Few-Shot Object Detection for Apple Leaf Diseases

BOHAN ZHANG

SID: 480170741

B.Eng (Hons)

Supervisor: Dr Mao Shan
External Supervisor: Dr Yongliang Qiao
External Supervisor: Dr Xinglong Lu

A thesis submitted in fulfilment of
the requirements for the degree of
Bachelor of Engineering (Hons)

School of AMME
Faculty of Engineering
The University of Sydney
Australia

8 February 2023

Abstract

Apple, as one of the most significant agriculture products, has shown its impact in both nutrition for human as well as economic contribution to the agriculture industry. Numerous researches have been reported to provide accurate detection solutions to apple leaf diseases at the early stage to improve the productivity. Deep learning as the state of the art in object detection, has shown its excellent performance on disease detection. However, due to the expert knowledge required for labelling various diseases and difficulties of capturing infected apple leaves in the wild, it usually takes significant effort to gather a sufficient number of images to prepare training dataset. In this thesis, we proposed a YOLOv5-based few-shot object detection method to detect apple leaf diseases to eliminate the requirement of many labelled images. The feasibility of the approach was analysed over a range of few images from 5 per class to 100, and a random selection of images with group of 20 was utilised to verify the accuracy. Additionally, we proposed a rating system for labels to provide the best selection for few-shot.

Experimental results showed that the proposed method had advantages of improved accuracy and running speed with comparison of traditional YOLOv5-style few-shot approach. Especially in very few shots, which increased 4.1% mean average precision (mAP) in 5-shot, 4% mAP in 10-shot overall. With the addition of rating system, our proposal reached 63.8% in 5-shot, 71.7% mAP in 10-shot, 73.9% in 20-shot, 75.9% in 30-shot, and 78.7% in 50-shot. Compared to target result of 80% mAP, our result at 50-shot were close enough for robotic spray. During the demo of sample images we proved the robustness of apple leaf disease detection, and verified the feasibility of robotic spray with our approach.

Statement of Student Contribution

I declare that the work done on this thesis is my own work. The summary of my work declared are as follows:

- I self-studied deep leaning and neural network at the beginning weeks, supported with the materials from Dr Yongliang Qiao.
- I carried out the literature survey to find achievements and challenges of apple leaf disease detection in recent 5 years. Dr Xinglong Lu provided advice for effective literature research skills.
- I proposed a few-shot based object detection neural network based on Yolov5 version 6.1, and designed relevant training methods running on Google Colab Pro. Dr Yongliang Qiao provided advice for the network design and tradeoff.
- I found over 4,000 images for apple leaf disease detection on Kaggle and labeled them with tool labelImg to construct dataset in thesis. Dataset was constructed for different purposes, including deep learning detection, few-shot object detection in both distributed shots and random shots, and rating system. Dr Yongliang Qiao provided advice for labeling technique.
- I proposed the rating system for optimising image selection in few-shot stage. I tested and found the most relevant factors for image labels, and designed linear relationship for them. I applied Matlab code to support data analysis on labels and rating methods. Dr Mao Shan provided guidance and support for designing rating system.
- I designed different experiments for apple leaf disease detection, recorded results on Excel, and plotted figures on Matlab. Dr Mao Shan provided guidance and support for random shot testing in 5 and 10 shots, and Dr Yongliang Qiao provided guidance and support for validation process.

- I applied Python code to support data preparation. These included image resize, label format conversion, random and distributed shot generation, and so on.
- I compared proposed method with peer methods, provided analysis of the performance compared to our design, and provided areas of improvements.
- I published my work in Github for future research with following links.
  - Proposed Network: https://github.com/bzha5848/ThesisNetowrk.git
  - Deep Learning Dataset: https://github.com/bzha5848/DeepLearnApple.git
  - Distributed few-shot Dataset: https://github.com/bzha5848/DistributFewShot.git
  - Random few-shot Dataset: https://github.com/bzha5848/RandomFewShot.git
  - Dataset of Rating system: https://github.com/bzha5848/RankFewShot.git
  - Other code covered in the thesis: https://github.com/bzha5848/SupportCode.git

The above represents an accurate summary of my contribution.

Signed by: Bohan Zhang

Acknowledgements

I greatly acknowledge my internal supervisor Dr Mao Shan who provided guidance and support during random shot testing and rating system, my external supervisor Dr Yongliang Qiao who provided guidance for Yolov5 few-shot network improvement and relevant testing design, and external supervisor Dr Xinglong Lu who provided academic support. I really appreciate their support over the thesis, thanks so much!!

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

As one of the most popular fruit, apple provides many nutrients to human body such as vitamin, antioxidants and potassium, brings significant contribution in the world economy and provides a lot of job positions in the apple industry. In the digital world, early diagnosis of apple leaf diseases becomes increasing important for orchardists, as it could provide a timely control of disease spread and reduce the cost of monitoring and degradation of apple qualities (Bin et al. 2017). Furthermore, it could also reduce the chemical pollution and maintain a stable development of apple industry (Bansal et al. 2021).

In this section, we introduced following topics to understand the background of the thesis. These included Apple Leaf Diseases and Current Challenges in Apple Leaf Diseases Detection. Based on the background, the aim of thesis was proposed and relevant questions were listed for research. We also highlighted the contribution of work and provided the structure of the thesis.

1.1 Apple Leaf Diseases

Common apple leaf diseases include rust, scab, powdery mildew, and frogeye leaf spot. Literature of these diseases would be provided to discuss the symptoms and causes, to obtain an understanding of the disease objects we would focus on.

1.1.1 Rust

Rust, or Cedar-apple rust, is a disease caused by the fungus Gymnosporangium. This disease shows yellow or orange-read spots on apple leaves and sometimes has a red boarder at the margin (Anderson and Uchida 2008). The symptoms of rust are shown in Fig. 1.1. As a result of the disease, it can result to severe leaf defoliation and infection, which reduces the production of apples as well as their qualities (Tummala 2021).

FIGURE 1.1: Rust (Tummala 2021)

1.1.2 Scab

As one of the most devastating diseases of apple leaf, scab has the symptoms of olive-green round spots, and becomes brown and dark with the time goes on (Chane and Boyraz 2017), as shown in Fig. 1.2. This disease is caused by fungus Venturia inaequalis. As a result, the disease would deform fruit and make leaf and make fruit drop more prematurely than other diseases. Without any control, it can infect over 90% apple leaves in the yield and cause massive economic loss (Chane and Boyraz 2017).

1.1.3 Powdery Mildew

Powdery Mildew is a symptoms caused by the fungus Podosphaera leucitrich. As shown in Fig. 1.3, the infected leaves with this symptoms is surrounded with the white fungal threads. As a result, it would attract every components of the apple plants, and prevent the creation of the fruit (Imre 2013a).

FIGURE 1.2: Scab (Sandskär 2003)

FIGURE 1.3: Powdery Mildew (Imre 2013b)

1.1.4 Frogeye leaf spot

As shown in Fig. 1.4, the symptoms of frogeye include tiny, purple specks; the spots are circular and has dark brown margins (Nicole Ward Gauthier 2016). The disease is caused by Cercospora sojina, which is a fungus that can be widespread more than 100 types of trees by wind, rain, and insects. As a result, it would cause over 30% reduction of the yields with yellow leaves and premature falls (Cochran et al. 2014).

1.2 Current Challenges in Apple Leaf Diseases Detection

Current leading researches focus on improving the detection of apple leaf diseases with deep learning. Although this method could reach a high accuracy in prediction, one of the biggest issues is that to achieve a good performance, hundreds or thousands of images would be required to be captured and labelled as the training objects. Unlike the common objects in the life, there are various of diseases in apple leaf; most of them require sufficient knowledge

FIGURE 1.4: Frogeye Leaf Spot (Nicole Ward Gauthier 2016)

to classify accurately and have to be labelled manually, which would cost many time and efforts. Another issue of deep learning is that it requires cost of much time in training due to the complexity of model. With various degrees of difficulties in detecting symptoms of diseases, these issues become much more obvious and affect the precision of accuracy (Hasan et al. 2020). Additionally, most of the literature utilises the experimental background for the leaf, which the results of accuracy would reduce with the background in wild.

The few-shot learning approach would be an alternative to the leaf disease detection. It could reduce the number of training images as well as efforts of modelling with transfer learning and fine-tuning process. But few researches have been conducted to apply this method in disease detection, and more of them focus on providing improved few-shot architecture and test the performance on classic COCO (Lin et al. 2014) and Pascal VOC dataset (Everingham et al. 2015).

1.3 Thesis Aim and Research Questions

With the challenges in previous section, we aimed to propose a few-shot object detection system for apple leaf diseases, with accuracy that was close to the result in deep learning.

So in this thesis we would consider these questions:

- Based on the existing few-shot objection methods, improve and provide an architecture that could achieve a better performance of detection on apple leaf diseases.
- Construct a dataset that uses the apple leaves taken in the wild instead of the lab. Use this dataset to make model and test the performance.
- Verify the proposed solution with a performance comparison of conventional deep learning methods, within the areas of accuracy, modelling time, input image and so on.
- Examine the robustness of the proposed model, including the accuracy with extremely few input images and performance of predicting diseases with complex symptoms.
- Develop the potential constraints of the proposed methods, analyse and find ways to reduce the impact.

1.4 Thesis Contribution

We contributed following work in the thesis:

- Carry out the literature survey to find achievements and challenges of apple leaf disease detection in recent 5 years.
- Propose a few-shot based object detection neural network based on Yolov5, and design relevant training methods with Google Colab Pro. The proposed network was published in Github.
- Find over 4,000 images for apple leaf disease detection and label them with tool labelImg to construct dataset in thesis. The dataset was published on Github.
- Propose the rating system for optimising image selection in few-shot stage. Test and find the most relevant factors for image labels, and design linear relationship for them. Apply Matlab to support data analysis on labels and rating methods.
- Design different experiments for apple leaf disease detection, record results on Excel, and plot figures on Matlab.

- Apply Python to support data preparation. These included image resize, label format conversion, random and distributed shot generation, and so on.
- Compare proposed method with peer methods and find areas of improvement.

1.5 Thesis Structure

An outline of the following contents in the thesis is shown as follows. In Literature review, we discussed the common classes of apple leaf diseases, and gave an overview of their symptoms as well as their impacts; an explanation of object detection with both deep learning and few-shot learning, and provided literature of current progress. In Method, we proposed our solution and designed the experiments. In Results, we demonstrated the results of experiments with corresponding data analysis. And in Conclusion, we discussed the achievements in our thesis and provided future work for fellow researchers.

CHAPTER 2

Literature review

This section reviewed the literature of object detection for deep learning & few-shot learning, to obtain an overview of the topics covered in this thesis. At the end of this section we provided questions for this thesis to solve.

2.1 Object Detection with Deep learning

Object detection refers to a computer vision approach that locates and shows the objects in the images or videos (Amit et al. 2020). A typical input of object detection would be an image with objects inside, and the output would be the bounding boxes of classes that align with the edges of objects detected, with a scale of the object instance included in the box. An example of the object detection is displayed in Fig. 2.1. The common ways of evaluating the accuracy of the object detection is to use precision, recall, and mean average precision (mAP).

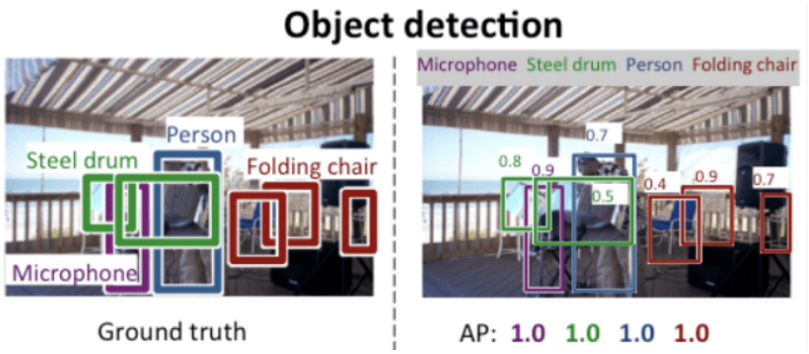

FIGURE 2.1: A demo of Object Detection (Russakovsky et al. 2014)

Recently, with the breakthrough of convolutional neural networks (CNN), more and more researches apply object detection with deep learning due to the following advantages CNN brings (Zhao et al. 2019).

- The capability of expression is significantly increased due to a deeper architecture
- Multi-task leaning manner that combines relevant tasks.
- Use hierarchical feature representation to increase learning abilities of high-level semantic features.

In recent 5 years, many researches have been conducted objection detection with model from 2 families: R-CNN and YOLO.

2.2 R-CNN

Region-based Convolutional Neural Networks (R-CNN) provides a human-like way of detecting object: firstly have an understanding of the scene, and then find the objects in a specific region we want (Zhao et al. 2019). As shown in Fig. 2.2, this way of thinking is accomplished by a 3-stage pipeline of object detection (Girshick et al. 2013):

- Region proposal generation: apply bounding boxes for objects to create regional proposal, which can reduce the efforts of space searching.
- Feature Extraction with CNN: use CNN to extract feature from each candidate region proposal.
- Localisation & Classification: classify features for one of the known classes.

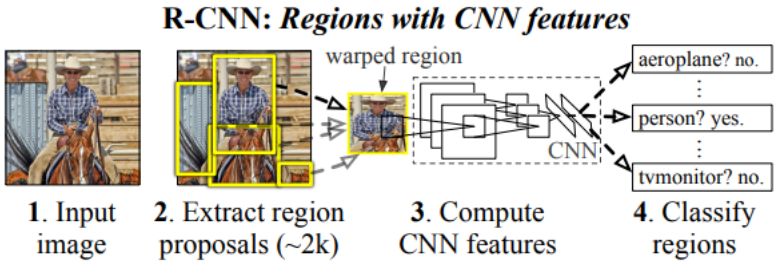

FIGURE 2.2: R-CNN (Girshick et al. 2013)

Compared to the old methods in past years, this approach improves over 30% accuracy in mAP and achieves an 53% accuracy.

2.2.1 Faster R-CNN

As one of the trending models in this family is Faster R-CNN. It is an extension of fast R-CNN that improves R-CNN in multi-stage pipeline, and reduces massive cost of training time & space, and low speed of object detection (Ren et al. 2015). As shown in Fig. 2.3, it utilises Region Proposal Network (RPN) to generating region proposals, and uses Region of Interest (ROI) Pooling that uses pre-trained network to extract features. As a result, this approach achieves 73.2% mAP in Pascal VOC 2007 and 70.4% mAP in Pascal VOC 2012 dataset (Ren et al. 2015).

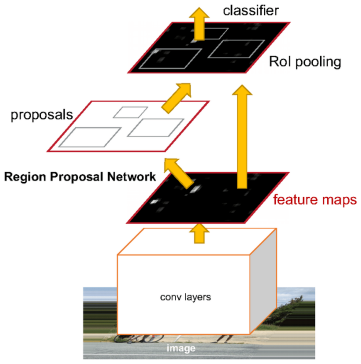

FIGURE 2.3: Faster R-CNN (Ren et al. 2015)

However, one of the limitations of this network family is operational speed.As shown in Fig. 2.4, most of the networks are slow to detect images and seem hard to be implemented in real-time detection (Ren et al. 2015).

