Detecting strawberries using different Convolutional Neural Networks

Jeroen Bechtold
Supervisor(s): Junhan Wen, Thomas Abeel
EEMCS, Delft University of Technology, The Netherlands
22-6-2022

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Abstract

This paper tries to combat the food waste of strawberries during the harvesting steps. An automatic pipeline must be established to combat this food waste. One of the steps needed in this pipeline is detecting strawberries in images. Therefore, this paper aims to find out which Convolutional Neural Network (CNN) can be best used to detect strawberries. Faster r-cnn, Mask r-cnn and RetinaNet are compared against each other using different setting. Mask r-cnn achieved the highest average bounding box and segmentation mAP with 51.63 and 73.20 respectively.

1 Introduction

Strawberries are part of the soft fruits, making them highly susceptible to waste. Research in Ontario, Canada, showed that 56% of all edible strawberries are wasted. [1]

Battling part of this waste may be done by having a more efficient and unbiased selection of which strawberries to harvest. Usually, humans, who are highly susceptible to personal bias, have to make these quick decisions. This selection may be improved by having an algorithm detect which strawberries are ready to be harvested or predict when they will be ready.

There first needs to be an algorithm that can segment strawberries from images. These segments can then later be used to predict their ripeness. Some successful research has already been done on selecting fruits from images. [2; 3] There are many models to detect objects [4]. Some research even tackles strawberry detection using some of these models [5; 6]. However, many more of these models can be used to detect strawberries.

Regardless, the following question remains: which CNN model does the best segmentation for the dataset provided by the supervisors? And as a sub-question, how do the training settings influence the final results?

To answer these questions, we will look into the following segmentation algorithms: Faster r-cnn [7], Mask r-cnn [8], RetinaNet [9]. The best accuracy of their prediction capabilities will be the primary goal of this research.

2 Methodology

An algorithm with visual prediction capabilities must be implemented to detect strawberries in images. This algorithm will then need data to train on so that the algorithm can make correct decisions.

In the field of strawberry detection, there has already been specific research to compare different algorithms to see which one produces the best results [5]. Out of all these algorithms, a CNN seemed the most promising, with an accuracy of 88 % However, this accuracy was achieved in different circumstances than our strawberries. That paper also only checked the performance of one CNN. Therefore this research focuses on comparing different CNNs by evaluating their accuracies. How CNNs operate and which CNNs are chosen will be discussed in section 2.1 the motivation for why those specific CNNs are chosen will also be discussed in their respective subsection. After the CNNs are presented and motivated, the data exploration will be discussed in section 2.2. Lastly, the research setup will be discussed in section 2.3.

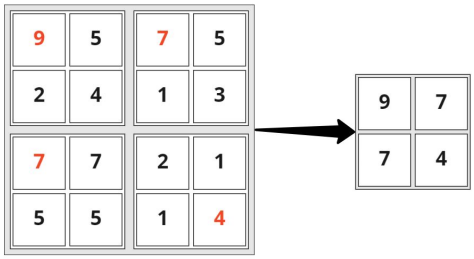


Figure 1: Left a 4 by 4 feature divided up in 2 by 2 areas can be seen. Right of the arrow you can see the result of the max pooling

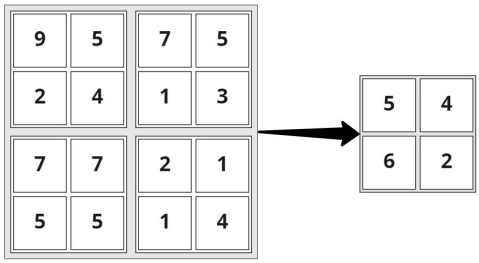


Figure 2: Left a 4 by 4 feature divided up in 2 by 2 areas can be seen. Right of the arrow you can see the result of the average pooling

2.1 Convolutional Neural Networks (CNN)

A convolutional neural network is a deep learning algorithm. This means that the algorithm consists of layers of neurons. These neurons can process the information they get and pass it on to the next layer. There are different types of layers which can be used. Pooling, Convolutional and Fully connected layers are the main layers used in such models.

Pooling layers reduce the resolution of the incoming image by doing either max pooling or average pooling. Max pooling is when, for example, a four-by-four feature comes in, and they are sliced into two-by-two squares. The max value is taken from this window and saved in the filtered output. This can be seen in figure 1 Average pooling is when the average of the window will be taken instead of the max. Which can be seen in figure 2

Convolutional layers learn about spatial features by convolving a filter over the input feature. These filters learn to detect specific features, for example edges. A convolution can be seen in figure 3.

Doing all these operations on the whole image would be time-consuming and inefficient. Therefore, most CNNs use a region proposal network that predicts regions containing objects. For this paper, a ResNet50 (based) backbone was chosen since all models are compatible with this model and ResNet50 is often used as Region Proposal Network (RPN).

