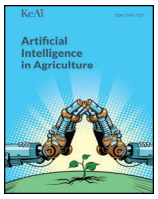
Artificial Intelligence in Agriculture 8 (2023) 30–45

Contents lists available at ScienceDirect

Artificial Intelligence in Agriculture

journal homepage: http://www.keaipublishing.com/en/journals/artificial-intelligence-in-agriculture/



Leguminous seeds detection based on convolutional neural networks: Comparison of Faster R-CNN and YOLOv4 on a small custom dataset

Noran S. Ouf *

Faculty of Engineering, Cairo University, Giza, Egypt

ARTICLE INFO

Article history:
Received 13 February 2022
Received in revised form 21 March 2023
Accepted 29 March 2023
Available online 12 April 2023

Keywords:
Machine learning
Object detection
Leguminous seeds
Deep learning
Convolutional neural networks
Faster R-CNN
YOLOv4

ABSTRACT

This paper help with leguminous seeds detection and smart farming. There are hundreds of kinds of seeds and it can be very difficult to distinguish between them. Botanists and those who study plants, however, can identify the type of seed at a glance. As far as we know, this is the first work to consider leguminous seeds images with different backgrounds and different sizes and crowding. Machine learning is used to automatically classify and locate 11 different seed types. We chose Leguminous seeds from 11 types to be the objects of this study. Those types are of different colors, sizes, and shapes to add variety and complexity to our research. The images dataset of the leguminous seeds was manually collected, annotated, and then split randomly into three sub-datasets train, validation, and test (predictions), with a ratio of 80%, 10%, and 10% respectively. The images considered the variability between different leguminous seed types. The images were captured on five different backgrounds: white A4 paper, black pad, dark blue pad, dark green pad, and green pad. Different heights and shooting angles were considered. The crowdedness of the seeds also varied randomly between 1 and 50 seeds per image. Different combinations and arrangements between the 11 types were considered. Two different image-capturing devices were used: a SAMSUNG smartphone camera and a Canon digital camera. A total of 828 images were obtained, including 9801 seed objects (labels). The dataset contained images of different backgrounds, heights, angles, crowdedness, arrangements, and combinations. The TensorFlow framework was used to construct the Faster Region-based Convolutional Neural Network (R-CNN) model and CSPDarknet53 is used as the backbone for YOLOv4 based on DenseNet designed to connect layers in convolutional neural. Using the transfer learning method, we optimized the seed detection models. The currently dominant object detection methods, Faster R-CNN, and YOLOv4 performances were compared experimentally. The mAP (mean average precision) of the Faster R-CNN and YOLOv4 models were 84.56% and 98.52% respectively. YOLOv4 had a significant advantage in detection speed over Faster R-CNN which makes it suitable for real-time identification as well where high accuracy and low false positives are needed. The results showed that YOLOv4 had better accuracy, and detection ability, as well as faster detection speed beating Faster R-CNN by a large margin. The model can be effectively applied under a variety of backgrounds, image sizes, seed sizes, shooting angles, and shooting heights, as well as different levels of seed crowding. It constitutes an effective and efficient method for detecting different leguminous seeds in complex scenarios. This study provides a reference for further seed testing and enumeration applications.

1. Introduction

Agriculture is vital for human survival and remains a major driver of several economies around the world. Increasing demand for food and cash crops, due to a growing global population, the challenges posed by climate change, and pandemics impacted communities and farmers worldwide. In a crisis like the one we are facing right now; seeds face a sharp rise in demand. Efforts to produce enough nutritious and affordable food are affected by the current health crisis, there is a pressing need to increase farm outputs while incurring minimal costs and human interactions. To address these challenges, big agricultural data and new deep learning technologies are needed to be applied in the agricultural field to help better understand, monitor, measure, and analyze various physical aspects and phenomena for both short-scale crop management as well as for larger-scale agricultural ecosystems' observation, to enhance the management and decision making by situation and context (Ouf, 2018).

In this research we focus on short-scale crop monitoring, specifically leguminous seeds detection using deep learning algorithms. The basic principle of machine learning (ML) is to construct algorithms that can receive input data and use statistical techniques to predict an output

* Corresponding author.
E-mail address: noransouf@gmail.com (N.S. Ouf).

https://doi.org/10.1016/j.aiia.2023.03.002

Table 1
Related studies.

Reference	Attributes	Classes	Crop	Dataset	ML algorithm	Accuracy measure
(Lawal, 2021, 123 CE)	Detection of tomatoes in natural daylight	Two (Ripe and unripe tomatoes)	Tomato	125 images	YOLO-Tomato models (A, B, C) (Modified YOLOv3)	YOLO Tomato-A with AP 98.3% YOLO-Tomato-B with AP 99.2%, YOLO-Tomato-C with AP 99.4%
(Roy et al., n.d.)	Real-time fine-grain object detection	Four (Different diseases in tomato plants)	Tomato	1200 images	Modified YOLOv4	mAP of 96.29
(Roy and Bhaduri, 2022)	Real-time growth stage detection	Four (Growth phases of mango)	Mango	420 images	DenseNet-Fused YOLOv4	mAP of 96.2
(Kundu et al., 2021)	Seeds Classification and Quality Testing	Four (Maize excellent, maize bad, pearl millet, clustered)	Seeds of pearl millet and maize	3954 images	YOLOv5	mAP of 98.3
(Mathew and Mahesh, 2022)	Leaf-based disease detection in bell pepper plant	Two (Healthy part, Bacterial spot)	Bell pepper plant	4000 images	YOLOv5	mAP of 90.7
(Li et al., 2022)	Detection of powdery mildew on strawberry leaves	Two (Infected leaves, powdery mildew)	Strawberry	1023 images	DAC-YOLOv4 (Modified YOLOv4)	mAP of 72.7

Fig. 1. The five different Backgrounds used in the dataset.

Fig. 2. The different crowdedness represented in the dataset.

Fig. 3. The different arrangements and combinations of seed types in the dataset.

while updating outputs as new data becomes available. In addition to investigating the recognition pattern and artificial intelligence, ML expands to the investigation of the construction of algorithms that can learn from and make predictions on data (Sarker, 2021).

Deep learning (DL) is a subset of ML inspired by the structure of the human brain (Elshawi et al., 2021). The central difference is that in traditional machine learning, all of the data analysis and theories development like decision trees, logistic regressions, naive Bayes, and support vector machines was done essentially by the programmer who looked carefully at a particular problem and then designed features that would be useful features for handling this problem. These kinds of systems would end up with millions of hand-designed features. It turns out that the machine was learning almost nothing but only running a learning numerical optimization algorithm to do numeric optimization by putting a parameter weight in front of each feature and adjusting those numbers to optimize performance. However, deep learning is part of this field which is called representation learning. The idea of representation learning is to feed to computers raw signals from the world, whether it is visual signals or language signals, and

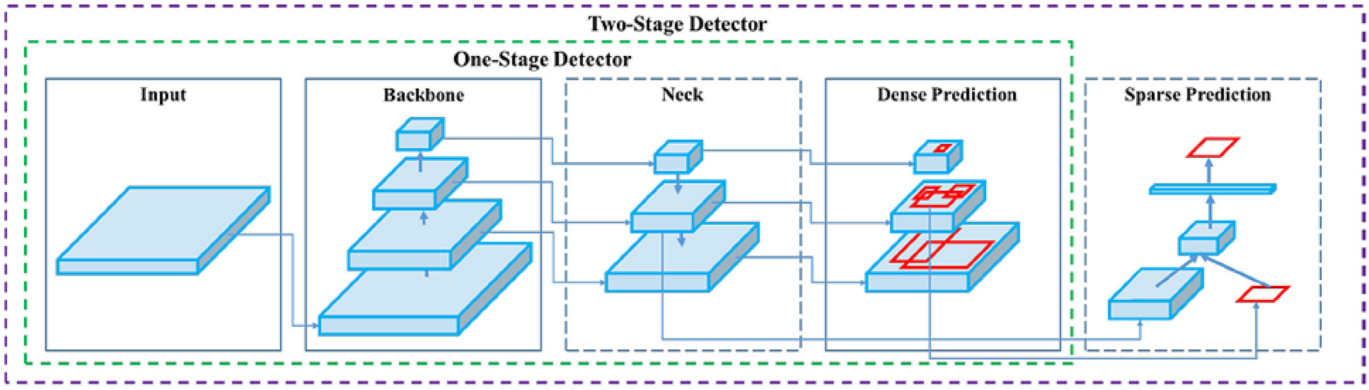


Fig. 4. Object detector architecture.

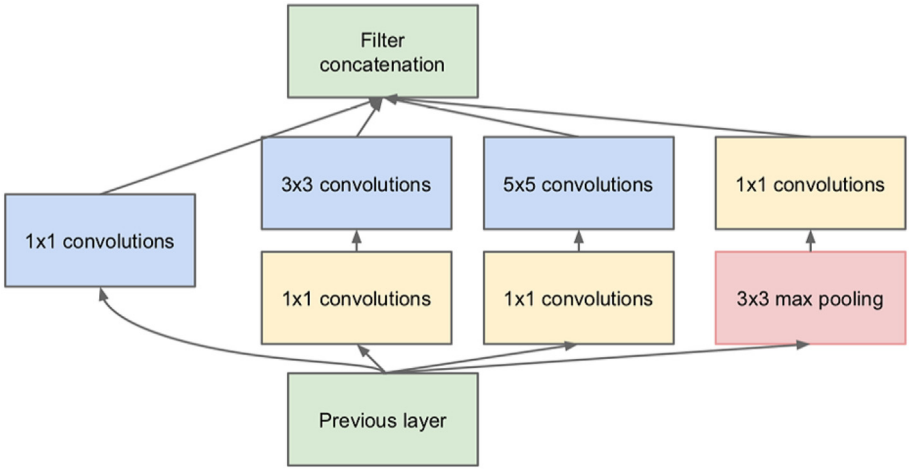


Fig. 5. Inception module.

then the computer can automatically, by itself, come up with good intermediate representations that will allow it to do tasks by inventing its own features in the same way that in the past the human being was inventing the features. Generally, manually designed features tend to be over-specified, incomplete, take a long time to design and validate, and only get to a certain level of performance. However, the learned features are easy to adapt, fast to train, and they can keep on learning so that they get to a better level of performance than has been achieved previously. Deep learning ends up providing this sort of very flexible, almost universal learning framework which is just great for representing all kinds of information.

