object.

The region proposals of the R-CNN architecture are produced by utilizing Selective Search, which groups together pixels of similar color and texture. The proposed region is then fed through a CNN structure based on AlexNet [5] for feature detection and this is fed through a SVM for multiple classification. The SVM outputs a class label and a confidence level for the prediction. As a final step, the proposed bounding box is refined with the help of a regressor, with bounding box coordinates and the predicted class of the object as inputs.

This initial model architecture was subsequently improved upon by introducing RoI Pooling. Region of Interest pooling greatly reduces the amount of forward passes needed, by forward passing the complete image through the network and saving the different convolutions, which can then be reused for the proposed regions. The model was further improved on by integrating the different steps into one single model, Fast R-CNN [10], which makes it much more convenient to train, since all the steps of the model are optimized concurrently in one session.

The next iteration for this architecture came with the Faster R-CNN [7] paper, which paved a way to heavily reduce the cost of the region proposal step. Ren et. al showed that the interesting regions in a given image is dependent on the extracted features from the CNN layers. They introduce a region proposal network, with the feature maps computed from the CNN layers as input. This means only one CNN needs to be trained, and the cost of the region proposal step is greatly reduced.

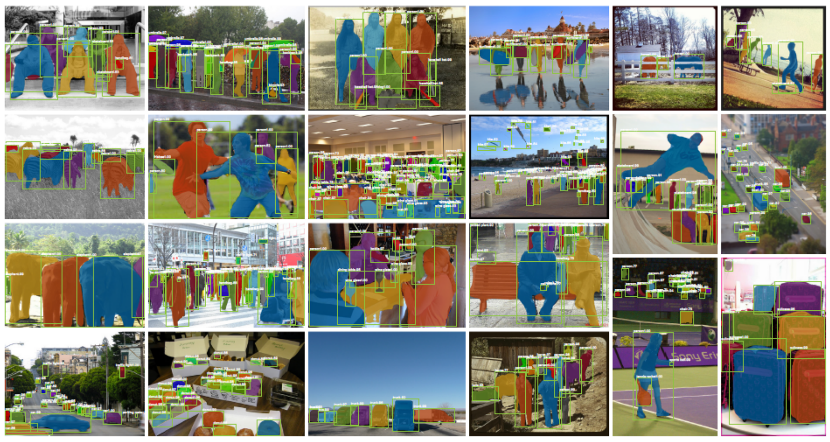

Figure 3: Example output from the Mask R-CNN model. Predicted masks with corresponding predicted class, for each instance of detected objects, overlayed on the original. [Mask R-CNN paper]

Finally, the Faster R-CNN architecture was further improved by enabling pixel-by-pixel segmentation masks. A parallel Fully Convolutional Network is added to the architecture, outputting a segmentation mask in binary form. Each pixel is classified as either a 1 or a 0, representing whether it belongs to a detected object or not.

2.3.2 YOLO

Unlike Faster R-CNN and Mask R-CNN, YOLO [11, 12] is a single-stage model. In the Faster R-CNN architecture regions of different sizes are first proposed by the RPN, based on the extracted features of the image, with the goal of proposing only viable and realistic boxes. In the second stage of the model, each of the candidate boxes are classified. Instead of proposing these boxes, a large number of boxes are always generated in a static way, without considering their viability.

This creates a class imbalance problem for the classification of the bounding boxes, since a large majority of the boxes are part of the background (image regions not belonging to any object class of interest) and thus the training of the models can become very biased towards this, while not actually learning the relevant non-background data, the objects to be detected. The model can have a low loss but still perform bad on the hard examples, which are what we actually want to detect.

This problem can be combated by for example online hard example mining, OHEM, which tries to balance the training by actively selecting the hard examples, in this case the non-background boxes corresponding to actual objects annotated in the training images.

2.3.3 RetinaNet

However, this approach only helps to an extent and single stage models usually have to limit the bounding boxes to ensure the model is still trained on the relevant hard examples. The main improvement with RetinaNet [13] is the introduction of Focal Loss, a loss function introduced to try to solve the problem with class imbalance.

$$\mathbf{FL}(p_t) = -(1 - p_t)^{\gamma} \log (p_t) \tag{2.1}$$

The focal loss function, Equation (2.1), achieves an adaptive total loss by weighting the easy examples and harder examples in a dynamic manner, prescribing higher weight to the harder examples, with a lower value of p_t, the probability of the prediction being the ground truth class. When p_t is high, for easy examples, the focal loss is close to zero and thus this does not contribute as much to the model training, but for hard examples with $p_t < 0.5$, the Focal Loss is significant and thus contributes to training a lot more. The parameter γ is called the focusing parameter and tunes how aggressive the focal loss disregards the easy examples. Setting the focusing parameter to 0 the focal loss simplifies to regular Cross Entropy loss.

2.4 Segmentation Mask or Bounding Box

Bounding boxes provide location and size of the detected object, whereas a segmentation mask label provides pixel accuracy of the object. The hypothesis is that this could aid the object detection, since it is trained on more precise information.

3 Methodology

This section describes the methods used for the data gathering process, how the object detection was evaluated in this paper as well as explanations on the experiments and the motivations as to why they were conducted.

3.1 Data gathering

The products to be used in the prototype was photographed by the same camera to be used for inference of the model, the Raspberry Pi camera. The Raspberry Pi camera was mounted in the same way as it is intended for inference, see Figure 4. An additional 50 images that were captured by a SLR camera at a higher resolution for previous development was also included in the dataset, with the reasoning that more data is better. Each object instance in every photo was labeled with pixel-accurate mask as well as the class of the product. The labelling tool VGG Image Annotator was used for labelling. The complete dataset consists of 486 images with an average of 3,8 objects per image. The mask labels can easily be converted to bounding boxes to train the models where the output is not instance segmentation, just bounding boxes. The dataset is stored in the COCO format in JSON files, and scripts for conversion to other data formats have been developed.