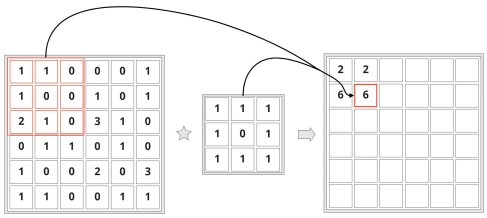


Figure 3: Left a 9 by 9 matrix can be seen. The red square indicates which values are taken into account for the current operation. In the middle you can see the filter which will be convolved by. The right most square contains the current results. Where the red square indicates the result of the convolution of the original red square with the filter.

Faster r-cnn

Faster r-cnn has been used to detect different kinds of fruits [10]. One of which was the strawberry. However, this research mainly focused on combining near-infrared and RGB images to detect sweet peppers. Therefore further research into faster r-cnn for strawberry detection is needed to know how well it will perform.

Faster r-cnn is based on the fast r-cnn, which is in itself based on r-cnn. [11; 12] As the name suggests, Faster r-cnn is a faster version of these algorithms and a more accurate version [7].

A region proposal network is used to extract interesting locations in the image. These patches consist of different aspect ratios and sizes. After the previously created anchor boxes by the RPN, faster r-cnn filters these proposed regions by applying Non-Maximum Suppression (NMS). This makes sure that there are no majorly overlapping anchor boxes.

These new regions are then fed into the Region Of Interest (ROI) pooling layer. ROI pooling layer performs a max-pooling operation so that the ROI can be transformed into the correct size for the fully connected layers at the end. From these fully connected layers, the prediction of whether it is a strawberry or not is made.

Faster r-cnn was chosen because it has proven itself and its accuracy by detecting different fruits in various studies [2]. However, strawberries were only tested in one study and only on a low number of Google images. Therefore we are interested in seeing how this CNN will perform in a more controlled environment with more data.

Mask r-cnn

Mask r-cnn has previously been used to detect ripe and unripe strawberries [6]. This research provided a highly promising precision rate of 95.78%. Therefore, in this research mask r-cnn will be one of the algorithms tested for our setup.

Mask r-cnn follows a similar design as the faster r-cnn. The main difference is that the CNN now does instance segmentation as well. For instance segmentation, a pixel-by-pixel classification is made. This classification is done in parallel with the classification of the bounding box. So the classification is not dependent on the mask but the original bounding box.

Mask r-cnn was chosen since there was a paper which mainly addressed this CNN using it for strawberry predictions, and they achieved a great result [5].

RetinaNet

RetinaNet is a one-stage detector. That means that the model produces the bounding box and predicts the class at the same time. Both faster r-cnn and mask r-cnn do not do this.

RetinaNet introduces the use of focal loss. Due to the focal loss, RetinaNet can produce a magnitude more anchor boxes than most CNNs. Usually, many small misclassifications could overwhelm the model. However, the focal loss counteracts this by punishing misclassifications more harshly and tiny errors less.

RetinaNet was chosen because it has previously proven its usefulness by providing fast and accurate object detection[9]. It has even been used to detect apples with great success[13]. Now the question remains how well RetinaNet will do on strawberry detection and how good it will be compared to the other CNNs?

2.2 Data exploration

Now the models still need some data to be trained on. For this research, a dataset is provided by the TU Delft[1], Birds.ai[2] and Delphy[3]. This dataset contains images from May 2021 until November 2021 from 3 different cameras. These images are taken every hour and in a controlled environment. The camera is in the exact location and takes pictures of the same strawberries. Then these pictures are taken every hour. So a total of 24 pictures a day.

These images are accompanied by a large JSON file which contains links between images and strawberry locations. This pixel wise locating is done with a polygon A polygon contains one strawberry and traces the outside of the strawberry. Bounding box information is also provided. One bounding box fits rectangularly around the accompanied polygon.

(a) One of the original images

(b) The ground truth of figure 4 a

Figure 4: Left in (a) you can see that the original image and (b) plots the ground truth over this image

Data preparation

Not all images are useable due to missing ground truth or a lack of strawberries. Thus, the data needs to be filtered. This needs to be done so that the CNNs have valid data to be trained on.

[1] https://www.tudelft.nl/
[2] https://birds.ai/
[3] https://delphy.nl/

In total, 12.155 images are provided. These images are filtered so that only images containing ground truths of strawberries are used. After this filtering, 4.370 images are left containing a total of 130.665 strawberries. One of these images and the ground truth have been visualised, can be seen in figure 4. As can be seen in this figure, the ground truth contains the bounding box information and pixel segmentation. This information will be used for the different CNN models.

The datasets of the different cameras are kept separate to ensure that the test set is not contaminated by having training data inside. Therefore the same camera is always used as the test dataset. This dataset contains 1.287 images and 33.274 strawberries. All other images can be used during the training.

2.3 Research Setup

Multiple CNNs must be tested on the dataset provided to answer the research question. Doing this efficiently will be done mainly using the detectron2 framework [14]. This framework is specifically designed for researchers. Detectron2 is therefore created so that the researchers can easily tweak and edit the models to fit the researchers' needs. Using this framework also provides a controlled environment for the models and ensures that they are easily comparable. The models we implemented using detectron2 are Mask r-cnn [8], Faster r-cnn [7] and RetinaNet [9].