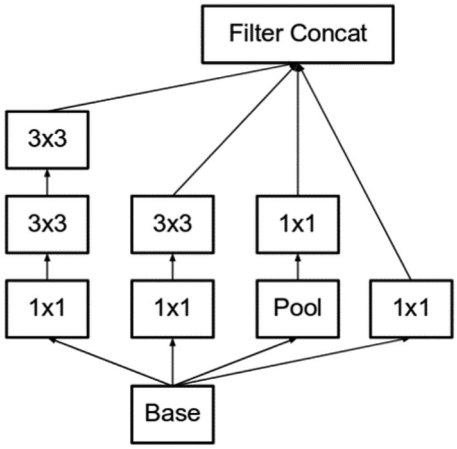


Fig. 6. Module A.

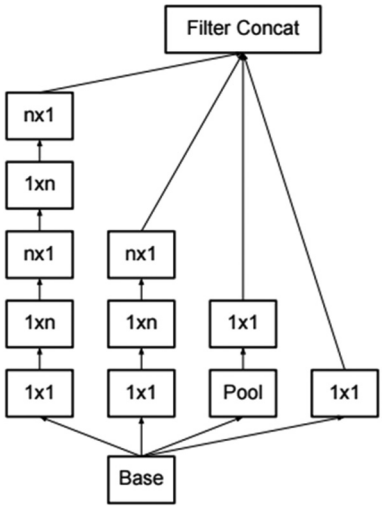


Fig. 7. Module B.

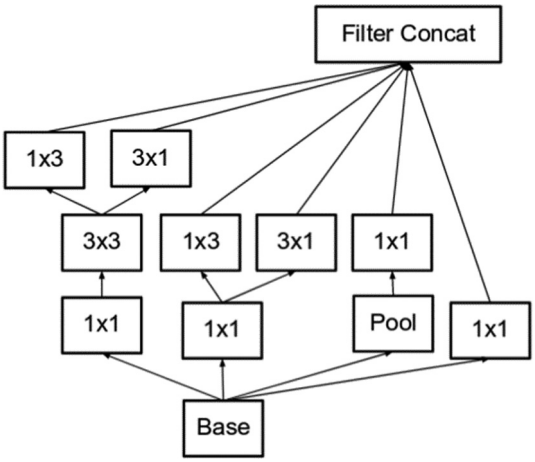


Fig. 8. Module C.

Table 2
Inceptionv2 parameters.

Type	Patch size/stride	Input size
Conv	3 × 3/2	299 × 299 × 3
Conv	3 × 3/1	149 × 149 × 32
Conv padded	3 × 3/1	147 × 147 × 32
Pool	3 × 3/2	147 × 147 × 64
Conv	3 × 3/1	73 × 73 × 64
Conv	3 × 3/2	71 × 71 × 80
Conv	3 × 3/1	35 × 35 × 192
3xInception	Module A	35 × 35 × 288
5xInception	Module B	17 × 17 × 768
2xInception	Module C	8 × 8 × 1280
Pool	8 × 8	8 × 8 × 2048
Linear	Logits	1 × 1 × 2048
Softmax	Classifier	1 × 1 × 1000

Table 3
Faster R-CNN configuration.

Model	Faster R-CNN
Number of classes	11
Min Input dimension	600
Max Input dimension	1024
Image resize	416*416
Feature extractor	Inceptionv2
First stage nms IoU threshold	0.7
First stage max proposals	300
Score converter	SOFTMAX
Batch size	12
Initial learning rate	0.0002
Momentum optimizer value	0.9
Number of steps	35,000
Augmentation	Random horizontal flip

DL has erupted over the last decade and enormously succeeded and expanded with continuous new improvements that are profoundly different from the vast majority of what happened in machine learning in the 80s, 90s, and 00s. Convolutional Neural Networks (CNN) have had a renaissance (Zhao et al., 2019), starting from approximately 2010, the field has been progressing quite so quickly in its ability to be sort of rolling out better methods month on month due to technological advances that have since happened that make this all possible. DL is employed with good performance in a variety of computational tasks such as image classification, object detection, and computer vision.

Object detection is the process of identification/classification and localization of an object or multi objects in an image based on previously defined classes or types. Among many factors and efforts that lead to the fast evolution of object detection techniques, notable contributions should be attributed to the vast amounts of data that favor deep learning models, the development of deep convolution neural networks, the advanced algorithms that provided better ways of learning intermediate representations, end-to-end joint system learning, and transferring information between domains and between contexts, GPUs computing power that allows parallel vector processing, and free cloud services that support a free GPU.

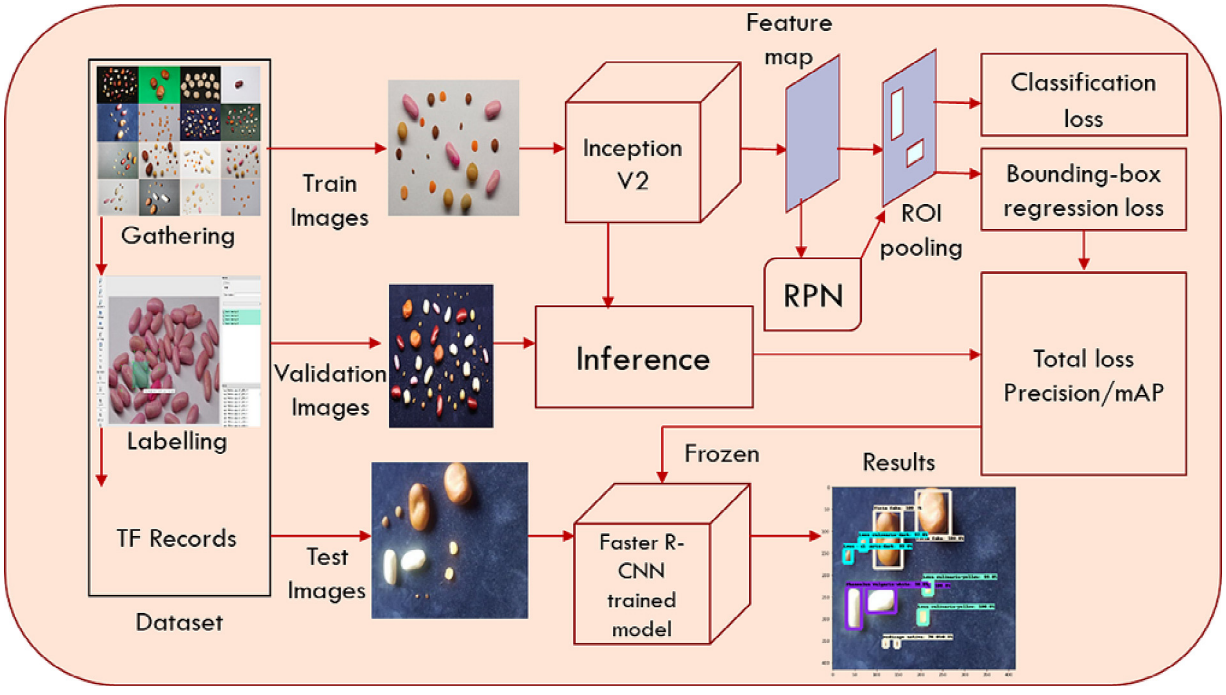


Fig. 9. The Faster R-CNN architecture.

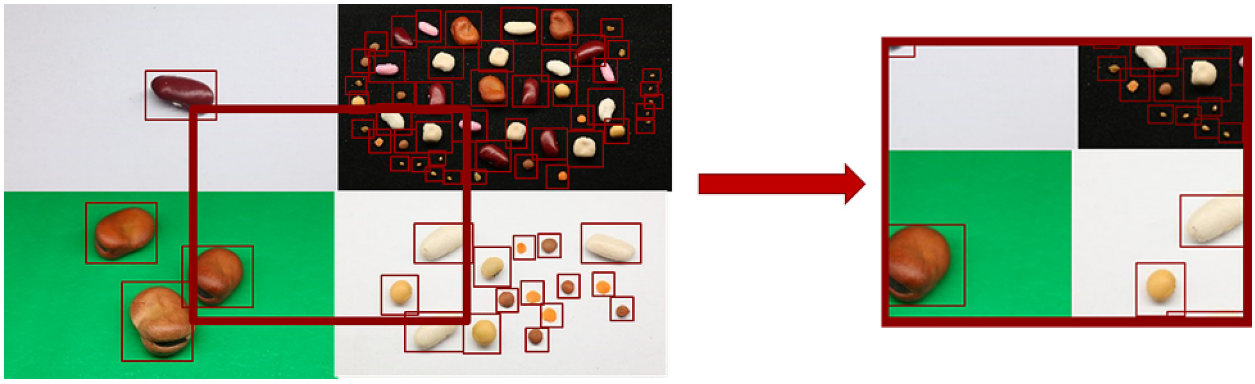


Fig. 10. Mosaic augmentation.

Table 4
YOLOv4 configuration.