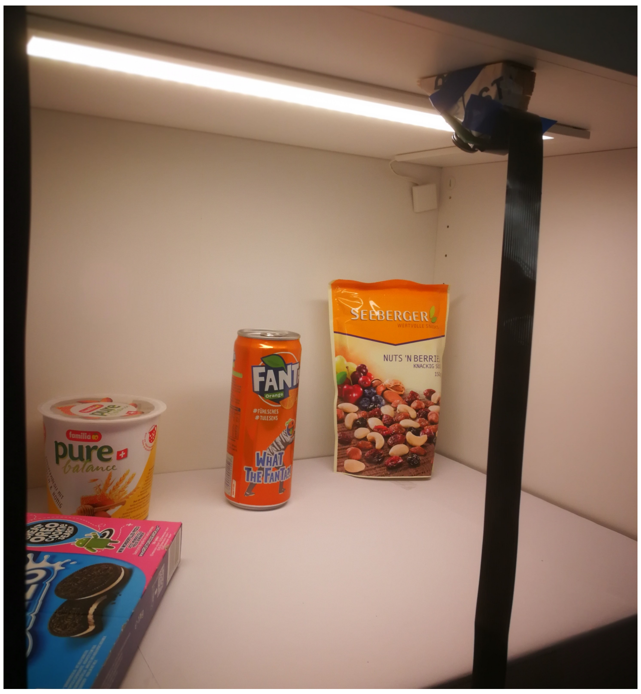

Figure 4: Setup for the data gathering process. A Raspberry Pi camera was mounted inside the shelf and a majority of the training data was captured this way, to mimic the run-time environment as close as possible.

3.1.1 Labelling

Each image in the data is labeled with the location, size and object class for all instances of objects in the corresponding image. The labels are in the form of pixel-by-pixel segmentation masks. The masks can be easily converted to bounding box labels, depending on the format of the model predictions. Examples from the dataset can be seen in Figure 5.

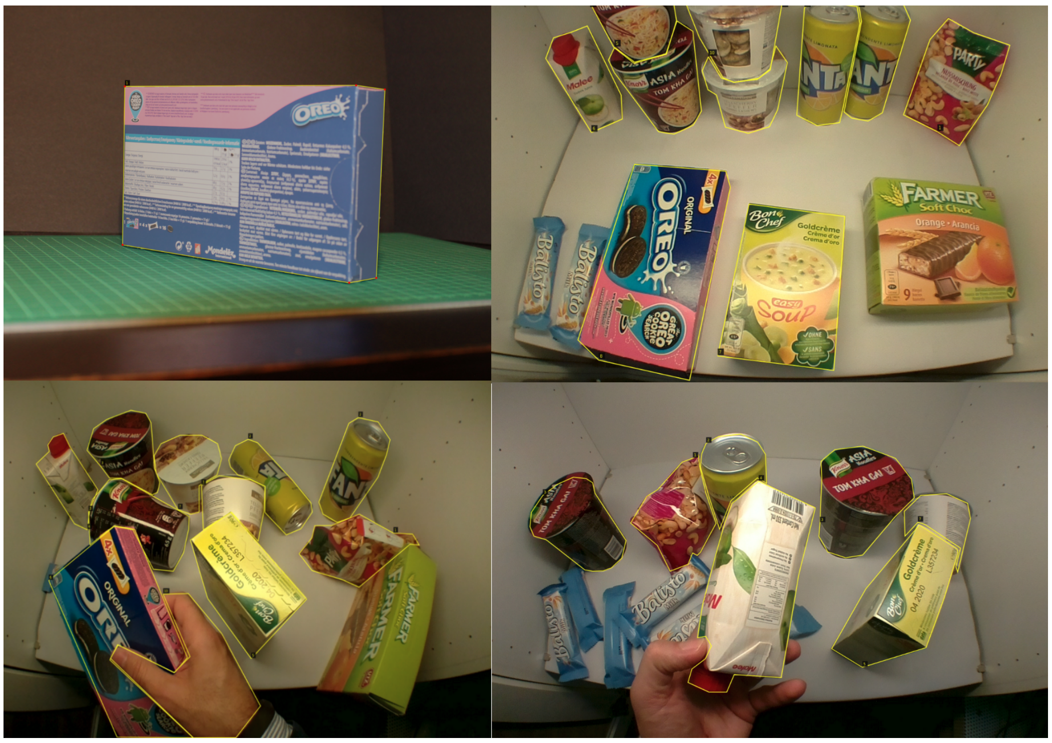
Figure 5: Example images of the labelled dataset. The outline of each mask can be seen in yellow. A majority of the images in the dataset contains several object intances.

3.1.2 Data augmentation

For the models used, more available training data generally leads to more accurate models. Because of this the dataset is augmented by adding modified copies of the images to the dataset. To decide what augmentations are effective an experiment was carried out, training a model on different data sets corresponding to he use of different augmentations. The augmentations that show to be effective in increasing the accuracy of the model will be used in the final prototype. The augmentations that were tested are the following operations:

- Scale (Resizing of image and annotations)
- Noise (Additive noise layered on top of the image)
- Brightness (up to 50% higher or lower pixel values)
- Shear (Distorted by shearing)
- Mirror (Mirrored in the horizontal and vertical plane)

The Python library *imgaug* [14] was used for the augmentations.

3.2 Evaluation metrics and benchmarking

There are several ways to evaluate an object detection model, depending entirely on the constraints and requirements of the application. In general, an accurate model is of course always better, but it will always be a compromise between accuracy and complexity, meaning accurate models will usually be more computationally heavy. With this in mind, the models will be evaluated both on accuracy, robustness, as well as run-time speed when running the model online.

3.2.1 Mean average precision (mAP)

Precision is a measure of how many of the total predicted positives are actually true positives. Average precision for an object class, AP, is a measure of the precision for all occurrences of a certain class in the test dataset. The mean average precision, mAP, is the mean AP for all object classes in the model.

3.2.2 Run-time

Run-time is a somewhat lesser discussed topic in object detection research, where accuracy of the model is the most important benchmark. In real world applications, however, the run-time of each model is crucial. The model needs to be able to run online, in a robust manner and not susceptible to customers trying to trick the system.

This does not necessarily mean that the run-time requirements are as high as in the autonomous vehicle industry, for example, where typical cameras are running at a FPS of at least 30 [15]. Thankfully a slightly delayed reaction or slow response does not mean life or death in a retail setting, but it is of course important that the model is robust and that the system is responsive enough.

3.3 FPS requirement

To find the approximate required FPS for our system, a number of different FPS levels were analyzed, running offline. The video snippets used for this experiment had sequences of items being removed and added in a manner where the speed of the action was varied, with examples of very fast actions intended to try and trick the system. The goal is to be able to detect the hand when it reaches in the cabinet. This is used as a measurement on how well the system performs, since this is a logical way to detect when a customer has picked up an object. It should be noted that if using a simplified way of keeping track of the number of products in the shelf, such as just observing the product count directly in each frame would negate the need of a high FPS even more. However, this would mean that the system would need complete vision of all the products on the shelf at once, something that might not be very practical for an actual application depending on camera position and the number of products on the shelf.