These models will be using the loss functions proposed in their original papers. The models will be initialized with randomized values as their weights. All other settings of the CNNs are kept the as the original authors proposed. The only real difference that is made is the learning rate.

The algorithms' learning rate has been made the same to make a more fair comparison. The new learning rate consists of 4 stages. The first stage is the warm-up stage which linearly increases the learning rate from close to 0 to the base learning rate value. This warm-up stage lasts for 10% of the iterations. The warm-up stage is used so that the model does not overtrain on the bias of the first few iterations. After 75% and 90% of the iterations, the base learning rate decreases by a factor of 10.

Training for all models will be done on the DelftBlue supercomputer [15]. The hardware on which all the models have trained consists of 1x AMD EPYC 7402 CPUs with 24 cores running at 2.80 GHz and 1x NVIDIA Tesla V100S GPU containing 32GB VRAM.

The comparison metrics used to analyze the different models will be discussed in the Results section of this paper. The training loss and the validation loss are calculated by summing the different loss values produced by the model. The FPN produces a localisation loss and a classification loss. The losses are calculated by the smooth L1 loss[4] and the cross-entropy loss[5] respectively. These same losses are calculated for the Bounding Box selection and classification part of the CNN. All these losses are then summed into the total loss. For mask r-cnn, the segmentation loss is also calculated by the smooth L1 loss and cross-entropy loss.

[4] pytorch.org/docs/stable/generated/torch.nn.SmoothL1Loss

[5] pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss

$$Precision = \frac{tp}{tp + fp}$$

Figure 5: The formula of how to calculate the precision. With TP being True Positive, and FP being False Positive.

3 Results

After generating the models, they finally were compared against each other. However, first, an objective measurement must be established. This measurement will be done by comparing the precision of the models against each other. The precision of a model is calculated by the formula above in figure 5. In this image TP = True positive, FP = false positive. Therefore, it calculates how many of the predicted strawberries are correct predictions and thus contain a strawberry.

However, the question remains: at what confidence level should the precision be calculated? When the model is 50% sure, it is a strawberry or at 90%? Choosing a specific confidence level would add bias to the evaluation. Therefore, the mean average precision (mAP) calculates the precision from a 50% confidence interval to a 95% confidence level. This is done in intervals of 0.05%. The average is then taken of these values, which produces the mAP. In object detection, the mAP is considered a standard metric to evaluate a model. The Bounding Box (BBox) mAP will be calculated for all the models. For mask r-cnn, the segmentation (segm) mAP will also be calculated.

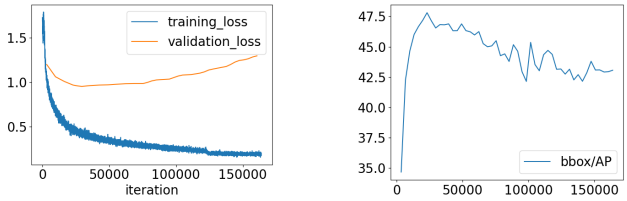
Since we have three cameras' datasets, we established that camera one is always the test set. Having a standard test set makes sure that any difference in results is due to the training of the CNNs and not test data variability. Many different training setups were used to gather data. All the generated results can be found in the appendix.

3.1 Single camera training

To start the testing, we started by creating a baseline. This baseline consisted of using one camera as a data set and testing it with 100 epochs (training the model on all the data 100 times) using different learning rates. The results can be seen in table 1. The training seemed to go well, and the training loss decreased during the training, which seemed promising. However, after further inspection, the model seemed to overfit. The overfitting can be seen in figure 6

Model	mAP BBox	mAP Seg	LR	Epoch
Faster r-cnn	44.73	-	0.2	100
Mask r-cnn	44.87	66.71	0.2	100
RetinaNet	45.91	-	0.2	100
Faster r-cnn	44.85	-	0.02	100
Mask r-cnn	44.88	67.07	0.02	100
RetinaNet	43.95	-	0.02	100
Faster r-cnn	43.06	-	0.002	100
Mask r-cnn	41.81	40.92	0.002	100
RetinaNet	43.64	-	0.002	100

Table 1: The results of training for 100 epoch on the data of camera 5.



(a) Training loss (b) mAP BBox

Figure 6: Left in (a) you can see that the validation loss increases even though the training loss keeps on decreasing. This shows that the model is overfitting. The training and validation loss is the same as the total loss of the respective classes. The overfitting is confirmed by the BBox mAP. At the start it increases, but after more training it decreases again. The model shown is the faster r-cnn trained with 0.002 Lr and 100 epochs trained on one camera.

