Erasmus School of Economics
Master Business Analytics and Quantitative Marketing

Product Region Detection using Faster R-CNN and Transfer Learning for Moonshop Technologies

Author:
Wander Marijnissen (458717)

Supervisor:
Paul Bouman

September 28, 2022

Abstract

Moonshop technologies aims to develop computer vision software using artificial intelligence that can be used in small autonomous supermarkets with Just Walk Out technology. Currently, however, this concept still requires a 'human in the loop'. To help achieve a fully autonomous algorithm, this paper aims to solve the subproblem of automatic product region detection, which would ease the job of the labelers. It uses a Faster R-CNN model with a ResNet backbone initiated with weights from training on the ImageNet dataset. Also, this paper contributes a dataset of 100 images annotated with product regions, which is used to train the model. The highest average precision (AP) was 0.6863 achieved with a Faster R-CNN model and ResNet-152 backbone. Furthermore, this paper discusses alternatives routes for Moonshop technologies for future research.

keywords: *object detection, neural network, Faster R-CNN, ResNet, transfer learning*

Contents

1 Introduction

After Turing (1950) was one of the first to discuss computer intelligence, machine learning methods have taken an important role in modern society, where enormous amounts of data are collected continuously. While comprehension of this data strongly surpasses human capacity, the application of artificial intelligence has proven important in industries like technology, banking and marketing. For decades, extensive domain expertise and thoughtful engineering was required to create a feature extractor to transform raw data into usable feature vectors for the subsequent learning system. Deep learning, however, allows computational models consisting of multiple processing layers to learn complex patterns from raw data (LeCun et al., 2015), which strongly improved many areas like speech recognition (Hinton et al., 2012), image classification (Krizhevsky et al., 2012), drug discovery (Ma et al., 2015) and even genomics (Leung et al., 2014; Xiong et al., 2015).

An important field of application of deep learning is computer vision. Today plentiful applications of computer vision are seen in many fields and industries, such as retail, automotive, medical and manufacturing industries. For example, computer vision is used in self-driving cars for real-time detection of different objects on the road (Agarwal et al., 2018). Numerous applications in the medical industry include skin-cancer detection (Jain et al., 2015), counting of various types of blood cells (Alam & Islam, 2019) and glaucoma diagnosis based on eye scans (Z. Li et al., 2018). In manufacturing, it is often used for anomaly detection (Scime & Beuth, 2018). Furthermore, computer vision has various applications in retail, such as generating heatmaps from customers to improve store layout (Deloitte Digital, 2018).

A prime example of a company that leverages artificial intelligence to support business decisions is Cape AI. Situated in Cape Town, this cutting-edge firm provides data-driven consultancy and creates independent ventures leveraging the power of artificial intelligence. One of these ventures that is currently in the process of being spun-off is Moonshop Technologies, which aims to increase customer experience in retail through its fully autonomous supermarkets.

Similar to Amazon GO (Ives et al., 2019) and Alibaba's Hema (Wang & Coe, 2021), many retailers worldwide are developing Just Walk Out technology (Cui et al., 2021). Moonshop aims to develop software that uses artificial intelligence to ensure the shops can run autonomously. It requires customers to identify themselves with the Moonshop app upon entering the store and is currently working on implementing an NFC (Near Field Communication) payment solution, such that customers can also enter the store with a creditcard. Customers can then grab everything they might want and simply walk out. A computer vision algorithm is applied to the footage of multiple cameras in the store, which can exactly identify what the customer took, after which the customer is automatically billed.

Reasons for the adoption of autonomous supermarket technologies are plentifold. First of all, this allows to drastically save on human labor costs. The checkout process in retail can

make up 30% of the total labor need (McKinsey, 2017). The Food Marketing Institute (2008) has shown that payroll and employee benefits together can account for more than half the gross margin of food retailers. While labor productivity is on the rise, and consequently unit labor costs decreasing (U.S. Department of Labor, 2020), there is still room for improvement. This is ever more relevant right now in a very tight labor market where employers are fighting over potential employees (Abraham et al., 2020; Forsythe et al., 2020).

Second, the use of computer vision provides large potential for monitoring and further analysis. For example, it can be used to efficiently keep track of in stock items and predict near-future stock requirements to make logistics more efficient. This way, on shelf availability in stores could be increased, reducing lost sales due to out of stock items (Higa & Iwamoto, 2018; Santra & Mukherjee, 2019; Yilmazer & Birant, 2021). Besides increased sales, this could reduce food waste. Annually, about 1.4 billion tons of food is wasted worldwide. In the US, this amounts to 30-40% of the entire food supply (RTS, 2022). Predicting and monitoring stock requirements as done in Yilmazer and Birant (2021) could reduce food waste, simultaneously contributing to increased profit margins and better sustainability.

Furthermore, computer vision can be an interesting basis for marketing related analysis (Wang & Coe, 2021), such as analysis of walking routes, basket and sentiment analysis. Moonshop provides an extensive interactive and customizable dashboard that provides important marketing insights. This can be used to make informed decisions on assortment and lay-out of the shops to maximize sales and customer satisfaction (Deloitte Digital, 2018).

Besides the additional marketing insights, the technology itself could lead to improved customer experience (Santra & Mukherjee, 2019) and higher purchase intent (Esch et al., 2021) compared to self-checkout. The rise of autonomous shopping could be similar to the introduction of Uber, which reinvented the taxi industry by erasing friction and improving customer experience (Ives et al., 2019). Through the use of automated checkout, queuing times are eliminated. This would be especially welcome at busy locations where people are often in a hurry, e.g. a train station. Furthermore, since autonomous shops are unmanned, they can still be profitable in less busy places, even if there would be no customers in the shops for a few hours each day. This opens up a whole range of possible new locations for supermarkets, which could be convenient for people in more remote areas.

Although a promising outlook, the concept currently still requires a so-called 'human in the loop', that is responsible for labeling customers' baskets. Different computer vision subproblems still have to be solved to create a fully autonomous algorithm. Many papers focus on individual product recognition in supermarkets (Goldman et al., 2019; Merler et al., 2007; Santra & Mukherjee, 2019). This approach often suffers from occlusion (Goldman et al., 2019), which makes it more difficult to pin a grab action to a specific product. As opposed to individual product recognition, this paper will focus on directly detecting product regions. This would help the current employers to more easily label which products are being taken from the shelves, and

would serve as an intermediate solution before a fully autonomous algorithm is in place.

Furthermore, rather than training the model to recognize specific SKU's (stock keeping units), we aim to let the model simply draw product regions, even if the product itself is unfamiliar. This could drastically reduce the amount of labeled data that is required and improve scalability. In the case of SKU specific product detection, the model needs to be trained again when it is applied to new shops, new products or even when a product's packaging is changed (Santra & Mukherjee, 2019).

This paper aims to answer the following research question;

How do we most effectively do automated product region detection?

To achieve automated product region detection, this paper uses a convolutional neural network with transfer learning and data augmentation to tackle the problem of data scarcity. It employs a Faster R-CNN architecture with ResNet backbone and created a database of 100 shelf images with annotated product regions for transfer learning. This is described in further detail in Section 3 and 4. The model is trained in multiple hyperparameter configurations. The best results are achieved with a ResNet-152 backbone, anchor sizes $32^2, 64^2, 128^2, 256^2, 512^2$ and aspect ratios 1:2, 1:1 and 2:1, which led to an average precision (AP) of 0.6863, with a detection speed of 0.816 frames per second (FPS) and training time of 74.55 seconds per epoch (SPE). Section 5 provides an extensive discussion of the results and implications for Moonshop. Furthermore, it places the problem in a wider perspective and discusses potential future research topics for Moonshop.

2 Literature

Concepts of autonomous supermarkets like Amazon GO (Ives et al., 2019), Alibaba's Hema (Wang & Coe, 2021) and Moonshop strongly rely on artificial intelligence and computer vision. This has come a long way, since Turing (1950) was one of the first to write about computer intelligence and philosophised about the question whether machines can think. Later, this concept of thinking machines was put into practice by Newell and Simon (1956) with their project the Logic Theorist, which was able to produce proofs for complex problems. Shortly after, the term artificial intelligence was conceived for the first time at the Dartmouth Summer Research Project on Artificial Intelligence (DSPRAI) (McCarthy & Minsky, 1956). This was the starting point for artificial intelligence, which has strongly developed with computers becoming faster, cheaper and more widely available.

2.1 Computer Vision

An important part of artificial intelligence is computer vision, which leverages machine learning models to extract information from image and video input. An instrumental role was played by

neurophysiologists Hubel and Wiesel (1959), who looked into behavior of visual cortical neurons and the effect of cat's visual inputs on their cortical architecture. The important discovery was that they first detect edges and simple shapes and structures, which is fundamental to the idea of deep learning and neural networks. This idea was further extended by Marr (1982), who discovered that vision is hierarchical; first simple shapes, edges and curves are detected, after which more details are filled in later. These conceptual frameworks form the basis for the deep learning architectures for computer vision that we see today.

2.1.1 Convolutional Neural Networks

Inspired by Hubel and Wiesel, Japanese computer scientist Kunihiko Fukushima developed Neocognitron (Fukushima, 1980), a self-organizing neural network model for pattern recognition. After the self-organizing step, this network has a hierarchical structure comparable to that proposed by Hubel and Wiesel (1959). It starts with an input layer, after which we see a cascading connection consisting of simple S-cells and more complex C-cells. In unsupervised fashion, repeated exposure to a certain stimulus pattern will lead to each C-cell in the final layer responding to only one stimulus pattern, regardless of the pattern's position and minor changes in shape or size. The neocognitron acts as the inspiration for convolutional neural networks we see today.

LeCun et al. (1989) proposed one of the first convolutional neural networks, which was later called LeNet-5. Convolutional neural networks (CNNs) today are a common neural network architecture for image related tasks, including object detection, image classification and pose estimation. LeNet-5, similar to convolutional neural networks we see today, has components like convolutional, pooling and fully connected layers. It consists of 7 layers with trainable parameters; 3 convolutional layers, 2 pooling layers and 2 fully connected layers. In a convolutional layer, a feature map is extracted from an image through multiplication of the input and weights, which is called a filter or kernel. Multiple of these filters can be applied in parallel. Furthermore, convolutional layers can be applied hierarchically to the output of other layers. This way, the filters that are applied in the first convolutional layers can extract lower-level features like lines and edges, and subsequent deeper layers can detect more complex shapes from the output of previous convolutional layers. Pooling layers are applied after a convolutional layer to reduce the dimension of the feature map by summarizing its most important features through applying a filter. This decreases sensitivity of feature maps to the location of features in the input. Last, in fully connected layers, as opposed to convolutional layers, all components of the input vector are connected to all components of the output vector. 10 years after the introduction in 1989, LeCun et al. (1998) found that convolutional neural networks outperformed various methods, such as K-nearest neighbors, hidden markov model and support vector machines on a standard handwritten digit recognition task.

A problem for early stage convolutional neural networks like LeNet-5 was the computational

intensity, that hardware at the time was not equipped for. However, the introduction of GPUs or graphic processing units dramatically increased the speed of CNNs by up to 4 times (Chellapilla et al., 2006). The GPU is a processor created to accelerate real-time graphics, with applications in gaming, artificial intelligence and more. Later, the GPU implementation of a deep CNN created by Ciresan et al. (2011) was shown to be 60 times faster compared to the CPU version, showing the incredible importance of GPUs. From the early 2000s, CNNs have had great success in object and region detection, segmentation and recognition (LeCun et al., 2015).

In 2012, Alexnet participated in the ImageNet Large Scale Visual Recognition Challenge, an image classification challenge with 1000 different classes. AlexNet achieved a top-5 error of 15.3%, 10.9 percentage points lower than the second-best model (Deng et al., 2012). Alexnet is a convolutional neural network consisting of five convolutional layers, some followed by max-pooling layers and three fully-connected layers with a final 1000-way max, which relates the output to the 1000 different class labels of the ImageNet dataset. The model has 8 trainable layers with 60 million parameters in total (Krizhevsky et al., 2012). The success of AlexNet proved that CNNs are an excellent fit for computer vision problems and they have been the standard to this day.

After the success of AlexNet, many different CNN model architectures arised. In 2014, GoogLeNet, also known as Inception, brought the top-5 error for the ImageNet Large Scale Visual Recognition Challenge down to 6.67% (Szegedy et al., 2015). A standard way of improving deep neural networks' performance is by increasing depth and width. However, this raises the risk of overfitting, especially with a small amount of training data. Furthermore, this dramatically increases computational intensity. To combat this, GoogLeNet, a 22 layers deep network, replaced fully connected layers by sparse layers, resulting in 12 times fewer parameters than the model of Krizhevsky et al. (2012). An important innovation is their so-called 'inception module', which adds an alternative parallel pooling path (Szegedy et al., 2015).

