Building a General Purpose Fruit-Detector with Faster R-CNN

Kasen Stephensen
Stanford University
450 Jane Stanford Way
albionst@stanford.edu

Abstract

In this report, I present my approaches to creating a general-use fruit-classification system by training a multi-class classifier as well as an orange citrus detection model. Since yield management is a predominantly manual task across many farms and orchards, this problem would help farmers reduce tedious tasks for their workers and minimize wasted produce. This problem space has inspired many researchers to apply cutting edge deep learning techniques to identify, classify, and count fruit in trees. For my classifier, I trained and evaluated several out-of-the-box models like the VGG-16, ResNet-34, and MobileNet v2. For the object detection system, I then fine-tune a Faster R-CNN pre-trained on the COCO dataset to detect oranges in trees. Each task required its own dataset. One is a dataset of 131 different fruit and vegetable varieties while the other contains annotated pictures of orange trees. Ultimately, the ResNet-34 is the most accurate classifier and the fine-tuned model outperforms the base object detector. I attempt, unsuccessfully, to combine the two models, but still succeed in producing two different models that are useful in classifying and detecting fruit.

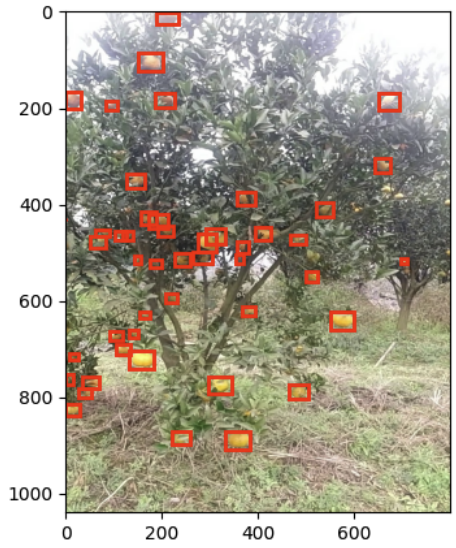

Figure 1. This is an annotated image from our dataset. Given an image, we want our model to predict these red boxes.

1. Introduction

Fruit yield management is a critical part of agriculture businesses. Yield monitoring systems provide data on the state of crops, which helps reduce waste, optimize efficiencies, and enhances sustainability. However, in many orchards and farms, yield management is still primarily manual, with teams of field workers walking the orchards and manually counting and sizing fruits. This is the perfect problem space for computer vision programs. Correctly solving this problem with computer-vision led analysis can automate tedious and time-consuming processes, compile real-time data on orchards and farms, eliminate waste, and detect infestation and fruit diseases.

In this project, I wish to focus primarily on fruit detection, which is the foundation of yield management. Later steps in the yield management process, which includes counting, monitoring and classifying features such as fruit size, growth over time, and disease, will not be addressed in this project for the sake of simplicity.

The input to my problem is a 800 x 1240 image of an orange tree. We will then use an image detection network (Faster R-CNN) to output boxes around detected oranges (see Figure 1 for an example). Three different approaches are introduced to detect oranges in trees. The first is applying a detection network pre-trained on the COCO dataset [11], the second is fine-tuning this detection network, and the third is implementing a classifier to augment the fine-tuned model. This third step required additional analysis to train, evaluate, and select a high quality classifier that can properly generalize to other datasets, domains, and fruits.

2. Related Work

Applications of deep learning in fruit detection tends to fall into a few overlapping categories. First, several papers solve inter-fruit classification by training classifiers that distinguish between different species of fruit or vegetables [15] [1]. Many researchers innovate by applying novel data sources to generic data infrastructures [14] [18]. Others focus solely on applying and modifying new technological methods and CNN architectures to fruit detection [8] [20]. Still others seek to solve niche problems in the fruit detection space, such as determining ripeness [12] [21]. Very few describe novel algorithms [23].

It appears that novel data sources are particularly helpful with improving classification. One that was particularly interesting involved combining RGB images with Infrared to better detect fruit in orchards [18]. Another state of the art paper constructed their own algorithm to track fruit across frames to avoid double counting [23].