3.2 Multi camera training

One hundred epochs and one camera as training data produced overfitting results. A new training setup was created to test if this is due to too few training data. Adding the dataset of an additional camera will create a more varied training set and could therefore improve the accuracy of the models and maybe negate some of the overfitting. However, training for 100 epochs with twice the data would be a bad comparison since the training iterations are not comparable. Therefore the training was done with 50 epochs. The result of this training can be seen in table 2

Model	mAP BBox	mAP Seg	LR	Epoch
Faster r-cnn	DIV	-	0.2	50
Mask r-cnn	DIV	DIV	0.2	50
RetinaNet	DIV	-	0.2	50
Faster r-cnn	48.14	-	0.02	50
Mask r-cnn	47.41	69.28	0.02	50
RetinaNet	48.68	-	0.02	50
Faster r-cnn	47.03	-	0.002	50
Mask r-cnn	47.69	67.91	0.002	50
RetinaNet	47.90	-	0.002	50

Table 2: The results of training for 50 epochs on the data of two different cameras. DIV means that during training, the training loss kept increasing and thus diverged.

When the double camera setup produces a result, it has a better mAP. Sometimes, the training on the two camera dataset did not provide a final model. No final model was reacted training loss diverged. The following formula was used to compare the one dataset results against the two camera dataset:

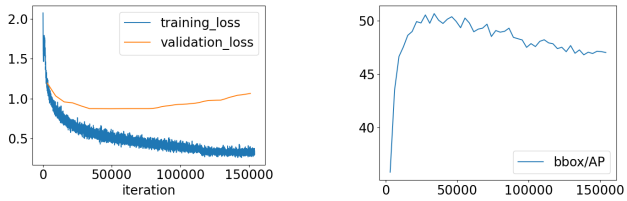
$$AVG_{improvement} = \frac{1}{n}\sum_{1}^{n}\frac{AP_{2cam}(e,lr) - AP_{1cam}(2*e,lr)}{AP_{1cam}(2*e,lr)} * 100$$

Figure 7: The formula used to calculate the average improvement of the model with two cameras as training data against one camera.

Individual improvements produced for all different CNN setups can be found in the appendix C.

The overall average improvement with the use of two cameras was 8.42%. The improvement from a one camera dataset to a two cameras dataset is slightly more significant for lower learning rates. For a learning rate of 0.002, the two camera setup was on average 9.13% better. While

However, the models are still overfitting as can be seen in figure 8.



(a) training loss (b) mAP BBox

Figure 8: Left in (a) you can see that the validation loss increases even though the training loss keeps on decreasing. This shows that the model is over fitting. This is confirmed by the BBox mAP which at the start increases but after a while it decreases after more training. The model shown is the faster r-cnn trained with 0.002 lr and 50 epochs and trained on 2 cameras

3.3 Epochs

The 100 epochs with one camera and 50 epochs with two cameras were overfitting the data. Therefore one of the solutions could be to train the models for fewer epochs. Training was done for all training configurations. A selection of these results is shown in table 3.

Model	mAP BBox	mAP Seg	LR	Epoch
Faster r-cnn	47.03	-	0.002	50
Faster r-cnn	49.22	-	0.002	25
Faster r-cnn	50.63	-	0.002	10
Faster r-cnn	51.00	-	0.002	5
Mask r-cnn	47.69	67.91	0.002	50
Mask r-cnn	50.22	70.83	0.002	25
Mask r-cnn	50.75	69.54	0.002	10
Mask r-cnn	51.63	70.03	0.002	5
RetinaNet	47.90	-	0.002	50
RetinaNet	47.89	-	0.002	25
RetinaNet	48.80	-	0.002	10
RetinaNet	48.45	-	0.002	5

Table 3: The results of training on the data of camera 3 and 5.

This table shows that decreasing the amount of epoch for Faster r-cnn and Mask r-cnn improves the BBox mAP score. RetinaNet has one outlier to this rule. The 10 epoch produces better results than the 5 epoch RetinaNet.

3.4 Learning rate

The last training variable that has been adjusted is the learning rate. This was done because the learning rate can influence the training models. Therefore different baseline learning rates have been tested. As discussed in section 2.3 the

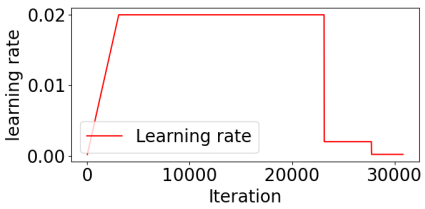


Figure 9: The Learning rate during training respective to the iteration for a base learning rate of 0.02

learning rate changes (warms up and drops) during training. This learning rate over time is visualised in Figure 9.

The accuracy of the learning rate for the single-camera test setup with 10 epoch shows that a higher base learning rate produces slightly better results. This can be seen in Table 4.

Model	mAP BBox	mAP Seg	LR	Epoch
Faster r-cnn	48.21	-	0.2	10
Faster r-cnn	47.98	-	0.02	10
Faster r-cnn	47.53	-	0.002	10
Mask r-cnn	48.05	71.56	0.2	10
Mask r-cnn	47.91	70.34	0.02	10
Mask r-cnn	47.82	66.12	0.002	10
RetinaNet	45.39	-	0.2	10
RetinaNet	44.92	-	0.02	10
RetinaNet	44.59	-	0.002	10

Table 4: The results of training for 10 epochs run on different learning rates. These results were from CNNs, which were trained on only the camera 5 dataset.