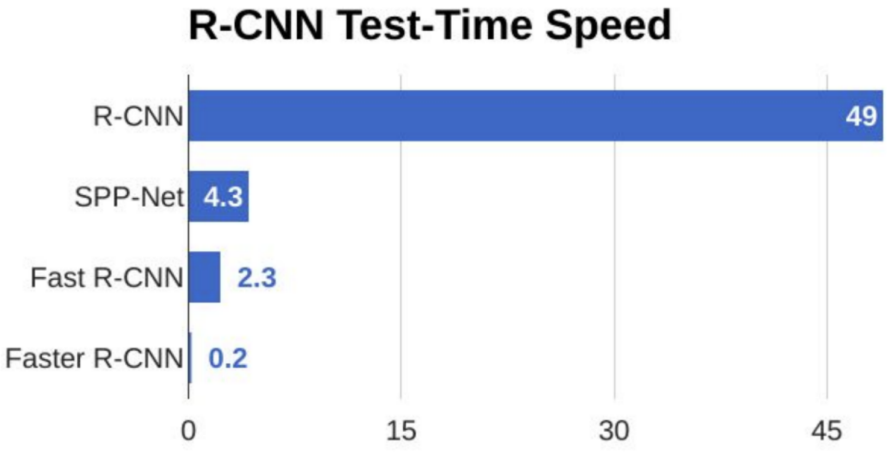

FIGURE 2.4: Detection Speed for R-CNN Family (Ren et al. 2015)

2.2.2 Application of R-CNN in apple leaf disease detection

In recent years, most of the research in this area focuses on applying Faster R-CNN with robust detection layers for extracting features. For example, Sardoğan et al. 2020 applied Faster R-CNN architecture with network Inception v2, to distinguish black spot from healthy leaves. As shown in Fig. 2.5, the way Inception v2 is applied was to perform as the feature extractor in the network, and then applied the feature map to Faster R-CNN to do the rest of the work. As a result, the proposed approach achieved an overall accuracy of 84%.

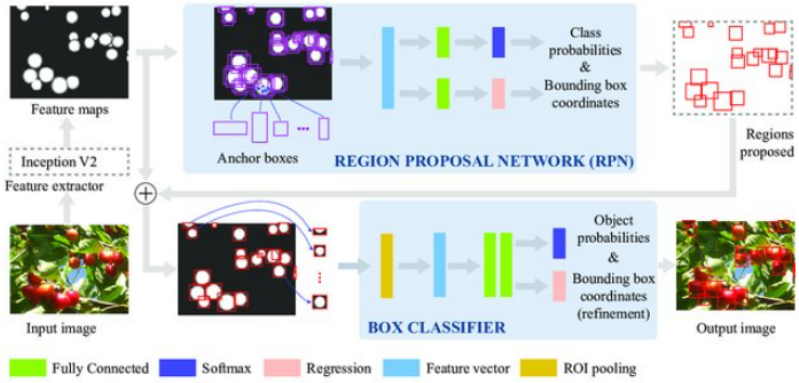

FIGURE 2.5: Faster R-CNN with Inception v2 (Villacrés and Auat Cheein 2020)

2.3 YOLO

Another popular family is YOLO. It refers to 'You only look once'. This method provides a single end-to-end network for training. An explanation of the network process is shown as follows (Handalage and Kuganandamurthy 2021). Firstly, it splits the input images into grids and for each cell it would predict the coordinate of the potential center of the bounding boxes within the cell. After this process Non Maximal Suppression would be used to combining duplicated bounding boxes and remove those which have a lower score of probability. A demo of this process is summarised in Fig. 2.6.

Due to this network structure, YOLO manages to achieve a high prediction speed at 45-155 frames per second (fps), which is favourable for real-time object detection (Handalage and Kuganandamurthy 2021).

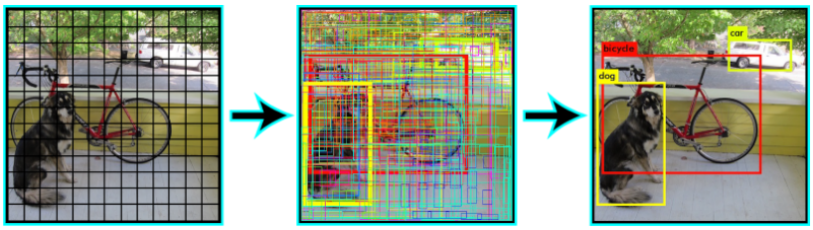

FIGURE 2.6: A demo of YOLO inter-work (Handalage and Kuganandamurthy 2021)

2.3.1 YOLOv5

One of the most popular model of YOLO in recent years is YOLOv5. As shown in Fig. 2.7, the backbone of YOLOv5 applied CSPDarknet that reduces the repeat of gradient information for feature map as well as the model size, improving running speed; the neck applied PANet that increases the detection of low-level features (Thuan 2021).

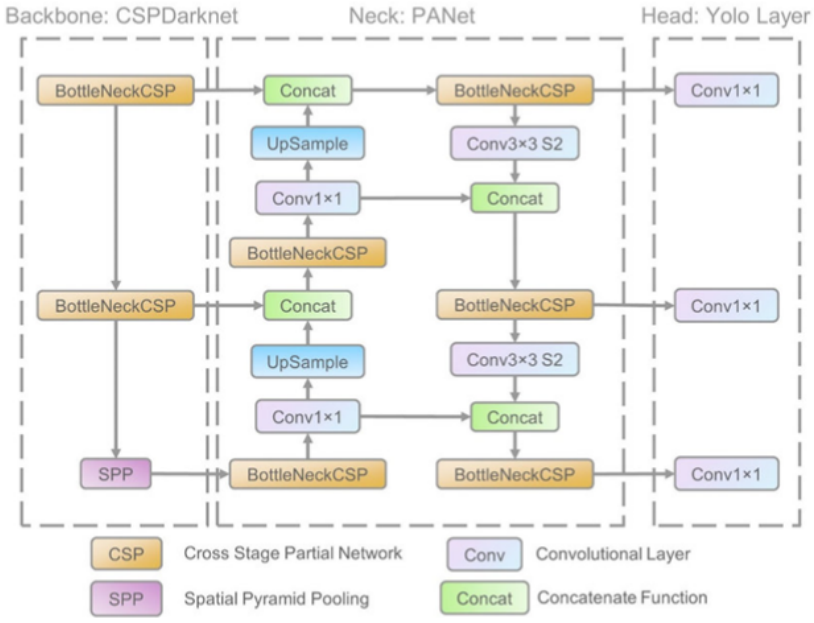

FIGURE 2.7: Network for YOLOv5 (Thuan 2021)

2.3.2 Application of YOLO in apple leaf disease detection

Besides applying the CNN networks as R-CNN, which were integrated with YOLO to extract more features, the improvements of YOLO for detection leaf diseases also included introducing ROI network to enhance feature extraction (Son 2021).

2.4 Attention Mechanism

One of the most significant approaches to increase the deep learning performance is to use attention mechanism. This method uses a specific algorithm to let model focus on key weights for feature extraction; which helps the model to achieve a better accuracy (Hernández and Amigó 2021).

For example, Adaptively Spatial Feature Fusion (ASFF), as one of the trending attention mechanisms used in Yolov3, improved the uniform of feature scale by filtering the conflict information from different levels of features (Li et al. 2020). It consisted of feature resizing and adaptive fusing.

Feature resizing, was applied to resize the channels and resolutions from 3 different levels into the same. To achieve this, a 1×1 convolution layer was applied for up-sampling; and a 3×3 convolution layer with a stride of 2 was implemented to down-sampling.

In terms of adaptive fusing, as shown in Fig. 2.8, each grid $y_{i,j}$ on feature map should be calculated with corresponding weight on each level as shown in E.q.(2.1), to get the final value for ASFF. And for weight on each level they followed the formula in E.q.(2.2). Softmax was used as the way to optimise these weight parameters.

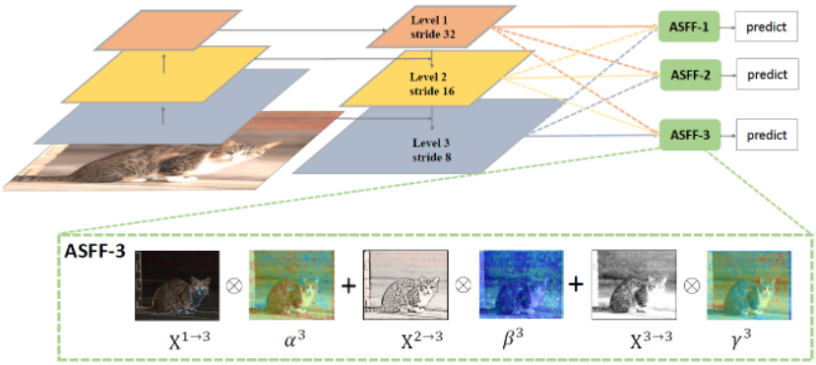

FIGURE 2.8: Adaptive Fusing (Li et al. 2020)

$$y_{i,j}^{n} = \alpha_{i,j}^{n} * x_{i,j}^{1\to n} + \beta_{i,j}^{n} * x_{i,j}^{2\to n} + \gamma_{i,j}^{n} * x_{i,j}^{3\to n} \tag{2.1}$$

$$\begin{cases} \alpha_{i,j}^n + \beta_{i,j}^n + \gamma_{i,j}^n = 1 \\ \alpha_{i,j}^n, \beta_{i,j}^n, \gamma_{i,j}^n \in [0,1] \\ \alpha_{i,j}^n = \frac{e^{\lambda_{\alpha,i,j}^n}}{e^{\lambda_{\alpha,i,j}^n} + e^{\lambda_{\beta,i,j}^n} + e^{\lambda_{\gamma,i,j}^n}} \end{cases} \tag{2.2}$$

As a result, ASFF managed to achieve an overall accuracy improvement of 10 points, shown in Fig. 2.9.

Method	Backbone	FPS	Avg. Precision, IoU: 0.5:0.95	0.5	0.75	Avg. Precision, Area: S	M	L
two-stages:								
Faster w FPN [20]	ResNet-101-FPN	11.0 (V100)	39.8	61.3	43.3	22.9	43.3	52.6
Mask R-CNN [9]	ResNext-101-FPN	6.5 (V100)	41.4	63.4	45.2	24.5	44.9	51.8
Cascade R-CNN [3]	ResNet-101-FPN	9.6 (V100)	42.8	62.1	46.3	23.7	45.5	55.2
SNIPER [34]	ResNet-101	5.0 (V100)	46.1	67.0	51.6	29.6	48.9	58.1
one-stages:								
SSD300* [25]	VGG	43 (Titan X)	25.1	43.1	25.8	–	–	–
RefineDet320 [42]	VGG	38.7 (Titan X)	29.4	49.2	31.3	10.0	32.0	44.4
RFB Net300 [23]	VGG	66.0 (Titan X)	30.3	49.3	31.8	11.8	31.9	45.9
YOLOv3 @320 [31]	Darknet-53	69 (V100)	28.2	–	–	–	–	–
YOLOv3 @320 + ASFF (Ours)	Darknet-53	**63** (V100)	**36.7**	**57.2**	**39.5**	**15.8**	**39.9**	**51.3**
YOLOv3 @320 + ASFF* (Ours)	Darknet-53	**60** (V100)	**38.1**	**57.4**	**42.1**	**16.1**	**41.6**	**53.6**
SSD512* [25]	VGG	22 (Titan X)	28.8	48.5	30.3	–	–	–
RefineDet512[42]	VGG	22.3(Titan X)	33.0	54.5	35.5	16.3	36.3	44.3
RFB Net512 [23]	VGG	33 (Titan X)	33.8	54.2	35.9	16.2	37.1	47.4
RetinaNet500 [21]	ResNet-101-FPN	17.8 (V100)	34.4	53.1	36.8	14.7	38.5	49.1
CornerNet-511 [18]	Hourglass-104	5.0 (Titan X)	40.5	56.5	43.1	19.4	42.7	53.9
CenterNet-DLA511 [44]	DLA-34	28 (Titan Xp)	39.2	57.1	42.8	19.9	43.0	51.4
YOLOv3 @416 [31]	Darknet-53	60 (V100)	31.0	–	–	–	–	–
YOLOv3 @416 + ASFF (Ours)	Darknet-53	**56** (V100)	**39.0**	**60.2**	**42.5**	**19.6**	**42.3**	**51.4**
YOLOv3 @416 + ASFF* (Ours)	Darknet-53	**54** (V100)	**40.6**	**60.6**	**45.1**	**20.3**	**44.2**	**54.1**
RetinaNet800 [21]	ResNet-101-FPN	9.3 (V100)	39.1	59.1	42.3	21.8	42.7	50.2
FCOS-800 [36]	ResNet-101-FPN	13.5 (V100)	41.0	60.7	44.1	24.0	44.1	51.0
NAS-FPN @640 [8]	ResNet-50	17.8 (P100)	39.9	–	–	–	–	–
NAS-FPN @1280[8]	ResNet-50	5.2 (P100)	**46.6**	–	–	–	–	–
YOLOv3 @608 [31]	Darknet-53	52 (V100)	33.0	57.9	34.4	18.3	35.4	41.9
YOLOv3 @608 + ASFF (Ours)	Darknet-53	**46.6** (V100)	**40.7**	**62.9**	**44.1**	**24.5**	**43.6**	**49.3**
YOLOv3 @608 + ASFF* (Ours)	Darknet-53	**45.5** (V100)	**42.4**	**63.0**	**47.4**	**25.5**	**45.7**	**52.3**
YOLOv3 @800 + ASFF* (Ours)	Darknet-53	**29.4** (V100)	**43.9**	**64.1**	**49.2**	**27.0**	**46.6**	**53.4**

FIGURE 2.9: ASFF Performance (Li et al. 2020)

2.5 Few-shot Object Detection

Although object detection with deep learning could achieve a high accuracy, one if the biggest issues is that this method requires a lot of images for training with a good result. In real case, however, this is hard to achieve for some classes which have insufficient images to be trained. In this case, few-shot object detection is proposed as a solution to take few images for training but manages to achieve a satisfying result compared to the deep learning.

The problem of few-shot object detection is defined as follow (Antonelli et al. 2022). Firstly, the training dataset includes base training and novel training. The base training dataset would

have sufficient number of images to train, and for the novel dataset for training is K*N images, where K is few number of image shots per class (normally we use 1-10 images, but Fan et al. 2021 and Wu et al. 2020 also implement 30 shots per class for COCO format); N is the number of novel classes. The approach suggested for training is to train the backbone with sufficient data such as COCO and Pascal VOC to generate low-level detection ability, and then fine-tuning with the novel classes to generate high level features of the objects. To enhance the feature extraction for novel class, most of the researches applied few shots for both base class and novel class in fine-tuning stage.

Besides, the dataset used for few-shot object detection in recent year focused on Pascal VOC and COCO (Fan et al. 2021). And 3-5 classes from these dataset would be taken as novel class, while the rest would be taken as base class(Antonelli et al. 2022).

Antonelli et al. 2022 categorises few-shot object detection into 2 main approaches: Meta Learning and Transfer Learning.

2.5.1 Meta Learning

Meta Learning is summarised as 'learn to learn'; which means to design a way to model the knowledge of current data that can also be applied to novel data, which has only few images to take (Liu and Fu 2021).