3.4 Model optimization

Using an established architecture based on the previously mentioned state of the art models is beneficial, since they are already proven to work well on the generalized benchmarks used in research. However the models are exactly that, very general, and to tune them to our specific application the architecture and the model parameters can be optimized to our needs and constraints.

4 Results

This section contains the results obtained from the different experiments conducted in the development process of the prototype. Results on the impact of data augmentation on the original dataset are reported with regards to different augmentation operations, the FPS requirement for the system was evaluated with the help of a FPS test and the result of the benchmarking of the three models are reported. The benchmark results consists of the accuracy, run-time and robustness to brightness variations.

Accuracy of the three state of the art models discussed in Section 2.3 are presented, measured by the mean average precision across all ten classes of the model. Results are reported in three columns, following the benchmark convention of the COCO challenge [3]. Instead of using just the standard precision at an intersection over union, IoU, of 50%, the COCO benchmark reports precision of the models as an averaged mAP over 10 different IoU thresholds of .50:.05:.95. It also reports the conventional precision metric mAP$^{IoU=50}$ as well as mAP$^{IoU=75}$. The intention is to better differentiate how good the different models are at localization.

4.1 Data augmentation test

The results to determine the impact of different augmentation methods are listed in Table 1. The different data sets were tested on Mask R-CNN at a resolution of 1024x720 pixels. It is assumed that the different augmentations would yield similar relative results for the other two models.

The different augmentations were applied in the following way:

- Scale - Resize each image and the corresponding annotations to something between 50 and 150% of its original size. (*imgaug.Resize*)
- Noise - Add gaussian noise (white noise) to an image, sampled once per pixel from a normal distribution. (*imgaug.AdditiveGaussianNoise*)
- Brightness - Multiply all pixels in an image with 0.5 to 1.5, thereby making the image darker or brighter. (*imgaug.Multiply*)
- Shear - Shear images and annotations by -20 to 20 degrees. (*imgaug.Affine(shear)*)
- Mirror - Mirrored images and annotations in the vertical and horizontal plane (*imgaug.Fliplr* and *imgaug.Flipud*)

Augmentation	mAP	mAP50	mAP75
Raw data set	39,3	69,3	56,1
Scale + raw	42,1	70,2	58,6
Noise + raw	44,8	72,9	60,1
Brightness + raw	41,2	72,7	59,0
Shear + raw	37,1	66,2	53,9
Mirror + raw	51,9	73,5	61,8
raw + augmentations (not shear)	60,1	86,5	70,0

Table 1: Results for a Mask R-CNN model trained on different data sets, including the raw collected data sets and various types of data augmentations.

Four out of the five tested augmentation operations yields improved results compared to the raw dataset. The *shear* operation was the only operation with negative impact, reducing the mAP with 2,2 percentage points. Including all four of the positive operations yielded an increase in mAP accuracy of 20,8 percentage points, which is a significant improvement.

4.2 FPS test

The results of the FPS test described in Section 3.3 are reported below in Table 2. The least accurate model (YOLO), was used in order to ensure that the resulting FPS requirement from this test is a worst case scenario. The reported percentages are the rate of detections for the *hand* class. The higher the rate of detection for the *hand* class, the better. The accuracy for the other classes was not evaluated in this experiment, since sensing the hand is seen as critical in the task of understanding if a customer has picked up an object or not. Input videos consist of reaching in and out of the shelf at various speeds.

FPS	Rate of hand detection (%)
2	74,8
5	93,4
10	95,7
15	96,1
30	96,3
60	96,3

Table 2: Rate of detections for the 'hand' class. Input videos of reaching in and out of the shelf at various speeds.

As expected, a very low FPS leads to a system unable to detect all the movements in and out of the shelf. Accuracies quickly taper out above 5 FPS, and anything above 10 seems to yield negligible results.

4.3 Model benchmark results

The benchmark conducted was based on the metrics discussed in Section 3.2, with the goals of measuring the accuracy and run-time of the tested models. The results of the benchmark are listed in the tables below. All the models were pre-trained on the COCO dataset, and utilizing transfer learning, the last few layers were trained and evaluated on the gathered dataset specific to our application. This dataset consisted of the complete training set with the added data from the four selected augmentations, meaning the total amount of training images added up to 2430 with an average of 3,8 object instances per image.

4.3.1 Accuracy

The accuracy of the three tested models are reported in Table 3-6 below, with the same test dataset for four different resolutions. The results of the models are given as percentages, in the COCO format, discussed in Section 4.1.

Model (1920x1080)	mAP	mAP50	mAP75
Mask R-CNN	72,3	93,1	79.3
RetinaNet	71,8	92,6	80,2
YOLOv3	59,9	82,2	70,0

Table 3: The three reported precision metrics for the input image resolution of 1920x1080.

Using the native image resolution of 1920x1080, Mask R-CNN achieves the highest accuracy, with a mAP of 72,3% as per Table 3. RetinaNet achieves similar results, while also surpassing Mask R-CNN on the mAP$^{IoU=75}$ metric. The accuracy of YOLO is around 10 percentage points lower on all three metrics.

Model (1024x720)	mAP	mAP50	mAP75
Mask R-CNN	60,1	86,5	70,0
RetinaNet	59,1	85,9	71,3
YOLOv3	53,2	74,4	66,3

Table 4: The three reported precision metrics for the input image resolution of 1024x720.

Model (800x600)	mAP	mAP50	mAP75
Mask R-CNN	37,4	58,9	42,4
RetinaNet	38,1	54,4	40,1
YOLOv3	32,9	52,2	39,0

Table 5: The three reported precision metrics for the input image resolution of 800x600.

The accuracies of the models seems to be affected roughly to the same degree when the input resolution is down-scaled to 1024x720 and 800x600, as can be seen in Table 4 and 5 respectively.

Model (600x400)	mAP	mAP50	mAP75
Mask R-CNN	23,2	56,9	47,3
RetinaNet	12,3	38,8	26,4
YOLOv3	21,2	31,2	28,0

Table 6: The three reported precision metrics for the input image resolution of 600x400.

