Machine Vision apple counting project

No.	Name of group members	Contribution to project	Contribution to report
1	Joshua Trimmer	RCNN Testing	2,6 and 7 Editing
2	Jeff Hannah	RCNN Training	1 and 3 Editing: 6
3	Santhosh Kanaga Sabapathy	UNet (Testing)	4, 6(Editing),Reformatting
4	Vishnu Hariharan Anand	RCNN (Test, Train), UNet (Test, Train).	4 and 5

Table of Contents

1. Introduction

Counting fruit, such as apples, before harvest is useful as it allows for the planning of the logistics along with estimation of yield and profit before harvest granting the farmers more flexibility in planning and financing and improving efficiency of the harvest and transportation of the crop (N. Häni, P. Roy and V. Isler, 2018). Additionally, any automated orchard harvesting system that could be deployed in the future must be able to identify apples for harvest and the most obvious method for performing this identification is visually, through a machine vision system.

This report aims to highlight the differences between two approaches for counting apples. The first of which being object detection and localisation, which uses a Region based Convolutional Neural Network (R-CNN) specifically we used an improved version of the Network called Faster-RCNN (Girshick, Donahue, Darrell, and Malik, 2014), the second being semantic segmentation, which uses U-Net (Ronneberger, Fischer and Brox, 2015) which is a convolutional neural network, originally developed for medical imaging and based on the 'fully convolutional network' (Shelhamer, Long and Darrel, 2017). This is done through a thorough investigation of both methods, including a rigorous review of related literature. A pre-existing database of apple trees in an orchard environment is used to conduct this investigation due to the infeasibility of collecting a full dataset ourselves due to equipment limitations, lack of apples during the winter and ongoing restrictions due to the COVID-19 pandemic. A thorough analysis of this database, including collection methods and equipment used, is presented in section 3 of this report along with an analysis of how a unique dataset may have been collected had it been possible. This is followed by a detailed methodology of the investigation discussing how the two variants of CNN were used to detect and count the apples, a demonstration of the implementation of both methods in Python and finally a well-rounded presentation and evaluation of our results along with a concise conclusion of the investigation.

2. Related works

A simplistic approach to apple counting would be to simply use histogram-based image segmentation (Shapiro, 2001). A colour image containing apples could be analysed and looking at the red image histogram it would be possible to set a threshold and this would isolate areas of high red intensity from the rest of the image. This image could then be plotted in binary form where everything above the threshold would be one and everything below would be zero. Erosion can then be used to separate where the apples overlap. The problem with such an approach is that apples are not always red and if this method would be applied to green apples the leaves of the tree would also be counted, there is also no way for this method to separate overlapping apples from elongated apples.

A more sophisticated approach to this problem is to use a neural network that can classify an object as an apple. A breakthrough in this area was made in 2010, the convolutional neural network (CNN) made for the ImageNet LSVRC-2010 contest was nicknamed Alex-net after one of its creators (Krizhevsky, Sutskever and Hinton, 2017). This CNN was one of the largest made at the time, with 60 million parameters and 650,000 neurones, it was also able to deal with much larger datasets than any previous neural network. A CNN such as this one works using a "sliding window" approach where a small section is processed at a time. An improved version of Alexnet was created in 2014 which had more layers thus making it deeper this was

referred to as VGG named after the Visual Geometry Group at the University of oxford (Simonyan and Zisserman, 2014). Unfortunately, although this made a CNN work with more precision and accuracy it also made it slower as it was more computationally intensive. Along with the problem of adding layers making CNN more demanding it was also noticed in 2016 that there was a limit to how much improvement could be achieved by simply adding more layers. More layers lead to more tracking error and degradation in accuracy, Resnet uses residual mapping to improve the speed and accuracy of the CNN (He, Zhang, Ren, and Sun, 2016).

To count the apples, it is necessary to not only know that there are apples present in the image but also to localise them. To localise objects within an image a bounding box is used, R-CNN is a process that outputs classification of objects along with location (Girshick, Donahue, Darrell, and Malik, 2014). R-CNN starts with producing areas of interest referred to as region proposals, there can be around two thousand region proposals. R-CNN selects these regions by using an algorithm referred to as selective search, this works using several different methods, the first of which is like the original proposal of separating by colour, but this is ineffective when separating objects from backgrounds of the same colour. To solve this problem the second method is introduced, this involves separating objects by texture, alas there is another problem that persists when these two methods are used, what if an object is made up of parts with different textures and colours? In this case composition of the object must be considered to merge parts of the same object together. Once these regions have been chosen the CNN is run on just these regions, this saves time and computational resources as previous methods would require inputting the whole image. For these regions to be processed they must all be the same dimensions, so they are stretched or squashed to fit a square of 227x227 pixels. These regions are then processed individually with a CNN, the output will be two parts, one that represents what the object is and the other representing the position of the object.