Parameter	Value
Classes	11
Batch size	64
Subdivisions	16
Max. Batches	22,000
Steps	17,600, 19,800
Filters	48
Input width and height	416*416

There are two types of object detection models, the one-stage (dense detection) model and the two-stage (sparse prediction) model (Jiao et al., 2023). A two-stage detector uses a preliminary stage where regions of importance are detected and then classified to see if an object has been detected in these areas. Two-stage detectors include the Region-based Convolutional Neural Network (R-CNN) algorithms that have truly been a game-changer for object detection tasks since 2013 when Girshick (Girshick et al., 2013) presented R-CNN that made major progress in the field of object detection in terms of accuracy. He followed it by Fast Region-based Convolutional Neural Network (Fast R-CNN) (Girshick, 2015), and Faster Region-based Convolutional Neural Network (Faster R-CNN) (Ren et al., 2017), which will be used in this research. On the contrary, a one-stage detector is capable of detecting objects without the need for a preliminary step. The advantage of a one-stage detector is the speed it can make predictions quickly allowing real-time use. One-stage detectors include Single Shot MultiBox Detector (SSD) (Liu et al., 2016), and You Only Look Once (YOLO) (Bochkovskiy et al., 2020; Redmon et al., 2015; Redmon and Farhadi, 2018) algorithms. These algorithms combined the field of object detection with deep learning, achieving significant improvements in accuracy, loss, and runtime.

The plant/Seed detection field has significant room for improvement in terms of seed types, scene recognition, accuracy, loss, runtime, and ease of use. This study aimed to use deep learning-based models to detect 11 different types of leguminous seeds using the free available COLAB platform.

Table 5
Faster R-CNN and YOLOv4.

	Faster R-CNN	YOLOv4
Framework	TensorFlow	Darknet
Phases	RPN + Fast R-CNN detector	Concurrent bounding-box regression and classification
Neural network type	Fully convolutional	Fully convolutional
Backbone feature extractor	Inceptionv2	CSPDarknet53
Location detection	Anchor-based	Anchor-based

Table 6
Computer configuration.

Parameter	Google colab
CPU	Intel(R) Xeon(R) CPU @ 2.30GHz
CPU cores	2
RAM	12GB
Disk space	25GB
GPU	Nvidia Tesla T4
GPU memory	16GB
GPU memory clock	1.59GHz
Performance	8.1 TFLOPS
Support Mixed Precision	Yes
RAM	12GB
Disk space	358GB

2. Related work

Various studies addressed crop/seed classification and detection by using deep learning. Recent studies have been conducted to perform classification, disease detection, and observation of growth stages of

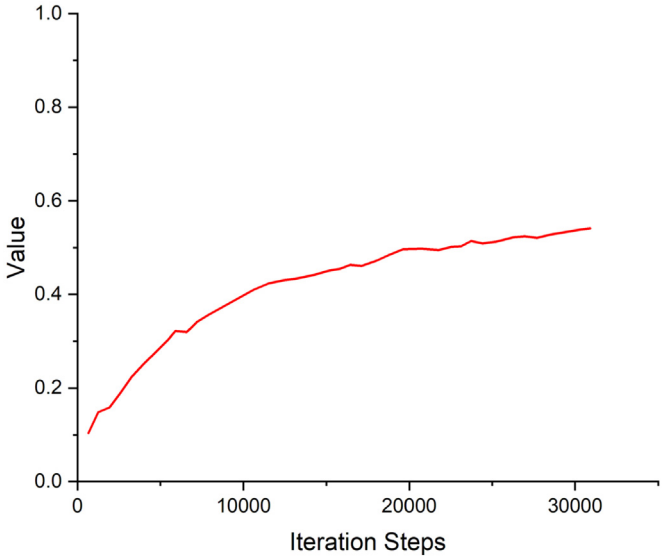


Fig. 11. Detection Boxes Recall/AR@100 (small).

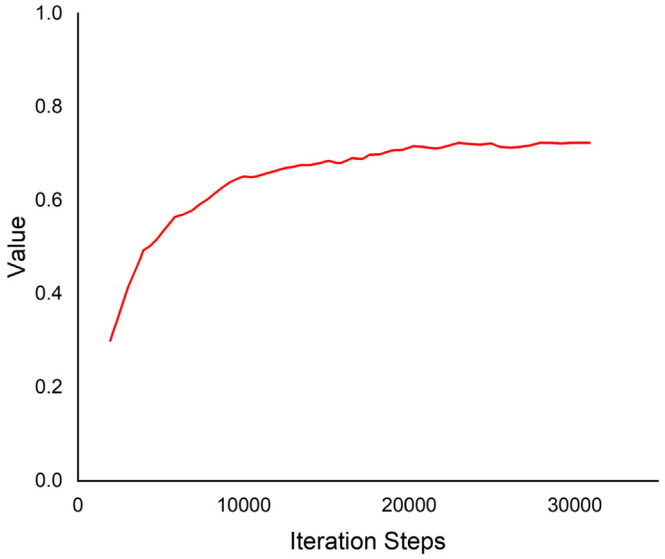


Fig. 12. Detection Boxes Recall/AR@100 (medium).

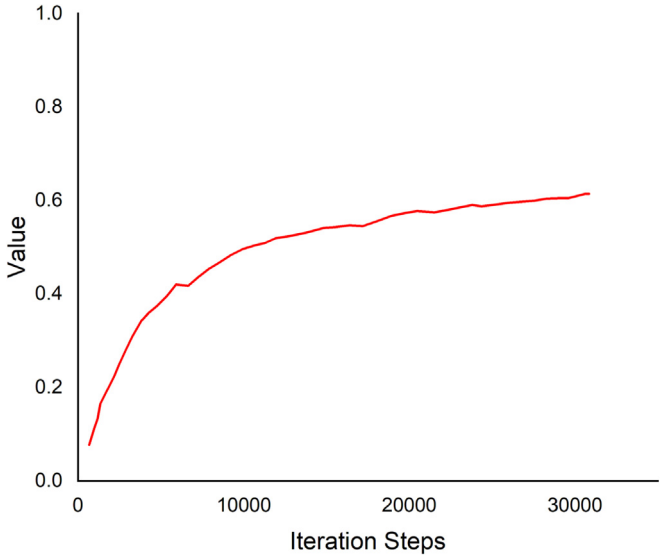


Fig. 14. Detection Boxes Recall/AR@100.

different seeds and crops by using deep learning techniques (See Table 1).

(Lawal, 2021, 123 CE) modified YOLOv3 model, the called YOLO-Tomato models used to detect two classes of ripe and unripe tomatoes, they constructed a small dataset of 125 images of tomatoes. The dataset was grouped into Raw, 0.5 ratio, and 0.25 ratio for training and testing. They kept all things the same as the YOLOv3 model with an increase in the concatenated features and an increase features in the FPN of the YOLO-tomato model. YOLO-Tomato-A was activated with Leaky Rectified Linear Unit (ReLU)31 having FDL × 3. The six layers of YOLOv3 were pruned as YOLO-Tomato-B was activated with Mish28 having FDL × 1, and YOLO-Tomato-C was activated with Mish28 having FDL × 2 and SPP26. The compared results of AP showed that YOLO-Tomato-A, B, and C outperformed the YOLOv3 model but not YOLOv4 for both raw data and 0.5 ratio. However, the AP of YOLO-Tomato-C was slightly increased compared to YOLOv4 for 0.25 ratio but the detection speed also increased. The performance of the models was high because of the very small datasets which also require further investigation.

(Roy et al., n.d.) proposed a real-time object detection model that was developed based on the YOLOv4 algorithm. They included CSP1-n block in the backbone, CSP2-n module in the neck, and DenseNet in the backbone to optimize feature extraction, transfer, and reuse. The proposed detection model was used to detect four different diseases in tomato plants. 300 images from each of the four different tomato plant diseases are collected from the publicly available Kaggle Dataset to construct a dataset consisting of 1200 images and augmented to obtain the custom dataset of 12,000 images. The model outperforms the existing state-of-the-art detection models in detection accuracy and speed with mean average precision (mAP) value of 96.29% compared to 92.84 for YOLOv4.

(Roy and Bhaduri, 2022) proposed a real-time object detection framework Dense-YOLOv4 based on an improved version of the YOLOv4 algorithm. The proposed model included DenseNet in the back-

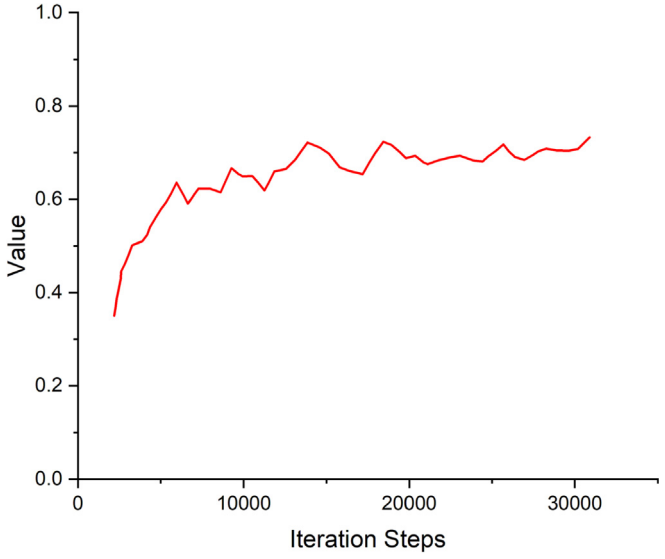


Fig. 13. Detection Boxes Recall/AR@100 (large).

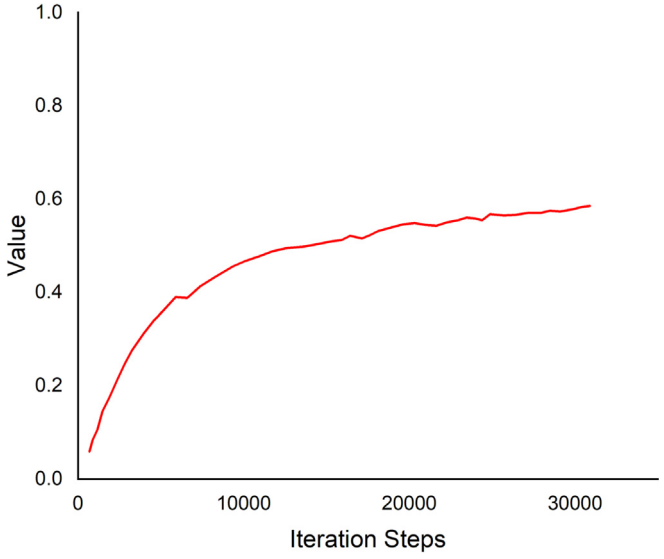


Fig. 15. Detection Boxes Recall/AR@10.