In this table, all higher learning rates produce better results than the 10 times lower counterparts. This trend is the same for most of the CNNs of the one camera training set, except for the 100 epochs. However, the CNNs trained on a two camera dataset seem to gravitate towards a lower learning rate. They sometimes prefer the lowest learning rate of 0.002 (Faster and Mask r-cnn for 10 epoch), the middle learning rate of 0.02 (all models on 50 epoch), but never 0.2. More on this in the discussion.

3.5 Bounding Box

Now that all of the settings have been explored, the final result can be analysed. In table 5 you can find the best mAP produced by each model family with the respective learning settings. validation loss

Model	mAP BBox	LR	Epoch	no. cams
Faster r-cnn	51.00	0.002	5	2
Mask r-cnn	**51.63**	**0.002**	**5**	**2**
RetinaNet	49.36	0.002	10	2

Table 5: The results of training on the dataset of one camera.

3.6 Segmentation

All that is left is to compare which model produces the best segmentation. Only Mask r-cnn contains the functionality to make pixel-wise predictions. All mask r-cnn results are shown in table 6. The best result is produced with the following settings: epoch: 10, learning rate:0.02, cameras:2.

Model	mAP Seg	LR	Epoch	No. cams
Mask r-cnn	**73.20**	**0.02**	**10**	**2**
Mask r-cnn	71.56	0.2	10	1
Mask r-cnn	71.27	0.02	5	1
Mask r-cnn	70.83	0.002	25	2
Mask r-cnn	70.34	0.02	10	1
Mask r-cnn	70.03	0.002	5	2
Mask r-cnn	69.54	0.002	10	2
Mask r-cnn	69.28	0.02	50	2
Mask r-cnn	67.91	0.002	50	2
Mask r-cnn	67.07	0.02	100	1
Mask r-cnn	66.83	0.002	5	1
Mask r-cnn	66.71	0.2	100	1
Mask r-cnn	66.12	0.002	10	1
Mask r-cnn	60.78	0.002	100	1
Mask r-cnn	DIV	0.2	50	2
Mask r-cnn	DIV	0.02	25	2
Mask r-cnn	DIV	0.2	10	2
Mask r-cnn	DIV	0.02	5	2
Mask r-cnn	DIV	0.2	5	1
Mask r-cnn	DIV	0.2	5	2

Table 6: All the Mask r-cnn results ordered on mAP Segmentation. DIV means that the training loss diverged and therefore no final model was created.

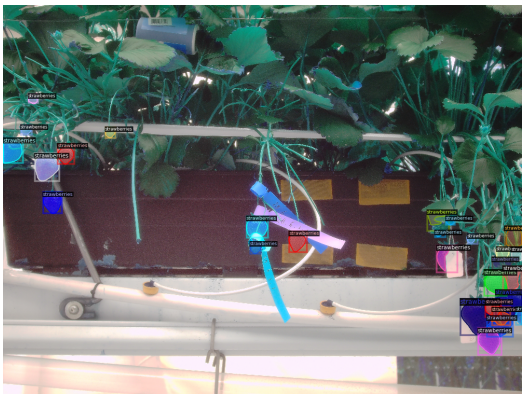
3.7 Visual Results

After all the technical analysis, it is time to see the visual results of the CNNs. First of all some images of the predicted strawberries from the best performing CNN. These images can be seen in Figures 10 and 11.

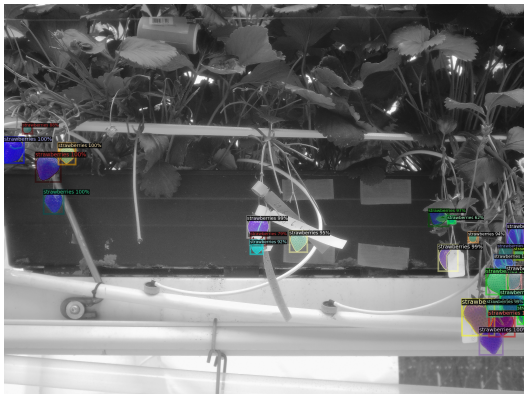
4 Discussion

As seen in the paper's results section, some models do not have an mAP. Instead, they have DIV as a result. The lack of mAP happened because the training did not produce a final model due to a diverging training loss. This divergence can occur when the learning rate is too large. The model then overshoots the local minima it is trying to learn towards and ends up higher on the loss curve than it previously was. Since the initial weights are initialized randomly for the main parts of the model, the divergence may happen due to unfortunate starting weights. An effort was made to ensure that all models have three tries to obtain a final model. If those three attempts were not enough, then the result of divergence was registered.

It can get stuck in a local minimum with only one run per hyperparameters. Therefore, the reported precisions may not be the best results a model could have achieved. Due to time constraints and hardware availability, testing each model multiple times was not an option.