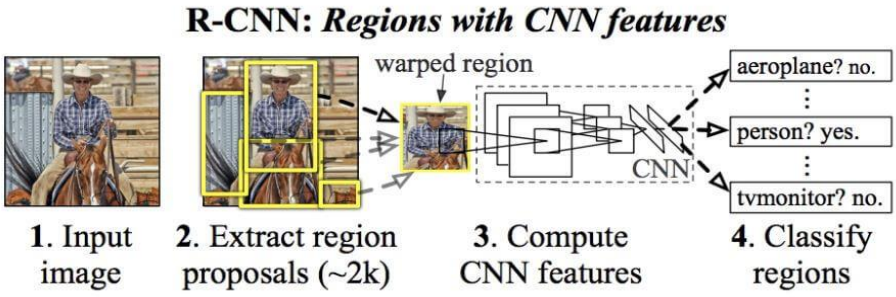

Figure 1: R-CNN Structure (Girshick, Donahue, Darrell, and Malik, 2014)

This method was improved the next year to make it faster, this latest version is called Fast R-CNN (Girshick, 2015). Fast R-CNN can be trained nine times faster and when testing it is 213 times faster than R-CNN, these improvements along with the fact that the accuracy is comparable to R-CNN makes R-CNN obsolete. The improvements made to the learning were firstly made by attempting to reduce the number of stages in the process and secondly to reduce storage requirements as R-CNN required a large amount of disk space. The process was simplified, and time was saved as whole images can now be taken as one input, meaning the CNN only runs once, instead of running for every region proposal separately. This is possible because of the use of spatial pyramid matching (Lazebnik, Schmid and Ponce, 2009) to create a feature map of the original image discussed further in the feature extractor section.

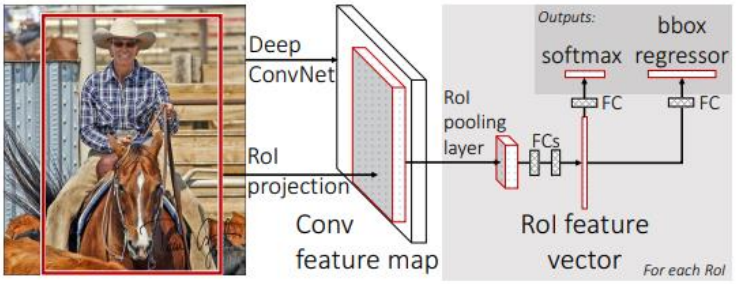

Figure 2: Fast R-CNN Structure (Girshick, 2015)

Since 2015 there has been a further improvement to R-CNN which is referred to as faster R-CNN, this takes the idea of reducing the steps required that fast R-CNN proposed and improves upon this idea. However, there is another approach that can be used for this problem and has a similar level of performance to faster R-CNN but is not based on R-CNN. This other method is called U-net and uses a vastly different structure to R-CNN based approaches, which includes two paths rather than one: the contracting path and the expanding path. These are the two approaches that will be used in this report and they will be discussed further later.

3. Data acquisition and datasets

Acquiring data for machine vision applications in an outdoor orchard environment presents several challenges: 1) variable illumination, 2) varied appearance of fruits on trees, I.e., not all apples will look quite the same, and 3) fruit occlusion with leaves or other fruit, I.e., apples can 'overlap' with each other from certain points of view or can be obstructed by leaves making it more difficult to isolate the fruit. These issues can be combatted in a few ways, the main method is to use machine learning and provide the learning algorithms with vast datasets for training, testing and the comparison of different techniques, these large datasets require long training times but also allow for the learning algorithm to adapt to and resolve the issues previously mentioned.