2.5.2 Transfer Learning

Transfer Learning refers to re-using the pre-trained weight from base model to construct a network that also manages to predict novel classes with few shots (Antonelli et al. 2022). Compared to meta learning, transfer learning has a easier approach and some easy modifications such as changes in loss function and adding branches on current architecture would help to improve the accuracy of transfer learning. Most of the methods in few-shot applied this learning to save modelling time.

2.5.3 Image Selection in Few-shot

All the literature we have researched did not focus on the image selection for each class; instead these literature focused on improving detecting performance with new networks and selected randomly in each class.

2.5.4 Few-shot Examples in Faster RCNN

Most of the researches in few-shot object detection focused on applying modification on Faster RCNN, by applying attention mechanism in some of the components like RPN to enhance feature extraction.

2.5.4.1 Few-Shot Object Detection (FSOD) with Attention-RPN and Multi-Relation Detector

One of the most recent literature was to apply Attention-Based Region Proposal Network (Attention RPN) & Multi-Relation Detector Fan et al. 2020. As shown in Fig. 2.10, this thesis provided a method that learned the relationship between the support set (images labelled) and query set (images to predict). It firstly utilised Faster RCNN as the weight-shared network to obtain the general knowledge between the support images and the query ones in the same categories.

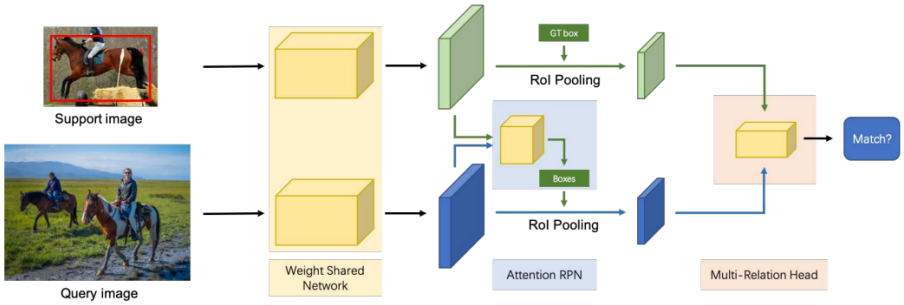

FIGURE 2.10: Network for Attention-RPN and Multi-Relation Detector (Fan et al. 2020)

Then the thesis applied the attention RPN to filter the irrelevant categories and background boxes, based on the information given from the support categories. As shown in Fig. 2.11,

this module averagely pooled the support features X into a 1*1*C vector, then computed deep-wise cross correlation with query feature Y, to generate the resultant attention feature map G. The formula of G was shown as below. The RPN model in this thesis adopted RESNet50 as the top layers of features.

$$G_{h,w,c} = \sum_{i,j} X_{i,j,c} * Y_{h+i-1,w+j-1,c} \tag{2.3}$$

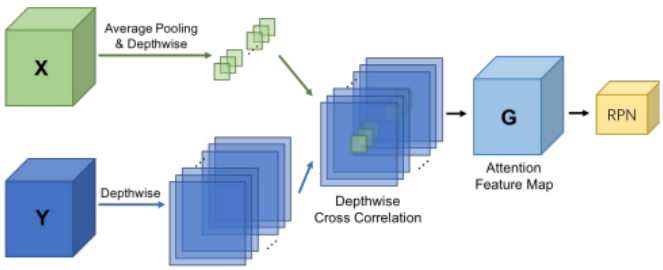

FIGURE 2.11: Attention-RPN

Then, Multi-Relation Detector was used to re-score the proposals and distinguish each categories based on the feature map extracted from attention RPN. As shown in Fig. 2.12, the detector consists of 3 heads:

- Global-relation head: match images in global representation.
- Local-correlation head: match relationship between support and query set in pixel-wise and depth-wise cross correlation.
- Patch-relation head: math one-to-many pixel relationship (match matching) with a deep non-linear metric.

With these methods, the proposed network achieved a satisfying result within very few shots, shown in Fig. 2.13

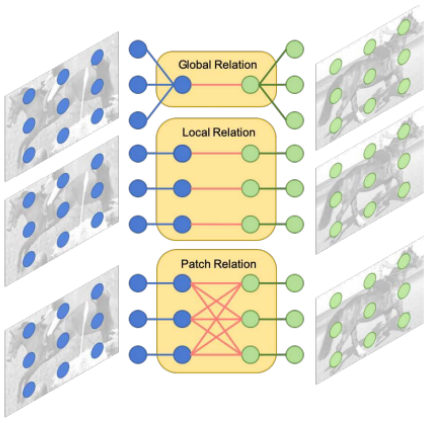

FIGURE 2.12: Multi-Relation Detector

Training Strategy	Attention RPN	AP_{50}	AP_{75}
1-way 1-shot		55.0	39.1
1-way 1-shot	✓	55.9	41.1
2-way 1-shot		63.8	42.9
2-way 5-shot		65.4	43.7
2-way 5-shot	✓	**67.5**	**46.2**
5-way 5-shot	✓	66.9	45.6

FIGURE 2.13: Result of FSOD with attention RPN

2.5.5 Few-shot Examples in Yolo

The few-shot application on Yolo family however, seemed much less than Faster RCNN. And most of the literature in this area only focused on using classic Yolo and Yolov2 as baseline network.

2.5.5.1 Few-shot Object Detection via Feature Reweighting

One of the most recent thesis in Yolo few-shot was to introduce a Feature Reweighting module in Yolov2. As shown in Fig. 2.14, the network firstly applied the backbone of Yolov2 (Darknet) as the feature extractor for the query images (images to predict), then the re-weighting module was applied for support images (labeled images) to re-weight the meta feature of each category, and produce new re-weighting vectors for the novel classes. Finally, the prediction layer would optimise the classification of each category and produce the results.

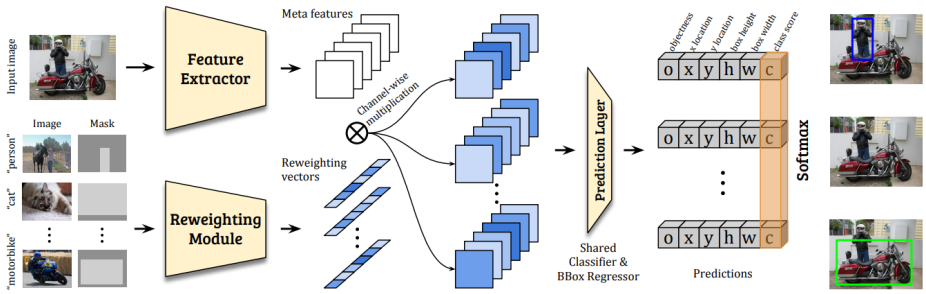

FIGURE 2.14: Network of Feature Reweighting on Yolov2 (Kang et al. 2019)

As a result, the proposed method achieved an obvious improvement compared to the existing few-shot methods for Yolo, shown in Fig. 2.15.

		Average Precision						Average Recall					
# Shots		0.5:0.95	0.5	0.75	S	M	L	1	10	100	S	M	L
10	YOLO-ft	0.4	1.1	0.1	0.3	0.7	0.6	5.8	8.0	8.0	0.6	5.1	15.5
	YOLO-ft-full	3.1	7.9	1.7	0.7	2.0	6.3	7.8	10.5	10.5	1.1	5.5	20
	LSTD(YOLO)	0.4	1.1	0.2	0.2	0.7	0.6	5.8	7.9	7.9	0.6	5.0	15.3
	LSTD(YOLO)-full	3.2	8.1	2.1	**0.9**	2.0	6.5	7.8	10.4	10.4	1.1	5.6	19.6
	Ours	**5.6**	**12.3**	**4.6**	**0.9**	**3.5**	**10.5**	**10.1**	**14.3**	**14.4**	**1.5**	**8.4**	**28.2**
30	YOLO-ft	0.6	1.5	0.3	0.2	0.7	1.0	7.4	9.4	9.4	0.4	3.9	19.3
	YOLO-ft-full	7.7	16.7	6.4	0.4	3.3	14.4	11.7	15.3	15.3	1.0	7.7	29.2
	LSTD(YOLO)	0.6	1.4	0.3	0.2	0.8	1.0	7.1	9.1	9.2	0.4	3.9	18.7
	LSTD(YOLO)-full	6.7	15.8	5.1	0.4	2.9	12.3	10.9	14.3	14.3	0.9	7.1	27.0
	Ours	**9.1**	**19.0**	**7.6**	**0.8**	**4.9**	**16.8**	**13.2**	**17.7**	**17.8**	**1.5**	**10.4**	**33.5**

FIGURE 2.15: Result of Feature Reweighting on Yolov2

2.6 Research Questions

Based on literature covered above, we derived the following questions.

- Based on our application, which deep learning family seems to be more feasible for object detection on robotics?
- What apple leaf diseases are feasible for few-shot object detection? Is there any structure modification required to be applied on selected network selected from last question?
- Based on selected network, does the methodology of few-shot suits the network well? Would there any modification required for the network?
- What is the scope of our research, including the scope of dataset, experiments, as well as time limit?

CHAPTER 3

Methods

3.1 Project Scope

We considered following aspects as our scope of the project:

- The selected performance metrics and target result was specified in section 3.2.3.
- All the images in dataset were labelled with optimal performance.
- Few-shot method had access to all the images and labels in ranking system.
- Version control of YOLOv5: version v6.1 was applied in this project.
- The working environment should agree with section 3.3

3.2 Dataset

3.2.1 Dataset Preparation

We selected the dataset from Plant Pathology 2021 (FGVC8) in Kaggle (https://www.kaggle.com/c/plant-pathology-2021-fgvc8/data). It collected over 18K RGB images of leaf diseases in unsprayed apple orchards, captured with different sources of cameras (Thapa et al. 2020). The dataset included 4 categories including frogeye leaf sopt, rust, scab, and powdery mildew. Due to the scope of the project, 1000 images in each class were selected, 700 of them for training, and the rest 300 for testing.

3.2.2 Image Labelling and Label Processing

According to Thapa et al. 2020, the dataset was originally for image classification, and all the images were required to be labelled with bounding boxes for object detection. To label disease objects in images, labelImg (Tzutalin 2015) was selected to create bounding boxes, as shown in Fig. 3.1. Labels were saved as Pascal VOC format (.xml).

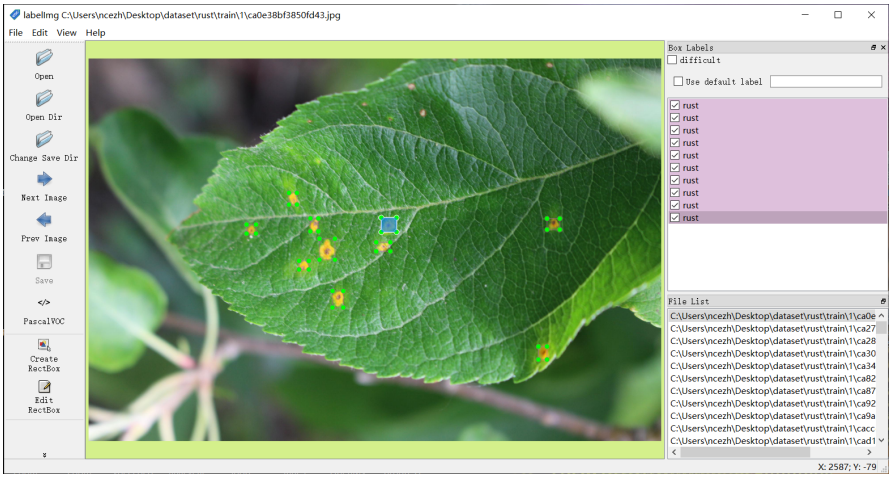

FIGURE 3.1: Use labelImg to label diseases

To deal with different purposes of implementation, the following codes were developed in language of Python & Matlab:

- Convert labels among various formats including Pascal VOC (.xml), COCO yolov5 (.txt), and COCO JSON (.json).
- Re-assemble dataset with specific order of classes.
- Find classes of interest based on their class number or name in dataset.
- Resize image into specific size (such as 416*416) and formats (including .jpeg) for different dataset.
- Select labels for different purposes including randomly selection, well-distributed selection, and ranking with their characteristics such as label size & number.

3.2.3 Leaf Diseases Selection

As shown in Table. 4.1 in section 4.1, the accepted disease class included frogeye leaf spot and rust.

Based on the results, we summarised the following target of performance:

- We selected mean average precision (mAP) as our main result of performance, although precision and recall are analysed as well.
- The overall mAP of frogeye and result were at 85%, so we set our target few-shot result as close to 80%.

3.3 Working Environment

We summarised our working environment into following subsections:

3.3.1 Colab

Due to the GPU required for deep learning and few-shot learning, all the model were built on Colab Pro (Bisong 2019), and all the code should have the packages which does not violate the Colab Environment. All the GPU-related setting such as batch size should be adjusted to ensure work can be finished during thesis.

3.3.2 Github

Github (github 2020) was utilised to store and load the dataset and modelling code of this project into repository. All the repositories were set as private to allow limited access. A demo of using Colab Pro with Github was shown in Fig. 3.2.

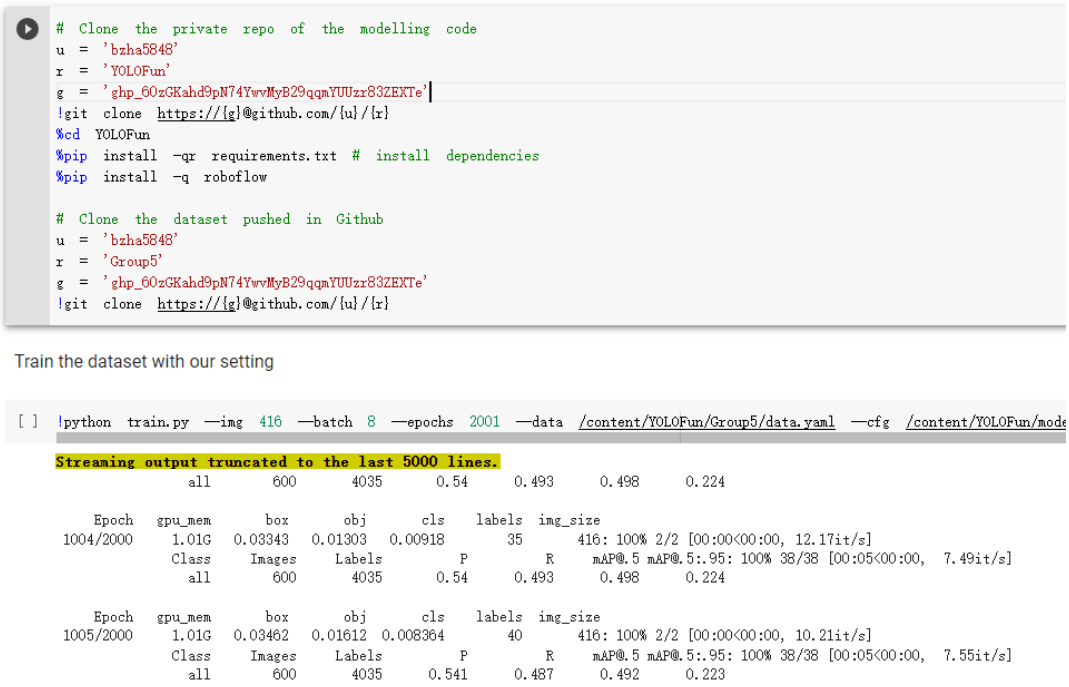

FIGURE 3.2: Illustration of code application on Colab Pro with Github

3.3.3 Google Drive

Google Drive was used to store all the checkpoints and weights trained from Colab, for testing and training resume.