(a) Ground Truth

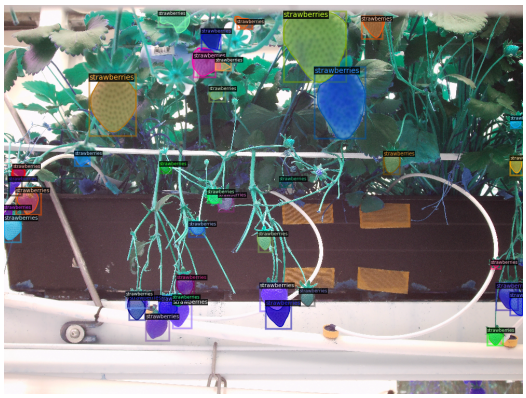


(b) Mask r-cnn

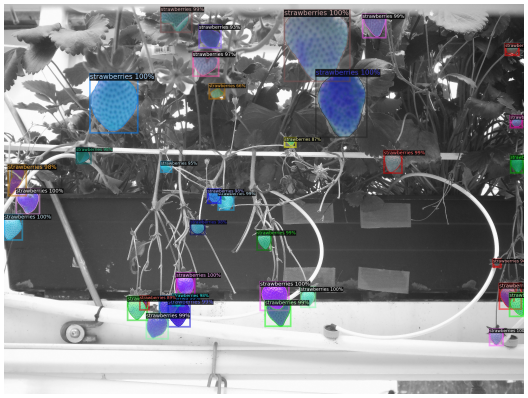
(c) Faster r-cnn

(d) RetinaNet

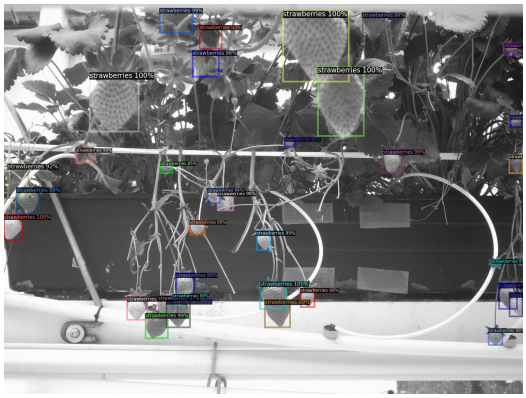
Figure 10: These images show the discrepancy between the ground truth and the predicted results of the best performing CNN in each category. These images show the overlap prediction capabilities (On the right of the images).



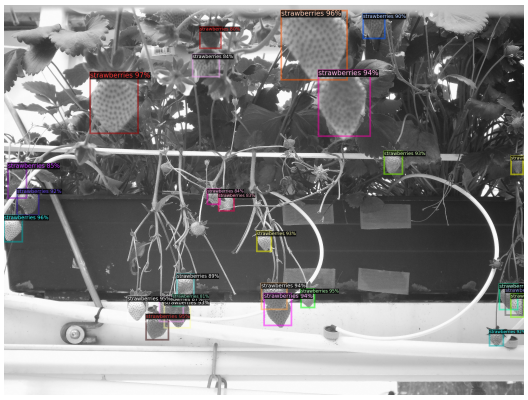
(a) Ground Truth



(b) Mask r-cnn

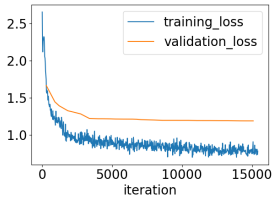


(c) Faster r-cnn

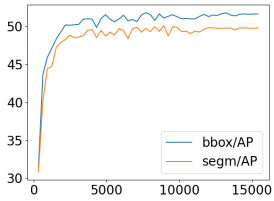


(d) RetinaNet

Figure 11: These images show the discrepancy between the ground truth and the predicted results of the best performing CNN in each category. These images represent strawberries of multiple scales.



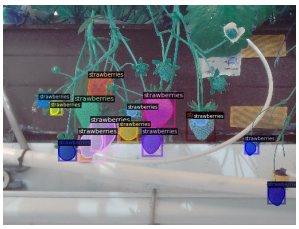
(a) Training loss



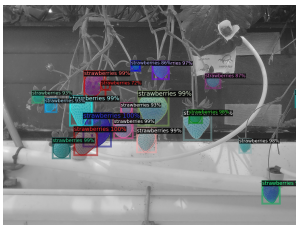
(b) mAP BBox

Figure 12: Left in (a) you can see that the validation loss increases even though the training loss keeps on decreasing. Right (b) shows that the mAP is stabilizing during training. The model shown is the Mask r-cnn trained with 0.002 Lr and 5 epochs on 2 cameras.

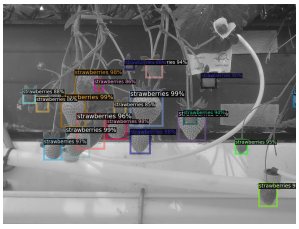
In the end, the CNNs seemed to prefer a lower amount of epochs due to overfitting at higher epochs. The best model for both BBox mAP and Segmentation mAP turned out to be a version of mask r-cnn. Moreover, they do not show overfitting, as shown in figure 12. In contrast to the initial tests using higher epoch training.