Unlike humans, a machine vision system is not limited to the 'visual spectrum' of light and can use other areas of the electromagnetic spectrum in their processes. For example, older machine vision counting methods often used a simple colour threshold for detecting the objects of interest, this method has some inherent limitations which were often compensated for by using thermal and Near Infrared camera (NIR) systems (Gongal A., et Al., 2015). However, for object detection network methods more comparable to those used in this investigation, either NIR (Near-Infrared) or standard visual spectrum RGB(Red-Green-Blue) images tend to be used (Bargoti S., Underwood J., 2017) (Häni, N., Roy, P. and Isler, V. (2019)).

The dataset used was MinneApple dataset (Häni, Roy and Isler, 2020) which is made up of 1000 varied images of apple trees in an orchard environment containing more than 41000 individual apples each annotated with a polygonal mask which simplifies precise object detection and localisation and aids in the implementation of segmentation. MinneApple contains a wide variety within its images as the images are collected over a two-year period and a single image contains anywhere between 1 and 120 apples consisting of varied species at various stages of the apple's ripening cycle. Data collection was performed by using a standard Samsung Galaxy S4 mobile phone camera to record video footage at a resolution of 1280 by 720 pixels, aiming the camera horizontally to a row of trees in the orchard, and walking

along the row at a slow speed of 1 m/s to reduce motion blur. Then every fifth frame of the subsequent video was extracted to produce the images of the data set. Over the course of the two years a total of 17 datasets were collected with varied pixel resolutions. Later an additional test set was collected from a further distance to test the algorithms ability to count lower resolution fruit.

The MineApple dataset was selected as it contains a higher number of images than comparable datasets which, of course, allows for a more accurate model as there is more data to train the neural networks with. In addition, the helpfulness of the annotation and polygonal masks will significantly improve our results when performing segmentation with the U-net architecture. Finally, the intentional inclusion of apples of different breeds, colours, and stages of growth with apples on the trees, on the ground with many being partially obstructed by leaves, branches, or other apples, allows us to ensure our solution is general enough to be practical in real-world applications.

The data set was used to train and test both the Faster R-CNN and the U-net network architectures investigated within this report. All images contained within the dataset were used and the ground truth of the images were measured against the numbers that the tested counting algorithms produced to grade the accuracy of our solutions.

4. Methodology

Object detection algorithms can be classified into 4 main categories based on the requirements:

- **Image Classification:** The algorithms that falls under this category can easily classify the entire image to either one of the classes. The dataset of this type of algorithm usually contains only the object of interest.
- **Semantic Segmentation:** These algorithms can classify each pixel of the image to multiple classes. But this does not always give proper localisation, e.g., when two objects of same class with overlap cannot be localised efficiently using these algorithms.
- **Object Detection and Localisation:** These algorithms can classify and localise all the objects in an image. These algorithms usually fit a bounding box to each object that got detected in the image.
- **Instance Segmentation:** This combines the semantic segmentation along with localisation. These algorithms give pixel level classification for each object and distinguish between different instances of the same object class in the same image.

To count the apples, it requires some information on localisation the apple in the image. So, this task falls between the category of semantic segmentation and object detection and localisation. Based on this we have chosen one algorithm from each of these categories.

4.1 Object Detection and Localisation Approach

The Faster R-CNN (Ren, He, Girshick and Sun, 2015) algorithm is used in this approach. This algorithm is one of the current state-of-the-art algorithms that perform object detection and localisation in images at extremely high accuracy. Faster RCNN comprises a feature extractor, a region proposal network (RPN), and a classifier.

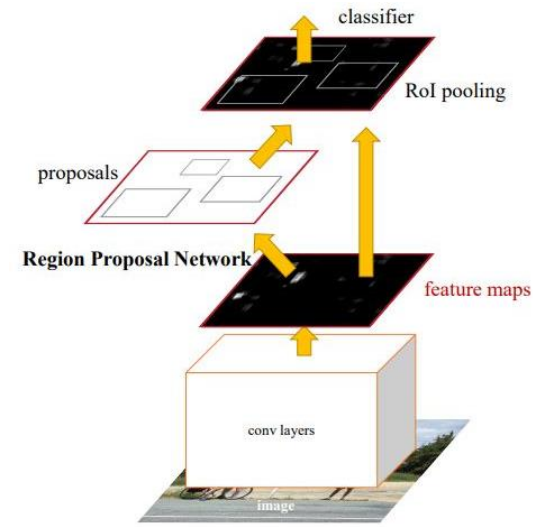

Figure 3: Faster RCNN (Ren, He, Girshick and Sun, 2015)