3.4 Performance Evaluation in Object Detection

We summarised following evaluation methods for the accuracy of deep learning and few-shot object detection methods.

3.4.1 Mean Average Precision (mAP)

Mean Average Precision (mAP) is one of the most common metrics for object detection. It compared the ground truth values between the labeled bounding boxes as well as the detected ones. It is calculated with the mean of weighted precision at each threshold, shown as the equation below. The threshold we used in this thesis was 50%.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP^i \tag{3.1}$$

3.4.2 Precision

Precision is used to measure how many labels are correctly predicted as diseases. It can be calculated with the following formula.

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

3.4.3 Recall

Recall is used to measure how many of the diseases in the actual case are predicted correctly. It can be calculated with the following formula.

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

3.4.4 F1 Score

F1 score is used as the harmonic mean of precision and recall, which would provide a balance performance of the results for the objects. It can be calculated with the following formula.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3.4}$$

3.5 Adapted YOLOv5 for Few-shot object detection

For a better result of few-shot detection with Yolov5, we applied the following modifications:

3.5.1 Version Control - v6.1

We selected Yolov5, version 6.1 (v6.1) (Liu et al. 2022b) as our version control. This version was published in Feb, 2022, which was the latest version of Yolov5 in our thesis. As shown in Fig. 3.3, this version had the following improvement compared to the original Yolov5 structure shown in Fig. 3.4:

- Replace Focus layer (at layer 1) in backbone with 6*6 convolution layer to increase computational efficiency.
- Replace Spatial Pyramid Pooling (SPP) He et al. 2015 with Improved Feature Fusion Network (SPPF) Liu et al. 2022a to reduce FLOPs of computation by using half of the parameter as SPP.
- Replace layer BottleNeckCSP with layer C3 to improve the feature extraction.

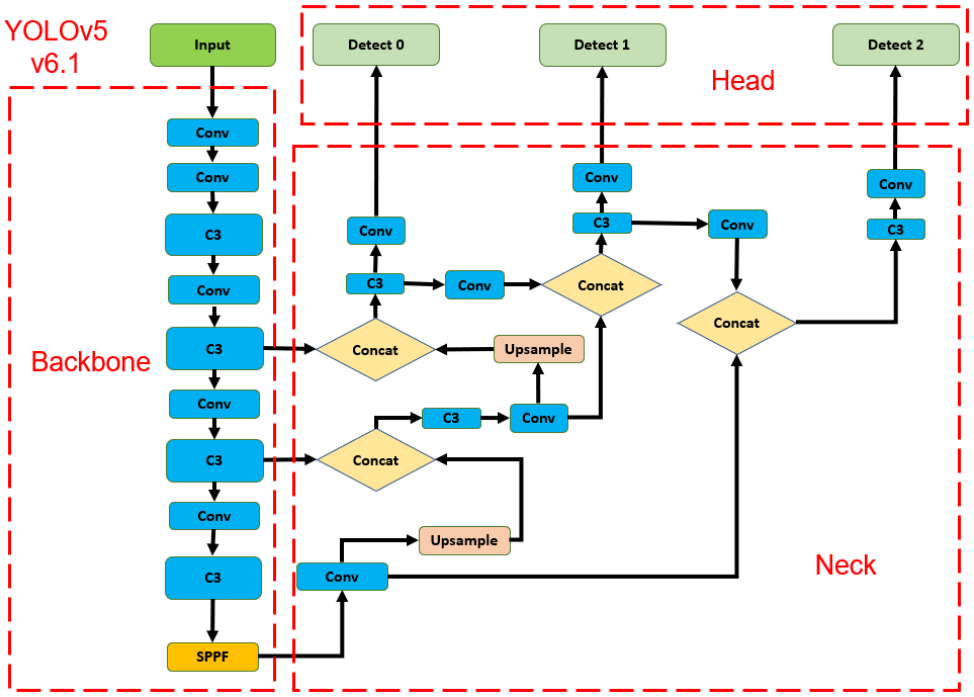

FIGURE 3.3: Network of Yolov5 v6.1 (Liu et al. 2022b)

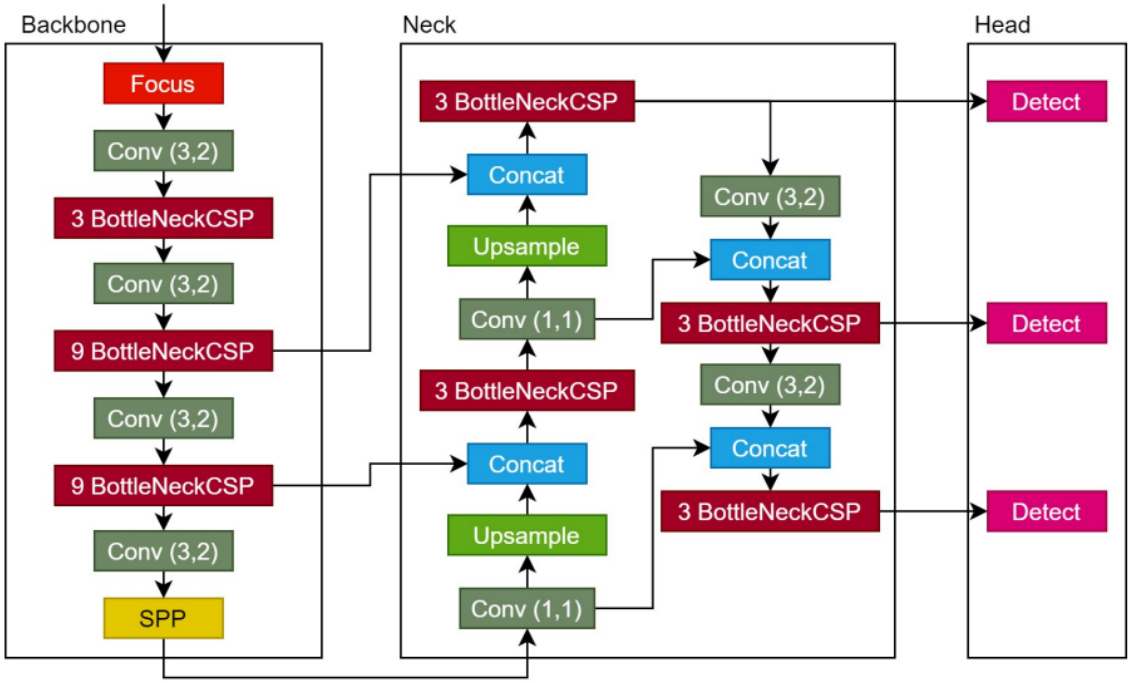

FIGURE 3.4: Network of Yolov5 Origin (Liu et al. 2022b)

3.5.2 Training method Adaption

We utilised following setting in our training and modelling process:

- Due to the GPU limit, we selected Yolov5 Medium Model (Yolov5m) in our design. The size of model decides the parameters in modelling process. Using the medium model could help us to use sufficient number of parameters of modelling while considering the GPU memory in Colab Pro.
- We used the backbone of Yolov5 v6.1 pre-trained by COCO dataset, with epoch of 300. The purpose of using freezed backbone was to obtain a model that had a good detection performance of the generic objects such as human and bike. As shown in Fig. 3.5, with over 500 epoch in training, the performance of model with freezed backbone achieved a close detect mAP result with the default model, which was suitable for our few-shot design.
- In training setting, we applied the batch size of 8, learning rate of 0.01, image size of 416*416, epoch of 2000, as well as using hypermater of high range (29 parameters) as our training settings from scratch to improve the feature detection.

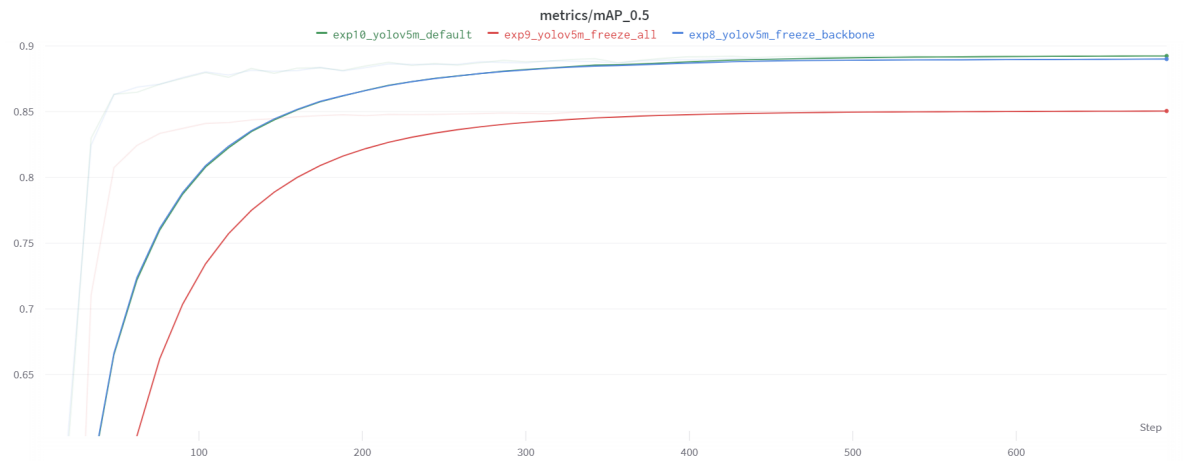

FIGURE 3.5: Detection performance of using model with freezed backbone and the default setting

3.5.3 Head Adaption

By applying the feature map shown in Fig. 3.7, the C3 layer 23 which was used as the C3 layer for Detection 2 in head, had a bad feature extraction. This was due to an over-convolution result that filtered the feature of the small targets. As shown in Fig. 3.6, most of the disease labels were very small, it would be less beneficial to use layer 23.

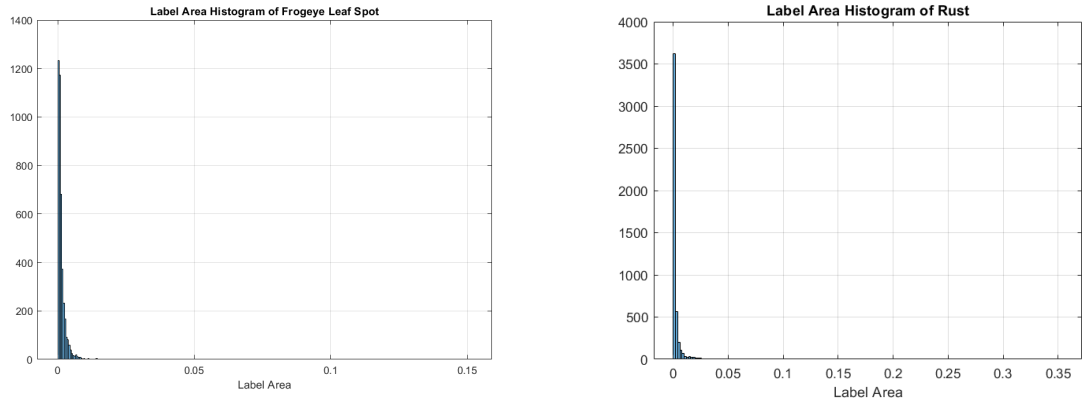

FIGURE 3.6: Label distribution for both classes

To match the Feature Pyramid Network (FPN) structure in head of Yolov5, which requires at least 3 layers of C3 for a feature comparison, we applied C3 layer 13 as an alternative, which showed a better performance of feature extraction in small targets.

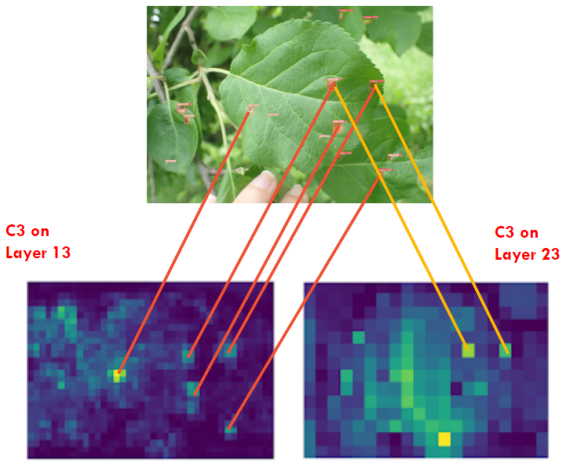

FIGURE 3.7: Feature map in C3 layer 23 compared to C3 layer in 13

3.5.4 Introduction of Attention Mechanism

We introduced attention mechanism in our design to enhance the performance of detection with fewer images.

3.5.4.1 Content-Aware ReAssembly of Features (CARAFE)

Based on many testing of mechanisms and analysis of the disease objects, we applied CARAFE as our improvement of upsampling in Yolov5. It applied following methods to improve the traditional upsampling method which was to extract features after convolution.

CARAFE consisted of 2 modules: kernel prediction module which generated the reassembly kernels with a content-aware method; and the content-aware reassembly module which reassembled the features within a local region (Wang et al. 2019).

As shown in Fig. 3.8, with the input of a feature map X with size of C*H*W and the integer ratio of upsampling σ, CARAFE would generate a new map X' with the size of $C * \sigma H * \sigma W$. The output map X' would have many target locations $L' = (i', j')$, corresponding to the source locations $L = (i, j)$ with the map X. And $(i', j') = (i\sigma, j\sigma)$ in that case. Here we denoted the neighbour of X with sub-region of k*k as $N(X, k)$, which centered at source location L.

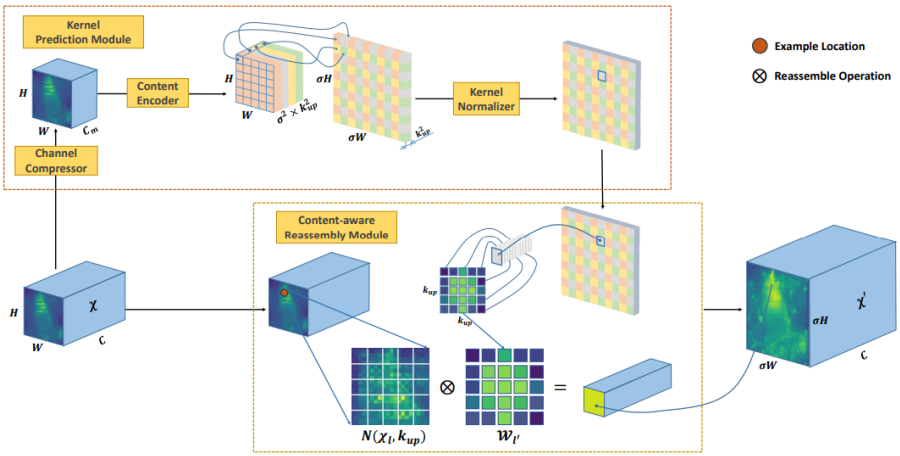

FIGURE 3.8: CARAFE structure (Wang et al. 2019)

Based on the previous formulations of the CARAFE framework, The kernel prediction module consisted of 3 sub-modules.