One year later, the Residual Neural Network (ResNet) did even better, with a top-5 error rate of 3.57% on the ImageNet dataset (He et al., 2016). Despite the model being even deeper, the researchers eased training by introducing a residual learning framework. Training deeper models with backpropagation and gradient-based methods often leads to the vanishing gradient problem, in which gradients become so small that certain weights are never changed. This especially affects gradients in deeper layers, and often prevents models from converging (Glorot & Bengio, 2010). This has been mostly overcome through normal initialization and intermediate normalization layers. However, despite the convergence of these models, a degradation problem arises; with additional layers accuracy gets saturated and even sharply decreases from a certain depth. ResNets combat these problems through residual learning and short-cut connections.

Residual learning entails that the model learns residual functions with reference to the layer inputs, instead of learning unreferenced functions. These models pile up residual blocks to form a network. He et al. (2016) created an extremely deep network with 152 layers, which

achieved high accuracy on the ImageNet dataset, but also showed great generalizability for other recognition tasks, such as ImageNet detection, ImageNet localization, COCO detection and COCO segmentation in the ILSVRC and COCO 2015 competitions.

2.1.2 Object Detection

For the task of object detection, an important development is region-based convolutional neural networks (R-CNN), invented by Girshick et al. (2014). Object detection essentially consists of two tasks; localization and classification of objects. In the R-CNN model, first 2000 region proposals are derived from the input image using selective search (Uijlings et al., 2013). Then, a CNN is applied to these regions after which the regions are classified. When all regions are given a score, the model uses greedy non-maximum suppression to prevent strongly overlapping regions for each class. For object detection, the R-CNN model presented a 30% improvement over the previous state-of-the-art on the PASCAL VOC 2012 dataset, commonly used for object detection. Although R-CNNs generally perform well in terms of accuracy, important drawbacks are the computational intensity for training and the slow test time, with the model taking around 47 seconds per image. Furthermore, no learning happens in the selective search algorithm. The architecture was further improved with the introduction of Fast R-CNN. In Fast R-CNN, the CNN is first applied to the input image to generate a feature map from which the region proposals are determined. Then, a region of interest (RoI) pooling layer is applied to transform the proposed regions into a fixed size. This dramatically increases the speed of fast R-CNN compared to R-CNN because the convolution is now only done once per image, instead of for 2000 region proposals. The test time excluding the region proposal went from 47 seconds to 0.32 seconds per image (Girshick, 2015). However, the main bottleneck now was region proposal computation, which increases test time from 0.32 to 2.3 seconds per image. To further decrease the influence of region proposals on train and test times, Ren et al. (2015) came up with Faster R-CNN, which replaces the selective search algorithm by a trainable region proposal network (RPN). With Faster R-CNN test time went down to 0.2 seconds per image, including region proposals. Besides a strong improvement in runtime, the learned RPN also enhances region proposal quality and consequently overall object detection accuracy.

Whereas all versions of the R-CNN architecture rely on region proposals for object detection, Redmon et al. (2016) came up with another approach. Their algorithm, You Only Look Once (YOLO) is a single CNN predicting bounding boxes and corresponding class probabilities. The input image is split into an SxS grid, after which B bounding boxes and confidence scores are predicted for each grid cell. The grid containing the center of an object is responsible for detecting the particular object. Bounding boxes with a confidence level above a certain threshold are then selected. This approach makes YOLO magnitudes faster than other object detection models (Redmon et al., 2016). In terms of accuracy, YOLO and Faster R-CNN are comparable, and performance depends on the specific application (Benjdira et al., 2019). M. Li

et al. (2020) achieved an accuracy in the detection of agricultural greenhouses of 86.0% and 90.4% with Faster R-CNN and YOLO respectively. On the other hand, Tan et al. (2021) found Faster R-CNN (87.69%) to outperform YOLO (80.17%).

2.2 Transfer Learning

Given scarcity in both time and resources, and the fact that machine learning problems sometimes require large amounts of data to achieve high accuracy, it is often not feasible to build a machine learning model from scratch (Pan & Yang, 2009; Rawat & Wang, 2017). An important solution was the prospect of transfer learning for training neural networks, first mentioned by Bozinovski and Fulgosi (1976), and first applied on a neural network learning to recognize letters (Bozinovski, 1981). Transfer learning is built on the assumption that training and future data must not necessarily be in the same data space (Pan & Yang, 2009). In transfer learning, weights from a previous learning task are copied and used as a starting point for the new learning task. Transfer learning has proven valuable in discovery of cancer subtypes (Hajiramezanali et al., 2018), building occupation using CO2 detection (Arief-Ang et al., 2018), text classification (Do & Ng, 2005), digit recognition (Maitra et al., 2015) and spam filtering (Bickel, 2006). It was also employed in the field of grocery product detection (Franco et al., 2017).

2.3 Approaches to Supermarket Product Recognition

To tackle the problem of product recognition in supermarket shelves, some have tried a sensor-based approach, using Radio Frequency IDentification (RFID), sensors or barcodes (Ives et al., 2019; Santra & Mukherjee, 2019; Wang & Coe, 2021). However, these approaches are often relatively expensive and tend not to scale well. Computer vision models present a viable alternative (Santra & Mukherjee, 2019). Gevers and Smeulders (1999) made the first attempt to recognize isolated retail products that were cropped from the images, but did not focus on localization. 8 years later, Merler et al. (2007) attempted localizing and recognizing products using rack images and a product image database.

While most studies use a product image database and shelve images (Santra & Mukherjee, 2019), Goldman et al. (2019) do not train their model on specific products. Rather, they apply object detection to densely packed retail scenes, treating all products as belonging to the same class. To do this, they created the SKU110K dataset, in which shelf images with various lighting and angles were annotated with bounding boxes around each item. Their approach, based on a base detection network, soft IoU layer and EM-merger achieved an average precision of 0.492. This shows the difficulty of the SKU110K dataset, which is composed of densely packed scenes suffering from occlusion.

A different approach was taken by Higa and Iwamoto (2018), who studied changes on the shelves using video footage and a CNN. With this, product amounts were computed to track on-shelf availability. They achieved an accuracy of 89.6% with an error margin less than one

product. In monitoring on-shelf availability, Yilmazer and Birant (2021) use semi-supervised learning in combination with one-stage detectors RetinaNet, YOLOv3 and YOLOv4 to detect and classify regions into 'product', 'empty shelve' and 'almost empty shelve'. First, the one-stage detectors are trained on 300 labeled images to see which method performs best in terms of mAP. Then, the best model is used to make predictions for the unlabeled data to create pseudo-labeled data. Last, the final classifier is built using both labeled and pseudo-labeled data. The study achieved an overall mAP of 89.27% with their best model using 80% labeled images.

In the domain of object detection of products in grocery shelves, no work has previously addressed product region detection.

3 Method

This paper focuses on product region detection, a form of object detection. In this case product regions rather than the individual products, as done by Goldman et al. (2019), are considered as objects. The algorithm will thus have to detect groupings of identical products and draw rectangular bounding boxes around these regions. This can range from one item to several items of the same product. No distinction is made between product regions of different products, all regions are seen as belonging to the same class. More information on the dataset and examples of some product regions can be found in Section 4.

Since no previously annotated data is at hand, this paper faces a data scarcity constraint. As previously discussed, transfer learning can combat this problem by using weights from a previously trained model in a different data space as starting point for the current object detection task (Pan & Yang, 2009; Rawat & Wang, 2017). This is due to the hierarchical nature of CNNs, where the first layers detect simple features like edges, and final layers can detect more complex and task specific features (LeCun et al., 1989). Many image classification models are based on transfer learning (He et al., 2016; Krizhevsky et al., 2012). In this paper, rather than training the entire model from scratch, weights from a model pre-trained on ImageNet will be used for initialization, after which the model will be trained on the dataset labeled with product regions. More specifically, this paper will use the Faster R-CNN architecture with a ResNet backbone for the CNN.

3.1 Faster R-CNN

The faster R-CNN architecture as proposed by Ren et al. (2015) consists of two parts, a so-called 'region proposal network' (RPN) and the Fast R-CNN detector (Girshick, 2015). This RPN is a fully convolutional network predicting object boundaries and objectness scores for each location. These RPNs are trained to create the most accurate region proposals, which are then fed to Fast R-CNN (Girshick, 2015) for detection. Since the RPN and Fast R-CNN should eventually come together, it is assumed that both networks have a common set of convolutional layers, which

leads to small marginal costs for creating proposals. To create region proposals, Ren et al. (2015) use a small network that slides over the feature map that is the output of the final shared convolutional layer. This network takes an $n \times n$ spatial window of the convolutional feature map as input, which is than mapped to a lower-dimensional feature. After, this feature is fed into two fully-connected layers; a box-regression layer and a box-classification layer.

At each point of the sliding window, a maximum of k region proposals is generated. This means the box-regression layer has 4k outputs representing the coordinates of k boxes. The box-classification layer is defined as a two-class softmax layer, an activation function which provides per class probabilities. Hence the box-classification layer has 2k scores, reflecting the probability of object or not object for each region. The k proposals relate to k anchors, which are centered at the particular sliding window and associated with scale and aspect ratio. A graphical representation of Faster R-CNN is shown in Figure 1.

Ren et al. (2015) use 3 scales and 3 aspect ratios, which results in $k = 9$ anchors at each sliding window. Given a convolutional feature map of size $W * H$, there are a total of WHk anchors. By considering anchor boxes with different scales and aspect ratios, this effectively deals with multi-scale predictions.

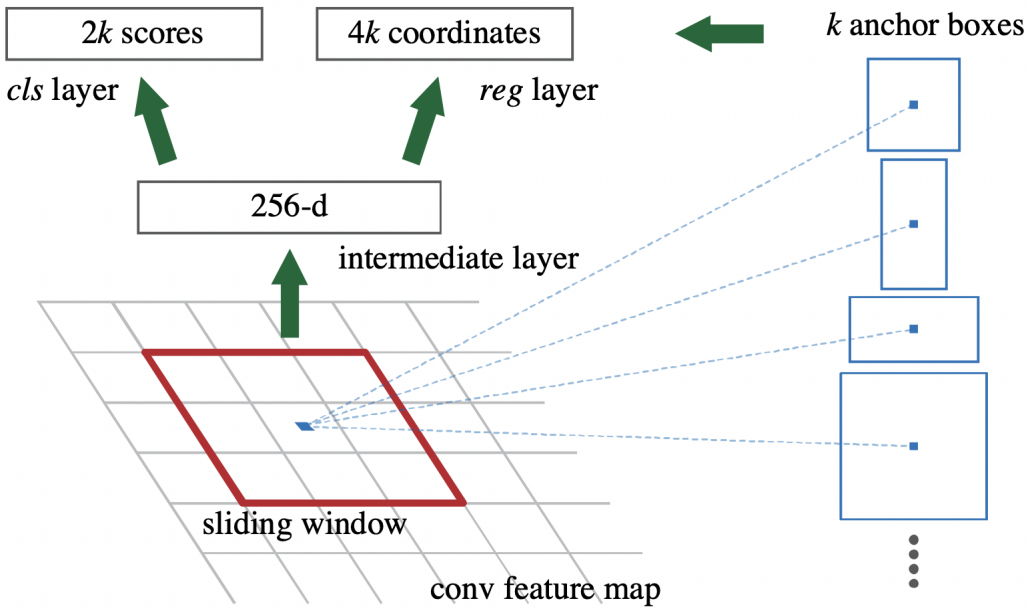


Figure 1: Illustration of Region Proposal Network (RPN) in Faster R-CNN architecture (Ren et al., 2015)

This approach leads to translation invariance. When an object in an image is translated, the proposal will translate in similar fashion. This results in a convolutional output layer with a dimension of $(4 + 2) * k = 54$ in the case of $k = 9$. This is smaller than methods that are not translation invariant such as the Multibox method (Szegedy et al., 2014), which has a fully connected output layer with dimension $(4 + 1) * 800 = 4000$. This reduces the risk of overfitting

on small datasets.