Faster R-CNNs and other Convolutional Neural Networks are commonly utilized in this problem space and will be utilized in this project.

One difficulty in comparing these methods is that there is no state-of-the-art dataset for fruit detection and classification. In fact, most of the papers describe how they created their own datasets, which can be very useful for researchers like me who don't have the resources to do so, but it also makes it difficult to compare methods. What could work well with one dataset may not generalize well to others, as we'll see later in this report.

3. Methods

There were two phases in my project. The first involved training and comparing different out-of-the-box networks to build out a robust fruit classifier. The second involves implementing the three object detection networks based on the Faster R-CNN network.

3.1. Fruit Classification

3.1.1 Model Architecture

I experimented with three different image processing architectures. The three selected were the VGG-16, Resnet-34, and MobileNetV2. These were selected because of the varying model innovations, network depths, and model complexity. Each of these models were compared with the CNN in the original paper (Figure 2), which is considered the baseline.

3.1.2 Learning Algorithm

For this problem, I implemented mini-batch gradient descent. In this algorithm, small batches of our training data

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 16, 100, 100]	1,216
ReLU-2	[-1, 16, 100, 100]	0
MaxPool2d-3	[-1, 16, 50, 50]	0
Conv2d-4	[-1, 32, 50, 50]	12,832
ReLU-5	[-1, 32, 50, 50]	0
MaxPool2d-6	[-1, 32, 25, 25]	0
Conv2d-7	[-1, 64, 25, 25]	51,264
ReLU-8	[-1, 64, 25, 25]	0
MaxPool2d-9	[-1, 64, 12, 12]	0
Conv2d-10	[-1, 128, 12, 12]	204,928
ReLU-11	[-1, 128, 12, 12]	0
MaxPool2d-12	[-1, 128, 6, 6]	0
Flatten-13	[-1, 4608]	0
Linear-14	[-1, 1024]	4,719,616
ReLU-15	[-1, 1024]	0
Dropout-16	[-1, 1024]	0
Linear-17	[-1, 256]	262,400
ReLU-18	[-1, 256]	0
Dropout-19	[-1, 256]	0
Linear-20	[-1, 131]	33,667
Softmax-21	[-1, 131]	0

Total params: 5,285,923
Trainable params: 5,285,923

Figure 2. This is the baseline Convolutional Neural Network we will compare our classifiers against.

are computed and used to compute the gradient. This gradient is then used in gradient descent to update the model's parameters. The algorithm will repeat this for all batches in the training data. Upon the conclusion of the training data, the model will be evaluated on the validation data. The validation loss will then be used to notify the scheduler. This entire process is one epoch and will repeat for all epochs. The equation for the parameter update rule is as follows:

$$\theta^{(t+1)} = \theta^{(t)} - \frac{\eta}{m} \sum_{i=1}^{m} \nabla_\theta J(\theta; x^{(i)}, y^{(i)})$$

m denotes the size of the mini-batch, where x is the input and y is the target output. η is the learning rate.

3.1.3 Loss Function

For this image classification problem, I used a binary cross entropy equation to quantify loss during model training. The equation is as follows:

$$L = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

This loss function encourages the model to minimize the difference between its predicted distribution and the correct distribution of classes, thereby encouraging the model to choose the correct class and no other.

3.2. Object Detection

3.2.1 Model Architecture

I utilized a ResNet-50 backbone architecture on a Faster R-CNN that has been pretrained on the COCO dataset. One

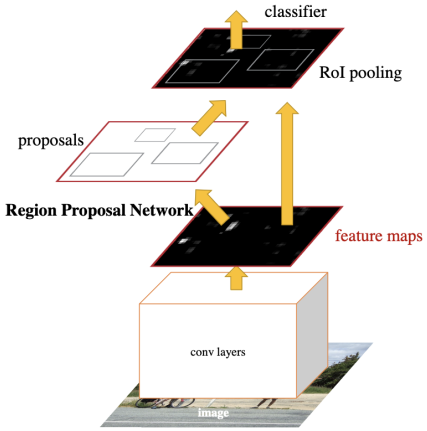

Figure 3. Faster R-CNN network as described in the original paper [17]

of the COCO classes is for an orange. We will use this pre-trained Faster R-CNN to make classifications on oranges in our orange tree dataset. This will be our baseline model and is available for use out-of-the-box. We fine-tune this baseline model on orange fruit trees. Finally, we will add in our fruit classifier to enhance the predictions that we make on regions.