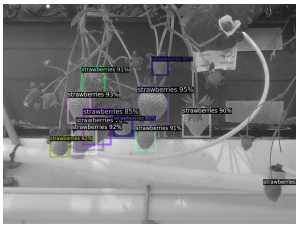
(a) Ground Truth



(b) Mask r-cnn



(c) Faster r-cnn



(d) RetinaNet

Figure 13: The ground truth (a) is missing some annotations for the smaller strawberries at the top. Both faster and mask r-cnn seem to detect these strawberries. While RetinaNet predicts some and misses others.

Lastly the data provided has some flaws. Some of the strawberries have not been annotated. This will hurt all the CNNs training, and the final scores will be lower when our models predict these strawberries. Figure 13 shows one of these instances.

5 Responsible Research

5.1 Ethical implications

The main goal of this project is to help combat food waste. The waste of strawberries can be partially attributed to the harvesting stages of the strawberries. By detecting the strawberries from images, other algorithms will be able to extract features from them and predict their ripeness score so that the strawberries that are ripe enough to be harvested are automatically detected. One of the downsides of an automatic system for strawberry harvesting is that in the future, strawberry harvesting jobs will possibly be done by machines. Therefore, humans can lose their (part-time) job.

5.2 Trustworthy results

To keep the results of this research as trustworthy as possible multiple things were taken into account while doing this research. All the implemented CNNs use the standard settings from the original papers of the proposal of the network. If any changes were made to these settings, these have been mentioned in this paper. An explanation of this deviation is then also provided. The precision of the CNNs was done with objective measurements, which are used in many of the original papers. This was done to remove as much bias as possible.

5.3 Reproducibility

During this research, a conscious effort has been made to build the system from an open-source library specifically designed for researchers. Doing this ensures that any researchers who want to implement the same system can easily access the same settings, making the research easier to reproduce. The code of the developed system is available on the TU Delft GitHub servers, and the images used for the training are also in possession of the TU Delft.

6 Conclusions

This paper aimed to find which CNN produces the best prediction data on where strawberries are located on the images. Multiple CNNs needed to be implemented and tested to arrive at this conclusion. For this research, we have chosen the following 3 CNNs: Faster r-cnn [7], Mask r-cnn [8] and RetinaNet [9].

Different learning rates and amounts of epochs have been tested. Moreover, from these results, the CNN with the most accuracy according to the bounding box mAP is Mask r-cnn with the following hyperparameters: LR: 0.002, epochs: 5, cameras used: 2, which resulted in an mAP of 51.63. And for the segmentation mAP, the best model is Mask r-cnn with the following hyperparameters: LR: 0.02, epochs: 10, cameras used: 2, which resulted in an mAP of 73.20.

7 Future Works

Due to limited time constraints, not all promising CNNs were able to be tested. Therefore we propose for future research to see how other CNNs like YOLOv5 perform [16]. A model which can be compared against the segmentation of mask r-cnn would be preferred.

Due to the random nature of the training, only a local minimum may have been reached. The models are only trained once per combination of hyperparameters. Rerunning the experiments may therefore result in slightly different results. Therefore, it would also be beneficial to research how starting weights influence the final prediction capabilities of the model.

References

[1] A. Siu, "An analysis of food waste in ontario's domestic fresh strawberry supply chain," -, 2014.

[2] S. Bargoti and J. Underwood, "Deep fruit detection in orchards," Proceedings - IEEE International Conference on Robotics and Automation, 2017.

[3] S. Wan and S. Goudos, "Faster r-cnn for multi-class fruit detection using a robotic vision system," Computer Networks, vol. 168, 2020.

[4] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021.

[5] Y. Fitter, "Strawberry detection under various harvestation stages," 2019.

[6] Y. Yu, K. Zhang, L. Yang, and D. Zhang, "Fruit detection for strawberry harvesting robot in non-structural environment based on mask-rcnn," Computers and Electronics in Agriculture, vol. 163, p. 104846, 2019.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2015.

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in Proceedings of the IEEE international conference on computer vision, pp. 2961–2969, 2017.

[9] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in Proceedings of the IEEE international conference on computer vision, pp. 2980–2988, 2017.

[10] I. Sa, Z. Ge, F. Dayoub, B. Upcroft, T. Perez, and C. McCool, "Deepfruits: A fruit detection system using deep neural networks," Sensors (Switzerland), vol. 16, 2016.

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

[12] R. Girshick, "Fast r-cnn," in Proceedings of the IEEE international conference on computer vision, pp. 1440–1448, 2015.

[13] Z. Ma and N. Li, "Improving apple detection using retinanet," in The International Conference on Image, Vision and Intelligent Systems (ICIVIS 2021), pp. 131–141, Springer, 2022.

[14] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2." https://github.com/facebookresearch/detectron2, 2019.

[15] Delft High Performance Computing Centre (DHPC), "DelftBlue Supercomputer (Phase 1)." https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1, 2022.

[16] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, TaoXie, J. Fang, imyhxy, K. Michael, Lorna, A. V, D. Montes, J. Nadar, Laughing, tkianai, yxNONG, P. Skalski, Z. Wang, A. Hogan, C. Fati, L. Mammana, AlexWang1900, D. Patel, D. Yiwei, F. You, J. Hajek, L. Diaconu, and M. T. Minh, "ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference," Feb. 2022.