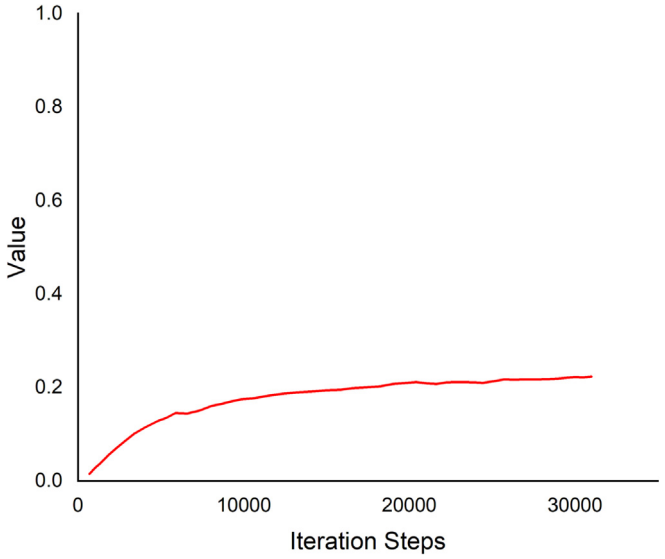


Fig. 16. Detection Boxes Recall/AR@1.

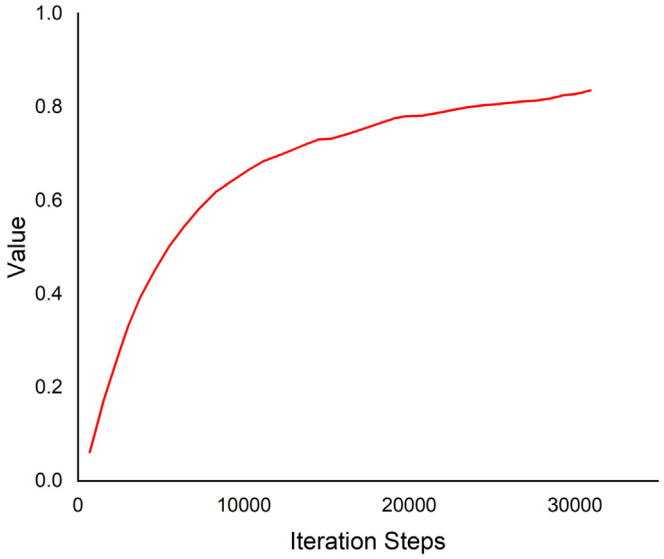


Fig. 18. Detection Boxes Precision/mAP@.50IOU.

bone, SPP block, and modified path aggregation network PANet in the YOLOv4 framework. The model was used to detect different growth stages of mango. A total of 420 original images consisting of 105 images from each of the four growth phases have been considered to construct the original dataset and augmentation has been applied expanding the original dataset tenfold. The mean average precision (mAP) of the proposed model has reached up to 96.20% at a detection rate of 44.2 FPS. The proposed Dense-YOLOv4 has outperformed the state-of-the-art YOLOv4 with a 4.73% increase in mAP.

(Kundu et al., 2021) used the YOLOv5 model for the classification and quality testing of seeds. The dataset consisted of 3954 images of seeds of pearl millet, healthy and diseased maize, and clustered was constructed. The model achieved the precision and recall of 99% in classifying the seeds into two classes pearl millet and maize with the quality of healthy or diseased.

(Mathew and Mahesh, 2022) work focused on the identification of diseases in bell pepper plant in large fields. YOLOv5 was used for detecting bacterial spot disease in the bell pepper plant from the symptoms seen on the leaves. The dataset consisted of a total of 4000 images, 2000 healthy and 2000 with bacterial spots. The YOLOv5 achieved 90.7%.

(Li et al., 2022) used a dataset of 6371 images of Strawberry powdery mildew (PM) and infected leaves (IL). The original YOLOv4 backbone and neck were replaced by their proposed backbone and neck with depth-wise convolution and hybrid attention mechanism. They combined the proposed backbone and neck, forming four new network structures, the best one was named DAC-YOLOv4. DAC-YOLOv4 used the depth-wise convolution and CSPNet structure concept. Compared with YOLOv4, the mean average precision (mAP) of DAC-YOLOv4 reaches 72.7%, while the size is greatly compressed.

3. Materials and methods

We chose Leguminous seeds from 11 types to be the objects of this study. Those types are of different colors, sizes, and shapes to add variety and complexity to our research. Since the objects of the study are not

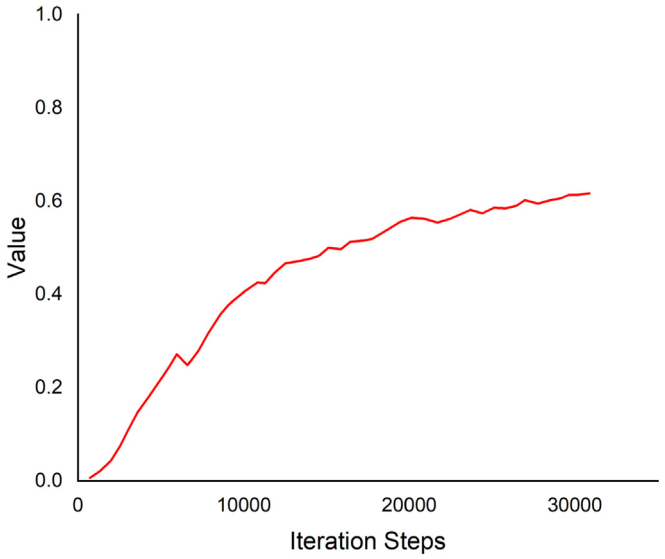


Fig. 17. Detection Boxes Precision/mAP@.0.75.

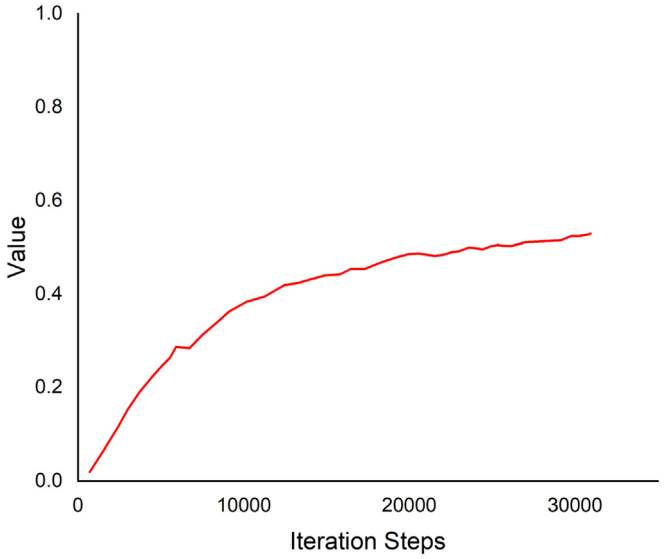


Fig. 19. Detection Boxes Precision/mAP.

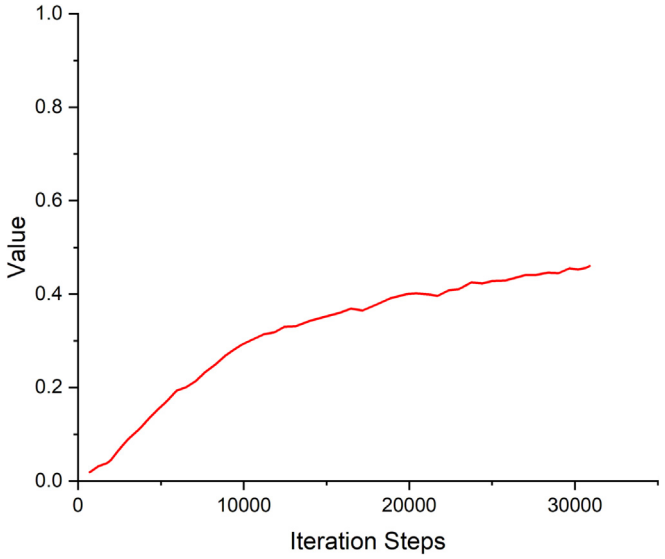


Fig. 20. Detection Boxes Precision/mAP (small).

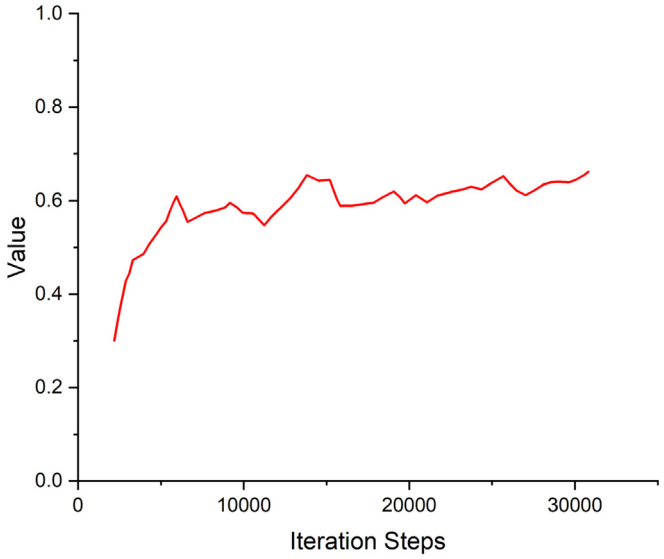


Fig. 22. Detection Boxes Precision/mAP (large).

on publicly available datasets designed and dedicated for machine learning and computer vision research as ImageNet (http://www.image-net.org/), PASCAL VOC (http://cvlab.postech.ac.kr/~Mooyeol/pascal_voc_2012/), COCO (http://cocodataset.org/), the images dataset of the leguminous seeds was manually collected, annotated, and then split randomly into three sub-datasets train, validation, and test (predictions), with a ratio 80%, 10%, 10% respectively.