For context, an R-CNN is a fully convolutional network that utilizes a selective search algorithm to efficiently select regions of an image that are likely to have an object in them. These regions are called Regions of Interest (RoI). These RoI are then converted into images, resized into a fixed size, and then passed through a classifier to get both classifications and "boxes" around the regions.

Faster-R-CNN improves this selective algorithm and image resizing step and by instead applying a "backbone" network onto the input image. These backbones can be any convolutional neural network, but excludes the fully-connected layers. The backbone network produces features that can then be cropped and resized into different regions and then passed through another convolutional neural network to get the object's category and box coordinates. See a diagram here 3 of the model's architecture.

There are other possible backbone models, but PyTorch has only two that are implemented in their pre-trained models. I selected ResNet because of my familiarity. If I had more time, I would've tested PyTorch's various implementations of ResNet and MobileNet Faster R-CNN models.

3.2.2 Learning Algorithm

I used mini-batch gradient descent, which is the same learning algorithm as described in 3.1.2.

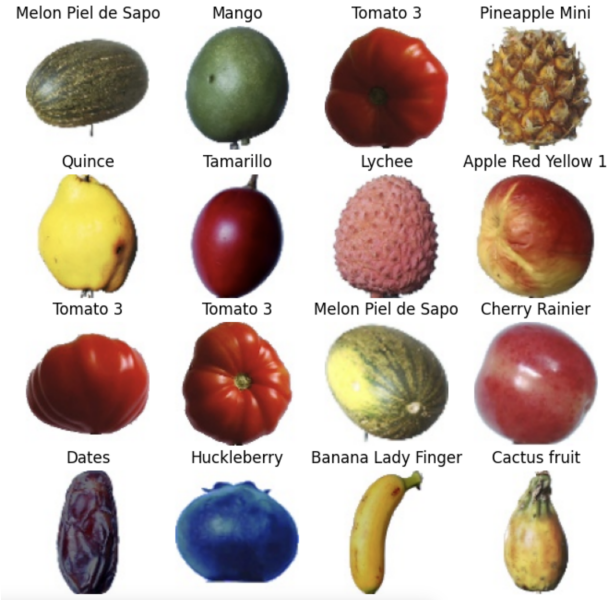

Figure 4. Examples from the fruits-360 dataset.

3.2.3 Loss Function

Object detection is a multi-task problem that involves both localization and classification. As such, the loss function combines regression and classification losses into a single loss function:

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

This is the loss function for a single image. The left-hand term computes to the classification loss of each Region of Interest and takes as inputs the difference between the predicted p_i and true p_i^* class probabilities. The second term calculates the difference between predicted t_i and t_i^* bounding box parameters for the i-th RoI.

4. Dataset and Features

4.0.1 Classifier: Fruits-360

There are 90,483 total images of size 100 x 100 x 3 and 131 classes of fruits and vegetables. Many fruits and vegetables include multiple varieties, for example, Apples include Crimson Snow, Golden, Granny Smith, Red Delicious, and others. These images were pre-split into a training set of 67,692 and 22,688 for testing. This dataset can be found here and was used in this paper [15]. See 4 for examples of the data.

The dataset description observes that: "Fruits and vegetables were planted in the shaft of a low-speed motor (3 rpm) and a short movie of 20 seconds was recorded... Behind the fruits, we placed a white sheet of paper as a back-

ground... However, due to the variations in the lighting conditions, the background was not uniform and we wrote a dedicated algorithm that extracts the fruit from the background."