A All single camera results

Model	mAP BBox	mAP Seg	LR	Epoch
Faster r-cnn	44.73	-	0.2	100
Mask r-cnn	44.87	66.71	0.2	100
RetinaNet	45.91	-	0.2	100
Faster r-cnn	44.85	-	0.02	100
Mask r-cnn	44.88	67.07	0.02	100
RetinaNet	43.95	-	0.02	100
Faster r-cnn	43.06	-	0.002	100
Mask r-cnn	41.81	60.78	0.002	100
RetinaNet	43.64	-	0.002	100
Faster r-cnn	48.21	-	0.2	10
Mask r-cnn	48.05	71.56	0.2	10
RetinaNet	45.39	-	0.2	10
Faster r-cnn	47.98	-	0.02	10
Mask r-cnn	47.91	70.34	0.02	10
RetinaNet	44.92	-	0.02	10
Faster r-cnn	47.53	-	0.002	10
Mask r-cnn	47.82	66.12	0.002	10
RetinaNet	44.59	-	0.002	10
Faster r-cnn	DIV	-	0.2	5
Mask r-cnn	DIV	DIV	0.2	5
RetinaNet	DIV	-	0.2	5
Faster r-cnn	49.36	-	0.02	5
Mask r-cnn	48.96	71.27	0.02	5
RetinaNet	DIV	-	0.02	5
Faster r-cnn	47.83	-	0.002	5
Mask r-cnn	48.53	66.83	0.002	5
RetinaNet	44.58	-	0.002	5

Table 7: The results of training on the data of one camera.

B All multi camera results

Model	mAP BBox	mAP Seg	LR	Epoch
Faster r-cnn	DIV	-	0.2	50
Mask r-cnn	DIV	DIV	0.2	50
RetinaNet	DIV	-	0.2	50
Faster r-cnn	48.14	-	0.02	50
Mask r-cnn	47.41	69.28	0.02	50
RetinaNet	48.68	-	0.02	50
Faster r-cnn	47.03	-	0.002	50
Mask r-cnn	47.69	67.91	0.002	50
RetinaNet	47.90	-	0.002	50
Faster r-cnn	46.67	-	0.02	25
Mask r-cnn	DIV	DIV	0.02	25
RetinaNet	48.12	-	0.02	25
Faster r-cnn	49.22	-	0.002	25
Mask r-cnn	50.22	70.83	0.002	25
RetinaNet	47.89	-	0.002	25
Faster r-cnn	DIV	-	0.2	10
Mask r-cnn	DIV	DIV	0.2	10
RetinaNet	DIV	-	0.2	10
Faster r-cnn	49.86	-	0.02	10
Mask r-cnn	50.48	73.20	0.02	10
RetinaNet	49.36	-	0.02	10
Faster r-cnn	50.63	-	0.002	10
Mask r-cnn	50.75	69.54	0.002	10
RetinaNet	48.80	-	0.002	10
Faster r-cnn	DIV	-	0.2	5
Mask r-cnn	DIV	DIV	0.2	5
RetinaNet	DIV	-	0.2	5
Faster r-cnn	DIV	-	0.02	5
Mask r-cnn	DIV	DIV	0.02	5
RetinaNet	49.34	-	0.02	5
Faster r-cnn	51.00	-	0.002	5
Mask r-cnn	51.63	70.03	0.002	5
RetinaNet	48.45	-	0.002	5

Table 8: The results of training on 2 camera datasets.

C The improvement percentage achieved by using the images of 2 cameras as training set

Model	mAP BBox	mAP Seg	LR	Epoch
Faster r-cnn	DIV	-	0.2	100-50
Mask r-cnn	DIV	DIV	0.2	100-50
RetinaNet	DIV	-	0.2	100-50
Faster r-cnn	7.34%	-	0.02	100-50
Mask r-cnn	5.64%	4.29%	0.02	100-50
RetinaNet	10.76%	-	0.02	100-50
Faster r-cnn	9.22%	-	0.002	100-50
Mask r-cnn	14.06%	11.73%	0.002	100-50
RetinaNet	9.10%	-	0.002	100-50
Faster r-cnn	DIV	-	0.2	10-5
Mask r-cnn	DIV	DIV	0.2	10-5
RetinaNet	DIV	-	0.2	10-5
Faster r-cnn	DIV	-	0.02	10-5
Mask r-cnn	DIV	DIV	0.02	10-5
RetinaNet	DIV	-	0.02	10-5
Faster r-cnn	6.35%	-	0.002	10-5
Mask r-cnn	7.97%	5.91%	0.002	10-5
RetinaNet	8.66%	-	0.002	10-5

Table 9: The improvement in percentage of using two cams instead of one. The first number in the epoch column represents the number of epochs of the one camera trained models and the second number in the epoch column is the number of epochs for the models trained on the two camera data set. DIV means either the one camera setups training diverged, the two camera setup diverged or both.