4.1.1 Feature Extractor

We have used pretrained ResNet-50 (He, Zhang, Ren, and Sun, 2016) along with FPN (Lin, Dollár, Girshick, He, Hariharan and Belongie, 2017) as the backbone for Faster RCNN which performs feature extraction from the input images. ResNet-50 has shown significant improvements in the accuracy for larger deep CNN's due to the residual connection and lower kernel size for convolution. Feature Pyramid Networks considers a pyramid of features rather than considering the last layer features which helps in dealing with the lost information from the previous layers for e.g., features of smaller object images vanishes/ diminishes during the forward pass of any CNN. So, combining the strengths of ResNet-50 and FPN structure for obtaining a feature pyramid gives us an upper hand on getting a good set of features that can be used by RPN and classifier later for apple detection and counting.

4.1.2 Region Proposal Network

The feature maps created by the feature extractor is further fed to the Region Proposal Network (RPN) to get the object proposals as bounding box and the 'objectness' score which is the confidence score of the presence of an object vs background of the image. RPN uses sliding window approach with anchors boxes of multiple size and ratio to generate bounding boxes on the feature map i.e., for each pixel in the feature map and k anchors there will be a total of k bounding boxes. The figure 4 represents the RPN which take feature maps as input and uses k anchors per pixel to generate a total of WHk bounding boxes per feature map (where WH is the size of feature map) by predicting $4k$ coordinates and $2k$ 'objectness' score for each bounding box. For a FPN this is repeated for each feature map.

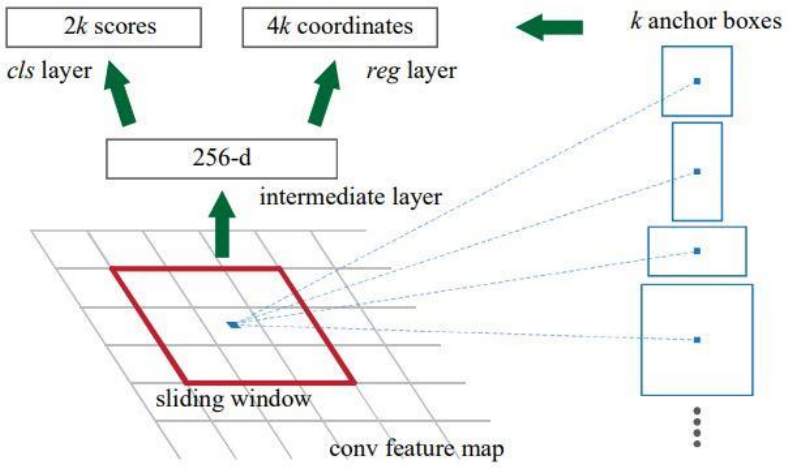

Figure 4: Region Proposal Network (Ren, He, Girshick and Sun, 2015)

4.1.3 Classifier

The function of the classifier is to classify each bounding box generated by the RPN to its corresponding classes i.e., it is performing an image classification task. A classifier takes each proposal, obtains its corresponding *region of interest* (ROI) from the feature map, runs ROI pooling (a special pooling layer which converts the multiple differently sized ROI's to a fixed, common size) and does classification and regression for generating more refined bounding box, it should be noted that this is not truly a prediction of an objects bounds as RCNN's purpose is the classify not bound (Ren, He, Girshick and Sun, 2015).

In detail, an ROI (region of interest) form feature map is extracted based on the proposal and using a scaling factor that can compensate the reduction of size of feature map w.r.t the input image size. This scaling factor is determined by the feature extractor that we use, in ResNet-50 FPN we use same convolution of 3x3 in all layer therefore, the scaling only depends on the number of max pooling layers. Since ROI form each proposal will be of varied sizes, the ROI pooling layer is used to down sample each ROI to fixed size ROI's. These fixed size ROI's are then fed to a regressor and a classifier to predict the error in bounding box generated by RPN and its corresponding object classes.