For my pre-processing, I first split the training data into training and validation (80-20 split). Then, after I compared models, I applied a ColorJitter transformation to ensure that the classifier did not become too dependent on specific lighting or hues. This was an important step because of the uniformity of lighting conditions for these fruits, which would not be seen on farms or orchards.

4.0.2 Detection: Orange

I have a smaller dataset of oranges found here and used in this paper [**?**]. There are 392 images of trees of size 800 x 1040 x 3 or 800 x 1240 x 3. Across these images, there are 6,967 total annotations of orange citrus. This averages to around 17 annotated oranges per image. See 1 for an example.

This data was not pre-split between training and testing, so I did a 70-15-15 split. This meant that I had 274 training images, 58 for validation, and 60 for testing.

For pre-processing, I had to process the annotation xml files accompanying every image. The images themselves were unaltered, although there were several images that had varying sizes and one image that contained no annotated oranges. If I had more time, I would've liked to have experimented with different image augmentations, especially since this is a smaller dataset.

5. Experiments, Results, and Discussion

5.1. Fruit Classification

5.1.1 Experiments

In order to ensure that these models are properly compared to the baseline model in the original dataset paper [15], the following hyper-parameters were utilized when training:

Hyperparameter	Description
Epochs	25
Batch Size	50
Learning Rate	0.1
Loss Function	Cross-Entropy
Optimizer	AdaDelta
Scheduler	ReduceLROnPlateau
Reduction Factor	0.5
Patience	3
Min. Learning Rate	0.00001

Table 1. Hyperparameters used in training

In order to ensure that proper comparisons were made across models, the hyper-parameters for the original paper [15] were utilized.

Each of the three models was trained according to these hyper-parameters and then tested on the same dataset. The training data was randomly split between training and validation data during training. The validation data was used in the scheduler.

Optimizer

The optimizer used in the original paper [15] is an AdaDelta optimizer [22]. It offers some improvements to the Adagrad optimizer. Specifically, it seeks to reduce Adagrad's aggressive, monotonically decreasing learning rate which often causes learning to stop too early. A key advantage is that there is no manual tuning of the learning rate. The update rule is as follows:

$$\Delta x_t = -\frac{\eta}{\sqrt{\sum_{\tau=1}^{t} g_\tau^2}} g_t$$

This update rule is used in gradient descent. The denominator computes the $l2$ norm of all previous gradients on a per-dimension basis, which is slightly different from Adagrad's update rule.

Scheduler

The scheduler in the original paper is Pytorch's "ReduceLROnPlateau". As its name suggests, it will automatically reduce the learning rate by the reduction factor if the validation loss delta over three consecutive epochs (as determined by the patience parameter) stays below a specified threshold. By default, this threshold is 0.0001.

5.1.2 Primary Metrics

For this problem, the primary metric is accuracy. The equation for accuracy is defined as follows:

$$\text{Accuracy} = \frac{\text{TP + TN}}{\text{TP + TN+ FP + FN}}$$

With TP, TN, FP, FN meaning true positive, true negative, false positive, false negative, respectively. We want to ensure that a fruit is correctly labeled, full-stop.

5.1.3 Results

See (Figure 5 and Figure 5.1.3). Note that while the test sets are exactly the same across models. The only thing that changes across these four models is the model architecture.

5.1.4 Discussion

Overall, the ResNet-34 appeared to edge out slightly the baseline convolutional network. However, this comes at a