Regarding RPN training, a binary class label is assigned to each anchor, indicating whether or not this is an object. This label is based on the Intersection-over-Union (IoU) between anchors and ground-truth boxes. The IoU between two regions is defined as the area of intersection of the two regions divided by the area of the union of the regions. A positive label is assigned to anchors with the highest IoU with a ground-truth box or any anchor with an IoU of at least 0.7 with any ground-truth box. This means that one ground-truth box can lead to multiple anchors with a positive label. A negative label is assigned to a non-positive anchor with an IoU below 0.3 for all bounding boxes. Neutral anchors are not considered for training. This leads to the following loss function, as described by Ren et al. (2015);

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{1}$$

where i is the index of an anchor in a mini-batch, p_i is the predicted probability of anchor i being an object, p_i^* is the ground-truth label, reg stands for the box regression and cls stands for the box classification, t_i the vector representating the 4 parameterized coordinates of the predicted bounding box and t_i^* that of the ground-truth box. L_{cls} is the classification log loss, $L_{reg}(t_i, t_i^*)$ the regression loss as defined in Girshick (2015);

$$L_{reg}(t_i, t_i^*) = smooth_{L_1}(t_i - t_i^*) \tag{2}$$

where

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & if|x| < 1 \quad (3) \\ |x| - 0.5 & otherwise \quad (4) \end{cases}$$

Because of $p_i^* * L_{reg}$, the regression loss is only calculated for positive anchors ($p_i^* = 0$). Furthermore, the two terms of the loss function are normalized by N_{cls} and N_{reg} and weighted by a parameter λ. In the paper by Ren et al. (2015), N_{cls} is set equal to the mini-batch size, N_{reg} is set to the number of anchor locations and $\lambda = 10$ by default. They have shown that the value of λ does not influence the results in a wide range. The 4 coordinates of the bounding box regression are parameterized as follows;

$$\begin{aligned} t_x = (x - x_a)/w_a, t_y = (y - y_a)/h_a \\ t_w = log(w/w_a), t_h = log(h/h_a), \\ t_x^* = (x^* - x_a)/w_a, t_y^* = (y^* - y_a)/h_a, \\ t_w^* = log(w^*/w_a), t_h^* = log(h^*/h_a), \end{aligned} \tag{5}$$

where x and y denote the box's middle coordinates and w and h its width and height. x, x_a and x^* stand for the predicted box, anchor box and ground-truth box respectively. The same holds for y, w and h. The features used for this bounding-box regression are of the same size (3 x 3) on the feature maps. Then k bounding-box regressors are learned, all responsible for

one scale and one aspect ratio to deal with different sizes. These k regressors have no common weights. This way, the model can still predict boxes of different sizes despite the fixed scale of the features.

This RPN can be trained using backpropagation and stochastic gradient descent (SGD) (LeCun et al., 1989). All mini-batches come from a single image containing multiple positive and negative example anchors. Rather than optimizing over the loss functions of all anchors, 256 anchors are randomly sampled from an image to compute the loss function. The ratio of positive to negative anchors is set to up to 1:1 to reduce the bias towards negative samples. All new layers are initialized by draws from a zero-mean Gaussian distribution with standard deviation 0.01 and all shared convolutional layers are initiated by loading the weights from a model pre-trained on the ImageNet dataset.

For the detection network, Faster R-CNN uses the network from the Fast R-CNN implementation (Girshick, 2015). The Fast R-CNN network takes a complete image and object region proposals as input. First, the image goes through several convolutional and max pooling layers to create a convolutional feature map. For each object proposal, a region of interest (RoI) pooling layer derives a fixed-length vector from the feature map. This way, differently sized region proposals are transformed into a standard size before going into the classifier.

These feature vectors are then led into a series of fully connected layers resulting in two output layers. One gives softmax probabilities for all object classes and one gives four real-valued numbers for all classes, relating to bounding-box coordinates.

This Fast R-CNN network and the previously described RPN are combined with some shared convolutional layers. However, the RPN and Fast-RCNN network are trained independently. The networks are trained in alternating fashion; first the RPN is trained, after which the proposals are used for training the Fast R-CNN. Then, the Fast R-CNN is used to initialize RPN, but the shared convolutional layers are frozen and only RPN specific layers are trained. Last, the unique layers of the Fast R-CNN are trained on the proposals from the RPN.

Since some region proposals might highly overlap, Ren et al. (2015) use non-maximum suppression (NMS) with an IoU threshold of 0.7, leading to about 2000 proposal regions.

Like Ren et al. (2015) this paper will evaluate the performance of the object detection model based on the PASCAL VOC benchmark (Everingham et al., 2008), a dataset of around 10.000 images over 20 categories.

3.2 Residual Neural Network (ResNet)

As the CNN backbone for the Faster R-CNN implementation, this paper uses the Residual Neural Network (ResNet), as introduced by He et al. (2016). ResNet is a widely adopted CNN structure and has led to good results in combination with the Faster R-CNN architecture in the COCO 2015 competitions (Ren et al., 2015). This CNN acts as a feature extractor. As discussed in Section 2, training deeper networks often leads to the degradation problem, where

accuracy gets saturated and even sharply decreases with additional depth added to a model. To be able to still train deep models whilst preventing the degradation problem, ResNet uses the residual learning framework. This approach leads to easier training and gained accuracy from the additional depth of these models.

Rather than fitting the desired underlying mapping $H(x)$, the model fits a residual mapping $F(x) := H(x) - x$. The original mapping then becomes $F(x) + x$. As the work by He et al. (2016) has shown, it is easier to fit the original unreferenced mapping by feedforward neural networks that make use of short-cut connections, referencing the function to the layer inputs. These connections skip one or multiple layers. In ResNet, these short-cut connections involve a simple identity mapping, which does not lead to any extra parameters or computational intensity. He et al. (2016) have shown that these networks are easy to train, while significantly reducing training error compared to deep networks that do not involve residual learning.

Residual learning is applied to every few stacked layers, making use of building blocks defined as;

$$y = F(x, \{W_i\}) + x \tag{6}$$

where x and y denote the input and output vectors of the considered layers and $F(x, \{W_i\})$ denotes the residual mapping. An example of such a building block is shown in Figure 2. This residual learning with short-cut connections is implemented in every couple of stacked layers. In the example of Figure 2, the building block consists of 2 layers, hence $F = W_2\sigma(W_1x)$. Batch normalization (Ioffe & Szegedy, 2015) is applied after every convolutional layer to normalize input for the activation function. The activation function used is a Rectified Linear Unit (ReLu), an activation function defined as $f(x) = x^+ = max(0, x)$ (Nair & Hinton, 2010) and denoted by σ. Biases are omitted to simplify notations. Then, a short-cut connection executes $F + x$ through element-wise addition, after which ReLu is applied again to $F + x$. Since the short-cut connections consist merely of identity mappings, they lead to no increased amount of parameters or computational complexity.

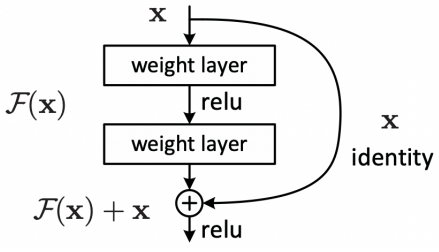


Figure 2: Residual learning building block (He et al., 2016)

The dimensions of F and x must be equivalent. If the dimensions differ, this can be solved through applying a linear projection W_s as follows;

$$y = F(x, \{W_i\}) + W_s x \tag{7}$$

An example of a ResNet compared to a plain network with 34 layers is shown in Figure 3. The dotted lines show a linear projection as described in Equation 7, which is required since the dimensions change in these layers. The full line shortcuts are simple identity mappings.

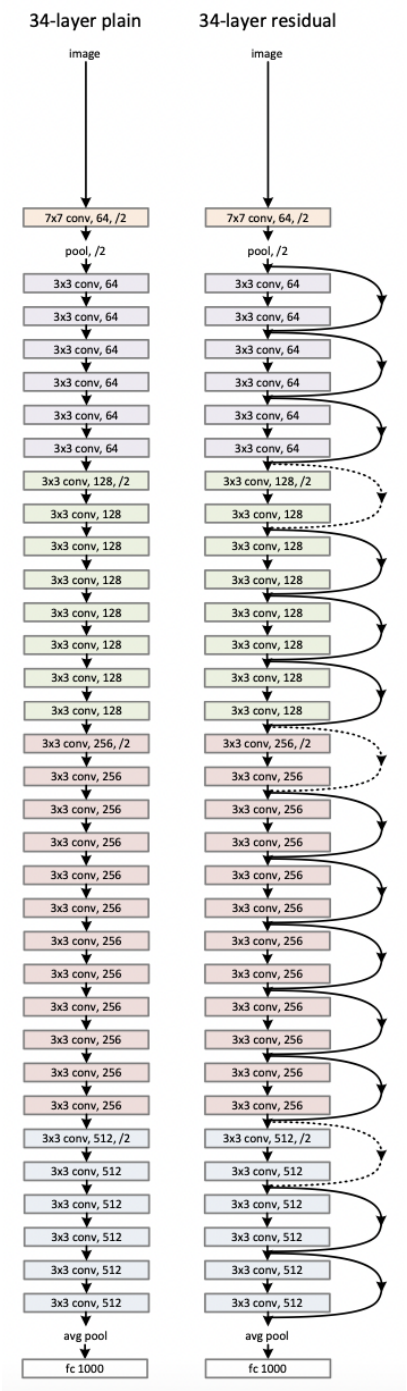


Figure 3: Plain network architecture (left) and ResNet architecture implementing residual learning (right) (He et al., 2016)

Deeper models such as ResNet-50, ResNet-101 and ResNet-152 use 3 layer building blocks for residual function F instead of 2. To limit required training time in these deeper models, He et al. (2016) came up with the bottleneck design, shown in Figure 4, which consists of $1x1$, $3x3$ and $1x1$ convolutions. The $1x1$ layers decrease and then increase/recover the dimensions, leaving the $3x3$ layer with smaller dimensions to work with.

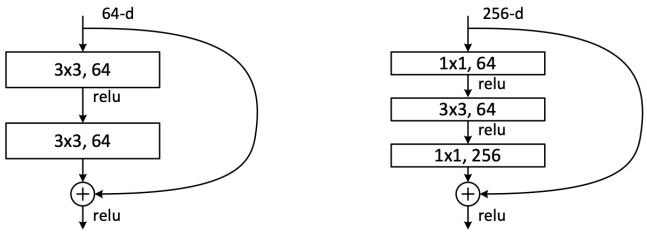


Figure 4: Regular building block (left) and bottleneck building blocks used in Resnet-50/101/152 (right) (He et al., 2016)

3.3 Model Architecture

As previously described in Section 3.1, the Faster R-CNN architecture consists of 3 parts; the feature extraction backbone, region proposal network (RPN) and region of interest (RoI) pooling layers. As a backbone, this paper uses the ResNet model as designed by He et al. (2016). ResNet-50, ResNet-101 and ResNet-152 have shown to be substantially more accurate than lower layer models. However, very deep models are also more prone to overfitting, especially in the case of a small dataset and generally have a longer train and test time. In order to explore the ideal depth of the ResNet backbone, this study uses the ResNet-18, ResNet-50 and ResNet-152 architectures, respectively. Table 1 shows the number of parameters in the backbone and the total number of parameters for Faster R-CNN with different ResNet backbones.

Table 1: Number of parameters in millions of Faster R-CNN with different ResNet backbones

Model	Total	Backbone	Non-Backbone
ResNet-18	40.3	11.2	29.1
ResNet-50	165.2	23.5	141.7
ResNet-152	199.9	58.1	141.7

Figure 5 shows the composition of different ResNet models in more debt. In the first convolutional layer, all models have a $7x7$ convolution with 64 filters and downsampling is performed through implementation of a stride of 2. Furthermore, in all models the second convolutional

layer starts with a $3x3$ max pool with stride 2. Then, we see a difference between ResNet-18 and the deeper models. ResNet-18 consists of a number of residual learning blocks of two convolutions stacked on top of each other, whereas ResNet-50 and ResNet-152 consist of three-layer residual learning blocks with the previously discussed bottleneck design. ResNet-50 and ResNet-152 merely differ in the number of residual learning blocks in convolutional layers 3 and 4. Downsampling is performed in the first convolution of convolutional layers 2, 3, 4 and 5 through a stride of 2. This convolutional backbone is shared by the RPN and Fast R-CNN.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 5: ResNet layers (He et al., 2016)

The feature map that is generated by the ResNet backbone is then fed into the RPN, which performs a box-regression and box-classification for all anchor sizes and aspect ratios to create region proposals. Next, the region proposals are reshaped by two fully connected RoI pooling layers, after which the box-regression and box-classification are performed for the region proposals.