For the lowest resolution of 600x400, the Mask R-CNN model stays around the same accuracy as for 800x600, whereas the RetinaNet model sees a large drop in all three metrics, and especially the averaged mAP value, from 38,1% to 12,3%, as can be seen in Table 5 and 6. This implies that the RetinaNet model is very inaccurate in terms of localization (large IoU thresholds which means high demands for localization) for lower resolution images.

4.3.2 Failure cases

Typical failure cases for all the models are images with a lot of overlap between the objects (see Figure 6), very dark images, distinction between two very similar products, images with a lot of glare and cases where an object was held very close to the camera, covering large parts of the frame. Handling occlusions and overlap between objects is an inherently hard problem, but some of the failure cases could possibly be avoided with a larger dataset, covering all the orientations of the products. When demonstrating the prototype live there was also some issues with reflections in the glass of the shelf, something that was not anticipated since the training data was gathered with the shelf door open at all times. This could be easily avoided by changing the camera angle.

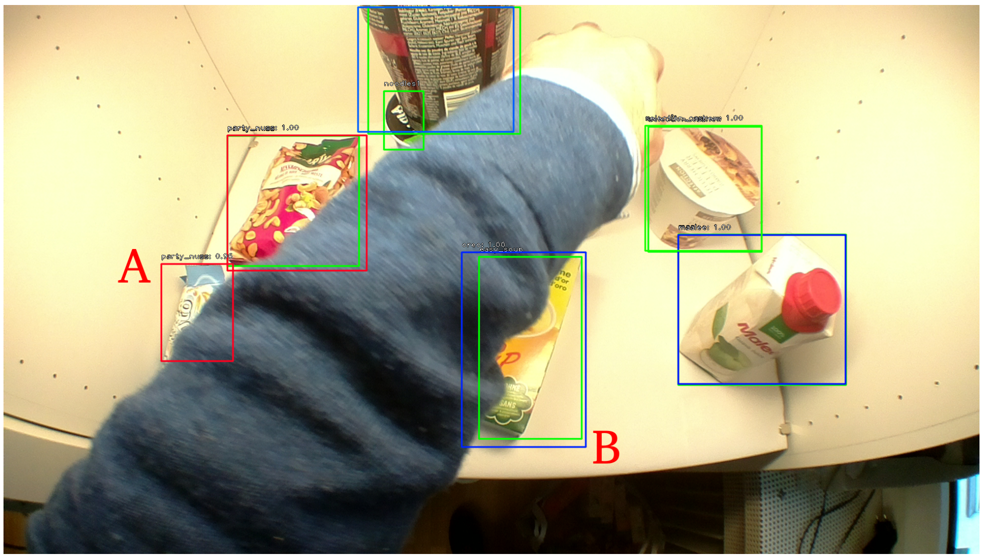

Figure 6: Example failure case for the RetinaNet model, box A is falsely classified and localized (this specific product is not included in the training dataset). Box B is correctly localized but falsely classified, probably due to the arm covering large parts of the product.

4.3.3 Run-time

The run-time for each model is tested on several different image resolutions, on a test set with scenes of different complexity (1 to 10 object instances per image). The results are reported in milliseconds (ms).

Model	1920x1080	1024x720	800x600	600x400
Mask R-CNN	590	343	284	231
RetinaNet	257	199	176	110
YOLOv3	127	83	71	68

Table 7: Runtime in milliseconds for each model, for 4 different resolutions.

4.4 Brightness variations

All three models were tested with regard to robustness against brightness levels. All three models was trained on the previously mentioned dataset including the augmented images. The brightness of the test set was varied between 50 to 150% of the base case, and the input resolution for the test images was 1024x720. The accuracy, here reported as the averaged mAP metric described in Section 4.1, is reported in Table 8 below.

Brightness (%) →	50	75	100	125	150
Model ↓	Accuracy (mAP)				
Mask R-CNN	37,4	55,9	60,1	58,3	47,8
RetinaNet	39,5	56,0	59,1	57,8	47,9
YOLOv3	32,9	50,2	53,2	52,5	44,2

Table 8: Accuracy for the different models at different brightness settings (% of 'optimal' light conditions used for training).

The results of the models are in line with the general mAP evaluations in Section 4.3.1, and no model seem to handle brightness variations exceptionally worse or better than the others.

5 Discussion, Conclusions, Recommendations

5.1 Data augmentation

The results in Table 1 show that extending the training data by the means of data augmentation greatly increases the overall accuracy of the models without much added effort. All methods except shearing the images helps the model generalize and results in more accurate predictions. Utilizing the complete dataset with all augmentations yields an increase in accuracy of over 17 percentage points for the classic mAP$^{IoU=50}$ metric and over 20 percentage points for the mAP averaged over 10 IoU thresholds. This implies that the augmentations are of even greater importance for the strictest IoU thresholds. The results are comparable with similar works such as the MVTec D2S Segmented Supermarket Dataset benchmark [16]. It can be argued that obtaining extra training data by means of augmentation was extra important seeing as the available dataset were relatively small, containing 486 images in total while the D2S dataset contains 4380 images. The D2S dataset also avoids including complex scenes in the training data, requiring the training images to "*only contain objects with no or marginal overlap without clutter, and a homogeneous background*", instead trusting the models to generalize well from simpler training data to complex scenes with occlusions and multiple stacked objects. In our case, including complex scenes in the training data yielded results comparable to [16], but it should be noted with much fewer classes to distinguish between in our case.

5.2 FPS requirements

Unlike other areas that rely heavily on machine vision to sense its environment, having a high and smooth FPS is not as critical for the AIRS prototype. For example, slow processing of the captured images or missing a few frames can be devastating for an autonomous vehicle relying on such systems detecting potential obstacles or dangers in its path. However, even if the consequences of a slow system is not as dire in this case, to ensure that the system can consistently register changes in the product count of the shelf, the computer vision system needs to be able to detect changes even in a fast-changing environment.

The experiments shows that a FPS of 5-10 is enough in principle, after 10 FPS the returns are very much diminishable. A higher FPS could still be preferable for responsiveness of the all-around system, i.e the customer should not have to wait for a long time to see the update in their virtual cart on the mobile application when a product is taken or put back on the shelf.