3.1. Image collection

Images for our dataset were manually collected. The images considered the variability between different leguminous seed types. The 11 types of leguminous seeds selected to be the research objects of this study are *Glycine* max, *Lens culinaris*-dark, *Lens culinaris*-yellow, *Lupinus albus*, *Medicago sativa*, *Phaseolus vulgaris*-pink, *Phaseolus vulgaris*-red, *Phaseolus vulgaris*-white, *Trifolium alexandrinum*, Trigonella foenun graecum, and *Vicia faba.*

The images were captured on five different backgrounds: white A4 paper, black pad, dark blue pad, dark green pad, and green pad. Different heights and shooting angles were considered. The crowdedness of the seeds also varied randomly between 1 and 50 seeds per image. Different combinations and arrangements between the 11 types were considered. Two different image-capturing devices were used: a SAMSUNG smartphone camera and a Canon digital camera. A total of 828 images were obtained, including 9801 seed objects (labels). The dataset contained images of different backgrounds, heights, angles, crowdedness, arrangements, and combinations as shown in Figs. 1, 2, and 3.

3.2. Object annotation

Object detection or object recognition is the task of identification and localization of the object in an image. The collected images were labeled by an object annotation tool. Among the commonly used tools like LabelImg, Imglab, LabelMe, Labelbox, and RectLabel, LabelImg (Tzutalin,

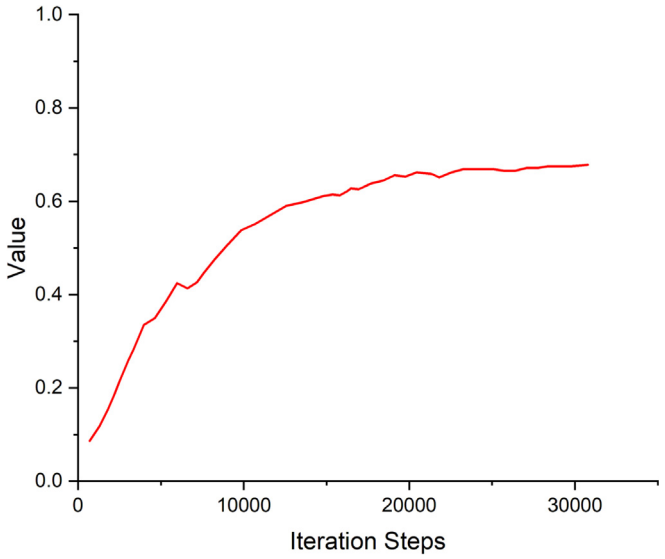


Fig. 21. Detection Boxes Precision/mAP (medium).

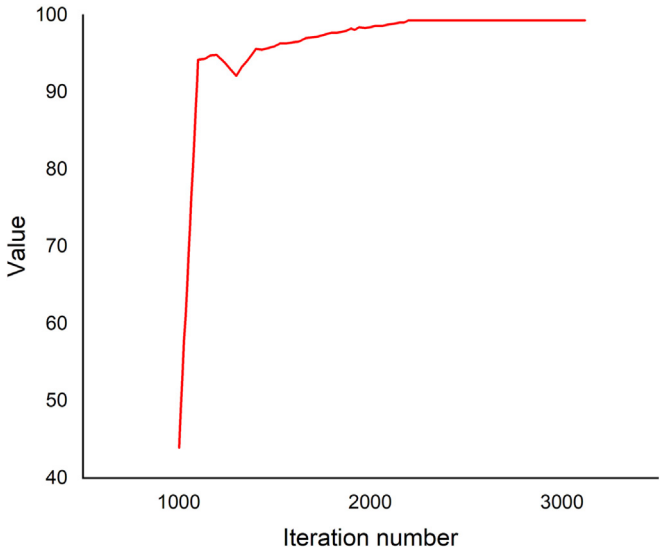


Fig. 23. Yolov4 mAP@0.5.

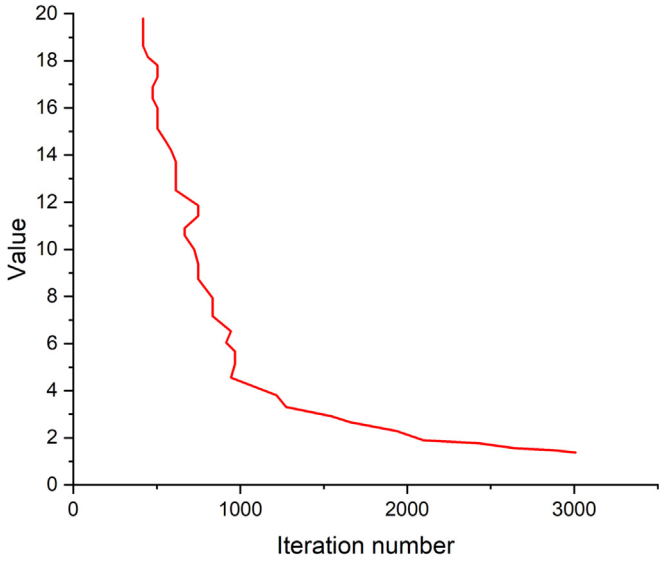


Fig. 24. Yolov4 loss.

https://github.com/tzutalin/labelImg) tool was selected. The image annotations generated were saved in both a .txt file format, the input for YOLO, and a .xml file format used with the PASCAL VOC dataset that can be easily converted to TFrecords.

3.3. Data standardization

We used the Darknet deep learning framework for the YOLOv4 model. Now ready, the images and annotations data were input into the model. For the Faster R-CNN model, we used TensorFlow deep learning framework, which needed the .xml annotations data to be converted into the TFRecord data type. Then the dataset was randomly split into train, validation, and test sets with ratios of 80%, 10%, and 10%, respectively.

3.4. Pre-processing

All images in our dataset were pre-processed before they were input into the models. The most important pre-processing was the input image resolution that would fit our models to avoid running out of memory, low speed, and low accuracy. Images were resized to 416×416 pixels.

3.5. Network models

All object detectors Consist of a backbone, neck, and detection head. First, the input image is fed to the backbone which compresses features down through a convolutional neural network. Unlike image classification, object detection backbones are not the end of the network. Predictions can't be made off only of them; Localization needs to be along with classification. Localization is the task of locating an object in the image by drawing multiple bounding boxes, so the feature layers of the convolutional backbone need to be mixed and held up in light of one another. The combination of backbone feature layers happens in the neck then the detection occurs in the head.

It is also useful to split object detectors into two categories, as shown in Fig. 4 (Bochkovskiy et al., 2020): one-stage detectors and two-stage detectors. While two-stage detectors decouple the task of object localization and classification for each bounding box, one-stage detectors make the predictions for object localization and classification simultaneously.

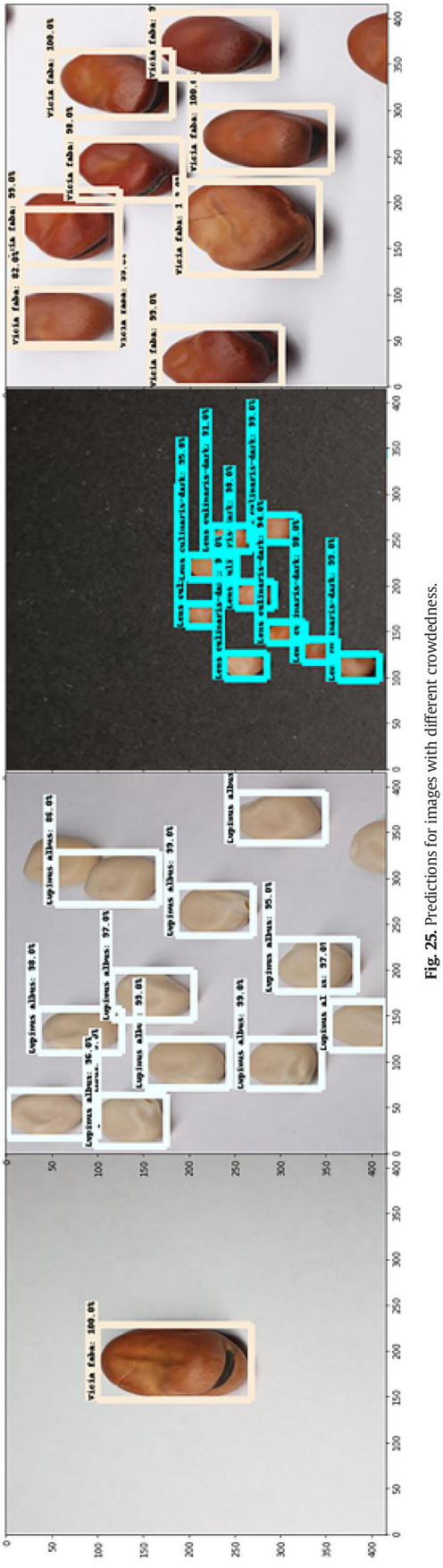


Fig. 25. Predictions for images with different crowdedness.