4.1.4 Loss Function in Faster-RCNN

In Faster-RCNN there are a total of 4 losses, two form RPN and two from classifier i.e., there are two classification loss and two regression loss in total. RPN uses a binary classification loss based on the presence of the object. A positive label is generated for the anchors with Intersection-over Union (IoU) overlap greater than 0.7. Anchors with less than 0.3 gets a negative label. All the other anchors do not contribute to the training. The classifier uses a multi class classification and considers the bounding box with maximum overlap with the ground truth while training as positive samples. The regression for the bounding box for both RPN and classifier uses smooth L_1 function. The classification and regression loss are combined using the below formula (Ren, He, Girshick and Sun, 2015):

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

where: i is the index of the anchor, p_i is the predicted probability, p_i^* is the ground truth label, t_i and t_1^* are the predicted and ground truth bounding boxes.

4.1.5 Apple Counting using Faster RCNN

For each image, the Faster RCNN model outputs all the bounding boxes and its corresponding scores representing the confidence of the class it detected. An appropriate threshold on the scores will limit the bounding boxes with high confidence detections. The apple count can then be easily calculated by counting the number of detections after thresholding.

4.2 Semantic Segmentation Approach

The semantic segmentation approach uses U-net (Ronneberger, Fischer and Brox, 2015). U-net has a vastly different architecture to the neural networks previously discussed partially due to it not being an entirely linear process. U-net was originally designed to be used in biomedical applications but works in various applications. This approach consists of two paths, the contracting path used for capturing context and the expanding path used for localisation.

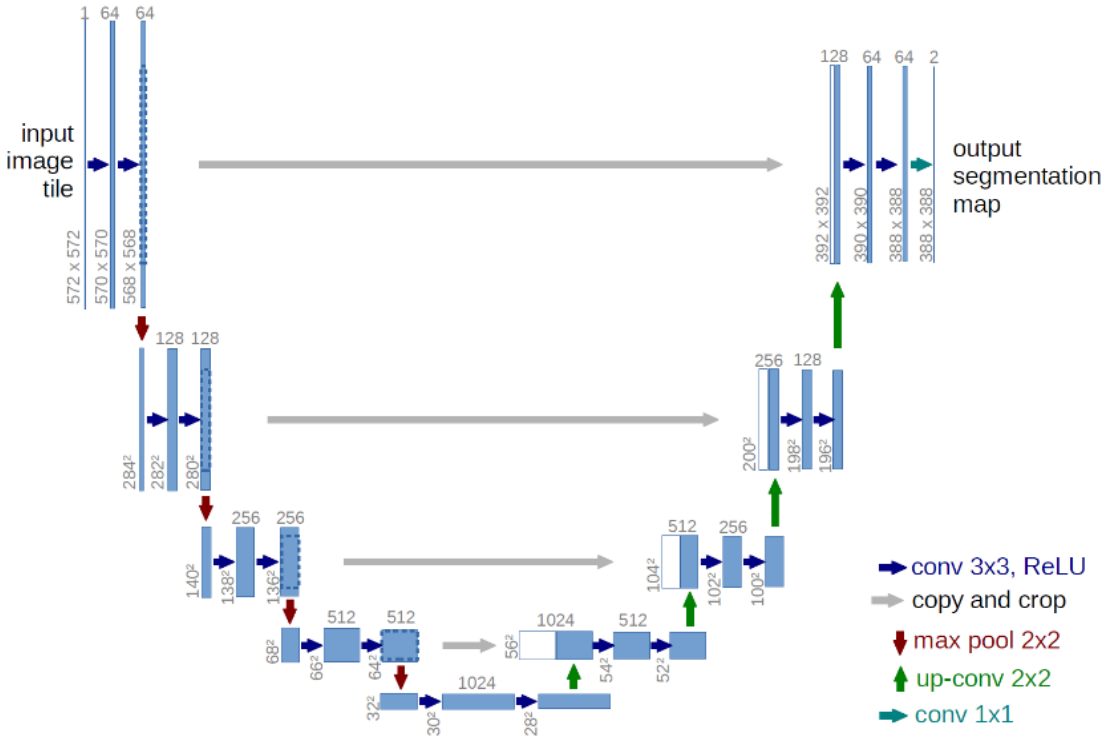

Figure 5: U-net Structure (Ronneberger, Fischer and Brox, 2015)

In semantic segmentation each pixel of an image is labeled with a related class of what is being represented. As we are determining a class for all the pixels in the image, this task is generally referred to as dense prediction.