5.3 Model benchmarking

5.3.1 Mask R-CNN

Mask R-CNN provides the most accurate predictions on the test dataset and is more robust to downscaled input images, outperforming the other models by a bigger margin for low resolution images. However, it is also the slowest model of them all, and the online performance is heavily dependent on the amount of objects detected in the frame. With 0-4 objects in the frame the model is able to run at 5 FPS on a Nvidia 2080, with the input video stream resolution of 1024 by 800 pixels. However, when more objects are present in the frame, as in the case of our prototype with 8-14 objects on a single level of the shelf, the FPS drops down to 2-3 FPS, which is generally too slow to provide a robust prediction of when a customer picks an item, as shown in the previous FPS test, see Figure 2.

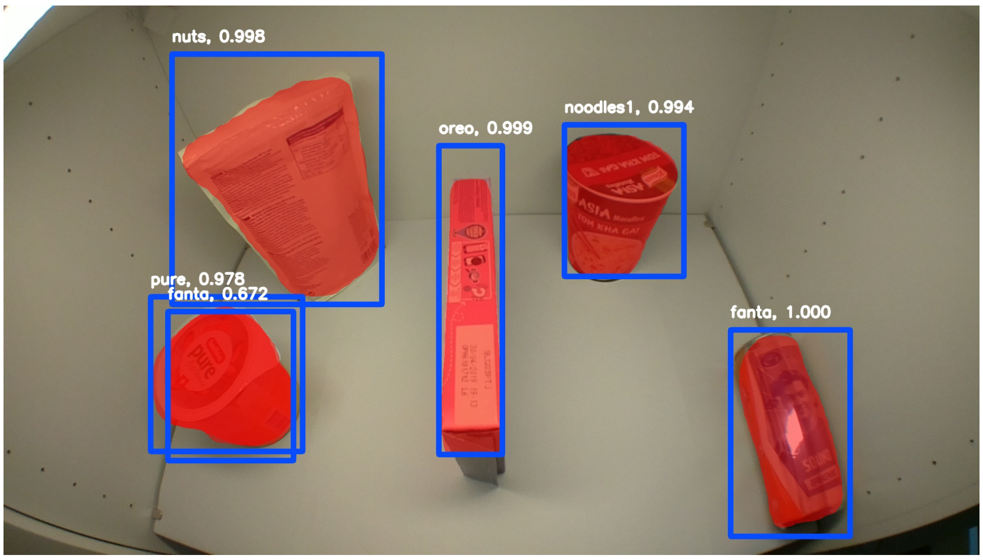

Figure 7: Example of Mask R-CNN inference output on the test dataset. Pixel-by-pixel segmentation masks as well as corresponding object classes are output on top of the input image. This is an example of a failure, where the model detected two objects on top of each other in the lower left corner.

5.3.2 RetinaNet

The RetinaNet model provides good accuracy, even surpassing Mask R-CNN for the IoU threshold of 75%. It also seems to be slightly more robust in lower light conditions, as per Table 8. The run-time of the model is higher than YOLO but significantly lower than Mask R-CNN while still maintaining good accuracy. However it should also be noted that the accuracy seem to fall off at a faster rate when input images are downscaled, comparing tables 3 to 6, and this should be taken into consideration for the final model recommendation. One of the challenges with the architecture was the inference predicting multiple overlapping bounding boxes. This can be valid if there are occlusions in the scene, for example when a customer is holding a product, but the model should not output two different classes for a single object instance, see figure 8. To combat this the original paper uses a non-max suppression threshold of 0,5 [13], but in our case this still resulted in multiple boxes, maybe because the bounding box regressor responsible for tightening the boxes was not trained to a high enough level.

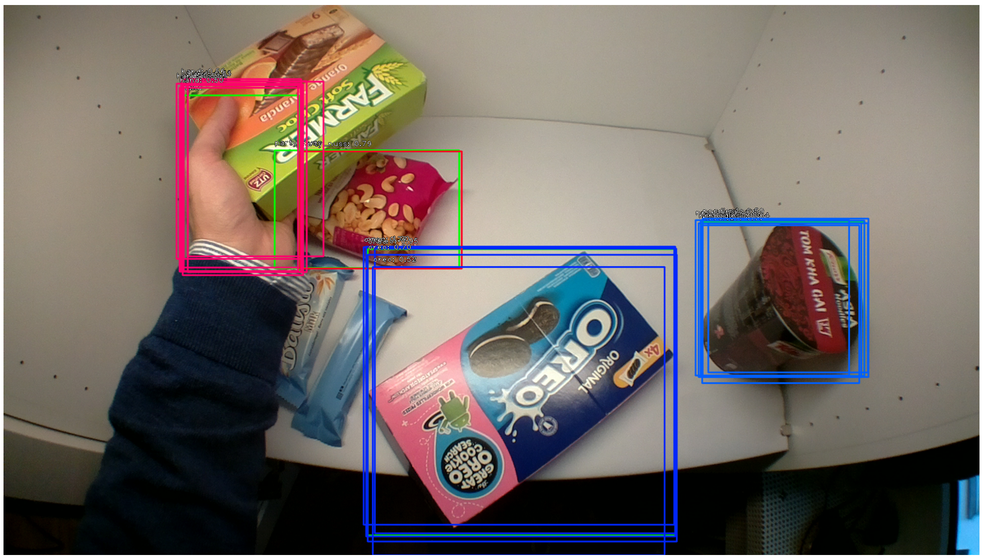

Figure 8: Example from RetinaNet inference, showing multiple bounding boxes for each object instance (undesirable).

5.3.3 YOLO

The main strength of the YOLO architecture is the good run-time. YOLO achieves the highest FPS of the three models for all tested resolutions. It is significantly faster than Mask R-CNN, and runs at more than double the FPS of RetinaNet for all tested resolutions but one. However the downside is that the achieved accuracy on our dataset is not on par with the other models, and the precision is at least 7-20 percentage points lower than the best model, depending on resolution.

5.4 Model optimizations

5.4.1 Anchor scaling

Our data set consists of products at approximately the same distance from the camera in every case, since the environment in the shelf is static with regards to camera position and the shelf level. Because of this, the objects in our data set is of approximately the same size, and we do not need to account for very large or very small objects. This knowledge can be used to simplify and optimize the model by making sure it proposes regions with a reasonable scale. This can be achieved by changing the scale of the anchor boxes used for the region proposals.

5.4.2 Image size

Another important factor for model performance is the size of the input images. A bigger image will retain more detail, but as a result the model will need to perform a lot more computations. Run-time is of course determined by the amount of computations needed for a forward pass, so this is a trade-off that needs to be explored, and several image resolutions are tested. The Raspberry Pi camera captures frames natively in the 1920x1080 format, but can be easily rescaled. For this specific prototype, the image re-scaling is done locally on the Raspberry Pi, which means this task is not something that will affect the run-time performance of the system.