As discussed in Section 2, transfer learning proves useful in problems that face data scarcity (Pan & Yang, 2009; Rawat & Wang, 2017). In the case of this research, weights from the ResNet model pre-trained on the ImageNet dataset (Deng et al., 2012) are used to initiate the backbone. After, the Faster R-CNN structure is initiated with the pre-trained ResNet backbone, a Region Proposal Network (RPN) and Region of Interest (RoI) pooling layer. When applying transfer learning to a large model with a small dataset, it can be beneficial to "freeze" some of the layers, meaning the gradient will not be calculated and hence the weights will not be updated. This way, weights learned from training on a comparable task in a different domain can be preserved, and computation time reduced. Furthermore, this can lead to better results with small datasets (Pan & Yang, 2009; Rawat & Wang, 2017). To test whether freezing some of the layers leads to a higher accuracy, the Faster R-CNN model with ResNet-50 backbone will be run in three different configurations; (1) with all backbone layers frozen, (2) with the backbone first 5 layers

(lower level features) frozen and last two layers (higher level features) unfrozen and (3) with the entire backbone unfrozen.

3.4 Data Transformations

As described in Section 4 (specifically Table 3), the available images vary greatly in input size in terms of pixels. Before training the model, some transformations are applied to the data when initiating the dataset. Before the images are used, they are rescaled. For this model, the minimum and maximum size of the images is set to 1024. If necessary, a padded margin will appear to limit image distortions. Furthermore, values of the 3 colour channels red, green and blue are normalized. For this, the image mean and image standard deviation of the ImageNet dataset are used, since this dataset was used to train the backbone. The values used for normalization are $(0.485, 0.456, 0.406)$ for the mean and $(0.229, 0.224, 0.225)$ for the standard deviation.

Besides, to maximize performance with the limited dataset at hand, data augmentation is employed following the albumentation module designed by Buslaev et al. (2020). As previously discussed, training deep models on small datasets can lead to overfitting. Through data augmentation the training data is increased both in volume and variety, while maintaining output labels. This is common practice to avoid overfitting and used to improve performance of CNNs (Krizhevsky et al., 2012; Santra & Mukherjee, 2019). Horizontal flips are applied to the training data with default probability 0.5. Additionally, inputs are randomly rescaled with scale limit 0.5 and probability 0.5. This means images are rescaled in the range $(0.5, 1.5)$.

Then, dataloaders are initiated. We specify three data loaders, for training, validation and testing. For training, we use a batch size of 2, which is the number of training samples used per iteration. Generally, a smaller batch size will lead to quicker convergence, however a larger batch size is more likely to lead to the global optimum and hence higher accuracy (Radiuk, 2017). Dataloaders for validation and testing work with a batch size of 1.

3.5 Anchors

Furthermore, as discussed in Section 3.1, the Faster R-CNN structure evaluates the sliding windows at anchors with different scales and aspect ratios. Ren et al. (2015) use scales $(128^2, 256^2, 512^2)$ and aspect ratios (2:1, 1:1, 1:2), however mention that these values are not carefully chosen for a particular dataset. They do show, however, that 3 aspect ratios and 3 scales lead to a higher accuracy compared to 1 aspect ratio and 3 scales, 3 aspect ratios and 1 scale or 1 aspect ratio and 1 scale. Additionally, for detection on the COCO dataset the paper adds 64^2 to the anchor sizes to aid in the detection of small objects. In our basic model, we will use anchor sizes $(128^2, 256^2, 512^2)$ and aspect ratio $(1:2, 1:2:1)$. However, we expect to gain from more anchor sizes and ratios, given the large variety of aspect ratio and size of the boxes in our dataset, which is shown in Section 4. Therefore we perform ablation experiments like Ren et al. (2015)

to assess the effect of different anchor sizes and aspect ratios on accuracy, by training the model in 4 different configurations. As described in Section 4, bounding boxes' height and width in the available dataset are positively skewed. Hence, it can be assumed that adding smaller anchor sizes could improve the results. Furthermore, the dataset shows very extreme aspect ratios, hence we experiment with adding these as well. Therefore, the model is run in the following configurations shown in Table 2. Results are shown in Section 5.

Table 2: Different configurations of anchor sizes and aspect ratios used in ablation experiment

Anchor sizes	Aspect ratios
$(128^2, 256^2, 512^2)$	(2:1, 1:1, 1:2)
$(32^2, 64^2, 128^2, 256^2, 512^2)$	(2:1, 1:1, 1:2)
$(128^2, 256^2, 512^2)$	(3:1, 2:1, 1:1, 1:2, 1:3)
$(32^2, 64^2, 128^2, 256^2, 512^2)$	(3:1, 2:1, 1:1, 1:2, 1:3)

3.6 Training

As discussed in Section 3.1, training relies on stochastic gradient descent (SGD) (LeCun et al., 1989) as its optimizer. Like He et al. (2016) and Ren et al. (2015), a momentum of 0.9 is used. Momentum is used to combat the stochastic nature of the SGD optimizer, which randomly picks a mini-batch at every iteration. Momentum ensures that the weight change of the current step not only depends on the gradient of the mini-batch, but also on the weight change of the previous step. This prevents oscillation and often leads to a smoother convergence with higher accuracy (Qian, 1999).

Furthermore, like Krizhevsky et al. (2012) and Ren et al. (2015) a weight decay of 0.0005 is employed. Weight decay acts as a regularizer, which penalizes parameters to prevent overfitting. Krizhevsky et al. (2012) has shown that this small weight decay is more than just a regularizer, it decreases the model's training error. Through weight decay, parameters are penalized, which prevents overfitting. The bigger the weight decay factor, the more parameters tend to zero. The update formula for the weights is shown in Equation 8.

To train the model, a learning rate of 0.001 is used as done by Girshick (2015) and Ren et al. (2015). Setting the learning rate is important, since a value too low can result in a slow training process, whereas a value that is too high can lead to an unstable process or result in

sub-optimal outcomes.

$$v_{i+1} := 0.9 * v_i - 0.0005 * \epsilon * w_i - \epsilon * \langle \frac{\partial L}{\partial w}|_{w_i} \rangle_{D_i}$$
$$w_{i+1} := w_i + v_{i+1} \tag{8}$$

Here, i is the iteration index, v is the momentum variable, ϵ is the learning rate and $\langle \frac{\partial L}{\partial w}|_{w_i} \rangle_{D_i}$ is the average of the derivative of the loss function with respect to w at w_i, over batch D_i.

During training, the model is saved periodically by monitoring the validation AP, which saves a model to the checkpoints when it achieves the maximum AP so far. Furthermore, the use of early stopping ensures that training is stopped when validation AP stops improving for a certain number of epochs. This is done to prevent overfitting (Ying, 2019) and avoid unnecessary training time. This study implements a patience of 30, which is the number of epochs to continue after the validation AP has stopped increasing. Due to this patience that prevents overfitting, we can specify a relatively high maximum number of epochs. It was observed that training never really improved after more than 150 epochs, which is why this is set as the maximum number of epochs. Furthermore, by choosing the best model based on the validation AP, rather than on training accuracy, overfitting is avoided.

As GPUs are known to drastically improve training time (Chellapilla et al., 2006; Ciresan et al., 2011), training is executed on a Quadro P5000 16 GB GPU offered by Paperspace Gradient. To compare training and detection speed between different models and evaluate their feasibility in employment, the number of seconds per epoch (SPE) during training and number of frames per second (FPS) during detection are logged and reported in Section 5.

4 Data

For dataset generation, this paper looked at the SKU110K dataset, which was generated by Goldman et al. (2019) and consists of 11,762 images of densely packed grocery shelves, varying in scale, angles, lighting conditions and noise levels. This dataset contains 110,712 different object classes, and on average contains 147.4 individually annotated products and 86 different classes per image. Items on grocery shelves are often tightly packed to maximize sales, which is reflected in the SKU110K dataset. The dataset contains images made in thousands of supermarkets around the world, usually made with cellphones.

From this dataset, 100 images where selected. These were divided in 80 images (80%) for training, 10 images (10%) for validation and 10 images (10%) for testing. With the task in mind, images consisting almost entirely of single products were not taken into account. This way, it is ensured that the images in the dataset provide sufficient opportunity for learning groupings of multiple identical products.

Since the SKU110K dataset was originally annotated with individual products, custom annotations where generated for this research. Rectangular bounding boxes where created around

all different product regions. This means there is one detectable class; 'region' which is labeled as 1, all background is labeled as 0. Pictures are saved in the .jpg format and the annotations are saved in .json format. This annotation file contains the labels and coordinates (x_top_left, y_top_left, x_bottom_right, y_bottom_right). The full dataset used for training, validation and testing can be found at https://github.com/wandermarijnissen/product-region-detection.

In Tables 3, 4 and 5 some summary statistics of the custom dataset are displayed. Images are inputted in (3, W, H) shape, where 3 represents the 3 colour channels red, green and blue, W and H stand for the width and height of the image in terms of pixels. Table 3 shows the minimum, maximum and average image sizes, showing a strong variety. This stems from the large variations in image size in the SKU110K dataset.

As shown in Table 4, the dataset consists of 100 labelled images, with a total of 5149 bounding boxes. The density of the images in terms of the number of product regions varies greatly, with a minimum of 8 (Figure 6) and maximum of 117 (Figure 7) regions per image.

Furthermore, Table 5 shows strong differences between bounding boxes. For example, height ranges from 36.0 to 2029.0, width ranges from 21.0 to 2097.0 and aspect ratio ranges from 0.0638 to 8.3734. Before using the data, some transformations are applied, as discussed in Section 3.4. Table 6 shows bounding box summary statistics after the images have been rescaled. Although this normalizes height and width, the high variation in aspect ratios remains. Per visualisation, the highest aspect ratio is shown in Figure 8.

Table 3: Summary statistics image height, width and size in terms of pixels

	minimum	maximum	average
Height	842	4208	2745
Width	1010	4160	2048
Size	1113020 (1010x1102)	13128960 (3120x4208)	5794601

Table 4: Image level summary statistics

Number of images	100
Total number of regions	5149
Minimum number of regions per image	8
Maximum number of regions per image	117
Average number of regions per image	51.49

Table 5: Bounding box level summary statistics before transformations

	minimum	maximum	average
Height	36.0	2029.0	278.1
Width	21.0	2097.0	238.5
Area	4826.0	1786785.0	73768.4
Aspect ratio	0.0638	8.3734	0.8982

Table 6: Bounding box level summary statistics after transformations

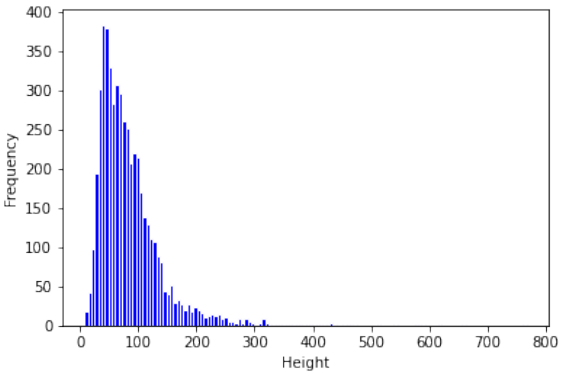
	minimum	maximum	average
Height	8.0	767.2	85.0
Width	10.3	690.5	102.2
Area	402.9	191102.0	9585.3
Aspect ratio	0.0638	8.3734	0.8982

Figure 6: Least dense image with 8 regions

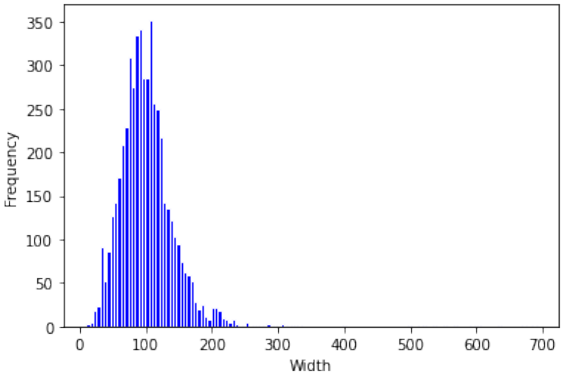
Figure 7: Most dense image with 117 regions

Figure 8: Region with the highest aspect ratio (8.37)

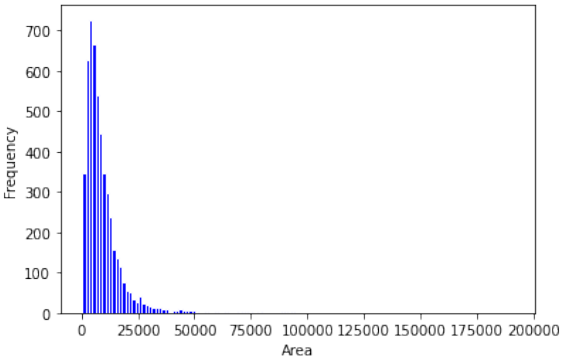
To give a better idea of the distribution of the bounding box summary statistics height, width, area and aspect ratio after the transformations, the histograms of these variables are shown in Figure 9. From these histograms and summary statistics (Table 6), it becomes clear that these variables are positively skewed. Despite the fact that there are some very large bounding boxes in the dataset, the majority of bounding boxes are on the smaller end of the spectrum. This has implications for the choice of anchor size and aspect ratios, described in more detail in Section 3.5.