Unlike faster RCNN, the output in semantic segmentation are not only bounding box parameters and labels. The output is a high-resolution image, where each pixel is categorized to a specific class. The output image is usually the same size as the input image.

The architecture UNET contains two paths. The first path is applied to determine the context of the image and is known as the contraction path, which is also called the encoder. The contraction path is a traditional stack of max pooling and convolution layers. In the second

path, transposed convolution is used to perform precise localization in the image, this is known as symmetric expanding or decoder. The combination of the paths makes this system a fully convoluted network (FCN).

4.2.1 Architecture in Detail

The Figure 5 shows the architecture that was implemented in the original paper. In our approach we have used a similar architecture with some minor modifications. The overall structure remains the same in terms of number of convolution and up convolution layers. Also, all the convolution kernels are same which are of size 3x3 expect for the output layer, which is 1x1 kernel. All the convolutions are same convolution which means the appropriate padding is done to make the output of convolution to be of same size as the input. The reduction in size happens only at the max pooling layer.

The up-convolution operation is done using Transpose convolution 2d method in pytorch. It does the opposite of max pooling to increase the resolution of the output (doubles in UNet). It uses a learnable non square kernel for achieving this. There is loss of information when doing up-convolution and this loss of information is regained using the concatenation of similar level layers during the encoder, which can be seen in the image. Also, it can be noted that the depth at each step in the encoder increases whereas the depth in decoder part is decreasing. This can also be a contribution factor to gain the loss of information.

The output of the UNet is a semantic segmentation map, which is obtained after applying sigmoid function. The output is of N channel, where N is the number classes as per the dataset/task. In our case for apple detection, we have only one class, so N is one. Each pixel value of the output represents the probability of the object class that is associated with the channel.

4.2.2 Dice Loss

Since the output layer is in the range of [0,1] we can apply dice loss. Dice coefficient measures the overlap between the ground truth and the predicted semantic map. The ground truth is also converted to binary format like semantic map of the UNet for easier comparison. The Dice coefficient is given by.

$$Dice = \frac{2|A \cap B|}{|A| + |B|}$$

where, A and B are the ground truth and the semantic map. Dice loss is given by the $1 - Dice$.

4.2.3 Apple Counting using UNet

The semantic map generated by the UNet is first converted to multiple binary images for each channel separately using some appropriate threshold in the accuracy. Morphological operations are then applied on the binary image trying to separate some overlapping objects or apples that got detected. The contours above certain size is then selected using the OpenCV and bounding boxes are detected. The apple count is given by the number of selected bounding boxes.

5. Experiment and Implementation

Both Faster RCNN and U-net require GPU (Graphic Processing Unit) for training and testing for better turnaround time. To train these deep learning networks we used Kaggle Notebook as the platform, which has Nvidia K80 16 GB GPU, 16GB RAM. Kaggle provides a free usage for about 40hrs per week. The testing code was written on PyCharm IDE (Integrated Development Environment) and testing on a hardware with Nvidia GTX 1650 4GB GPU and 8 GB RAM. Some of the main libraries used while training are PyTorch and Torchvision as the machine learning framework, OpenCV as the computer vision library, NumPy for numerical calculations, matplotlib for plotting.

The apple dataset contained both training and test images, but the ground truth data was only available for training dataset publicly. To train both networks, the entire training images were split into train, validation, and test datasets in the ratio of 0.6:0.2:0.2 randomly.

Below is the pseudocode of the training done for both the networks. For Faster RCNN the image was scaled down by half whereas for U-Net the input images were reduced to 512 x 512 resolution. Training was done for 150 epochs for both the networks and the best model was chosen for testing based on the validation accuracy.

5.1 Faster RCNN: Training

1. Define Apple Dataset Class which takes images and masks as input and returns image as tensor object and bounding box coordinates as required by the Faster RCNN algorithm.
2. Create two objects of Apple Dataset as train and validation dataset.
3. Define data loader using torch library for the train and validation dataset created with a batch size of 4 and shuffle enabled.
4. Create Faster RCNN model object with ResNet-50-FPN as backbone using torchvision library.
5. Create a list of parameters that are required to calculate gradients while backpropagation is occurring.
6. Create a Stochastic Gradient Descent optimizer object using the parameters and with an initial learning rate of 0.005.
7. Initialize total epochs.
8. For each epoch:
    a. For dataset from categories [train or validation]
        i. For each batch of images and targets from the data loader
            1. Losses = model (images, targets)
            2. Take sum of all losses.
            3. IF Training:
                a. Set all gradients to zero.
                b. Generates the gradients steps for all parameters.
                c. Update the parameters.
            4. IF Validation:
                a. Save model with if the epoch has the best validation accuracy.