5.4.3 Non-max suppression

As shown in Figure 3 and 8, both Mask R-CNN and RetinaNet had problems with outputting multiple bounding boxes for the same object instance. This is not desirable for obvious reasons, and to combat this non-max suppression is used to only include the bounding box with the highest predicted validity score. The non-max suppression threshold was tuned in order to still allow multiple bounding boxes in the cases of occlusions, but still avoid the case of two or more boxes over a single object. The threshold disregards the box with a lower score, if they overlap above a certain value. Figure 9 below shows the result after correctly tuning the non-max suppression threshold.

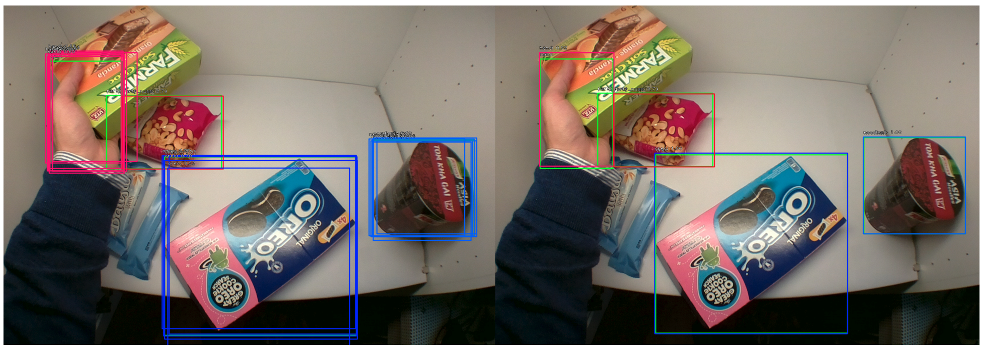

Figure 9: NMS solves the issue with multiple bounding boxes for a single object instance. Left is before proper NMS threshold is applied, right is after.

5.5 Handling occlusions

The system keeping track of the product count in each category is responsible for updating the back-end and in succession the virtual shopping cart in the mobile application on a customer's phone. A big challenge for the prototype is to distinguish between occluded objects and an actual removal of an item from the shelf. The system needs to be robust, and it would not be acceptable to bill a customer for a product she did not actually remove from the machine, that the computer vision system falsely detected as a product removed.

The evaluated models are able to inherently handle occlusions to an extent, since the complete object does not need to be visible at all times in order for it to be detected. However, if a large part of the object is covered the system will lose track of the object. To combat this, a system to try to distinguish the two cases was developed. Since the models are trained on human hands, the system is able to detect when a customer reaches in to grab a product and detects her hand. Every time the system detects a change in the product count, the intersection over union, IoU, for the bounding box for the hand and the detected product are calculated. This measurement gives an indication for whether a customer was actively interacting with an object at the time it can no longer be detected. If this is the case, we conclude that the customer actually removed the object. If we lose track of the object without having a strong indication for the customer interacting with it, i.e a low IoU score for the hand and the object, we conclude that we lost track of it because of an occlusion, and thus do not add it to the bill of the customer.

This idea worked quite well in principle, but it was hard to find the correct threshold to determine if the user actually grabbed a product or not. Especially smaller products which can be completely grasped by the hand

and thus not detected at all, turned out to be tricky. For bigger products they were always detectable even when held, and thus the IoU score was consistent. Because of these reasons, it was found to not be robust enough on its own, but this method together with other logic could potentially be used to handle occlusions.

5.6 Trade-off: Speed vs. Accuracy

As expected, the more accurate model is also slower. This is an important trade-off, since minimizing the computing power for the system is crucial, while still maintaining an acceptable accuracy. The current stock losses for retailers in Germany, a comparable market to Switzerland, is approximately 1,2% [17, 18], so the overall accuracy of the model should be close to 98% if it wants to lower the losses. One has to take into account that the customer is able to add or subtract products that they deem a mistake by the system, again giving the customer the benefit of the doubt as in current self-scanning solutions available.

5.7 Sources of errors

The results from the benchmark are realistic and in line with works in the same area [16]. Possible sources of errors are mis-labeled images, class imbalance in the dataset, though RetinaNet should tackle this issue especially well with its focal loss [13]. Overfitting [19] can always be an issue with deep learning models, where the model memorizes the training data instead of learning a general mapping that works outside of the training dataset as well. This is combated with reguralizations of the weights, validation sets and early stopping in all of the tested models, but it can still be a factor.

5.8 Integration with prototype system

A Raspberry Pi was mounted in the prototype shelf together with a Raspberry Pi camera. The camera is able to capture video at 1920 by 1080 at 60fps. This video stream is then fed to the computer vision system over a websocket connection. The PC handling the object detection model is also acting like a webserver with one or more Raspberry Pi's as clients, sending continuous video over a WiFi connection. The computer vision system is responsible for the object detection, but this information needs to be provided to the rest of the system logic. A backend in Java with a connected MongoDB database is handling the state of the system. A frontend application is provided to the customer as well as the manager or stocking personnel of the product shelf. The computer vision system, the front end interface and the backend system is able to communicate with each other through a RESTful API. The complete prototype can be seen in figure 10, setup in the Zürich central station to showcase the idea and gather feedback from potential investors as well as end customers.