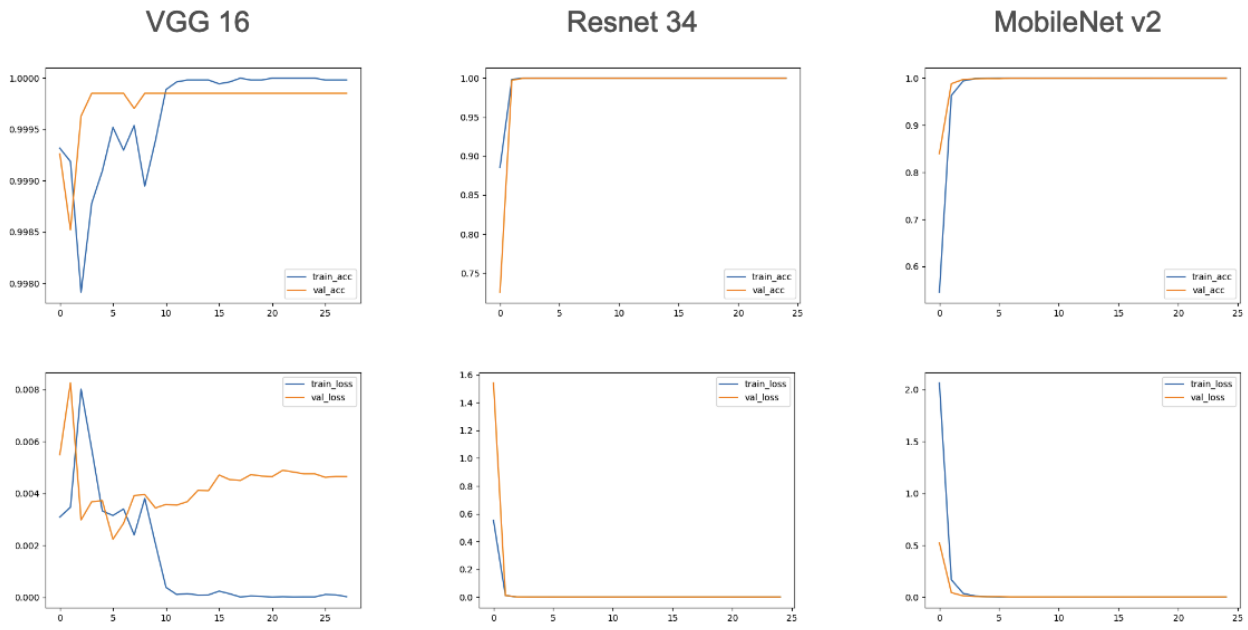

Figure 5. Loss and accuracy saturated quickly for the classifier.

Model	Training Set Accuracy	Test Set Accuracy
Baseline	99.98%	98.66%
VGG-16	99.98%	94.5%
ResNet-34	99.98%	98.77%
MobileNetv2	99.98%	98.09%

Table 2. Comparison of models on Val Set and Test Set

cost. The ResNet-34 is much more complex than the baseline model, which means that it may introduce unnecessary complexity both computationally and in memory.

These models appear to saturate very quickly 5 and achieve extremely high test accuracy. This is most likely a result of the uniformity of conditions across fruits during the creation of the dataset. Specifically, in image generation, a fruit was attached to a drill and rotated. Images were captured from the frames of webcam footage. This may also cause some training set leakage in that the same fruit may be present in both training and test, but at a different angle.

This hypothesis is supported by the original paper [15]. They found that if they converted all images to grayscale, test performance declined, suggesting over-dependence on the colors of the same fruit. If they augmented the data through hue/saturation modifications and horizontal flips, test performance declined further. In other words, this classifier may be over-fit to the exact fruit in this dataset, which will likely not generalize well to other fruit datasets.

5.2. Object Detection

5.2.1 Experiments

Three models were tested on the Orange Tree dataset. The Orange Tree dataset was split into training, validation, and test data. The following hyperparameters were used to fine-tune our model:

Hyperparameter	Description
Epochs	10
Batch Size	8
Learning Rate	0.0001
Optimizer	Adam
Scheduler	ReduceLROnPlateau
Reduction Factor	0.5
Patience	3
Min. Learning Rate	0.000001

Table 3. Hyperparameters used in training the orange detector.