5.2 U-Net: Training

1. Define Apple Dataset Class which takes images and masks as input and returns image as tensor object and coverts mask into pure black and white semantic tensor object.
2. Define U-Net Model class which implements the architecture described in the previous section. We use "ConvTranspose2d" function form PyTorch for up convolution.
3. Define Dice Loss class which works for binary classifiers.

$$Dice\_Loss = \frac{2 * |prediction \cap target|}{|prediction \cup target|}$$

4. Create two objects of Apple Dataset for train and validation dataset.
5. Define data loader using torch library for the train and validation dataset created with a batch size of 4 and shuffle enabled.
6. Create U-Net model object of U-Net model class.
7. Create Adam optimizer object for the model parameters and with an initial learning rate of 0.0001.
8. Initialize total epochs.
9. For each epoch:
    a. For dataset from categories [train or validation]
        i. For each batch of images and targets from the data loader
            1. predictions = model (images, targets)
            2. loss = dice loss (predictions, targets)
            3. IF Training:
                a. Set all gradients to zero.
                b. Generates the gradients steps for all parameters.
                c. Update the parameters.
            4. IF Validation:
                a. Save model with if the epoch has the best validation accuracy.

Once the models are trained, the models are ready to be tested against the test dataset. Faster RCNN results a bounding box and its corresponding confidence score. We have used a threshold of 0.9 on the confidence of the bounding box for selecting an apple based on which the count is calculated. The mean Average precision values of different IoUs (Intersection over Union) is also recorded using a python library (mean_average_precision). Since U-Net generates a mask image, which is processed using a series of morphological operations trying to increase the gap between overlapping apples. Then all the contours are detected using OpenCV and the count is taken based on the number of contours with area greater than some threshold.

6. Results and Evaluation

To evaluate object detection-based approaches such as Faster R-CNN a metric called *average precision* is used. To calculate *average precision* the *recall* and *precision* must first be calculated. To do this the results must be compared to the true values (ground truth). *Precision* is the number of positives that are true divided by the total number of positives, so in this case the number of correctly identified apples divided by the total number of objects that have been identified as apples. *Recall* is the number of true positives divided by the true positives added to the false negatives, so in this case the number of apples detected divided by the total number of apples present in the image (Powers, 2020).

$$Recall = \frac{True\ positives}{True\ positives + False\ negatives}$$

Now the *average precision* (AP)can be found by taking the integral of the *precision* against the *recall.*

$$AP = \int_0^1 p(r)dr$$

The *average precision* is used over different *Intersection-Over-Union* (IoU) thresholds. IoU measures the overlap between two boundaries. By looking at what this overlap is between the models and the ground truth the precision can be determined. IoU is found by dividing the area of overlap with the area of union of the two boundaries. The average precision is taken across a range of IoU values, from 0.5 up to 0.95 with 0.05 increments. As the objects being detected are quite small the average precision at IoU = 0.5 and 0.75 have also been included in the results as can be seen in the table below. Comparing the results from our Faster R-CNN model to the ones seen in the MinneApple paper (Häni, N., Roy, P. and Isler, V. (2019)) it can be said that for the range of IoU values and at IoU=0.5, our Faster R-CNN model has remarkably similar values to the models tested by Häni et al. The performance is however slightly worse at IoU=0.75 but this could be due to our dataset being split.

AP at IoU=0.5	AP at IoU=0.75	AP at IoU=0.5:0.05:0.95	Count Accuracy
0.766	0.29	0.405	0.66

Table1: Faster R-CNN results

While the R-CNN based approach produces the bounding boxes as part of the process, the U-net approach only adds the bounding boxes in post processing. This may lead to a bias towards the R-CNN approach when directly comparing the average precision scores, thus, to evaluate U-net another approach is used. IoU as described earlier is used as a metric to evaluate U-net, when comparing to similar approaches mentioned in the MinneApple paper (Häni, N., Roy, P. and Isler, V. (2019)) this model performs well. Pixel accuracy is also used which, measures how accurately the model has classified each individual pixel when compared to the ground truth. It can be seen in table 2 below that the result for pixel accuracy is remarkably high. Pixel accuracy This could be because of how similar the images are within the dataset, so this could lead to overconfidence.