- Channel Compressor. CARAFE utilised a 1 * 1 convolution layer to compress the channel C into C_m, which would reduce parameters and computational cost.
- Content Encoder. CARAFE applied a $k_{encoder}$ * $k_{encoder}$ convolution layer to generate reassembly kernels from last sub-module, which would result to the encoder parameters with number of $k_{encoder} * k_{encoder} * C_m * C_{up}$. By testing, CARAFE selected the relationship between the encoder kernel and upsampling kernel as $k_{encoder} = k_{up} - 2$.
- Kernel Normalizer. This module normalised the reassembly kernel with softmax before implementing to the feature map.

With these operations in sub-module, the kernel prediction module managed to predict a local-wise kernel $W_{l'}$ with following equation.

$$W_{l'} = \psi(N(X_l, k_{encoder})) \tag{3.5}$$

The second module was Content-aware Reassembly module, which took $W_{l'}$ from the kernel prediction module and reassembled the features of neighbour $X'_{l'}$ with function ϕ. The equation of this module was shown as below.

$$X'_{l'} = \phi(N(X_l, k_{up}), W_{l'})$$

$$X'_{l'} = \sum_{n=-r}^{r} \sum_{m=-r}^{r} W_{l'(n,m)} * X_{(i+n,j+m)}$$

$$r = k_{up}/2$$

With the following structure, CARAFE managed to apply the better kernels of adapted shape instead of the fixed ones, which was shown in Fig. 3.9. Based on the irregular shape of diseases, CARAFE seemed more suitable in this project compared to the traditional upsampling method like deconvolution and interpolations.

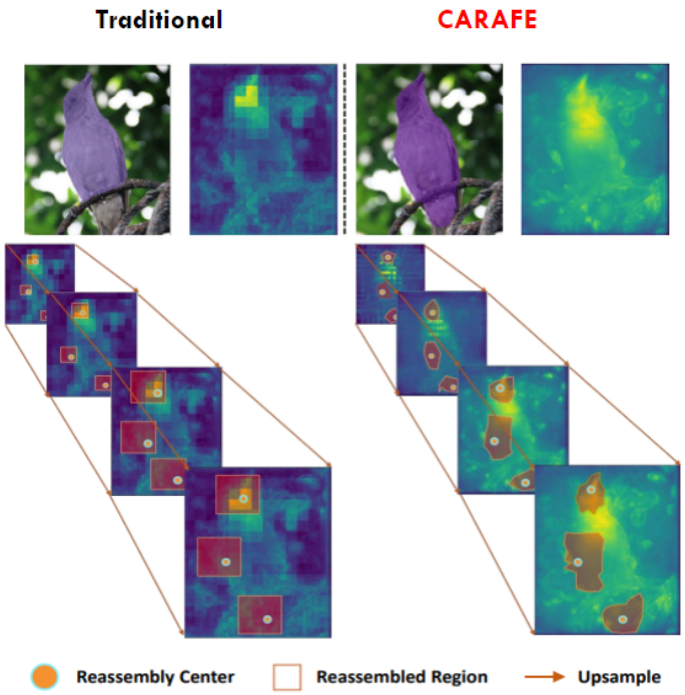

FIGURE 3.9: Feature kernels in CARAFE compared the ones in traditional deconvolution upsampling (Wang et al. 2019)

With these modifications in upsampling structure, CARAFE improved the process in the following areas:

- Large field of view. CARAFE provided a larger receptive field to extract contextual information instead of using information from pixel neighbourhood.

- Content-aware handling. CARAFE applied an adapted kernels in feature extraction instead of fixed kernel, with the structure of instance-specific content-aware handling.
- Lightweight and fast to compute. With Channel Compressor sub-module in kernel prediction module, the parameters and computational cost was largely reduced.

As a result of applying CARAFE as upsampling module in Faster RCNN, the one with CARAFE achieved the best mAP performance and a relatively low parameters for computation, as shown in Fig. 3.10.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	FLOPs	Params
Nearest	36.5	58.4	39.3	21.3	40.3	47.2	0	0
Bilinear	36.7	58.7	39.7	21.0	40.5	47.5	8k	0
N.C.	36.6	58.6	39.5	21.4	40.3	46.4	4.7M	590k
B.C.	36.6	58.7	39.4	21.6	40.6	46.8	4.7M	590k
Deconv [27]	36.4	58.2	39.2	21.3	39.9	46.5	1.2M	590k
P.S.[32]	36.5	58.8	39.1	20.9	40.4	46.7	4.7M	2.4M
GUM[23]	36.9	58.9	39.7	21.5	40.6	48.1	1.1M	132k
S.A.[3]	36.9	58.8	39.8	21.7	40.8	47.0	28k	2.3k
CARAFE	**37.8**	**60.1**	**40.8**	**23.1**	**41.7**	**48.5**	199k	74k

FIGURE 3.10: Performance of CARAFE in Faster RCNN (Wang et al. 2019)

3.5.5 Summary of proposed Network

A summary of proposed network based on Yolov5 was shown in Fig. 3.11. The improvement included:

- Replace C3 layer 23 with C3 layer 13 for feature detection of small target.
- Replace CARAFE with upsampling in neck for an adapted kernel size of features, and other relevant benefits from CARAFE.

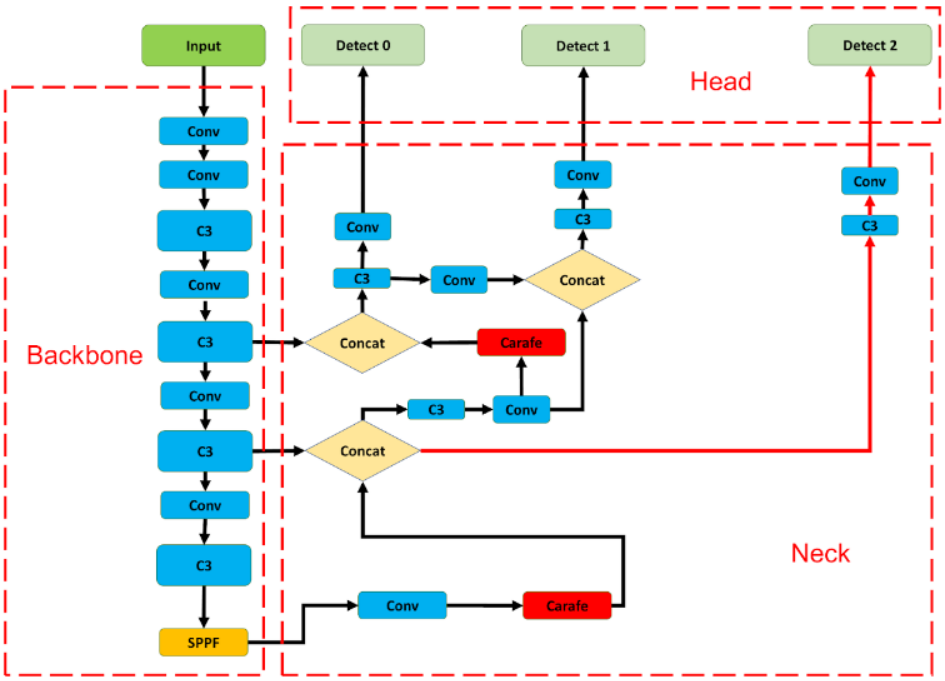

FIGURE 3.11: Proposed Network Structure

3.6 Rating System

Based on the results shown in section 4.3, we could see that in few-shot the selection of labels could affect the performance dramatically, so in this case we introduced the rating system of labels for the image selection of the few shot. The advantages of the rating system included:

- Reduce the uncertainty of the performance in few-shot selection and verify the maximum performance of the proposed network.
- Reliable and robust method to optimise the selection compared to the random selection method, helping researchers to re-produce the result and implement the method into other few-shot dataset.
- The development of the rating system could help researchers to understand the characteristics behind the images and labels, which could make more specific design for the few-shot object detection.

On the other hand, the limitation of the rating system included:

- Constrain of dataset: as currently we only designed the rating system based on labels, all the images in the dataset must be labelled correctly. As most of the recent literature were using fully labelled dataset in few-shot as well, this issue would not be obvious at current stage.
- Cost of development: as most of the past literature only focused on detecting performance of the network and rarely worked on rating system, we had to start from scratch, and conducted many tests on different rating methods.
- Transferability: this issue happened when more than 1 factor in the rating system, in that case numerous tests were required to balance different factors. As this process required the detection result to verify, a lot of time would be used on testing the model and verify the weights.

The strategy of designing and validating the rating systems was summarised as follows:

- Analyse the characteristics of the image labels and intuitively generate the rating systems with base factors like label areas and label number
- All the scores were obtained by re-scaling the output of the factors from 0-100. A higher score would be considered as better to be selected. For few-shot selection we select the top images with number of images we want.
- Perform test of system on 5-shot first as it has the shortest modelling time, if any method achieves a better performance than peer methods, then extend the test on shots 30 and 50, as these shots were considered as milestone in few-shot method. If the method is verified as good in these 2 types of shots, then finalise the test on 10 and 20 shots. For 100-shot, as this type of shot method was close to medium shot, so we did not apply it in rating system testing.
- After verifying the base factors, we considered applied modified factors based on them. These included label intensity ($\frac{labelArea}{labelNumber}$), count intensity ($\frac{1}{labelIntensity}$), filtered label area (effective label area), and so on.
- Finally, we considered introducing multiple factors in 1 system. As we did not know the relationship among them, in our thesis we only considered combining 2 effective factor in system, with simple relationship, which can be formulated as follows. In the formula, a and b were the weight of the factor, ranging from 0-1.

$$imageScore = a * FactorOne + b * FactorTwo \tag{3.6}$$

3.6.1 Method 1: Label Numbers

We scored the images simply by ranking the number of labels in each of them. The advantages of this method could be summarised as follows:

- An increased number of labels could bring more samples of diseases to be analysed in the network, compared to images with less labels, this could saved image numbers to be trained and achieve the purposed of few-shot.
- The image of many labels means compared to peer images, this image would be more representative than others, and more variance of symptoms of certain disease could be detected.

With the analyse of the labels in all the images of the dataset, as shown in Fig. 3.12, we could see that the distribution of the labels in images varied from 1 shots to about 50 shots, and most of the images in both classes distributed around 0-15 shots. Although in deep learning due to the large number of training images, this difference would be eliminated, in few-shot we would prefer minor images which had many labels like 50, as this would provide over 10 times of the samples for training compared to ones with 1-5 shots.

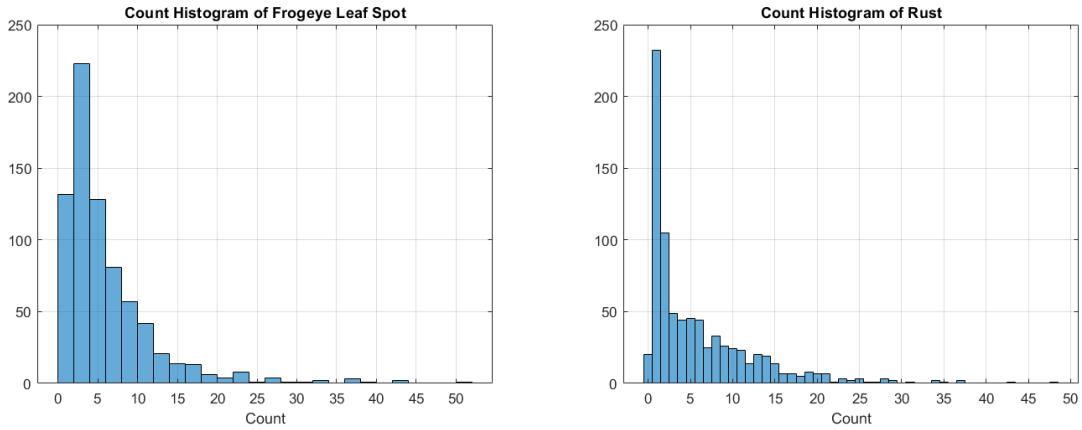

FIGURE 3.12: Label analysis of each class

3.6.2 Method 2: Label Numbers + Effective Label Areas

The second method was developed from the factor of label areas, as a bigger areas was supposed to provide more features to analyse than others. Based on our design and testing, we provided effective label areas, and combined this method with label numbers with corresponding weights.

3.6.2.1 Effective Label Areas

This method was derived from the base method total label areas. Based on our testing, the original method of total label areas failed to provide a decent selection of the images because of the mix of extreme large labels. These labels not only reduce the available number of labels in the images when ranking the total areas but also providing less features as expected, as our design focused on small labels.

Accordingly, as shown in Fig. 3.13, we selected the effective areas as our factor of interests. The threshold value of these areas in each class was 0.15 for frogeye leaf spot and 0.3 for rust. In our method we only considered the areas of these labels within the threshold and ignored the bigger labels. Based on the testing the result was good in 5-shot, but compared to the method 1, the performance of this factor was relatively lower. So we considered this as acceptable factor for rating system, and tried to combine both label number and effective label area to provide a better performance.

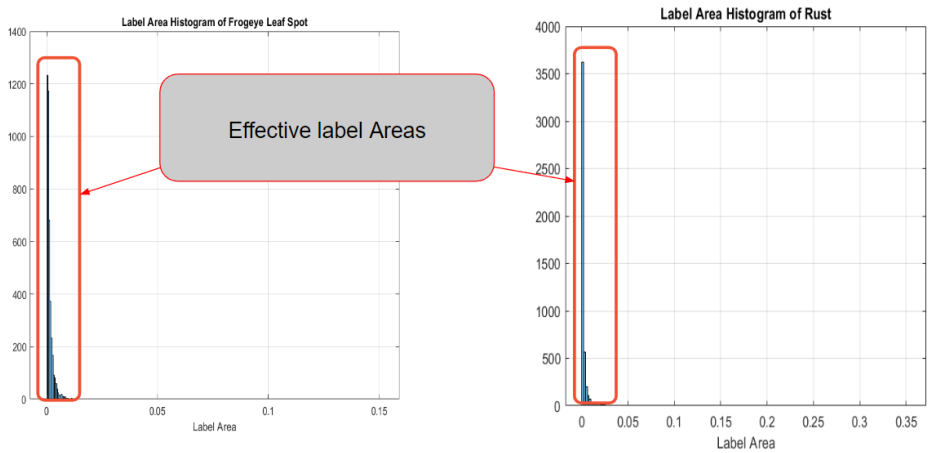

FIGURE 3.13: Effective label Area

3.6.2.2 Label Numbers + Effective Label Areas

In this section, we considered combining these factors, the method of that was by changing the weights of a and b. We considered using a and b with the value selection of 1, 0.5, and 0.25. Based on our testing of 5 shots, we found the best performance with following linear relationship.

$$imageScore = 1 * labelNum + 0.25 * EffectiveLabelArea \tag{3.7}$$

3.7 Peer Methods

In this section we introduced some of the the most recent literature of few-shot, to compare the performance with our network.