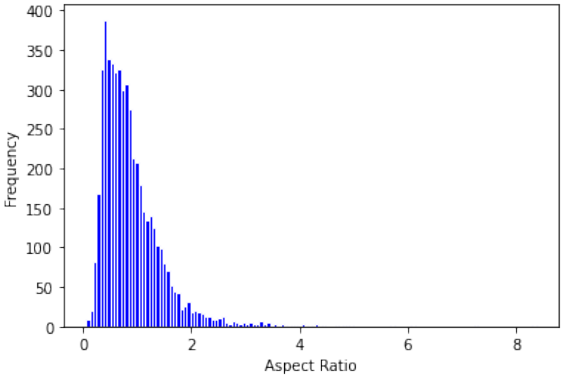
(a) Height



(b) Width



(c) Area



(d) Aspect ratio

Figure 9: Histograms showing distribution of bounding box height (a), width (b), area (c) and aspect ratio (d) in the dataset, after transformations as described in Section 3.4. The x- and y-axes are based on the range in which these values occur in the dataset, showing a strong positive skew for height, width and area

Products included in the dataset fall into product, brand and sub-brand (Goldman et al., 2019). Due to the variety of products, the detector faces enormous within-class variability. This is even further increased by focusing on object regions as opposed to individual objects. In this case, variability not only comes from the difference in products, but also from the difference in number of products in a product region. Sub-brands, however, can differ only by very slight packaging differences, making it an even harder challenge. An example of this is shown in Figure 10.

Figure 10: Example of sub-brands posing challenging detection environment

5 Results

To evaluate the effectiveness of the different models, the results are discussed in terms of average precision (AP), considered the most important measure of object detection (Ren et al., 2015). After bounding boxes are predicted, they are counted as true or false positives, depending on their Intersection-over-Union (IoU) with ground-truth boxes. In accordance with the PASCAL VOC challenge (Everingham et al., 2008), boxes with an IoU greater than 0.5 are counted as true positives. Furthermore, different detections of the same object are counted as false positives. After the true and false positives are defined, all detections are ranked based on the confidence level of the prediction. Then, precision is calculated as the number of true positives over the number of predictions, and recall is the number of true positives over the total number of ground-truth bounding boxes. Now, the interpolated Average Precision (AP) is calculated as done by Salton and McGill (1983). The AP summarizes the precision/recall curve, being the mean precision at eleven equally spaced recall levels [0, 0.1, ..., 1]. This precision at each recall level is interpolated by taking the maximum precision at all recall levels exceeding the current one;

$$p_{interp}(r) = \underset{\geq r}{\max} p(\tilde{r}) \tag{9}$$

The average precision is then calculated as;

$$AP = \frac{1}{11} \sum_{r \in \{0.0, \ldots 0.1\}} p_{interp}(r) \tag{10}$$

Besides accuracy, it is important to consider runtime analysis. Like Goldman et al. (2019) and Ren et al. (2015), this paper reports frames per second (FPS) during inference. Also, graphs of the validation AP during training are shown in the Appendix, showing the learning curves of the model. Last, it is important to consider training speed in order to evaluate whether the model could easily be trained on a larger dataset. Therefore, for each training run the number of seconds per epoch (SPE) is reported.

First, as discussed in Section 3.3, the model is trained with a ResNet-50 backbone with different numbers of layers frozen. Results are displayed in Table 7. Looking at the reported

AP, it becomes clear that freezing the entire backbone leads to a considerably lower accuracy (0.2398), compared to a partly frozen (0.6097) or unfrozen (0.6100) backbone. As expected, the training time in seconds per epoch is slightly lower for the frozen backbone (40.45) compared to the partly frozen (49.84) and unfrozen (54.68) backbone. However, the differences in FPS and SPE are not deemed large enough to compromise on accuracy. Therefore, from here on all models are trained with a completely unfrozen backbone.

Table 7: Results from training Faster R-CNN model with ResNet-50 backbone with; 1) the weights completely frozen, 2) the first 5 layers of the backbone frozen and 3) the backbone completely unfrozen, respectively. Results show average precision (AP), frames per second (FPS) and seconds per epoch (SPE) as described above.

	AP	FPS	SPE
ResNet-50 frozen	0.2398	0.6434	40.45
ResNet-50 partly frozen	0.6097	0.6353	49.84
ResNet-50 unfrozen	0.6100	0.6232	54.68

Furthermore, as discussed in Subsection 3.3, this study compares performance between the Faster R-CNN model with a ResNet-18, ResNet-50 and ResNet-152 backbone, respectively. Results are displayed in Table 8. These backbones led to an AP of 0.6096, 0.6100 and 0.6531. Whereas the accuracy with ResNet-18 and ResNet-50 are very similar, it is noteworthy that the accuracy with the ResNet-152 backbone is considerably higher. Furthermore, what is interesting and perhaps somewhat unexpected, is the fact that the detection speed is highest with the ResNet-152 backbone (0.8415 FPS) compared to ResNet-18 (0.3174 FPS) and ResNet-50 (0.6232 FPS). Although the model with ResNet-152 backbone is deeper, increased detection speed is likely due to higher quality region proposals. Regarding training time, the model with ResNet-152 backbone needed considerably longer to train with 72.33 SPE. However, total training was performed in 3 hours, which is still within reasonable margins.

Table 8: Results from training Faster R-CNN model with ResNet-18, ResNet-50 and ResNet-152 backbone, respectively. The weights of the backbone are all unfrozen. Results show average precision (AP), frames per second (FPS) and seconds per epoch (SPE) as described above.

	AP	FPS	SPE
ResNet-18	0.6096	0.3174	25.84
ResNet-50	0.6100	0.6232	54.68
ResNet-152	0.6531	0.8415	72.23

Finally, ablation experiments were performed to establish the individual influence of anchor sizes and aspect ratios on model performance, as done by Ren et al. (2015). Since the ResNet-152

backbone led to the highest AP in the previous experiments, this backbone is also used for the ablation experiments. Results are shown in Table 9. As previously mentioned, the model with ResNet-152 and 3 anchor sizes and 3 aspect ratios led to an accuracy of 0.6531. Whereas two extra aspect ratios only slightly improved accuracy (0.6537), the addition of two extra anchor sizes increased accuracy to an AP of 0.6863. This AP was almost the same as the configuration with 5 anchor sizes and 5 aspect ratios (0.6819). Considering detection speed (FPS) and training speed (SPE), all configurations are feasible.

Table 9: Results from training Faster R-CNN model with ResNet-152 backbone with different anchor scales and anchor sizes. The weights of the backbone are all unfrozen. Results show average precision (AP), frames per second (FPS) and seconds per epoch (SPE) as described above.

anchor sizes	aspect ratios	AP	FPS	SPE
$(128^2, 256^2, 512^2)$	(2:1, 1:1, 1:2)	0.6531	0.8415	72.23
$(32^2, 64^2, 128^2, 256^2, 512^2)$	(2:1, 1:1, 1:2)	0.6863	0.816	74.55
$(128^2, 256^2, 512^2)$	(3:1, 2:1, 1:1, 1:2, 1:3)	0.6537	0.725	71.01
$(32^2, 64^2, 128^2, 256^2, 512^2)$	(3:1, 2:1, 1:1, 1:2, 1:3)	0.6819	0.7693	58.18

All in all, the best accuracy was achieved by a Faster R-CNN model with ResNet-152 backbone, anchor sizes $32^2, 64^2, 128^2, 256^2, 512^2$ and aspect ratios 2:1, 1:1, 1:2. To give an idea of what this model can achieve, Figure 11 shows an example of predictions this model makes on unseen data. A score threshold of 0.85 is applied to the predictions to avoid redundant bounding boxes.

Figure 11: Example predictions of Faster R-CNN model with ResNet-152 backbone, anchor sizes $32^2, 64^2, 128^2, 256^2, 512^2$, aspect ratios 2:1, 1:1, 1:2 and a score threshold of 0.85.

6 Conclusion and Discussion

This study employed a Faster R-CNN (Ren et al., 2015) model with a ResNet (He et al., 2016) backbone trained on ImageNet (Deng et al., 2012) to solve the task of automatic product region detection. The model was trained on 100 labelled images (see Section 4) and uses transfer learning to combat the issue of data scarcity. This approach is aimed at directly drawing up regions of identical products and is not trained on specific SKU's. Results are displayed in Tables 7, 8 and 9.

6.1 Summary of Results

Whilst some hyperparameters were previously determined (see Section 3.4 & 3.6), some other hyperparameters were experimented with. First of all, the number of frozen layers in a ResNet-50 backbone was experimented with, of which results are shown in Table 7. The highest accuracy was achieved with all backbone layers unfrozen, which led to an AP of 0.61 at 0.623 FPS.

Then, the depth of the model was experimented with, when a Faster R-CNN model with ResNet-18, ResNet-50 and ResNet-152 were compared, with all weights unfrozen. ResNet-18 and ResNet-50 backbones led to roughly the same accuracy, whereas the model with a ResNet-152 backbone outperformed these with an AP of 0.6531.

Last, ablation experiments were performed to assess the influence of extra anchor sizes and aspect ratios. The highest accuracy was achieved with anchor sizes $32^2, 64^2, 128^2, 256^2, 512^2$ and aspect ratios 2:1, 1:1, 1:2, which led to an AP of 0.6863 at 0.816 FPS. This is the highest AP that was achieved overall.

6.2 Discussion of Results

Figure 11 in Section 5 shows predictions of the model with the highest AP. In this example, it becomes clear that the model picked up grouping of similar products quite well. This makes for a promising outlook. However, more work is necessary to further improve the accuracy of the model.

The term "Garbage in, Garbage out", first used by Mellin (1957), shows that researchers have long been aware of the influence of data quality on machine learning results. Feed low quality data into a model, no matter how sophisticated, and results will be poor (Geiger et al., 2021). In this regard, it is important to critically think about the data used for this problem, described in Section 4.

First of all, it is important to consider the size of the dataset. Where models in other studies are trained on very large amounts of data (Goldman et al. (2019) use 11,762 images, Ren et al. (2015) train on PASCAL VOC 2007 (9,963 images), PASCAL VOC 2012 (11,530 images), COCO (328K images) and combinations of these), this study faces a data-scarcity problem with 100 labelled images. This makes it difficult to compare results of this study with those presented in Goldman et al. (2019) and Ren et al. (2015). To combat the problem of data scarcity, Yilmazer and Birant (2021) used a semi-supervised approach, where the model is first trained on the labeled dataset, then used to make predictions for the unlabeled data which are then added to the dataset and used to train the model again. However, unsatisfactory detection results in the first step likely make this an unfeasible approach for this study, since the predictions are too inaccurate and would feed to much noice into the dataset.

To optimally use the data at hand, data augmentation is performed as described in Section 3.4. Furthermore, transfer learning drastically reduces the amount of data required to train object detection models (Pan & Yang, 2009). Since the model was originally trained on ImageNet, which consists of over 1 million images (Deng et al., 2012), a small dataset in combination with data augmentation should suffice in learning the current task. However, since the results show that the model does pick up on grouping of products, it would be interesting to assess whether expanding the dataset could further improve accuracy.

Also, data was labelled by hand. Bounding boxes were applied to fit regions as closely as

possible, but not too close such that edges could be better detected. Data labelling was not outsourced, but everything was done in-house to have full control over the labelling process. The annotations were carefully checked after labelling was done. However, it is still vulnerable to human inconsistencies. Northcutt et al. (2021) have shown that even the test sets of the 10 most common computer vision, natural language programming and audio datasets contain on avarage at least 3.3% errors. For ImageNet, this is at least 6%. This study found that in real-world applications with high proportions of wrongly annotated data lower complexity models could be more beneficial than more complex models.

Besides human inconsistencies, there are some inconsistencies caused by the definition of a product region. Whereas for an individual product it is generally clear what the boundaries should be, this is not always the case for a product region. Sometimes product regions can be very clear (Figure 12). However, in some cases it becomes more complex to label product regions. An important issue is the fact that product regions can strongly overlap. In some situations this overlap is simply caused by the fact that rectangular bounding boxes are used to annotate product regions that are not necessarily rectangular (Figure 13). This becomes even more of a problem when images are strongly angled.