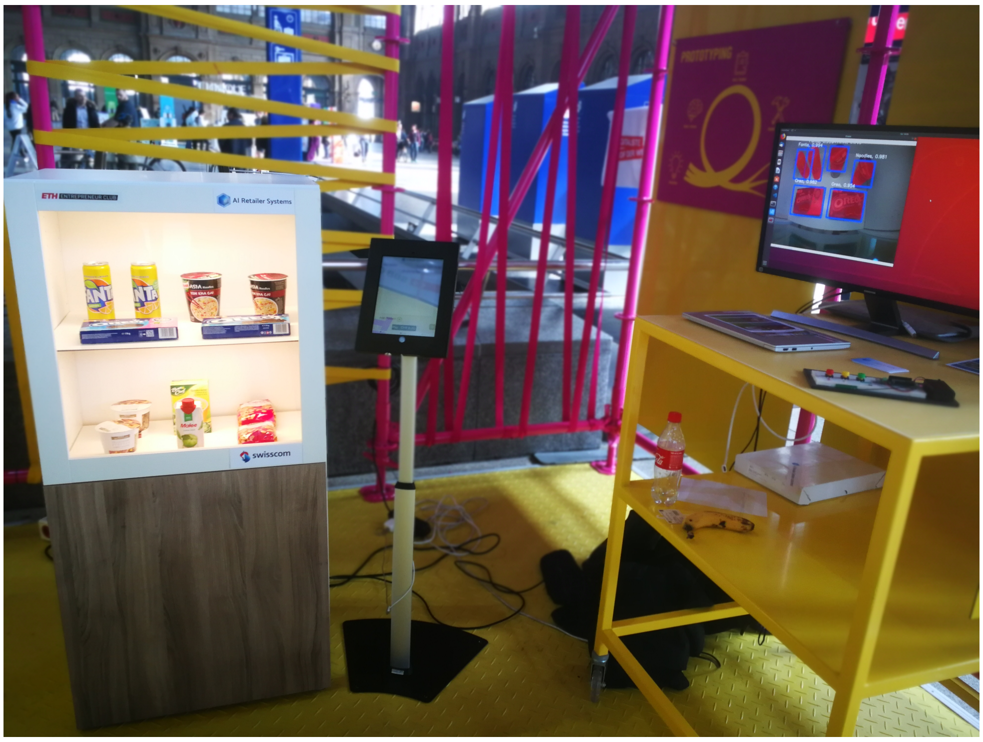

Figure 10: The complete prototype with the shelf with products, the front-end application running on the iPad in the stand as well as a visualization of the model running live on the big PC screen.

The logic for computer vision system contains a hysteresis with regards to the product count, in order to be more robust against flickering detections between frames. Only when there has been a consistent change over several frames in a row is it reported as a change in product count.

5.9 Recommendation for model to be used

Even if Mask R-CNN provides a slightly more accurate prediction, the computational complexity is high and leads to a slow performing model when run online. Having a high FPS is not critical, as previously discussed, but it means the prototype would be harder to scale up to several modules. If the run-time is faster than needed, the computations from several camera streams could be done on one server / GPU. These considerations together with the extremely time-consuming task of labeling pixel-by-pixel segmentation masks, which can take up to 10 times as long as for regular bounding boxes annotations, means the Mask R-CNN model is not optimal for this type of prototype and product.

RetinaNet seem to offer better accuracy, but with added computation time. The achieved FPS is still within acceptable boundaries but if an optimized YOLO architecture can yield similar accuracy results, it would be a clear winner. AIRS is therefore recommended to look more closely into the YOLO architecture as well as other similar models such as SSD [20]. For the time being they are recommended to use the RetinaNet object detection architecture for the next step of the prototype, as it provides a solid baseline with a good balance between good accuracy and acceptable run-time. An input resolution of 1024x720 or 1920x1080 seem to be necessary to reach an acceptable accuracy, which is higher than what most networks use in other models. The model is robust enough to showcase the idea, but the system as a whole have a lot of optimizations to be done before it would be commercially viable.

The hypothesis of complete segmentation mask aiding in the distinction between very small stacked products does not seem to hold in a significant manner, since RetinaNet with its bounding box annotations achieved an accuracy very close to the one of Mask R-CNN with complete segmentation masks.

5.10 Future work and next steps

The results from this work are promising, but there are still a lot of things to be done and to optimize before a commercialization of the prototype is possible. The requirements on accuracy are high, since the security of the system is lost if the system has to let the customer adjust the products picked themselves. If the customer is allowed to add or remove products by themself in the front end application, the convenience of the system is lost. At that point the solution reduces to a self-scan solution with a lot higher implementation costs, which is not very useful. Future research into single shot models such as optimizing a YOLO model or looking into SSD would be beneficial. Another question that remains unanswered is why RetinaNet and YOLO dropped off so much in accuracy for the tests with lower input images.

5.10.1 Other object detection models

This work has focused on established state of the art models on well known optimized for object detection benchmarks, but this is a field that is moving at a very fast pace, with extremely high research interest from both academia as well as large corporations. Because of this there are already interesting works to look into, such as [1] and [2]. Works that optimize for the specific problems in this application such as the Fribourg Grocery Dataset [21] and the recent work on Data Priming for Automatic Checkout [22] that optimizes the training process and data augmentation steps are also of high interest.

Further validation on the recommended model is needed, with regards to accuracy on a dataset with more classes, further validation on the failure cases of the model as well as runtime optimizations to reduce the computational resources needed. Finally, the need of more labeled data is a requirement, especially for models with more classes and higher complexity scenes.

6 Conclusions

A prototype to showcase the use case of deep learned-based object detection in the retail space have been successfully developed and showcased to potential investors and customers. The use of state of the art object detection models optimized for this specific use case have shown to be accurate enough to show the general concept, but still needs further development and optimization to be ready for commercialization. The project implemented a complete solution, from data gathering, annotation, image processing, hardware solutions and front end interfaces facing the end-customer. Models providing full segmentation masks did not provide significantly better results than the bounding box models and the best out of the three tested models is RetinaNet.

The goal of purely vision based automation of the retail checkout process is deemed to be a challenging problem, since even the most accurate models do not provide accuracies close to 100%, which a customer and retail outlets would expect in order to implement this system in a real environment. As always the case with supervised learning, this project again highlights the need of a lot of annotated data, an area where data augmentation proves to be a key tool to extend the dataset. There are a lot of future work possible in object detection models, data augmentation techniques and optimizations for the retail sector.

Figure 11: Prototype showcase in the Zürich HB (central station), the team behind the prototype.

References

[1] TrigoVision. *Evolution of the retail checkout experience.* URL: https://www.trigovision.com/evolution-of-the-retail-checkout-experience/ (visited on 04/07/2019).

[2] Amazon Website. *Amazon Go".* URL: https://www.amazon.com/b?ie=UTF8&node=16008589011 (visited on 06/07/2019).

[3] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *CoRR* abs/1405.0312 (2014). arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312.

[4] Open Source Initiative. *MIT License.* URL: https://opensource.org/licenses/MIT (visited on 04/07/2019).

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25.* Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[6] J. Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09.* 2009.

[7] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *CoRR* abs/1506.01497 (2015). arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497.

[8] Ross B. Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *CoRR* abs/1311.2524 (2013). arXiv: 1311.2524. URL: http://arxiv.org/abs/1311.2524.