3.7.0.1 FsDet

FsDet was one of the most classic examples of transfer learning in few-shot. It applied a 2 stage process of modelling. The network of this model was Faster R-CNN, with custom network for feature extraction in backbone (Wang et al. 2020). As shown in Fig. 3.14, in stage 1, the base model would be constructed by training dataset Pascal Pascal VOC 2007+2012 (Everingham et al. 2010). And for stage 2, both base class and novel class with few shots were required to fine-tune the last layer of the model with cosine similarity as the classifier of the boxes, and other layers were kept frozen.

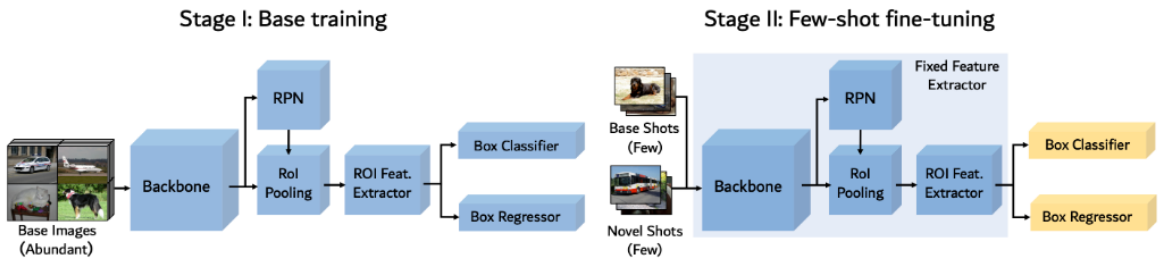

FIGURE 3.14: Network for FsDet (Wang et al. 2020)

As shown in Fig. 3.15, this method achieves 2-20 improvements of mAP@0.5 over the current few-shot object detection methods.

Method / Shot	Backbone	Novel Set 1					Novel Set 2					Novel Set 3				
		1	2	3	5	10	1	2	3	5	10	1	2	3	5	10
YOLO-joint (Kang et al., 2019)	YOLOv2	0.0	0.0	1.8	1.8	1.8	0.0	0.1	0.0	1.8	0.0	1.8	1.8	1.8	3.6	3.9
YOLO-ft (Kang et al., 2019)		3.2	6.5	6.4	7.5	12.3	8.2	3.8	3.5	3.5	7.8	8.1	7.4	7.6	9.5	10.5
YOLO-ft-full (Kang et al., 2019)		6.6	10.7	12.5	24.8	38.6	12.5	4.2	11.6	16.1	33.9	13.0	15.9	15.0	32.2	38.4
FSRW (Kang et al., 2019)		14.8	15.5	26.7	33.9	47.2	15.7	15.3	22.7	30.1	40.5	21.3	25.6	28.4	42.8	45.9
MetaDet (Wang et al., 2019b)		17.1	19.1	28.9	35.0	48.8	18.2	20.6	25.9	30.6	41.5	20.1	22.3	27.9	41.9	42.9
FRCN+joint (Wang et al., 2019b)	FRCN w/VGG16	0.3	0.0	1.2	0.9	1.7	0.0	0.0	1.1	1.9	1.7	0.2	0.5	1.2	1.9	2.8
FRCN+joint-ft (Wang et al., 2019b)		9.1	10.9	13.7	25.0	39.5	10.9	13.2	17.6	19.5	36.5	15.0	15.1	18.3	33.1	35.9
MetaDet (Wang et al., 2019b)		18.9	20.6	30.2	36.8	49.6	21.8	23.1	27.8	31.7	43.0	20.6	23.9	29.4	43.9	44.1
FRCN+joint (Yan et al., 2019)	FRCN w/R-101	2.7	3.1	4.3	11.8	29.0	1.9	2.6	8.1	9.9	12.6	5.2	7.5	6.4	6.4	6.4
FRCN+ft (Yan et al., 2019)		11.9	16.4	29.0	36.9	36.9	5.9	8.5	23.4	29.1	28.8	5.0	9.6	18.1	30.8	43.4
FRCN+ft-full (Yan et al., 2019)		13.8	19.6	32.8	41.5	45.6	7.9	15.3	26.2	31.6	39.1	9.8	11.3	19.1	35.0	45.1
Meta R-CNN (Yan et al., 2019)		19.9	25.5	35.0	45.7	51.5	10.4	19.4	29.6	34.8	**45.4**	14.3	18.2	27.5	41.2	48.1
FRCN+ft-full (Our Impl.)	FRCN w/R-101	15.2	20.3	29.0	40.1	45.5	13.4	20.6	28.6	32.4	38.8	19.6	20.8	28.7	42.2	42.1
TFA w/ fc (Ours)		36.8	29.1	43.6	**55.7**	**57.0**	18.2	**29.0**	33.4	**35.5**	39.0	27.7	33.6	42.5	48.7	**50.2**
TFA w/ cos (Ours)		**39.8**	**36.1**	**44.7**	**55.7**	56.0	**23.5**	26.9	**34.1**	35.1	39.1	**30.8**	**34.8**	**42.8**	**49.5**	49.8

FIGURE 3.15: mAP@0.5 Results of FsDet (Wang et al. 2020)

3.8 Experiment Design

In the experiment, we considered the mean average precision (mAP) as our main parameter of the performance, as it represents the performance of the bounding boxes. Other parameters including precision, recall, and f1 score were considered as well, but as the limitation of the peer literature, some of them did not consider these factors, so we only used mAP for comparision in Stage 5.

We summarised the experiments into following stages.

3.8.1 Stage 1: Class Selection Test with Deep Learning

This stage applied YOLOv5 to train the dataset with the deep learning method. The accepted classes based on section 3.1 would be decided to be applied in following stages, as a high performance could help us to understand the loss of performance in few-shot method.

3.8.2 Stage 2: Experiment of fine-tuning with base class few-shot

This stage tested the performance of the Yolov5 few-shot in fine-tuning stage. We compared the difference between fine-tuning with few-shot of novel classes and the one with novel class & base class. We chose COCO and Pascal VOC as the base class in our experiment. We conducted the experiment on distributed 30 shots.

3.8.3 Stage 3: Few-shot Testing with proposed method

This stage tested and analysed the results of our proposed network compared to Yolov5 adopted in few-shot. In this stage we firstly tested the networks on images on the dataset which was selected in a distributed position. Based on what we got during the results, we found some interesting results at 5-10 shots and then performed the tests of these shots with a random selection.

3.8.4 Stage 4: Experiments of Rating System on image labels

In this stage, we tested our rating systems with shots of 5, 10, 20, 30, and 50. We compared the systems with the baseline results of distributed image for Yolov5 few-shot used in last stage.

3.8.5 Stage 5: Experiment of few-shot from Peer

In this stage, we performed testing on the approaches from peer. Due to the limitation of GPU, we only performed the test on 5 shots. All the methods of few-shot were compared to their deep learning results. Due to the fact that some of the methods only used mAP as the results, here we only compared the mAP as our result of performance.

CHAPTER 4

Results

4.1 Stage 1: Class Selection Test with Deep Learning

The result of Stage 1 was shown in Table. 4.1. Only frogeye leaf spot and rust achieved a good performance of mAP over 85%, which could show a distinct view of loss in few-shot method. So we took theses classes as our class for few-shot object detection.

We also discussed reasons of bad performance in powdery mildew and scab:

- Similar features as leaf, which enhanced the difficulties of labeling and detection.
- Lack of samples: Since only 4000 out of 18K images were selected from the dataset, the training images for both dataset might not be sufficient enough to get a high accuracy.

TABLE 4.1: Performance of object detection with different models

Method	Class	Labels	Precision (%)	Recall (%)	F1 (%)	mAP@0.5 (%)
YOLOv5 Deep Learning	all	8660	75.3	67.8	71.35	71.7
	frogeye leaf spot	2221	86.2	79.3	82.6	86.0
	powdery mildew	1045	66.7	57.7	61.9	60.2
	rust	1948	84.4	84.4	84.4	87.5
	scab	3446	64.0	49.6	55.9	52.9

4.2 Stage 2: Experiment of fine-tuning with base class few-shot

As shown in Table. 4.2, only fine-tuning the novel classes (leaf diseases) turned out to have a better performance than the fine-tuning with both novel classes and base classes. This might because compared to the novel classes that focused on specific features of leaf diseases, the base classes paid more attention on general objects like human and cars. So using base classes in COCO and Pascal VOC could train a good backbone for lower features, but it would not contribute to extract high level in head and neck when fine-tuning with novel classes, as these two types of classes have obvious difference and features. In this case, we chose only fine-tuning novel classes in our approach.

TABLE 4.2: Performance of fine-tuning with base class few-shot in 30 shots of Yolov5 few-shot

Method	Class	Labels	Precision (%)	Recall (%)	F1 (%)	mAP@0.5 (%)
Only Novel	all	4169	74.5	66.2	70.11	71.6
	frogeye leaf spot	2221	76.8	65	70.41	72.2
	rust	1948	72.2	67.3	69.66	71
Novel + COCO Base	all	4169	75.6	65.4	70.13	68.5
	frogeye leaf spot	2221	81.5	61.3	69.97	70.6
	rust	1948	69.7	69.5	69.60	66.3
Novel + Pascal VOC Base	all	4169	74.6	56.7	64.43	67.6
	frogeye leaf spot	2221	77.7	56.7	65.56	69.7
	rust	1948	71.5	56.7	63.25	65.5

4.3 Stage 3: Few-shot Testing with proposed method

In this stage, we firstly conducted the distributed testing from 5-100 shots. Based on the result, we figured out an interesting problem in 5 and 10 shots then conducted test on 5 and 10 shots with random selection of images, which explore a deeper problem behind the selection of the images in few-shot.

4.3.1 Distributed shot Test

As shown in Fig. 4.1 and Table. 4.3, we found that compared to the Yolov5 in few-shot, our proposed network achieved the most obvious increase of mAP results in fewer shots, especially in 5 shots, which increased 7.6%. In terms of other results in overall, the proposed network also surpassed the Yolov5 few-shot obviously in 5-shot, which increased 6.1% in precision, 4.6% in recall, and 5.28% in F1 score. As for each class, for example, the result of frogeye leaf spot in 5 shots increased 4.5% in mAP, 7.4% in precision, and 3.06% in F1 score. And for rust, the proposed method increased 10.6% in mAP, 8.2% in precision, 7% in recall, and 7.56% in F1 score.

Similarly, at 30-shot in Fig. 4.1, we could observe the second most obvious improvement compared to the others. At 30-shot, our method increased 2.9% in mAP, 3.5% in precision, 1.4% in recall, and 62.21% in F1 score. And for separate classes it increased 1.5% in mAP, 2.5% in recall, and 1.09% in F1 score in frogeye leaf spot; and increased 6.3% in mAP, 7.9% in precision, 0.4% in recall, and 3.72% in F1 score for rust. On the other hand, the least improvement of the results happened at 10-shot and 100-shot, where only around 2% improvement in mAP can be observed in overall. This seemed to be reasonable, as with the increase of the images selected, the network would have more samples to be trained compared to the very few shots, and the carrying effect of our proposed method would be less obvious than the Yolov5 few-shot.

However, at 100 shots, the overall mAP achieved at 77.6, which was close to our target result (80%). We could verify that it would be possible for our network to provide a few-shot method which could achieve a close accurate result as the deep learning method.

Furthermore, from Fig. 4.1, an interesting drop of mAP was observed from 5-shot to 10-shot in our proposed method. In principle this would not happen, as with an increase of the images to be trained, the more samples would be analysed, and the result should be better, but in our case this did not work. As this problem did not happen in the Yolov5 few-shot, we noticed that there might be some issues in selection of the images that might affect the performance dramatically in our proposed method.

To validate our idea, we performed few-shot test on 5 and 10 shots, with a range of random selection of the images to be trained in the model.

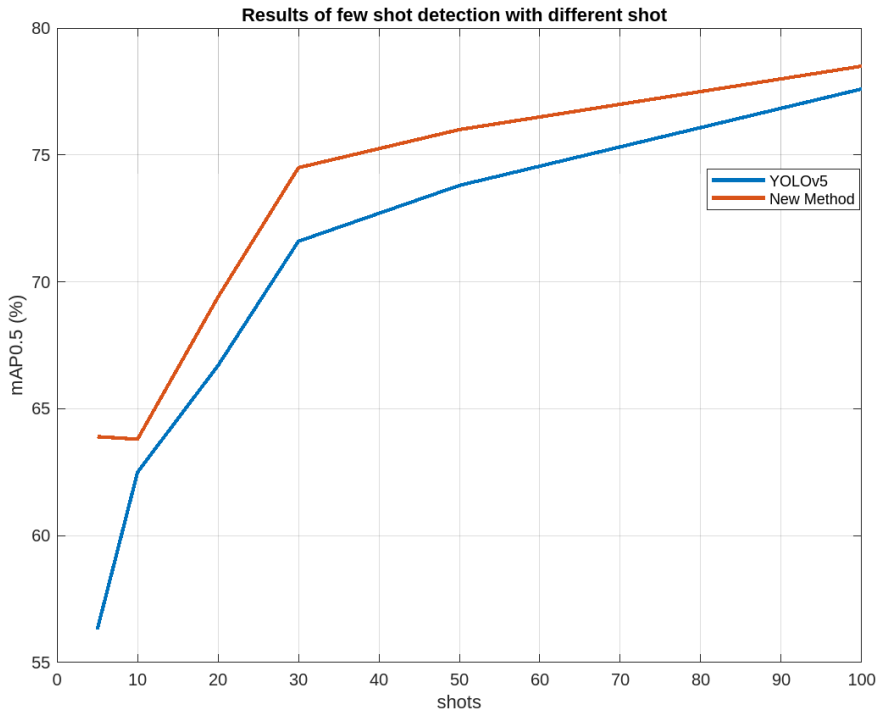

FIGURE 4.1: Performance on distributed images

Besides, for each class, we noticed the trend of the result in class frogeye leaf spot and rust. From 5 shots to 20 shots in both models, the results in frogeye leaf spot seemed better than rust. For example, at 5 shots, the result of frogeye leaf spot in Yolov5 few-shot model

surpassed a lot than rust, with a difference of 8.2% in mAP, 7.6% in precision, 6.6% in recall, and 7.07% in F1 score.

However, with an increase of images, this gap was decreased, and at 20 shots, the results between these classes are close, and in our proposed method at 20 shots, the result of rust took over frogeye leaf spot. And surprisingly, over 50 shots, the results of rust was much better than frogeye leaf spot. For instance, with our proposed method in 100 shots, compared to frogeye leaf spot, the result of rust increased 3.6% in mAP, 4.7% in precision, 0.9% in recall, and 2.59% in F1 score.

With these interesting results between frogeye leaf spot and rust, we could see that although the result of rust was lower than frogeye leaf spot with very few shot, the increase rate of performance in rust was much better than the frogeye leaf spot. This meant that the ability increase of feature extraction in rust seemed better than the one in frogeye leaf spot, although frogeye leaf spot seemed better in very few-shot object detection. This gave us some ideas of how each class performed during few-shot tests.