Figure 12: Example of clearly defined bounding boxes

Figure 13: High product region overlap due to rectangular bounding boxes

Next to strong overlap between regions, there are some more general concerns regarding product regions. For example, do we see the cardboard box that the products are in as part of the product region, or only the products themselves? Products in the background also often

present some ambiguity. In the example of Figure 14 the 6 products in the front clearly belong to the same product region. However, it is debatable whether the products lying in the back should be included or not. This ambiguity leads to two different ways to label the product region, with highly different outcomes in terms of the region. If this is not done consistenly, the model will have difficulties learning the desired outcome.

Figure 14: Products in background making product region ambiguous

Even when assuming data annotation is done without any errors, the data still presents a very challenging detection environment. First of all, shelves in the images are densely packed (Goldman et al., 2019), with a high number of products and sometimes also product regions per image (Figure 7). Furthermore, there is a large variability within the products in terms of shape, size and colours. This variability is enlarged through the use of product regions contrary to individual products. Now variability is not only caused by the many different products, but also by different numbers of products in every product region. This is made even more difficult by products being slightly turned, more in the background, shown from a different side or put upside down. Last, minor packaging differences in sub-brands (Figure 10) make it even harder to distinguish between different products to detect product regions (Goldman et al., 2019; Santra & Mukherjee, 2019). This seems to be an important driver of mistakes in the predictions shown in Figure 11.

The question is; would the model perform a lot better if images were less cluttered and densely packed? In the few pilot shops that Moonshop has launched, the layout is a lot more spacious, with more distance between different products. Since these shops are relatively small in terms of size and assortment, there are less very similar sub-products than in the current dataset. All in all, detection should be somewhat easier in the setting of the current Moonshops. However, if performance were a lot better in these circumstances, this would mean that the model is highly sensitive to changes in store layout. This has two important implications. First of all, if an unmanned shop gets restocked once a day, large clutter can build up during that

day, which could strongly influence the accuracy of the model. If customers become aware of this, they could mess with the algorithm by misplacing items and use this to their advantage. Furthermore, Moonshop offers a "white-label" solution, which is to be implemented in any small supermarket or vendor. This means that Moonshop will not always have control over store layout and assortment.

6.3 Future Research for the Concept of Moonshop

Further hyperparameter tuning leaves some room for improvement. For example, the hyperparameters that were experimented with in this study, like anchor sizes and aspect ratios, could be further studied in more different configurations. Also, experimenting with the learning rate and batch size can influence training (Smith et al., 2017). Besides further hyperparameter tuning, as previously mentioned, it would be interesting to evaluate the influence of expanding the dataset on detection accuracy. While the use of transfer learning greatly reduces the amount of data needed to achieve satisfactory detection results (Pan & Yang, 2009; Rawat & Wang, 2017), the high variability within product regions could mean that more data than usual is required to learn the specific task.

A different approach would be to first pre-train the model on individual product recognition with the SKU110K dataset (Goldman et al., 2019). Then, the weights from this could be used for the ResNet backbone, instead of those pre-trained on ImageNet. Since these weights are derived from training on a dataset even more similar to the current one, it could be hypothesized that the backbone would extract features that are even more relevant to this specific task. However, as shown by Pan and Yang (2009), transfer learning does not necessarily require the two tasks to be in the same domain, so it is uncertain whether pre-training the model on the SKU110K dataset would lead to an improved accuracy.

6.3.1 Different Approaches to Product Region Detection

Besides thinking about how to reach the maximum accuracy with the current approach, it is important to consider different approaches that could be beneficial to the concept of Moonshop. For example, the model could be trained to recognize individual products such as in Goldman et al. (2019). With this output, the labelers could then click on all the items of the same product, after which a product region is automatically drawn around the boundaries of these products. While this is a relatively simple and straight-forward approach, it suffers from some drawbacks. First of all, this involves extra action from the labelers, which is not certain to speed up the work over the labelers drawing up the product regions themselves. Especially in product regions with many products this would be a problem. Furthermore, Goldman et al. (2019) still fail to detect a lot of products in their approach.

As an extension of this approach, a model could be trained to learn whether two products are the same or not. This is often done using siamese neural networks (Chicco, 2021), which

consist of two networks with equal weights. Two images are passed through the networks, after which the two outcomes are compared through a distance measure. An example of this is DeepFace, where a siamese neural network is used for face verification using two images. Trained on four million facial images belonging to more than 4000 identities, this approach achieved an accuracy of 97.35% (Taigman et al., 2014). A similar structure as that used in DeepFace could be employed, where weights are copied and the new task is learned through transfer learning.

This could then be combined in a two-stage detector. First all the individual products are detected using an approach similar to Goldman et al. (2019). Then, all detections are compared with each other to say if they are identical products or not. With the outcomes of the second stage, product regions can be drawn and compared against the ground-truth boxes in our dataset. This way outliers within a product region could also be automatically detected.

It would be important to think about how to draw product regions. For example; what to do when the transitivity constraint does not hold? If *product a* is identical to *product b*, *product b* is identical to *product c*, but *product a* is not identical to *product c*, a problem could arise. This is very likely to occur in product regions with a large number of products.

This approach also suffers from some other important drawbacks. First of all, as previously discussed, accuracy of individual product detection is not extremely high (Goldman et al., 2019). When a lot of individual product detections are missing in the first stage, it would be hard to draw accurate product regions in stage two. Furthermore, this would mean that the current dataset would have to be relabeled in a different way. Also, identical products can look very dissimilar if they are shown from completely different angles. Last, the problem of occlusion will play an important role here, since many products are partially covered. Therefore it is unlikely that the second stage approach will lead to an accuracy as high as that of Taigman et al. (2014).

6.3.2 Broader Perspective on the Problem

A completely different avenue of product region detection would be to train SKU specific object detection models. With these individual products, product regions can be drawn. SKU specific approaches have shown to lead to considerably higher accuracy (Merler et al., 2007; Santra & Mukherjee, 2019). However, this approach also faces some drawbacks, the main one being that it requires lots of product specific data. This means that whenever a new shop is opened with a different assortment or when products' packaging is changed, Moonshop must go through the whole process of gathering and labelling data and training the model again (Santra & Mukherjee, 2019). This is a time-consuming process and could limit scalability. However, if this leads to a much higher accuracy, this could be a trade-off worth considering.

More generally, what also deserves some thought, is the choice between product regions or individual products. In the first place, this study decided to focus on detecting product regions rather than individual products to combat the problem of occlusion. However, this approach also poses some practical limitations. How do we deal with overlapping regions? What do we

do if a customer takes a product from a spot where two or more regions overlap? When the footage is judged by a labeller, they can simply see to which region the product belongs and click on this region. For making the shops fully autonomous, however, this could be a problem. In the end, the idea is to couple a customer action classified as taking a product to a specific product region. This way, we can conclude that a customer has taken a certain product. In this scenario, there are two ways to deal with overlapping regions. First of all, shops can be designed in such a way that there will never be overlapping regions. However, this might not always be possible in future shops and is again very sensitive to customers misplacing products. Second, another model could be applied to determine which product region a product more likely belongs to if it sits in overlapping regions. This, however, creates another step that would increase computational intensity and the possibility for more errors. Both ways to deal with this are not desirable. Therefore, product region detection can be a good intermediate step to aid labelers in doing their work more efficiently in the near future. In the transition to fully autonomous supermarkets, however, SKU specific individual product based methods might be a more efficient approach.

More generally, Moonshop should think about other measures to increase detection, rather than just computer vision. For example, like Amazon GO (Ives et al., 2019), it is important to consider alternative forms of input to aid in detecting which products a customer takes. The use of scales could not only help detect how many products a person takes, it would also pave the way for weighted products, such as fresh cheese, fruits and vegetables. Another option is Radio-Frequency IDentification (RFID) (Bottani et al., 2017; Santra & Mukherjee, 2019), which can be seen as the successor of the bar code and is for example used by Hema (Wang & Coe, 2021). This is mainly used to track on-shelf availability and product freshness (Bottani et al., 2017). However, sensor-based approaches are often very expensive and face scalability issues (Santra & Mukherjee, 2019).

All in all, the problem can be approached from many different angles, all with their own drawbacks and advantages. For now, product region detection could be a helpful tool to aid labelers. As Ives et al. (2019) argues, a well developed Just Walk Out technology could disrupt not only grocery shopping, but the wider retail sector. Therefore, Moonshop should carefully think about the right approach, strongly keeping scalability in mind. When this is done well, a bright future lies ahead.

References

Abraham, K. G., Haltiwanger, J. C., & Rendell, L. E. (2020). How tight is the us labor market? *Brookings Papers on Economic Activity*, *2020*(1), 97–165.

Agarwal, N., Chiang, C.-W., & Sharma, A. (2018). A study on computer vision techniques for self-driving cars. *International Conference on Frontier Computing*, 629–634.

Alam, M. M., & Islam, M. T. (2019). Machine learning approach of automatic identification and counting of blood cells. *Healthcare technology letters*, *6*(4), 103–108.

Arief-Ang, I. B., Hamilton, M., & Salim, F. D. (2018). A scalable room occupancy prediction with transferable time series decomposition of co2 sensor data. *ACM Transactions on Sensor Networks (TOSN)*, *14*(3-4), 1–28.

Benjdira, B., Khursheed, T., Koubaa, A., Ammar, A., & Ouni, K. (2019). Car detection using unmanned aerial vehicles: Comparison between faster r-cnn and yolov3. *2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS)*, 1–6.

Bickel, S. (2006). Ecml-pkdd discovery challenge 2006 overview. *ECML-PKDD Discovery Challenge Workshop*, 1–9.

Bottani, E., Bertolini, M., Rizzi, A., & Romagnoli, G. (2017). Monitoring on-shelf availability, out-of-stock and product freshness through rfid in the fresh food supply chain. *International Journal of RF Technologies*, *8*(1-2), 33–55.

Bozinovski, S. (1981). *Teaching space: A representation concept for adaptive pattern classification* (tech. rep.). COINS Technical Report, University of Massachusetts at Amherst.

Bozinovski, S., & Fulgosi, A. (1976). The influence of pattern similarity and transfer learning upon the training of a base perceptron b2. *Proceedings of Symposium Informatica*, 3–121.

Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., & Kalinin, A. A. (2020). Albumentations: Fast and flexible image augmentations. *Information*, *11*(2), 125.

Chellapilla, K., Puri, S., & Simard, P. (2006). High performance convolutional neural networks for document processing. *Tenth international workshop on frontiers in handwriting recognition.*

Chicco, D. (2021). Siamese neural networks: An overview. *Artificial Neural Networks*, 73–94.

Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M., & Schmidhuber, J. (2011). Flexible, high performance convolutional neural networks for image classification. *Twenty-second international joint conference on artificial intelligence.*

Cui, Y. G., van Esch, P., & Jain, S. P. (2021). Just walk out: The effect of ai-enabled checkouts. *European Journal of Marketing.*

Deloitte Digital. (2018). Connected stores. transforming store fleet through connectivity [Last accessed 29 May 2022]. https://www2.deloitte.com/content/dam/Deloitte/ar/Documents/Consumer_and_Industrial_Products/Connected_Stores_Deloitte_POV.pdf

Deng, J., Berg, A., Satheesh, S., Su, H., Khosla, A., & Fei-Fei, L. (2012). Imagenet large scale visual recognition competition 2012 (ilsvrc2012). *See net. org/challenges/LSVRC*, *41*.

Do, C. B., & Ng, A. Y. (2005). Transfer learning for text classification. *Advances in neural information processing systems*, *18*.

Esch, P. v., Cui, Y., & Jain, S. P. (2021). Stimulating or intimidating: The effect of ai-enabled in-store communication on consumer patronage likelihood. *Journal of Advertising*, *50*(1), 63–80.

Everingham, M., Zisserman, A., Williams, C. K., Van Gool, L., Allan, M., Bishop, C. M., Chapelle, O., Dalal, N., Deselaers, T., Dorkó, G., et al. (2008). The pascal visual object classes challenge 2007 (voc2007) results.

Food Marketing Institute. (2008). Marketing costs [Last accessed 11 July 2022]. https://www.fmi.org/docs/facts-figures/marketingcosts.pdf?sfvrsn=2

Forsythe, E., Kahn, L. B., Lange, F., & Wiczer, D. G. (2020). *Searching, recalls, and tightness: An interim report on the covid labor market* (tech. rep.). National Bureau of Economic Research.

Franco, A., Maltoni, D., & Papi, S. (2017). Grocery product detection and recognition. *Expert Systems with Applications*, *81*, 163–176.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, *36*, 193–202.