Dice Loss	IoU	Pixel Accuracy	Count Accuracy
0.252	0.662	0.989	0.682

Table 2: U-net results

To directly compare the results of the two methods we calculated the *count accuracy*, this is the average accuracy of the number of apples the model counts in each image when compared to the ground truth. This is calculated in an equivalent way to the recall as mentioned above, the number of apples detected divided by the total number of apples present in the image. It can be seen in the above tables that both models performed well with results just under 70% but the results from U-net were slightly better with a greater accuracy of more than 2%. It is important to note that the count accuracy does not include false positives.

Figure 6: Initial Results from U-NET

Figure 7: Faster R-CNN results compared with UNET results with bounding boxes added.

One image has been chosen here to demonstrate how the two different approaches yield different results. Bounding boxes have been added to the U-net image to allow a more direct comparison. It can be seen that the Faster R-CNN approach detects much more of the fallen apples including one in the foreground that is obviously not fit for consumption. The reason for this could be due to how Faster R-CNN works, with regions of interest not being chosen based on whether they contain an apple just whether they contain an object. These regions are then fed into the CNN with no context such as location information, this means it cannot tell if apples are on the tree or the ground. The U-net approach also detects some of the fallen apples but far less than the Faster R-CNN approach. The U-net approach also detects apples that are obscured by leaves far better than the Faster R-CNN approach although this does lead to some occasions where the same apple is detected multiple times. However, the Faster R-CNN approach is better at separating individual apples, this can be seen towards the lower right of the Image where U-net has combined apples into one instance, but Faster R-CNN has recognized the same cluster of apples are separate instances, this cluster is circled in red in figures 6 and 7.

7. Conclusions and Future works

As can be seen in the results section, both approaches give incredibly good apple detection, the U-net method gives a slightly better count score so if the purpose of the user is just to count the number of apples then U-net is recommended, despite this U-net has been proven to have less accurate localisation and separation of different apples so for use cases where localisation is important U-net is not as preferable. An area where these approaches both show limitations is when identifying if an apple is on the tree or on the ground. This problem could lead to apples that are over ripe or rotting being harvested. A solution to this problem could be adding depth perception to the images fed to the networks, this additional information could make identifying fallen apples much easier. It could also make detecting which tree the apples belong to much easier, current methods currently detect apples that are not relevant as they are on another tree.

8. References

Girshick, R., Donahue, J., Darrell, T. and Malik, J., 2014. 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).

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

Gongal, A., Amatya, S., Karkee, M., Zhang, Q. and Lewis, K., 2015. Sensors and systems for fruit detection and localization: A review. *Computers and Electronics in Agriculture*, *116*, pp.8-19.

Häni, N., Roy, P. and Isler, V., 2020. Minneapple: A benchmark dataset for apple detection and segmentation. *IEEE Robotics and Automation Letters*, *5*(2), pp.852-858.

Häni, N., Roy, P. and Isler, V., 2020. A comparative study of fruit detection and counting methods for yield mapping in apple orchards. *Journal of Field Robotics*, *37*(2), pp.263-282.

Häni, N., Roy, P. and Isler, V., 2018, October. Apple counting using convolutional neural networks. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 2559-2565). IEEE.

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

Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2017. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), pp.84-90.

Lazebnik, S., Schmid, C. and Ponce, J., 2009. Spatial pyramid matching. *Object Categorization: Computer and Human Vision Perspectives*, *3*(4).

Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B. and Belongie, S., 2017. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2117-2125).

Powers, D.M., 2020. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*.

Ren, S., He, K., Girshick, R. and Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).

Ronneberger, O., Fischer, P. and Brox, T., 2015, October. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234-241). Springer, Cham.

Shapiro, L.G., 2001. Computer Vision/Linda G. *Shapiro, George C. Stockman–Pearson*.

Shelhamer, E., Long, J. and Darrell, T., 2017. Fully convolutional networks for semantic segmentation. *IEEE transactions on pattern analysis and machine intelligence*, *39*(4), pp.640-651.

Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Bargoti, S. and Underwood, J., 2017, May. Deep fruit detection in orchards. In *2017 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 3626-3633). IEEE.