TABLE 4.3: Performance of object detection with distributed images

Method	Class	Labels	Precision (%)	Recall (%)	F1 (%)	mAP@0.5 (%)
Yolov5 Few-shot - 5 shots	all	4169	61	55.2	57.96	56.3
	frogeye leaf spot	2221	64.8	58.5	61.49	60.4
	rust	1948	57.2	51.9	54.42	52.2
Proposed Method - 5 shots	all	4169	67.1	59.8	63.24	63.9
	frogeye leaf spot	2221	68.8	60.8	64.55	64.9
	rust	1948	65.4	58.9	61.98	62.8
Yolov5 Few-shot - 10 shots	all	4169	66	58.1	61.80	62.5
	frogeye leaf spot	2221	72.2	55.8	62.95	64
	rust	1948	67	59.8	60.05	61.1
Proposed Method - 10 shots	all	4169	66.7	59.4	62.84	63.8
	frogeye leaf spot	2221	75.2	52.9	62.11	64.6
	rust	1948	58.1	65.9	61.75	62.9
Yolov5 Few-shot - 20 shots	all	4169	69.9	61.6	65.49	66.7
	frogeye leaf spot	2221	69.7	62.2	65.74	66.5
	rust	1948	70	61.1	65.25	66.8
Proposed Method - 20 shots	all	4169	72.3	64	67.90	69.4
	frogeye leaf spot	2221	73.9	61.4	67.07	68.6
	rust	1948	70.6	66.6	68.54	70.1
Yolov5 Few-shot - 30 shots	all	4169	74.5	66.2	70.11	71.6
	frogeye leaf spot	2221	76.8	65	70.41	72.2
	rust	1948	72.2	67.3	69.66	71
Proposed Method - 30 shots	all	4169	78	67.6	72.43	74.5
	frogeye leaf spot	2221	76	67.5	71.50	73.7
	rust	1948	80.1	67.7	73.38	77.3
Yolov5 Few-shot - 50 shots	all	4169	76.2	68	71.87	73.8
	frogeye leaf spot	2221	77.1	68	72.26	74
	rust	1948	75.4	67.9	71.45	73.6
Proposed Method - 50 shots	all	4169	77.4	69.8	73.40	76
	frogeye leaf spot	2221	73.8	71.7	72.73	75.2
	rust	1948	81.1	68	73.97	76.9
Yolov5 Few-shot - 100 shots	all	4169	79.8	70.6	74.92	77.6
	frogeye leaf spot	2221	81.7	67.6	73.98	76.7
	rust	1948	77.8	73.6	75.64	78.4
Proposed Method - 100 shots	all	4169	79.8	71.1	75.20	78.5
	frogeye leaf spot	2221	77.4	70.7	73.90	76.7
	rust	1948	82.1	71.6	76.49	80.3

4.3.2 Random shot Test

Based on the finding in last section, we performed a new experiment of few shots on randomly selected images. Due to the limitation of the running time in Colab Pro, maximum 10 hr would be allowed to run the model with GPU, and each of the model would require 2-3hr to be fully trained and constructed. With consideration, in our project, we decided to make 20 random tests on 5 and 10 shots. After the testing, we summarised the 20 random shot selection into boxplot shown in Fig. 4.2 and averaged the results into the Table. 4.4.

Based on the results shown in Table. 4.4, we could see that the overall results in 10-shot of our proposed method was better than the one in 5-shot, which increased 5.9% in mAP, 4.4% in precision, 2.6% in recall, and 4% in F1 score. This proved that 10-shot performed better in 5-shot with our method, which met the principle. Meanwhile, our proposed method surpassed the Yolov5 few shot. In 5-shot, our method increased 4.1% in mAP, 5.4% in precision, 2.3% in recall, and 3.7% in F1 score in overall.

And for separate classes, in 5 shots, the proposed method increased 4.2% in mAP, 3.1% in precision, 2% in recall, 2.6% in F1 score, for frogeye leaf spot; and for rust, it increased 6.7% in mAP, 7.7% in precision, 2.6% in recall, and 4.8% in F1 score. However, in 10 shots, the increment dropped for both classes. For frogeye leaf spot, it increased 2.7% in mAP, 5.2% in precision, and 2% in F1 score; while for rust, it increased 5.5% in mAP, 3.5% in precision, 5.5% in precision, and 4.7% in F1 score.

As for each class, what we explored in last section got proved in this table as well. Although at the very few-shot the result in frogeye leaf spot was better than rust, the increase of the result in rust seemed better. For example, the difference between frogeye leaf spot and rust in Yolov5 few-shot was 6.1% in mAP but in 10 shots the difference decreased to 5.4%.

However, compared to Yolov5 few-shot, our proposed method seemed better in bridging the gap of the results between these classes. For instance, in 5-shot with our method, the difference of classes between these classes was 3.6% in mAP, 3% in precision, and 3% in F1

TABLE 4.4: Average Performance of 20 random shots 5-shot and 10-shot

Method	Class	Labels	Precision (%)	Recall (%)	F1 (%)	mAP@0.5 (%)
YOLOv5 Few-shot - 5 Random	all	4169	57.0	55.5	56.3	55.4
	frogeye leaf spot	2221	57.8	59.8	58.8	57.5
	rust	1948	56.2	51.2	53.6	51.4
Proposed Method - 5 Random	all	4169	62.4	57.8	60.0	59.5
	frogeye leaf spot	2221	60.9	61.8	61.4	61.7
	rust	1948	63.9	53.8	58.4	58.1
YOLOv5 Few-shot - 10 Random	all	4169	63.6	60.4	60.8	61.4
	frogeye leaf spot	2221	61.8	64.0	62.9	63.4
	rust	1948	65.5	52.5	58.3	58.8
Proposed Method - 10 Random	all	4169	68.0	60.4	64.0	65.4
	frogeye leaf spot	2221	67.0	62.9	64.9	66.1
	rust	1948	69.0	58.0	63.0	64.3

score; while for Yolov5 the difference was 6.1% in mAP, 8.6% in precision, and 5.2% in F1 score.

On the other hand, as we could see in the Fig. 4.2, the mAP gap among 5-shot was obvious. As we could see in Yolov5 5-shot in the figure, the gap was around 10%, which could not be ignored, but in 5-shot of Yolov5, the gap was closed to around 5%. Similarly, with our proposed method, the gap in 5-shot was around 5%, while in 10-shot it deduced to around 3%, with some outliers. With this trend we could see the uncertainty of the results from the selection of images, and this explained well why in the last section the result of 5-shot wit our method was slightly better than the on in 10-shot. With the exploration in random selection of shots we then derived the rating system and related experiments.

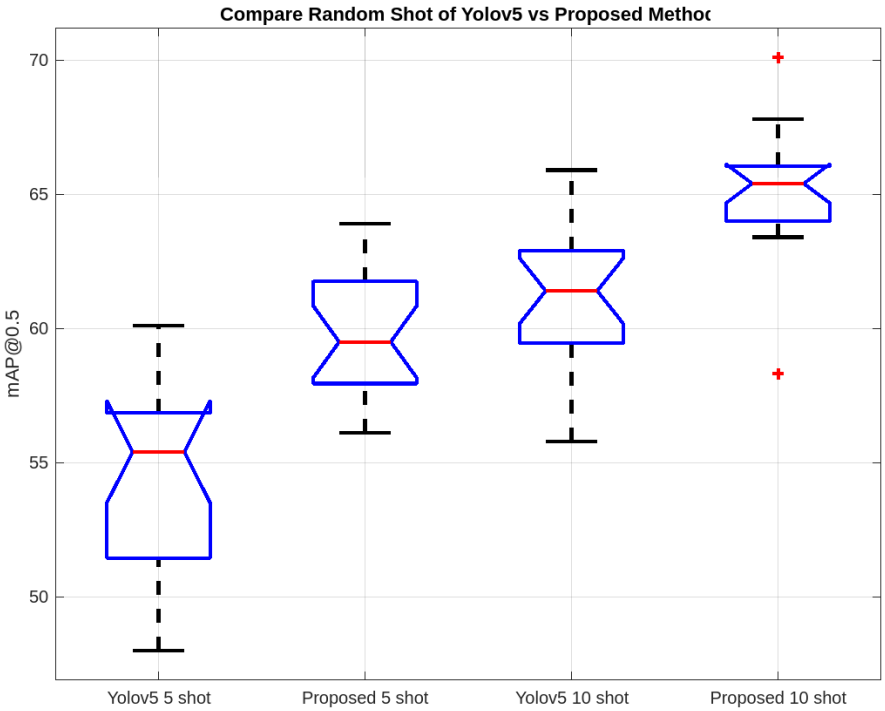

FIGURE 4.2: Performance on random shots for 5 and 10

4.4 Stage 4: Experiments of Rating System on image labels

As shown in Fig. 4.3, compared to Yolov5 with few-shot method, the proposed network with rating system increased significantly. For example, from 5-shot to 30-shot, the addition of the rating system on our method increased approximately 10% compared to the baseline method of Yolov5 few shot. Although this gap reduces to around 5% after 30 shots, the improvement still could be consider as obvious. This gap reduction might due to an increase of disease samples. Also, compared to the random selection test in 5 shots and 10 shots in last section, we could see that both of the rating system approached the best results of selection, which proved that the introduction of the rating system could optimise the best possible results for our proposed network, and the results seemed much better compared to the baseline method of Yolov5 in few-shot.

On the other hand, by comparing two methods of rating system, we could observe following characteristics. Firstly, based on both Fig. 4.3 and Table. 4.5, both methods achieved the same result at 5 shot, which was due to the same selection of images. This meant that at 5 shots the result in 5 shots would be the best possible solution for the network. Then, we could see

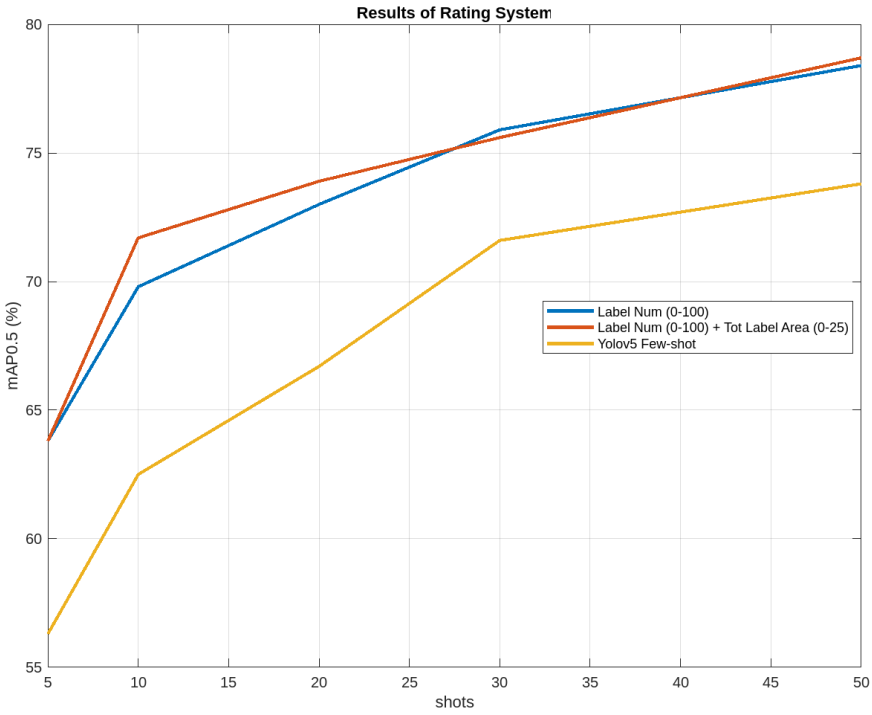

FIGURE 4.3: Performance of rating system

that from 5-25 shots in Fig. 4.3, the second method (Label number + Total effective label area) showed a relatively better detection performance than the one with method 1 (Label number). The most obvious difference happened at 10 shots, where in overall the method 2 increased 1.9% in mAP, 2.4% in precision, 1.6% in recall, and 1.96% in F1 score. As for separate classes, in frogeye leaf spot, the method 2 increased 1% in mAP, 4.6% in recall, and 1.93% in F1 score; while in rust, it increased 2.8% in mAP, 6.8% in precision, and 2.5% in F1 score. As for other classes, the differences between two method were small. And at 30-50 shots, the mAP results of two method seemed much closer.

Another obvious improvement happened at 50 shots. With the rating systems, the mAP results reached 78.4% and 78.7%, which surpassed the baseline method at 100 shots and met our purpose of target 80% result of mAP. This result dramatically decreased our requirement of images for few-shot and contributed to provide a close accurate result with much fewer images.

As shown in Fig. 4.4, after applying the weights of method 2 in 50-shot, the proposed network managed to correctly label most of the diseases in the images with a relatively high

TABLE 4.5: Performance of object detection with different rating methods

Method	Class	Labels	Precision (%)	Recall (%)	F1 (%)	mAP@0.5 (%)
Method 1 - 5 shots	all	4169	67.7	60.3	63.79	63.8
	frogeye leaf spot	2221	72	62.2	66.74	67.6
	rust	1948	63.5	58.3	60.79	60.1
Method 2 - 5 shots	all	4169	67.7	60.3	63.79	63.8
	frogeye leaf spot	2221	72	62.2	66.74	67.6
	rust	1948	63.5	58.3	60.79	60.1
Method 1 - 10 shots	all	4169	71.8	64.4	67.90	69.8
	frogeye leaf spot	2221	76.5	61	67.88	71.1
	rust	1948	67	67.8	67.40	68.5
Method 2 - 10 shots	all	4169	74.2	66	69.86	71.7
	frogeye leaf spot	2221	74.6	65.6	69.81	72.1
	rust	1948	73.8	66.4	69.90	71.3
Method 1 - 20 shots	all	4169	76	67.5	71.50	73
	frogeye leaf spot	2221	77.4	68.1	72.45	74.3
	rust	1948	74.6	66.9	70.54	71.6
Method 2 - 20 shots	all	4169	76.2	67.8	71.76	73.9
	frogeye leaf spot	2221	75.4	70.2	72.71	75.1
	rust	1948	77.1	65.4	70.70	72.8
Method 1 - 30 shots	all	4169	77.8	71.2	74.35	75.9
	frogeye leaf spot	2221	76.8	72.4	74.54	77
	rust	1948	78.7	70	74.10	74.9
Method 2 - 30 shots	all	4169	76.1	69.9	72.87	75.6
	frogeye leaf spot	2221	72.7	71.4	72.04	74.4
	rust	1948	79.4	68.5	73.55	76.7
Method 1 - 50 shots	all	4169	80.5	72.2	76.12	78.4
	frogeye leaf spot	2221	80.9	70.8	75.51	77.6
	rust	1948	80.2	73.6	76.76	79.3
Method 2 - 50 shots	all	4169	79.7	71.1	75.15	78.7
	frogeye leaf spot	2221	80.2	68.3	73.77	77.6
	rust	1948	79.2	74	76.51	79.9

confidence. This could prove that with 50-shot the network could be utilised to perform a relatively accurate detection on apple leaf diseases, which suited our purpose: to use fewer images to achieve a satisfactory detection of the diseases.