Geiger, R. S., Cope, D., Ip, J., Lotosh, M., Shah, A., Weng, J., & Tang, R. (2021). "garbage in, garbage out" revisited: What do machine learning application papers report about human-labeled training data? *Quantitative Science Studies*, *2*(3), 795–827.

Gevers, T., & Smeulders, A. W. (1999). Color-based object recognition. *Pattern recognition*, *32*(3), 453–464.

Girshick, R. (2015). Fast r-cnn. *Proceedings of the IEEE international conference on computer vision*, 1440–1448.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 580–587.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 249–256.

Goldman, E., Herzig, R., Eisenschtat, A., Goldberger, J., & Hassner, T. (2019). Precise detection in densely packed scenes. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5227–5236.

Hajiramezanali, E., Zamani Dadaneh, S., Karbalayghareh, A., Zhou, M., & Qian, X. (2018). Bayesian multi-domain learning for cancer subtype discovery from next-generation sequencing count data. *Advances in Neural Information Processing Systems*, *31*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Higa, K., & Iwamoto, K. (2018). Robust estimation of product amount on store shelves from a surveillance camera for improving on-shelf availability. *2018 IEEE International Conference on Imaging Systems and Techniques (IST)*, 1–6.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, *29*(6), 82–97.

Hubel, D. H., & Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, *148*, 574–591.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International conference on machine learning*, 448–456.

Ives, B., Cossick, K., & Adams, D. (2019). Amazon go: Disrupting retail? *Journal of Information Technology Teaching Cases*, *9*(1), 2–12.

Jain, S., Pise, N., & Jagtap, V. (2015). Computer aided melanoma skin cancer detection using image processing. *Procedia Computer Science*, *48*, 735–740.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), 541–551.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Leung, M. K., Xiong, H. Y., Lee, L. J., & Frey, B. J. (2014). Deep learning of the tissue-regulated splicing code. *Bioinformatics*, *30*(12), i121–i129.

Li, M., Zhang, Z., Lei, L., Wang, X., & Guo, X. (2020). Agricultural greenhouses detection in high-resolution satellite images based on convolutional neural networks: Comparison of faster r-cnn, yolo v3 and ssd. *Sensors*, *20*(17), 4938.

Li, Z., He, Y., Keel, S., Meng, W., Chang, R. T., & He, M. (2018). Efficacy of a deep learning system for detecting glaucomatous optic neuropathy based on color fundus photographs. *Ophthalmology*, *125*(8), 1199–1206.

Ma, J., Sheridan, R. P., Liaw, A., Dahl, G. E., & Svetnik, V. (2015). Deep neural nets as a method for quantitative structure–activity relationships. *Journal of chemical information and modeling*, *55*(2), 263–274.

Maitra, D. S., Bhattacharya, U., & Parui, S. K. (2015). Cnn based common approach to handwritten character recognition of multiple scripts. *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, 1021–1025.

Marr, D. (1982). *Vision: A computational investigation into the human representation and processing of visual information*. MIT press.

McCarthy, J., & Minsky, M. (1956). The dartmouth summer research project on artificial intelligence. *Artificial intelligence: past, present and future*.

McKinsey. (2017). The future of grocery — in store and online [Last accessed 13 July 2022]. https://www.mckinsey.com/industries/retail/our-insights/the-future-of-grocery-in-store-and-online.pdf

Mellin, W. D. (1957). Work with new electronic 'brains' opens field for army math experts. *The Hammond Times*, *10*, 66.

Merler, M., Galleguillos, C., & Belongie, S. (2007). Recognizing groceries in situ using in vitro training data. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 1–8.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. *Icml*.

Newell, A., & Simon, H. A. (1956). The logic theory machine - a complex information processing system. *IRE Transactions on information theory*, *2*, 61–79.

Northcutt, C. G., Athalye, A., & Mueller, J. (2021). Pervasive label errors in test sets destabilize machine learning benchmarks. *arXiv preprint arXiv:2103.14749*.

Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, *22*(10), 1345–1359.

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, *12*(1), 145–151.

Radiuk, P. M. (2017). Impact of training set batch size on the performance of convolutional neural networks for diverse datasets.

Rawat, W., & Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, *29*(9), 2352–2449.

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

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, *28*.

RTS. (2022). Food waste in america in 2021 [Last accessed 11 July 2022]. https://www.rts.com/wp-content/uploads/2021/04/RTS_Food_Waste_Guide_2021.pdf

Salton, G., & McGill, M. J. (1983). *Introduction to modern information retrieval*. mcgraw-hill.

Santra, B., & Mukherjee, D. P. (2019). A comprehensive survey on computer vision based approaches for automatic identification of products in retail store. *Image and Vision Computing*, *86*, 45–63.

Scime, L., & Beuth, J. (2018). Anomaly detection and classification in a laser powder bed additive manufacturing process using a trained computer vision algorithm. *Additive Manufacturing*, *19*, 114–126.

Smith, S. L., Kindermans, P.-J., Ying, C., & Le, Q. V. (2017). Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

Szegedy, C., Reed, S., Erhan, D., Anguelov, D., & Ioffe, S. (2014). Scalable, high-quality object detection. *arXiv preprint arXiv:1412.1441*.

Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1701–1708.

Tan, L., Huangfu, T., Wu, L., & Chen, W. (2021). Comparison of yolo v3, faster r-cnn, and ssd for real-time pill identification.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, *49*, 433–460.

Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, *104*(2), 154–171.

U.S. Department of Labor. (2020). Productivity and costs by industry. wholesale trade and retail trade industries [Last accessed 11 July 2022]. https://www.bls.gov/news.release/pdf/prin1.pdf

Wang, Y., & Coe, N. M. (2021). Platform ecosystems and digital innovation in food retailing: Exploring the rise of hema in china. *Geoforum*, *126*, 310–321.

Xiong, H. Y., Alipanahi, B., Lee, L. J., Bretschneider, H., Merico, D., Yuen, R. K., Hua, Y., Gueroussov, S., Najafabadi, H. S., Hughes, T. R., et al. (2015). The human splicing code reveals new insights into the genetic determinants of disease. *Science*, *347*(6218), 1254806.

Yilmazer, R., & Birant, D. (2021). Shelf auditing based on image classification using semi-supervised deep learning to increase on-shelf availability in grocery stores. *Sensors*, *21*(2), 327.

Ying, X. (2019). An overview of overfitting and its solutions. *Journal of physics: Conference series*, *1168*(2), 022022.

Appendix

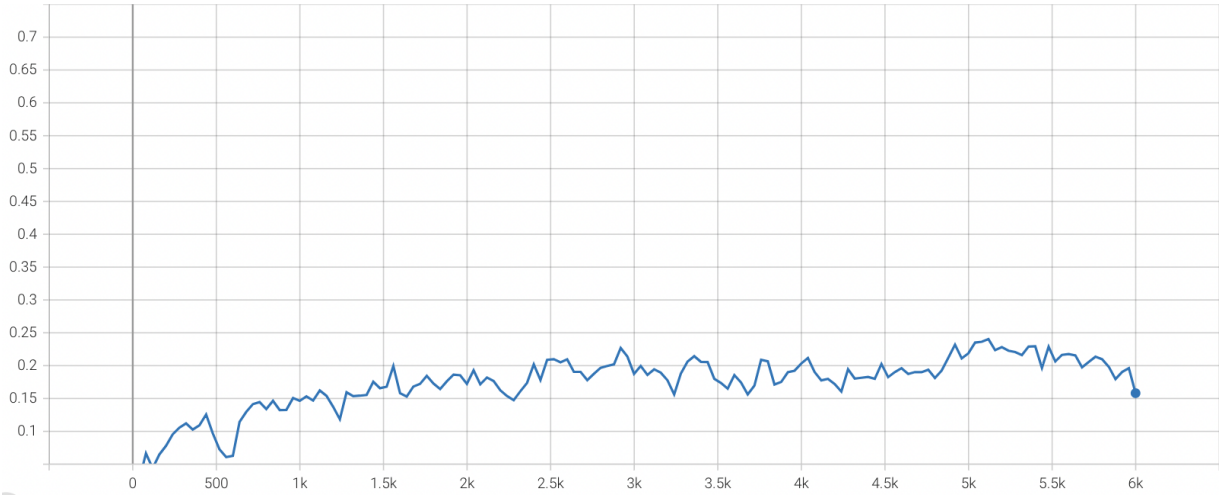


Figure 15: Learning curve in terms of validation AP of Faster R-CNN model with ResNet-50 backbone. Weights of the entire backbone are frozen. Anchor sizes $128^2, 256^2, 512^2$ and aspect ratios 2:1, 1:1, 1:2. Y-axis shows validation AP, x-axis shows the number of training steps.

Figure 16: Learning curve in terms of validation AP of Faster R-CNN model with ResNet-50 backbone. Weights of the first 5 layers of the backbone are frozen. Anchor sizes $128^2, 256^2, 512^2$ and aspect ratios 2:1, 1:1, 1:2. Y-axis shows validation AP, x-axis shows the number of training steps.

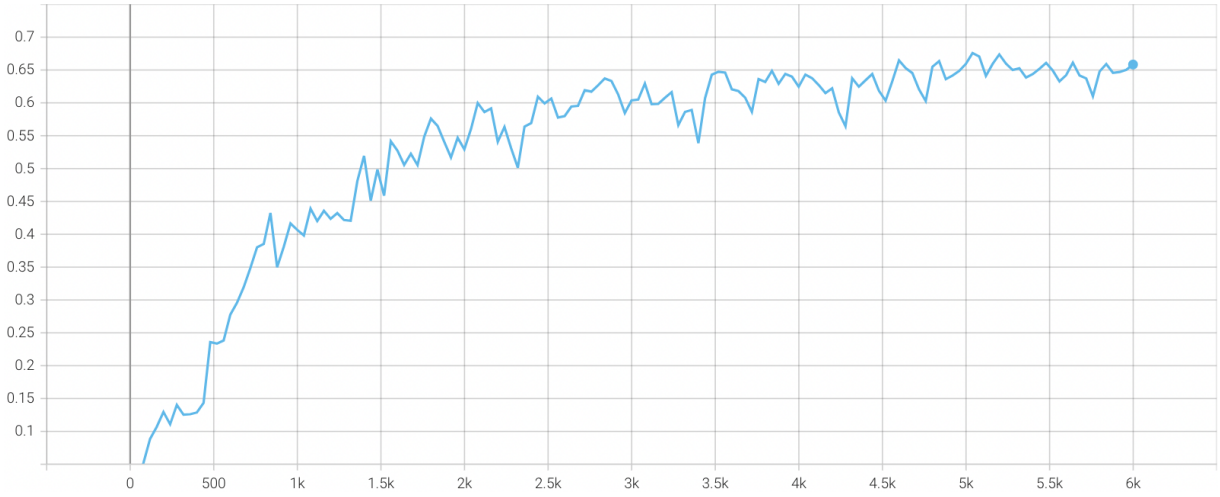


Figure 17: Learning curve in terms of validation AP of Faster R-CNN model with ResNet-50 backbone. Weights of the entire backbone are unfrozen. Anchor sizes $128^2, 256^2, 512^2$ and aspect ratios 2:1, 1:1, 1:2. Y-axis shows validation AP, x-axis shows the number of training steps.

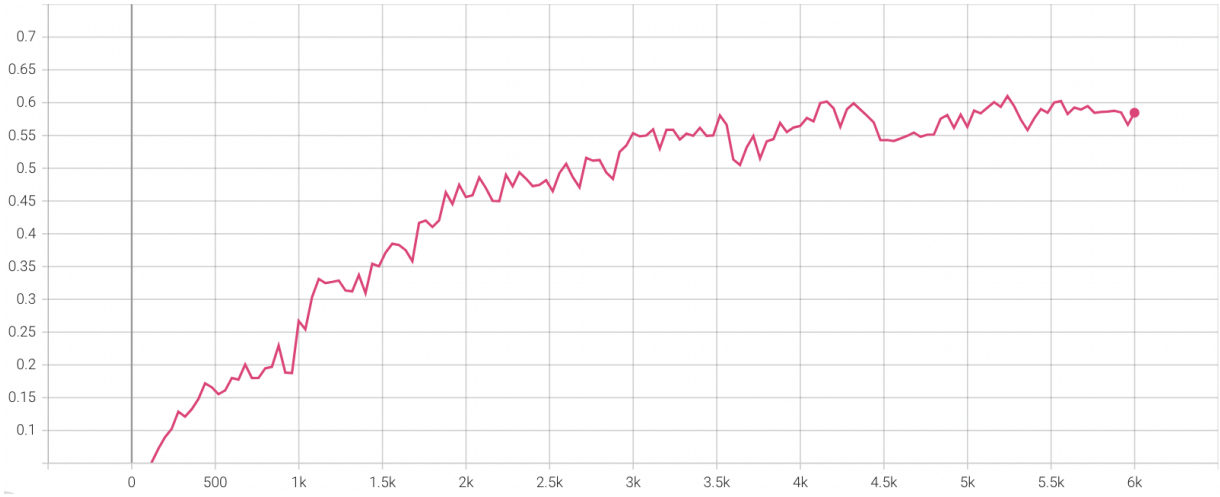


Figure 18: Learning curve in terms of validation AP of Faster R-CNN model with ResNet-18 backbone. Weights of the entire backbone are unfrozen. Anchor sizes $128^2, 256^2, 512^2$ and aspect ratios 2:1, 1:1, 1:2. Y-axis shows validation AP, x-axis shows the number of training steps.