Epochs were selected to give the validation or training accuracy enough time to plateau. After which, the epochs were adjusted to prevent over-fitting. The batch size was selected to be smaller in order to improve generalization while still being feasible given GPU constraints (8 was too small a batch). The Adam optimizer is a good default choice in that it combines the benefits of the AdaGrad and RMS prop optimizers while performing stochastic gradient de-

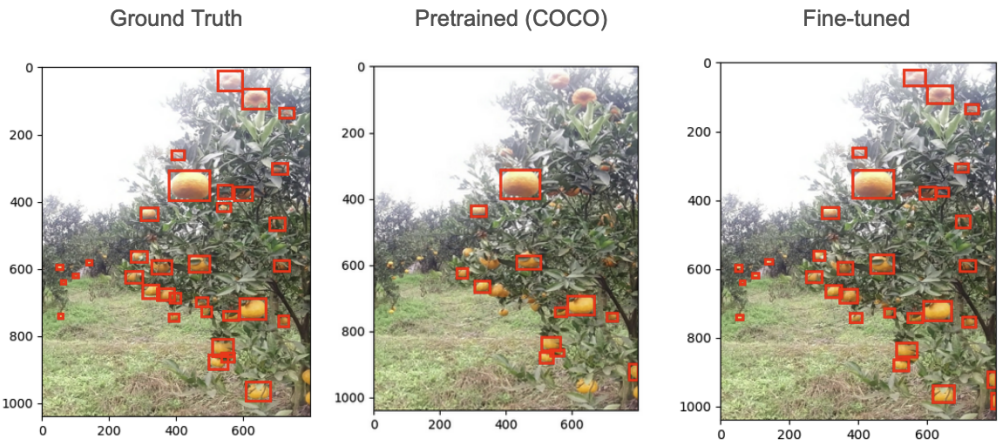

Figure 6. The finetuned model best approximated the ground truth.

scent. I paired this optimizer with a scheduler, the same scheduler from classification training, in order to improve performance by tuning the learning rate.

5.2.2 Metrics

For this problem, the key metric is mean Average Precision (mAP). This metric is calculated by analyzing three other metrics. First, note that precision is defined as the proportion of correctly identified objects among all identified objects (true positives + false positives). Recall is related in that it is the proportion of correctly identified objects among all actual objects (true positives + false negatives). These two metrics can be plotted on a Precision-Recall Curve, which plots precision against recall for different threshold values. Average Precision is the area under this precison-recall curve and it effectively summarizes precision-recall curve. This metric is calculated for each class (in this case, only oranges), and then averaged across all classes. This allows us to summarize detection performance in a single score.

5.2.3 Results

After running the experiment, the following scores were generated.

Model	mAP
Baseline	0.1111
Finetuned	0.4403
Finetuned + Classifier	N/A

Table 4. The finetuned model performed better than the baseline.

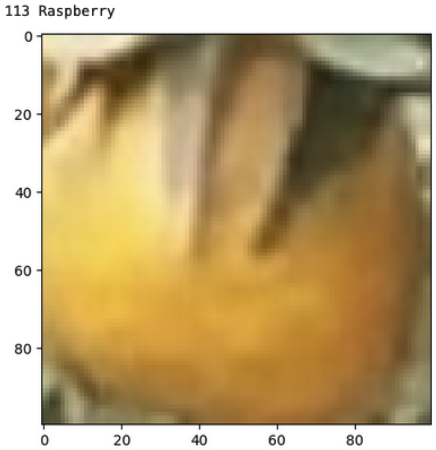

Figure 7. The fruits-360 classifier correctly classified so few of the oranges as to be unusable. It classified this orange as a ”raspberry”.

5.2.4 Discussion

The fine-tuned model demonstrated improvement over the pre-trained COCO detector. This is evident both numerically through the mAP scores (Figure 5.2.3) and visually 6 This makes sense because of the principle of ”transfer learning.” The pre-trained model could already recognize objects such as oranges, as demonstrated by the baseline model. By fine-tuning this pre-trained model, we could take advantage of that knowledge and ”transfer” some of its learning onto this new task. Unfortunately, the classifier didn't prove to generalize outside of the fruits-360 dataset and didn't correctly classify any of the detected oranges. Therefore, it was left unimplemented.

6. Conclusion / Future Work

In this project, two things were done. The first was the training of a highly accurate fruit classifier capable of distinguishing between 131 different variety of fruit with high accuracy. The second was the implementation and training a fruit detection model that could detect oranges in trees. In the first part of the project, the ResNet-34 performed the best on the test set. In the second, the fine-tuned model performed much better than the baseline. I was unsuccessful in connecting the two.