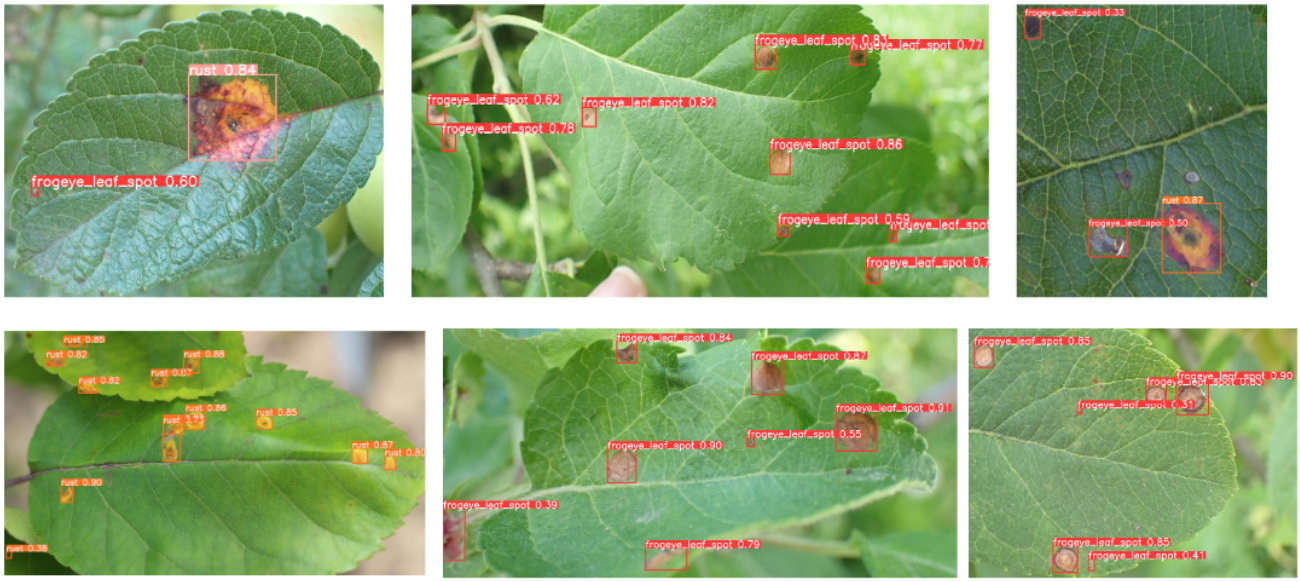

FIGURE 4.4: Examples of detection

4.5 Stage 5: Experiment of few-shot from Peer

As shown in Table. 4.6, we could see that compared to both deep learning method and few-shot method, our method surpassed the peer method. For deep learning method Yolov5 performed much better than the Faster R-CNN with Resnet101 for the peer method, with an increase of 27.05% in overall mAP, 9.7% in frogeye leaf spot, and 44.5% in rust; while in 5-shot, our method increased 36.9% in overall, 36.6% in frogeye leaf spot, and 37.2% in rust.

There were two reasons of low performance of the peer results. Firstly, it applied a old-fashion network compared to Yolov5, which seemed less effective in detecting leaf diseases. Secondly, as the peer method only fine-tuned the last layer of the network, which meant that the performance of the network highly relied on the base dataset used. As the peer applied Pascal VOC as the dataset for base classes, which had much difference of the foliar disease dataset we analysed. That explained the low performance in this case.

Inspired by the situation above, we also made the experiment of only fine-tuning the last layer of Yolov5 network with both base (COCO) and novel classes (Our diseases classes). The

TABLE 4.6: Performance of FsDet compared to Ours

Method	Class	mAP@0.5 (%)
Faster R-CNN Deep Learning	all	59.7
	frogeye leaf spot	76.3
	rust	43.0
FsDet 5-shot	all	19.4
	frogeye leaf spot	23.8
	rust	15.0
Yolov5 Deep Learning	all	86.75
	frogeye leaf spot	86.0
	rust	87.5
Our Method - 5-shot	all	56.3
	frogeye leaf spot	60.4
	rust	52.2

reason we did not fine-tuned the proposed network was because the Yolov5 has the pre-trained network, so it would provide more pre-trained parameters for the few-shot detection.

As shown in Table. 4.7, we could see that by fine-tuning the last layer of the network, Yolov5 showed a lower detection performance than the peer method. This proved that the feature difference between the base and novel class would affect the prediction dramatically.

TABLE 4.7: Performance of Yolov5 following FsDet to only fine-tune last layer

Method	Class	mAP@0.5 (%)
FsDet 5-shot	all	19.4
	frogeye leaf spot	23.8
	rust	15.0
Our Method - 5-shot	all	56.3
	frogeye leaf spot	60.4
	rust	52.2
Yolov5 fine-tune last layer - 5-shot	all	10.4
	frogeye leaf spot	13.1
	rust	7.76

CHAPTER 5

Conclusion

This session provides an overall discussion of our design and experiments, and provides ideas of future work.

5.1 Thesis Contributions and Discussions

In this thesis, we provided a novel few-shot object detection method for apple leaf disease detection, built on Yolov5. We analysed our investment with following components.

5.1.1 Network

In this thesis, we developed a Yolov5 style network for few-shot object detection. Based on the characteristics of the apple leaf diseases, we applied CARAFE as our upsampling process to provide adapted kernels for feature extraction with faster operation speed, and made modification in head to reduce over convolution for samll size of diseases which took the majority in the label contribution.

However, there were some limitations for our network. As the modification of head focused on reducing over-convolution for the small target, this process however, seemed not suitable for the very large labels. Although this type of the labels were rare and we could reduce this issue by introducing effective label areas in rating system, a better network which is friendly for both small and very large disease objects would be desired in the future.

5.1.2 Dataset

We applied 2 types of apple leaf diseases in this thesis for our few-shot object detection, which achieved an excellent performance in deep learning detection. Although this gave us different aspects of the comparison of the performance, the classes in this thesis seemed limited and more diseases could be included to verify our achievements in tests. These classes would not only limited in apple leaf diseases, but also the leaf diseases in other crops such as grape leaves.

Also, for the other two disease classes that got less accurate results, we could analyse the performance in few-shot and see the loss of the accuracy with our methods.

5.1.3 Rating System

For rating system, we explored the possible factors based on the labels of images and rate the top images for best performance. In this thesis we find two most important factors and constructed linear relationship to optimise the result. In the future, we could find more factors related to detection and find more factors on images instead of labels. And more work should be done to explore the best relationship among these factors.

5.1.4 Experiments for few-shot

We have many limitations in our experiments. Firstly, due to the GPU and time limitation of this project, we only made the random shot tests on 5 and 10 shots, with 20 random selections. In the future, we could do tests for images over 10 shots, and extend the random selections into 40, to verify the results we got.

Also, in the future we should explore the results on fewer shots like 1 shot or even zero shot, to see the detection ability for our approach.

Furthermore, we could apply our model in robotic system to see the performance of disease detection and spray effects.

Besides, more peer experiments would be required for comparison.

5.2 Future outlook

Based on the last section, we summarised the following work to be done in the future.

- Extend our approach on other foliar disease detection.
- Explore fewer shot and zero-shot detection for leaf disease detection.
- Apply our system on robotic spray.
- Explore rating system, find factors from images and build more robust model of factor relationship.
- Improve network for both very large and small disease labels

Bibliography

Amit, Yali, Pedro Felzenszwalb and Ross Girshick (Jan. 2020). 'Object Detection'. In: pp. 1–9. ISBN: 978-3-030-03243-2. DOI: 10.1007/978-3-030-03243-2_660-1.

Anderson, Robert and Janice Uchida (Jan. 2008). 'Disease Index of the Rust Puccinia psidii on Rose Apple in Hawai'i'. In.

Antonelli, Simone et al. (Feb. 2022). 'Few-Shot Object Detection: A Survey'. In: *ACM Computing Surveys*. DOI: 10.1145/3519022.

Bansal, Prakhar, Rahul Kumar and Somesh Kumar (June 2021). 'Disease Detection in Apple Leaves Using Deep Convolutional Neural Network'. In: *Agriculture* 11, p. 617. DOI: 10.3390/agriculture11070617.

Bin, Liu et al. (Dec. 2017). 'Identification of Apple Leaf Diseases Based on Deep Convolutional Neural Networks'. In: *Symmetry* 10, p. 11. DOI: 10.3390/sym10010011.

Bisong, Ekaba (2019). 'Google Colaboratory'. In: *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*. Berkeley, CA: Apress, pp. 59–64. ISBN: 978-1-4842-4470-8. DOI: 10.1007/978-1-4842-4470-8_7. URL: https://doi.org/10.1007/978-1-4842-4470-8_7.

Chane, Tibebu and Nuh Boyraz (May 2017). 'Critical Review on Apple Scab (Venturia inaequalis) Biology, Epidemiology, Economic Importance, Management and Defense Mechanisms to the Causal Agent'. In: 5. DOI: 10.4172/2329-955X.1000166.

Cochran, A. et al. (Nov. 2014). 'Impact of fungicide modes of action on Frogeye Leaf Spot'. In: pp. 28–28.

Everingham, M. et al. (Jan. 2015). 'The Pascal Visual Object Classes Challenge: A Retrospective'. In: *International Journal of Computer Vision* 111.1, pp. 98–136.

Everingham, Mark et al. (June 2010). 'The Pascal Visual Object Classes (VOC) challenge'. In: *International Journal of Computer Vision* 88, pp. 303–338. DOI: 10.1007/s11263-009-0275-4.

Fan, Qi et al. (2020). 'Few-Shot Object Detection with Attention-RPN and Multi-Relation Detector'. In: *CVPR*.

Fan, Zhibo et al. (May 2021). *Generalized Few-Shot Object Detection without Forgetting*.

Girshick, Ross et al. (Nov. 2013). 'Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation'. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. DOI: 10.1109/CVPR.2014.81.

github (2020). *GitHub*. URL: https://github.com/.

Handalage, Upulie and Lakshini Kuganandamurthy (May 2021). *Real-Time Object Detection Using YOLO: A Review*. DOI: 10.13140/RG.2.2.24367.66723.

Hasan, Reem Ibrahim, Suhaila Yusuf and Laith Alzubaidi (Oct. 2020). 'Review of the State of the Art of Deep Learning for Plant Diseases: A Broad Analysis and Discussion'. In: *Plants* 9. DOI: 10.3390/plants9101302.

He, Kaiming et al. (2015). 'Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.9, pp. 1904–1916. DOI: 10.1109/TPAMI.2015.2389824.

Hernández, Adrián and José Amigó (Feb. 2021). 'Attention Mechanisms and Their Applications to Complex Systems'. In: *Entropy* 23, p. 283. DOI: 10.3390/e23030283.

Imre, Holb (July 2013a). 'Apple powdery mildew caused by Podosphaera leucotricha: some aspects of biology'. In: *International Journal of Horticultural Science* 19. DOI: 10.31421/IJHS/19/3-4./1096.

— (July 2013b). 'Apple powdery mildew caused by Podosphaera leucotricha: some aspects of biology'. In: *International Journal of Horticultural Science* 19. DOI: 10.31421/IJHS/19/3-4./1096.

Kang, Bingyi et al. (2019). 'Few-shot Object Detection via Feature Reweighting'. In: *ICCV*.

Li, Wei et al. (July 2020). 'Object detection based on an adaptive attention mechanism'. In: *Scientific Reports* 10. DOI: 10.1038/s41598-020-67529-x.

Lin, Tsung-Yi et al. (May 2014). 'Microsoft COCO: Common Objects in Context'. In.

Liu, Haiying et al. (Aug. 2022a). 'SF-YOLOv5: A Lightweight Small Object Detection Algorithm Based on Improved Feature Fusion Mode'. In: *Sensors* 22, p. 5817. DOI: 10.3390/s22155817.

Liu, Jiangbo and Zhenyong Fu (2021). 'Curriculum Meta Learning: Learning to Learn from Easy to Hard'. In: *Proceedings of the 2021 5th International Conference on Electronic Information Technology and Computer Engineering*. EITCE 2021. Xiamen, China: Association for Computing Machinery, pp. 1571–1576. ISBN: 9781450384322. DOI: 10.1145/3501409.3501686. URL: https://doi.org/10.1145/3501409.3501686.

Liu, Yukuan et al. (2022b). 'NRT-YOLO: Improved YOLOv5 Based on Nested Residual Transformer for Tiny Remote Sensing Object Detection'. In: *Sensors* 22.13. ISSN: 1424-8220. DOI: 10.3390/s22134953. URL: https://www.mdpi.com/1424-8220/22/13/4953.

Nicole Ward Gauthier, Paul Andrew Rideout (2016). 'Frogeye Leaf Spot & Black Rot of Apple'. In: *Plant Pathology Fact Sheet*.

Ren, Shaoqing et al. (June 2015). 'Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39. DOI: 10.1109/TPAMI.2016.2577031.

Russakovsky, Olga et al. (Sept. 2014). 'ImageNet Large Scale Visual Recognition Challenge'. In: *International Journal of Computer Vision* 115. DOI: 10.1007/s11263-015-0816-y.

Sandskär, Boel (Jan. 2003). 'Apple scab (Venturia inaequalis) and pests in organic orchards'. In: *Apple Scab (Venturia Inaequalis) and Pests in Organic Orchards*.

Sardoğan, Melike, Yunus Ozen and Adem Tuncer (2020). 'Detection of Apple Leaf Diseases using Faster R-CNN'. In.

Son, Chang-Hwan (2021). 'Leaf Spot Attention Networks Based on Spot Feature Encoding for Leaf Disease Identification and Detection'. In: *Applied Sciences* 11.17. ISSN: 2076-3417. DOI: 10.3390/app11177960. URL: https://www.mdpi.com/2076-3417/11/17/7960.

Thapa, Ranjita et al. (2020). 'The Plant Pathology Challenge 2020 data set to classify foliar disease of apples'. In: *Applications in Plant Sciences* 8.9, e11390. DOI: https://doi.org/10.1002/aps3.11390. eprint: https://bsapubs.onlinelibrary.wiley.com/doi/pdf/10.1002/aps3.11390. URL: https://bsapubs.onlinelibrary.wiley.com/doi/abs/10.1002/aps3.11390.

Thuan, DoCong (2021). 'Do Thuan EVOLUTION OF YOLO ALGORITHM AND YOLOV5: THE STATE-OF-THE-ART OBJECT DETECTION ALGORITHM EVOLUTION OF YOLO ALGORITHM AND YOLOV5: THE STATE-OF-THE-ART OBJECT DETECTION ALGORITHM'. In.

Tummala, Subhash (2021). 'Classification of Multi Diseases in Apple Plant Leaves'. In.

Tzutalin (2015). *LabelImg*. Free Software: MIT License. URL: https://github.com/tzutalin/labelImg.

Villacrés, Juan and Fernando Auat Cheein (June 2020). 'Detection and Characterization of Cherries: A Deep Learning Usability Case Study in Chile'. In: *Agronomy* 10. DOI: 10.3390/agronomy10060835.

Wang, Jiaqi et al. (May 2019). 'CARAFE: Content-Aware ReAssembly of FEatures'. In.

Wang, Xin et al. (Mar. 2020). *Frustratingly Simple Few-Shot Object Detection*.

Wu, Jiaxi, Songtao Liu and di Huang (July 2020). *Multi-Scale Positive Sample Refinement for Few-Shot Object Detection*.

Zhao, Zhong-Qiu et al. (Jan. 2019). 'Object Detection With Deep Learning: A Review'. In: *IEEE Transactions on Neural Networks and Learning Systems* PP, pp. 1–21. DOI: 10.1109/TNNLS.2018.2876865.