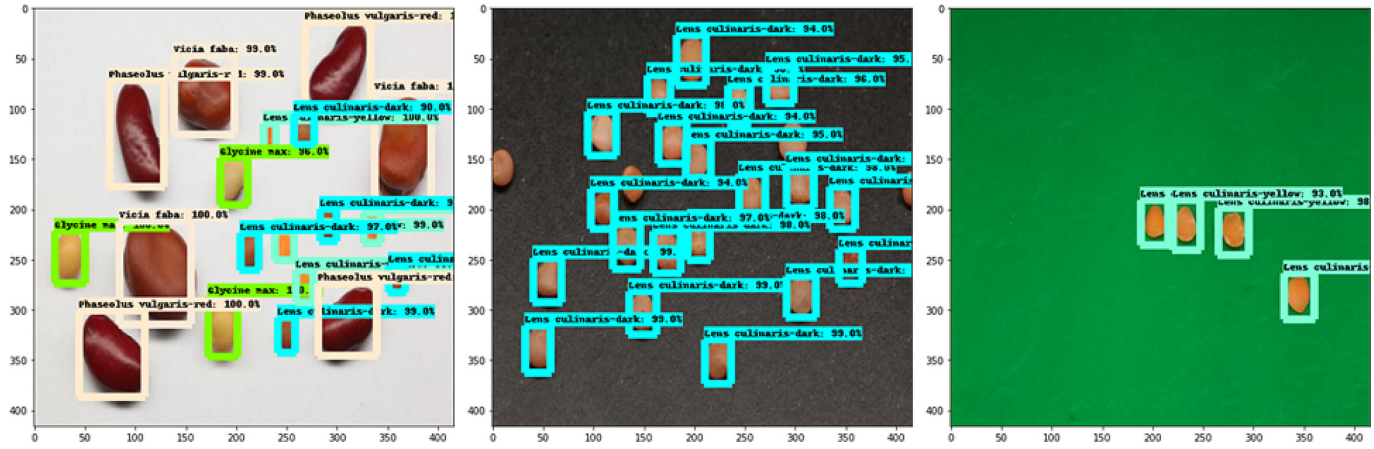


Fig. 26. Predictions for images with different background.

3.5.1. Faster R-CNN

Two models were used and compared for object detection. The Faster R-CNN Model was developed from R-CNN and Fast R-CNN. Like all the R-CNN family, Faster R-CNN is a region-based well-established two-stage object detector, which means the detection happens in two stages. The Faster R-CNN architecture consists of a backbone and two main networks or, in other words, three networks. First is the backbone that functions as a feature extractor by running a convolutional neural network on the original map to extract basic features and generate a feature map. In this study, Inceptionv2 pre-trained on the MS COCO dataset was chosen as the backbone. The two main networks, the first network is a simple regional proposal network (RPN) that proposes a set of regions of interest not using a selective search algorithm like its predecessors. The second network is an evaluation network or a detection network that processes both the feature map and the regions of interest generated by the previous networks by a classification layer and a bounding box regression layer, generating the class and bounding box.

There are multiple different feature extractors available to choose from, including VGG16, Inception, ResNet, and MobileNet. Fig. 5 shows the Inception module of GoogLeNet used to build the Inceptionv1 model. The model's convolutional layers used several filter kernel sizes 1×1, 3×3, and 5×5 (blue blocks) along with max-pooling (pink block). Additional 1×1 convolutions were added before the 3×3 and 5×5 convolutions to reduce computations. The filter concatenation block (in green) concatenates the output depth of the convolutions and max pooling. Inceptionv1 consisted of nine linearly stacked modules, a deep model of 27 layers. Inceptionv2 used three modified module types different from the one used in Inceptionv1. Module A, shown in Fig. 6, replaced or factorized the 5×5 computationally expensive convolutions with two 3×3 convolutions. Module B, shown in Fig. 7, replaced the nxn convolutions with a 1xn convolution followed by nx1 convolution. Eventually, Module C is shown in Fig. 8, where the module filters are expanded to prevent loss of information due to dimension reduction. Our choice of Inceptionv2 as the backbone network was based on its architecture which helps reduce the problem of vanishing gradient and increasing loss faced by deeper models that can cause overfitting, especially with our small dataset. The parameters of Inceptionv2 42 layers are shown in Table 2 based on the three modules described above.

The RPN is a simple convolutional network with anchors of fixed dimensions and ratios that generates a set of bounding boxes called regions of interest (ROI) at each anchor location. The detection network receives the ROI and looks for objects within those regions—eventually, the detector returns proposals of final bounding boxes with a confidence score. The architecture of our Faster R-CNN model is shown below in Fig. 9, and the Faster R-CNN configuration is shown in Table 3.

3.5.2. YOLOv4

The original YOLO (You Only Look Once) used a Darknet flexible framework that was written in low-level languages and has produced a series of the best real-time object detectors in computer vision. The YOLO series moves ever forward with the publication of YOLOv4 in the past couple of months. YOLOv4 outperforms other object detection models by a significant margin in inference speed. The Original YOLO was the first object detection network to combine the problem of drawing bounding boxes and identifying class labels in one end-to-end differentiable network. YOLOv2 made a number of iterative improvements on top of YOLO, including BatchNorm, higher resolution, and anchor boxes. YOLOv3 built upon previous models by adding an abjectness score to bounding box prediction, added connections to the backbone network layers, and made predictions at three separate levels of granularity to improve performance on smaller objects.

CSPDarknet53 is used as the backbone for YOLOv4 based on DenseNet designed to connect layers in convolutional neural networks with the following motivations to alleviate the vanishing gradient problem, bolster feature propagation, encourage the network to reuse features, and reduce the number of network parameters.

The neck then is to mix and combines the features formed in the ConvNet backbone to prepare for the detection step. The components of the neck typically flow up and down among layers and connect only a few layers at the end of the convolutional network. Of all the options as FPN, PAN, NAS-FPN, BiFPN, ASFF, and SFAM, YOLOv4 chooses PANet for the feature aggregation of the network. Additionally, YOLOv4 adds an SPP block after CSPDarknet53 to increase the receptive field and separate out the most important features from the backbone.

Finally, the detection head used by YOLOv4 is the same as YOLOv3 with anchor-based detection steps and three levels of detection granularity.

Bag of Freebies is so termed because they improve the network's performance without adding to inference time in production. Mostly Bag of Freebies has to do with data augmentation. YOLOv4 uses data augmentation to expand the size of its training set and expose the model to semantic situations that it would not have otherwise seen. Most of the augmentation was already known in the field of computer vision except the mosaic data augmentation, which combines four images together in one image, resizes along with the BB and put them on one grid, then takes a random crop from the center as shown in Fig. 10. The technique help with teaching the model to find smaller objects and pay less attention to surrounding scenes that are not immediately around the object. Our custom dataset is a good candidate for mosaic, which helps a lot with real-world objects, small objects, and mobile objects (the position of objects changes in an image).

Mosaic was optimized by equally sampling all parts of the image (including objects not just at the center). Since larger upfront objects are

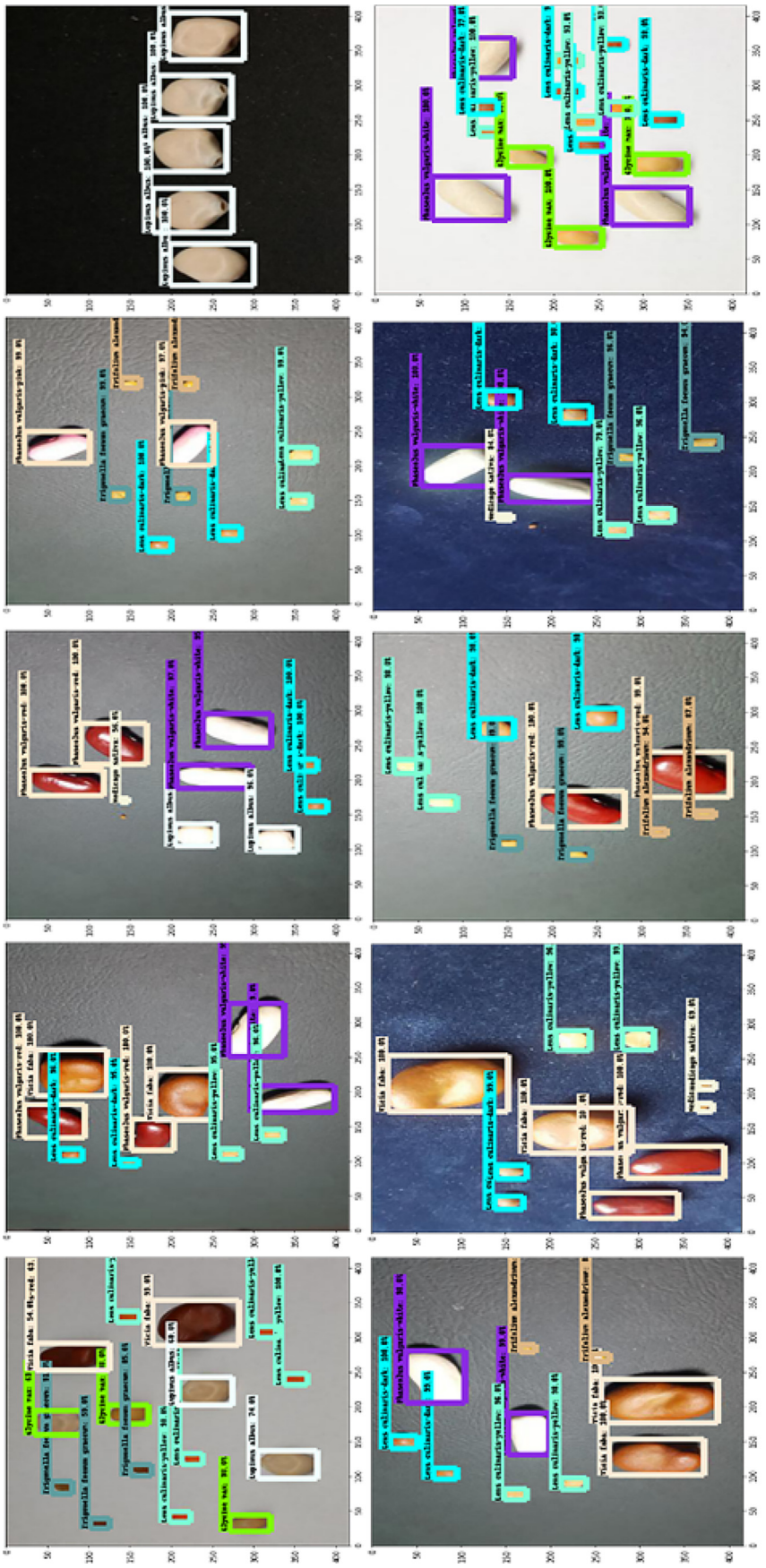


Fig. 27. Predictions for images with different combinations and multiple types.