If I had more time (or any other team-members), there are a few categories of things I would've liked to have done. First, I would've liked to have successfully merged the two models into one all-purpose fruit classifier. I would like to integrate the fruit classifier into the classification architecture of the Faster R-CNN. If that is infeasible, I would like to instead to try and layer it on top to enhance predictions.

Second, I would like to improve the fruit classifier by adding more image augmentations. This would make it more generalizable, particularly regarding hue, brightness, and occlusion. This would've overcome the shortcoming with the dataset.

Third, I would like to improve the object detection system. Many of the papers I read described using YOLO and reported good performance. It would also be nice to run some more experiments on hyperparameters and backbone models to improve my detector.

This was a great project where I learned a lot. I think I was too ambitious, given my experience and that I'm working by myself, but I learned enough to implement and train both classifiers and object detection models and I'm excited to build better systems in the future.

7. Appendix

N/A

8. Contributions Acknowledgement

I did the entire project by myself.

References

[1] H. Basri, I. Syarif, and S. Sukaridhoto. Faster r-cnn implementation method for multi-fruit detection using tensorflow platform. In *2018 International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC)*, pages 337–340, 2018.

[2] G. Bradski. The opencv library. *Dr. Dobb's Journal of Software Tools*, June 2000.

[3] A. Buslaev, V. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. Kalinin. Albumentations: Fast and flexible image augmentations, 2020.

[4] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber. High-performance neural networks for visual object classification, 2011.

[5] A. Clark and Contributors. Pillow - the friendly pil fork. `https://python-pillow.org/`, 2015.

[6] C. da Costa-Luis. tqdm: A fast, extensible progress bar for python and cli, 2019.

[7] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy, 2020.

[8] J. Hu, C. Fan, Z. Wang, J. Ruan, and S. Wu. Fruit detection and counting in apple orchards based on improved yolov7 and multi-object tracking methods. *Sensors (Basel)*, 23(13):5903, 2023.

[9] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[10] A. Koirala, K. B. Walsh, Z. Wang, and C. McCarthy. Deep learning – method overview and review of use for fruit detection and yield estimation. *Computers and Electronics in Agriculture*, 162:219–234, 2019.

[11] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft coco: Common objects in context, 2015.

[12] X. Mai, H. Zhang, and M. Q.-H. Meng. Faster r-cnn with classifier fusion for small fruit detection. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7166–7172, 2018.

[13] W. McKinney. Data structures for statistical computing in python. *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, June 2010.

[14] H. Mirhaji, M. Soleymani, A. Asakereh, and S. Abdanan Mehdizadeh. Fruit detection and load estimation of an orange orchard using the yolo models through simple approaches in different imaging and illumination conditions. *Computers and Electronics in Agriculture*, 191:106533, 2021.

[15] H. Mureșan and M. Oltean. Fruit recognition from images using deep learning. *Acta Universitatis Sapientiae, Informatica*, 10:26–42, 06 2018.

[16] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035, 2019.

[17] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[18] I. Sa, Z. Ge, F. Dayoub, B. Upcroft, T. Perez, and C. McCool. Deepfruits: A fruit detection system using deep neural networks. *Sensors (Basel)*, 16(8):1222, 2016.

[19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[20] S. Wan and S. Goudos. Faster r-cnn for multi-class fruit detection using a robotic vision system. *Computer Networks*, 168:107036, 2020.

[21] S. Zeeshan, T. Aized, and F. Riaz. The design and evaluation of an orange-fruit detection model in a dynamic environment using a convolutional neural network. *Sustainability*, 15(5), 2023.

[22] M. D. Zeiler. Adadelta: An adaptive learning rate method, 2012.

[23] W. Zhang, J. Wang, Y. Liu, K. Chen, H. Li, Y. Duan, W. Wu, Y. Shi, and W. Guo. Deep-learning-based in-field citrus fruit detection and tracking. *Horticulture Research*, 9:uhac003, 02 2022.