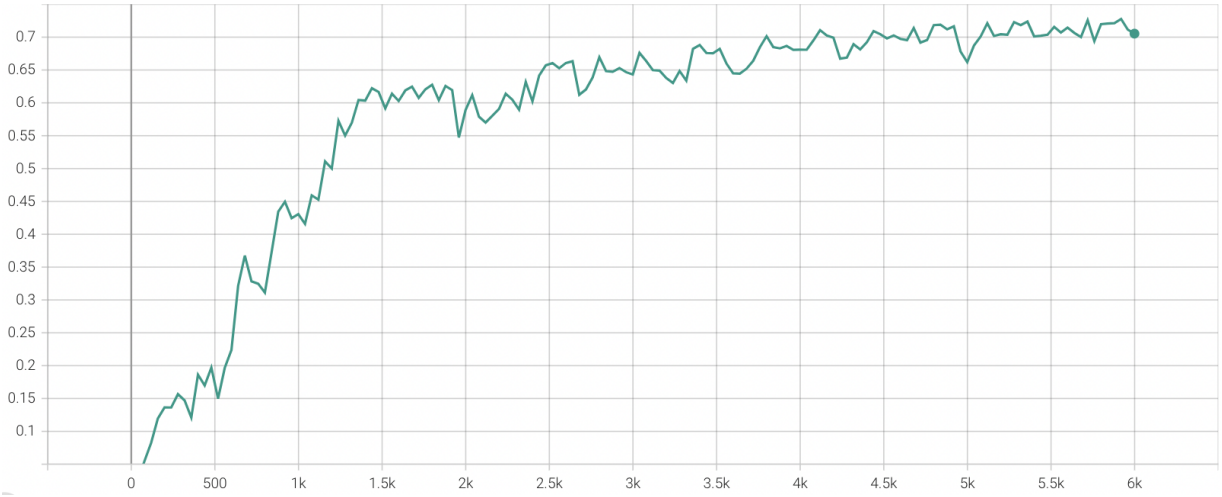


Figure 19: Learning curve in terms of validation AP of Faster R-CNN model with ResNet-152 backbone. Weights of the entire backbone are unfrozen. Anchor sizes $128^2, 256^2, 512^2$ and aspect ratios 2:1, 1:1, 1:2. Y-axis shows validation AP, x-axis shows the number of training steps.

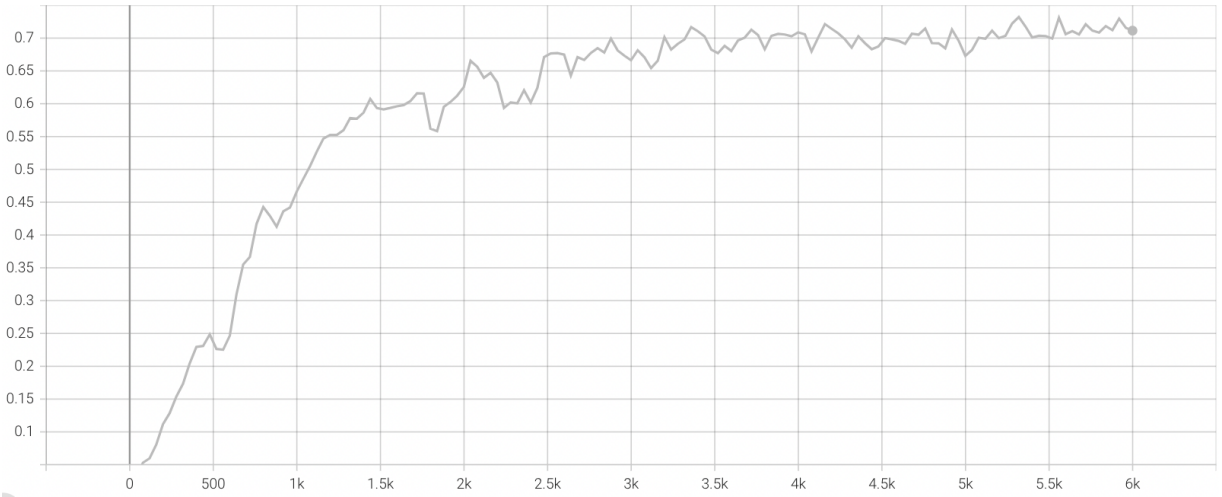


Figure 20: Learning curve in terms of validation AP of Faster R-CNN model with ResNet-152 backbone. Weights of the entire backbone are unfrozen. Anchor sizes $32^2, 64^2, 128^2, 256^2, 512^2$ and aspect ratios 2:1, 1:1, 1:2. Y-axis shows validation AP, x-axis shows the number of training steps.

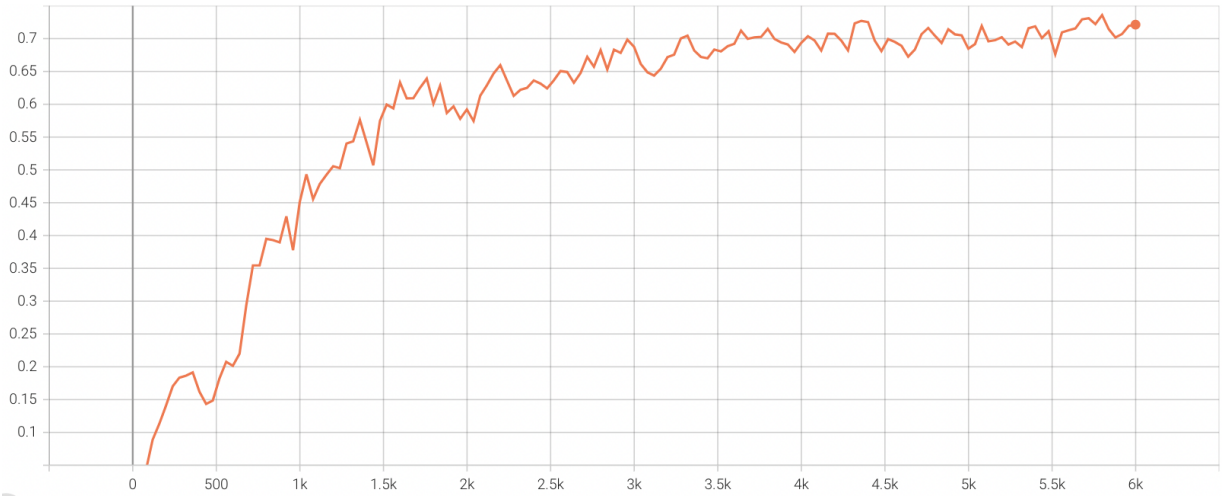


Figure 21: Learning curve in terms of validation AP of Faster R-CNN model with ResNet-152 backbone. Weights of the entire backbone are unfrozen. Anchor sizes $128^2, 256^2, 512^2$ and aspect ratios 1:3, 2:1, 1:1, 1:2. 1:3. Y-axis shows validation AP, x-axis shows the number of training steps.

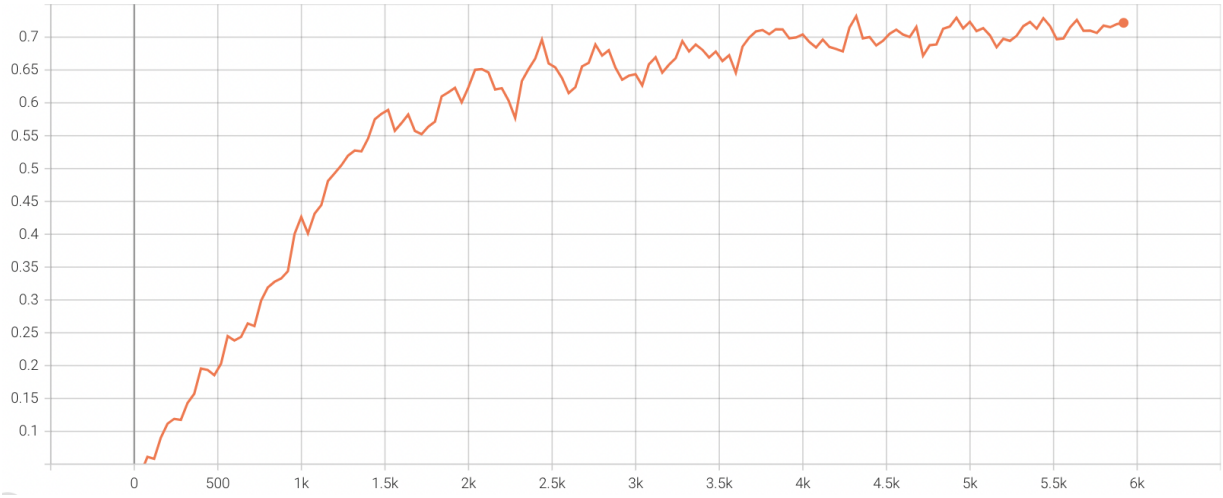


Figure 22: Learning curve in terms of validation AP of Faster R-CNN model with ResNet-152 backbone. Weights of the entire backbone are unfrozen. Anchor sizes $32^2, 64^2, 128^2, 256^2, 512^2$ and aspect ratios 1:3, 2:1, 1:1, 1:2. 1:3. Y-axis shows validation AP, x-axis shows the number of training steps.

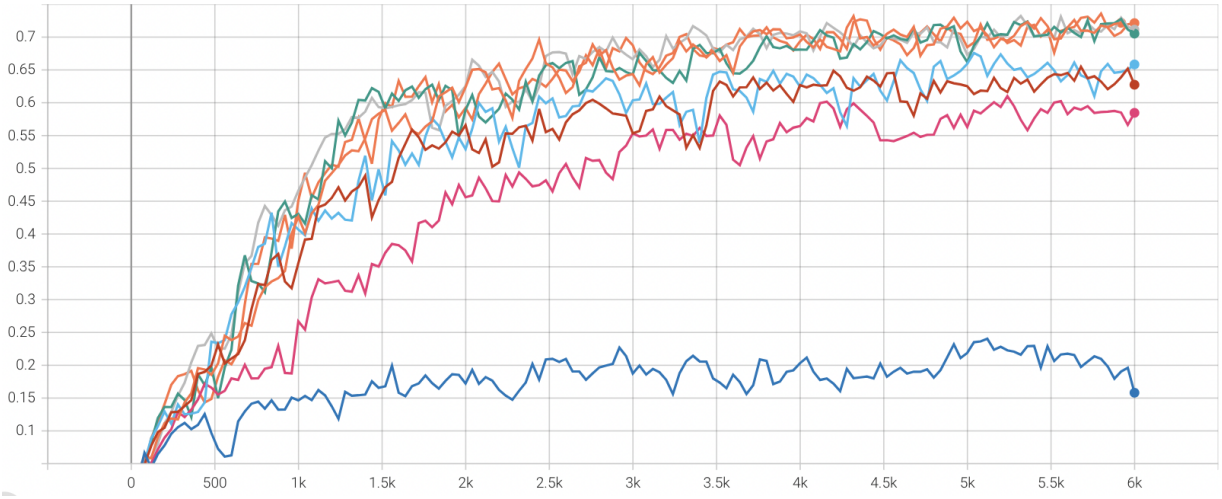


Figure 23: Overview of all learning curves in terms of validation AP in one graph. Y-axis shows validation AP, x-axis shows the number of training steps. Different colours correspond to the same models as in the figures above, in the same order as previously described; *blue* = ResNet-50 with backbone frozen, *red* = ResNet-50 with backbone partially frozen, *light blue* = ResNet-50 with backbone unfrozen, *pink* = ResNet-18 with backbone unfrozen, *green* = ResNet-152 with backbone unfrozen, *grey* = ResNet-152 with backbone unfrozen & extra anchor sizes, *dark orange* = ResNet-152 with backbone unfrozen & extra aspect ratios, *orange* = ResNet-152 with backbone unfrozen, extra anchor sizes & aspect ratios