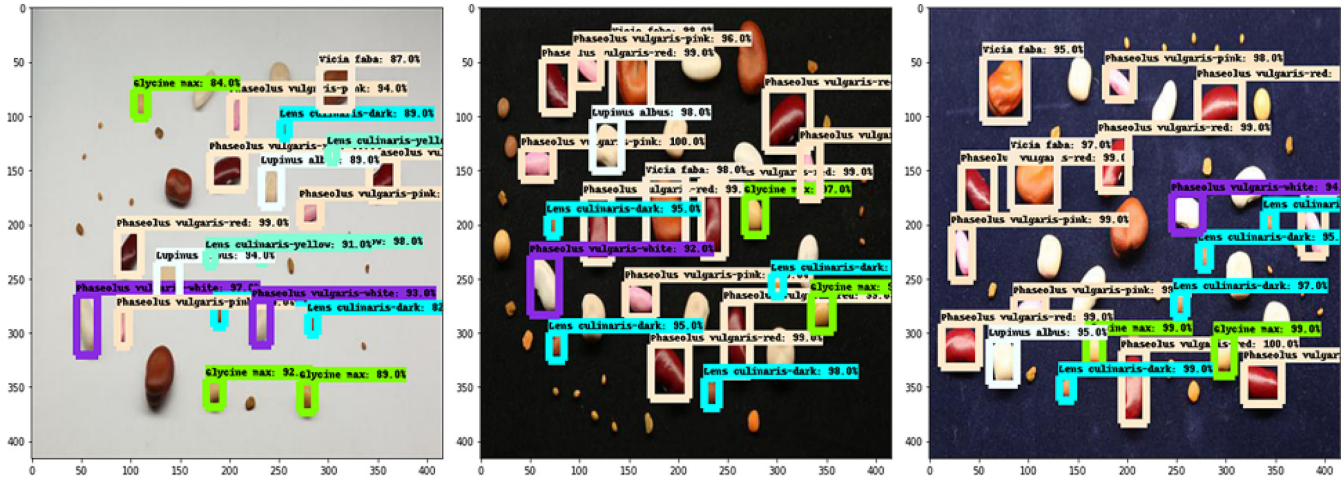


Fig. 28. Predictions for images with all types.

more commonly labeled (detected) than smaller further away objects by a detector, we placed smaller objects at the edge up and down in an image so the mosaic random rectangular crop from the center will capture them (zoom in on smaller objects).

YOLOv4 configuration is shown in Table 4. The maximum batches are set to be the number of classes * 2000. The training steps are set to be 80% of maximum batches and 90% of maximum batches, and the filters in the three convolutional layers before the YOLO layer are set to be (number of classes+5) *3 (See Table 5).

3.6. The assessment method

3.6.1. Loss function

The performance of a model is assessed by a cost function or a loss function. The smaller the loss function is, the better the model fits. Similar to fast R-CNN. Faster R-CNN is optimized for a multi-task loss function (Wu et al., 2020). The loss function combines the losses of classification and bounding box regression as follows:

$$L = L_{cls} + L_{box} \tag{1}$$

Fig. 29. YOLOv4 predictions.

Fig. 30. YOLOv4 predictions with multiple types.

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \frac{\lambda}{N_{box}}\sum_i p_i^* . L_1^{smooth}(t_i - t_i^*) \quad (2)$$

where i represents the index of an anchor in a batch, p_i is the predicted probability of anchor i being an object, p_i^* is the Ground truth label (binary) of whether anchor i is an object; when the anchor is a positive sample, $p_i^* = 1$, and when it is a negative sample $p_i^* = 0$. It can be seen that the regression loss term is only activated if the anchor is positive; t_i is the predicted four parameterized coordinates of the positive sample anchor; t_i^* is the ground truth coordinates of the positive sample anchor; λ is a balancing parameter used to weigh classification loss L_{cls} and bounding box regression loss L_{box} so that both terms are roughly equally weighted, and the default value of λ set to be ~10; N_{cls} and N_{box} are normalization terms used to normalize classification loss item L_{cls} and regression loss item L_{box}, respectively. Where L_{cls} is the log loss function over two classes, as we can easily translate a multi-class classification into a binary classification by predicting a sample being a target object versus not. L_1^{smooth} is the smooth L_1 loss. The classification loss function L_{cls} is a Boolean classifier (object or not), and the formula is as follows:

$$L_{cls}(p_i, p_i^*) = -p_i^* \log p_i - (1 - p_i^*) \log (1 - p_i) \quad (3)$$

The bounding box regression loss function L_{box} is used to calculate the difference between the two transformations, and the formula is as follows:

$$L_{box}(t_i, t_i^*) = R(t_i - t_i^*) \quad (4)$$

Table 7
Model performance evaluation.

Model	mAP@0.5	Time/ms
YOLOv4	98.52%	47.2
Faster R-CNN	84.56%	53.1

where R function is defined as

$$Smooth_{L1}(x) = \begin{cases} 0.5x^*, & if|x| < 1 \\ |x| - 0.5 & otherwise \end{cases} \quad (5)$$

3.6.2. Precision, recall, and mAP

The mAP is a popular metric used to assess the performance of an object detection model. It involves two concepts: precision and recall. For an object, precision, also known as the positive predicted value, is the ratio of correctly predicted positive observations to the total predicted positive observations, which means high precision relates to the low false-positive rate

$$Precision = \frac{TP}{TP + FP} \quad (6)$$

While recall, also known as true-positive rate or sensitivity, is the ratio of correctly predicted positive observations to all observations in an actual class

$$Recall = \frac{TP}{TP + FN} \quad (7)$$

A trade-off between precision and recall performance can be adjusted by the model's final layer softmax threshold. Increasing the threshold would decrease the number of FP which will lead to higher precision and lower recall. Similarly, to increase recall we need to decrease the number of FN which will reduce precision. Commonly in object detection tasks, precision needs to be high (predicted positives to be TP). Precision and recall are widely used along with other metrics such as accuracy, which is simply a ratio of correctly predicted observation to the total observations

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (8)$$

and F1-score also known as true negative rate (TNR) (Boracchi et al., 2017) is the weighted average of Precision and Recall not as simple as

accuracy but more useful than accuracy. Accuracy is a great measure but only when you have symmetric datasets where values of false positives and false negatives have a similar cost. However, the F1 score takes both false positives and false negatives into account. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

$$F1Score = \frac{2^{*}(\text{ Re } call^{*}Precision)}{Recall + Precision} \tag{9}$$

Intersection over Union (IoU) defines the calculation of AP for object detection. The IoU is given by the ratio between the area of intersection and the area of the union of the predicted bounding box and ground truth bounding box as shown in the equation.

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{10}$$

The IoU would be used to determine if a predicted bounding box (BB) is TP, FP, or FN. The TN is not evaluated as each image is assumed to have an object in it. Traditionally, IoU is set to 0.5. when the object detection model run on an image, a predicted bounding box would be defined to be a TP if the IoU is >0.5, FP if either IoU < 0.5 or the bounding box is duplicated, and FN If the object detection model missed the target either because there is no detection at all or the predicted BB has an IoU > 0.5 but has the wrong classification, the predicted BB would be FN.

Precision and recall were calculated for a given class across the test set. Each BB would have its confidence level, usually given by its softmax layer, and would be used to rank the output.

3.6.3. Interpolated precision

Before we plot the PR curve, we need first need to know the interpolated precision. The interpolated precision, p_{interp}, is calculated at each recall level, r, by taking the maximum precision measured for that r. The formula is given as such:

$$p_{interp}(r) = \max_{\tilde{r}:\tilde{r}\geq r} p(r) \tag{11}$$

where $p(\tilde{r})$ is the measured precision at recall $\tilde{r}$.

Their intention of interpolating the PR curve was to reduce the impact of "wiggles" caused by small variations in the ranking of detections, then we can plot the PR curve. For each example, the corresponding precision, recall, and interpolated precision are calculated by the formulas defined above. The AP is then calculated by taking the area under the PR curve. This is done by segmenting the recalls evenly into 11 parts: {0,0.1,0. 2, …,0.9,1}. We get the following:

$$AP = \frac{1}{11} \sum_{r\in\{0,0.1,0.2,0..,0.9,1\}} p_{interp}(r) \tag{12}$$

3.6.4. COCO metrics

The COCO dataset is typically used to train and validate object detection models. It contains a broad range of 80 object classes that will help a model generalize. Transfer learning then can be used to expose pre-trained models on the COCO dataset to new training data and new object detection tasks. COCO provided six new methods of calculating AR and mAP at different IoU thresholds and object sides. The metrics are mAP at IoU = 0.5 which is IoU of BBs needs to be above 0.50, mAP at IoU = 0.75 which IoU of BBs needs to be above 0.75, mAP at IoU = 0.50: 0.05: 0.95, mAP for small objects that have an area below 32^2 px, mAP for medium objects that have an area between 32^2 and 96^2 px, mAP for large objects that have an area above 96^2 px. The most important metric mAP is evaluated by calculating AP at starting IoU = 0.5 to an IoU = 0.95, with incremental steps of 0.05. The results are then averaged.

Other metrics involve AR at different object sizes and number of detections. The metrics are AR for small objects that have an area below 32^2 px, AR for medium objects that have an area between 32^2 and 96^2 px, AR for large objects that have an area above 96^2 px, AR with the number of detections below 100, AR with the number of detections below 10, AR with only one detection.

This would allow better differentiation of models as some datasets have more small objects than others. In this case, precision and recall can be calculated. The average precision (AP) and average recall (AR) curves can be retrieved through multiple calculations and trials for each class, and the area under the curve is the AP value. The mAP for object detection is the average of the AP calculated for all the classes as shown in the following formula:

$$mAP = \frac{1}{|Q_R|} \sum_{q\in Q_R} AP(q) \tag{13}$$

where Q is the number of queries.