[9] Kaiming He et al. "Mask R-CNN". In: *CoRR* abs/1703.06870 (2017). arXiv: 1703.06870. URL: http://arxiv.org/abs/1703.06870.

[10] Ross B. Girshick. "Fast R-CNN". In: *CoRR* abs/1504.08083 (2015). arXiv: 1504.08083. URL: http://arxiv.org/abs/1504.08083.

[11] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *CoRR* abs/1506.02640 (2015). arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640.

[12] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: *CoRR* abs/1804.02767 (2018). arXiv: 1804.02767. URL: http://arxiv.org/abs/1804.02767.

[13] Tsung-Yi Lin et al. "Focal Loss for Dense Object Detection". In: *CoRR* abs/1708.02002 (2017). arXiv: 1708.02002. URL: http://arxiv.org/abs/1708.02002.

[14] Alexander Ljung. *imgaug library*. https://github.com/aleju/imgaug. Version 0.2.9. 2019.

[15] Lex Fridman et al. "MIT Autonomous Vehicle Technology Study: Large-Scale Deep Learning Based Analysis of Driver Behavior and Interaction with Automation". In: *CoRR* abs/1711.06976 (2017). arXiv: 1711.06976. URL: http://arxiv.org/abs/1711.06976.

[16] Patrick Follmann et al. "MVTec D2S: Densely Segmented Supermarket Dataset". In: *ECCV*. 2018.

[17] IMCO-Berlin. *Wake Up Call for Retail: Organized Crime Winning the "Shoplifting War"*. URL: https://www.imco-berlin.de/en/blog/article/view-article/ein-weckruf-fuer-den-einzelhandel-organisierte-bandenkriminalitaet-gewinnt-oberhand/ (visited on 06/07/2019).

[18] Trading Economics. *Germany Retail Sales YoY*. URL: https://tradingeconomics.com/germany/retail-sales-annual (visited on 06/07/2019).

[19] Rich Caruana, Steve Lawrence, and C Lee Giles. "Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping." In: vol. 13. Jan. 2000, pp. 402–408.

[20] Wei Liu et al. "SSD: Single Shot MultiBox Detector". In: *CoRR* abs/1512.02325 (2015). arXiv: 1512.02325. URL: http://arxiv.org/abs/1512.02325.

[21] Philipp Jund et al. "The Freiburg Groceries Dataset". In: *CoRR* abs/1611.05799 (2016). arXiv: 1611.05799. URL: http://arxiv.org/abs/1611.05799.

[22] Congcong Li et al. "Data Priming Network for Automatic Check-Out". In: *CoRR* abs/1904.04978 (2019). arXiv: 1904.04978. URL: http://arxiv.org/abs/1904.04978.

Appendices

A Images of Prototype & product demo day

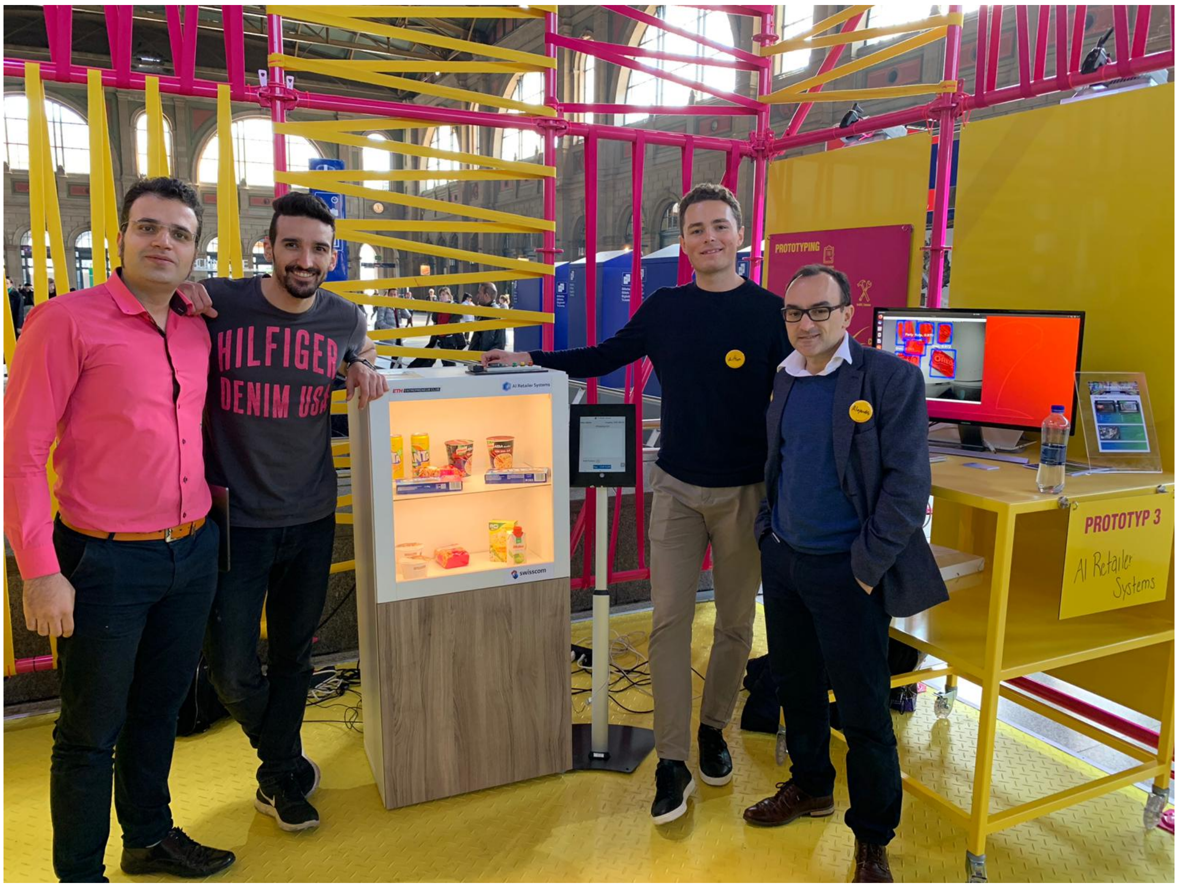

Figure 12: Prototype showcase in the Zürich HB (central station), the team behind the prototype.

Figure 13: Prototype showcase in the Zürich HB (central station) with investors from a major retail chain in Switzerland.