3.7. Computer configuration

Colaboratory or Colab is an AI browser-based platform that allows writing and executing notebooks written in python code. It allows us to run our code on a free GPU. The specifications of the machine are shown in Table 6.

4. Model training and results

4.1. Model training

4.1.1. The faster R-CNN

We monitored the training and verification process, and the parameters were adjusted and tuned accordingly. For the feature extraction, we used Inceptionv2, as mentioned earlier, which was pre-trained on the COCO dataset and tuned for our custom dataset; this process of transfer learning greatly reduced the training time. The model began to converge quickly at 5000 iterations. The number of iterations had a great influence on the training effect of the model. As shown in the Figures below, the performance of the model was significantly improving with iterations meaning the model is learning and there is no underfitting. Reaching 30,000 iterations, the curves start to reach a point of stability. A choice of 30,000 iterations steps was perfect in order for the model to generalize and don't overfit with our data. Figs. 11, 12, and 13 Show the Detection Boxes Recall/AR@100 (small), Detection Boxes Recall/AR@100 (medium), and Detection Boxes Recall/AR@100 (large) that represent the average recall for small objects, average recall for medium objects, and average recall for large objects with 100 detections.

Figs. 14, 15, and 16. Show the Detection Boxes Recall/AR@100, Detection Boxes Recall/AR@10, and Detection Boxes Recall/AR@1 that represent average recall with 100 detections, average recall with ten detections, and average recall with one detection.

The above 6 metrics were based on recall or mAR (mean average recall). Detection Boxes Recall/AR@(1,10,100) are mean average recalls sliced by the number of detections in the image. AR@1 means that it will compute the mean average recall across all images with at most one detection (i.e., 0 or 1), across all classes, and all IoU thresholds. For AR@10, it would do the same, but across all images with at most 10 detections (i.e. $0 \leq n \leq 10$). AR@100 is for at most 100 detections. The figures converged and stabilized quickly. AR@10 and AR@100 show the highest value and reached above 0.6 since most of the images contain ten or more detections (objects).

Detection Boxes Recall/AR@100 (small, medium, large) are mean average recalls sliced by the size of the detected bounding box for at most

100 detections. This means that it only takes images with at most 100 detections (most of the images). Sizes of the small is (0, 32*32), medium (32*32, 96*96), large (96*96, 1e5*1e5).

Detection Boxes Precision/mAP, as mentioned earlier, is computing the precision over all images, classes, and IoU thresholds and then taking the average (See Fig. 19).

Detection Boxes Precision/mAP@.50IoU specifies the IoU, so it doesn't go over all IoU thresholds as general mAP. It computes only the average precision at the 0.5 IoU threshold. The idea of this metric is to give you a rough sense of precision, not super strict about the position of bounding boxes (only require at least IoU = 0.5 to count as positive). The mAP@0.5 reached above 80%, according to Fig. 18. Also, Detection Boxes Precision/mAP@.75IoU computed at IoU = 0.75 instead of IoU = 0.5. The idea of this metric is to give a rough sense of precision, being too strict about the position of bounding boxes (requiring at least IoU = 0.75 to count as positive). Fig. 17 shows that it reached above 60%.

Detection Boxes Precision/mAP (small, medium, large) are essentially the same as mAP above but sliced by the size of the bounding boxes. The small one is only computing mAP for bounding boxes that are small (area < 32*32 pixels). Medium is for bounding boxes with 32*32 < area < 96*96. Large is for area > 96*96 (in reality the implementation for large is 96*96 < area < 1e5*1e5). These metrics allow you to get a sense if your model is performing better/worse in specific sizes of bounding boxes. As shown, our Faster R-CNN model performs the best with medium-sized objects (most of our 11 categories) (See Figs 20, 21 and 22).

4.1.2. YOLOv4

Figs. 23 and 24 Show the loss and mAP@0.5, respectively. After 2200 iterations, the average loss drops below 2.0, and mAP stabilized pretty quickly, eventually reaching almost 99%.

4.2. Predictions

The Predictions clearly show that our Faster R-CNN model was accurate with the different crowdedness, backgrounds, combinations, and multiple types, as shown in Figs. 25, 26, and 27. However, it struggles with the detection of small-sized *Medicago sativa*, *Trifolium alexandrinum*, and Trigonella foenun graecum when there are many predictions in an image or they are placed next to relatively larger size seeds, as shown in Fig. 28.

YOLOv4 predictions with all types of seeds, even the small-size types was incredibly accurate. The error rate was approximately zero, as shown in Figs. 29 and 30.

5. Discussion

In previous plant/seed detection studies, it has been observed that the datasets were very small and they focused on detecting a few categories which contributed to the high mAP achieved. In this study, two models were trained and tested for leguminous seed recognition. Their performances were compared in identifying 11 different categories of leguminous seeds, some are very small and have similar colors against different backgrounds and within the same image. As shown in Table 7. the YOLOv4 was the most successful one with a 16.5% increase in mAP than Faster R-CNN, and an inference speed of 47.2 ms compared to 53.1 ms for Faster R-CNN.

Small object detection has always been a research hotspot in the field of object detection. In agricultural production, many seeds are very small and mostly have similar colors, making seed detection a challenge. This research used object detection models for the accurate identification and positioning of small seeds. Mosaic data augmentation in YOLOv4 was utilized in the training stage, which mixed four images into one image. The utilization of Mosaic data augmentation has proven to help with the detection of the smallest leguminous seed types in our dataset and also with the detection of very close seeds with similar colors.

6. Conclusion

In this paper, the leguminous seeds dataset was collected and with the use of transfer learning two deep learning-based models were trained. YOLOv4 proved to improve the accuracy and the runtime with less computation load. The model can be applied to the detection of seed images in a variety of complex backgrounds at multiple scales, and in multi-angle environments. With an error rate of <2% and a running time of <2 s, the YOLOv4 model constitutes an effective tool for the detection of leguminous seeds.

Future work will be focused on building larger datasets and keep testing new algorithms to further optimize the model and improve the mAP and speed of detection. Moreover, current work can be extended to different seed detection, seed disease detection, real-time seed detection and enumeration, and various automated agricultural detection processes.

CRediT authorship contribution statement

Noran S. Ouf: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y.M., 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection.

Engineering applications of neural networks. In: Boracchi, G., Iliadis, L., Jayne, C., Likas, A. (Eds.), Communications in Computer and Information Science. 744. https://doi.org/10.1007/978-3-319-65172-9.

Elshawi, R., Wahab, A., Barnawi, A., Sakr, S., 2021. DLBench: a comprehensive experimental evaluation of deep learning frameworks. Clust. Comput. 24, 2017–2038. https://doi.org/10.1007/S10586-021-03240-4.

Girshick, R., 2015. Fast R-CNN. IEEE International Conference on Computer Vision. IEEE Computer Society, USA, pp. 1440–1448.

Girshick, R., Donahue, J., Darrell, T., Malik, J., 2013. Rich feature hierarchies for accurate object detection and semantic segmentation. Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. 580–587.

Jiao, L., Zhang, F., Liu, F., Member, S., Yang, S., Member, S., 2023. A Survey of Deep Learning-based Object Detection. pp. 1–30.

Kundu, N., Rani, G., Dhaka, V.S., 2021. Seeds classification and quality testing using deep learning and YOLO v5. ACM Int. Conf. Proc. Ser. 153–160. https://doi.org/10.1145/3484824.3484913.

Lawal, M.O., 2021. 123AD. Tomato detection based on modified YOLOv3 framework. Sci. Report. 11, 1447. https://doi.org/10.1038/s41598-021-81216-5.

Li, Y., Wang, J., Wu, H., Yu, Y., Sun, H., Zhang, H., 2022. Detection of powdery mildew on strawberry leaves based on DAC-YOLOv4 model. Comput. Electron. Agric. 202, 107418. https://doi.org/10.1016/J.COMPAG.2022.107418.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C., 2016. SSD: single shot multibox detector. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 9905 LNCS. pp. 21–37. https://doi.org/10.1007/978-3-319-46448-0_2.

Mathew, M.P., Mahesh, T.Y., 2022. Leaf-based disease detection in bell pepper plant using YOLO v5. Sign. Image Video Process. 16, 841–847. https://doi.org/10.1007/S11760-021-02024-Y/FIGURES/12.

S. Ouf, N., 2018. A Review on the Relevant Applications of Machine Learning in Agriculture. IJIREEICE 6, 1–17. https://doi.org/10.17148/IJIREEICE.2018.681

Redmon, J., Farhadi, A., 2018. YOLOv3: An Incremental Improvement.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2015. You only look once: unified, real-time object detection. Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. 2016-December, pp. 779–788.

Ren, S., He, K., Girshick, R., Sun, J., 2017. Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. 39, 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031.

Roy, A.M., Bhaduri, J., 2022. Real-time growth stage detection model for high degree of occultation using DenseNet-fused YOLOv4. Comput. Electron. Agric. 193, 106694. https://doi.org/10.1016/J.COMPAG.2022.106694.

Roy, A.M., Bose, R., Bhaduri, J., n.d. A fast accurate fine-grain object detection model based on YOLOv4 deep neural network. https://doi.org/10.1007/s00521-021-06651-x.

Sarker, I.H., 2021. Machine learning: algorithms, real-world applications and research directions. SN Comput. Sci., 1–21. https://doi.org/10.1007/S42979-021-00592-X.

Wu, W., Le Yang, T., Li, R., Chen, C., Liu, T., Zhou, K., Sun, C. Ming, Li, C. Yan, Zhu, X. Kai, Guo, W. Shan, 2020. Detection and enumeration of wheat grains based on a deep learning method under various scenarios and scales. J. Integr. Agric. 19, 1998–2008. https://doi.org/10.1016/S2095-3119(19)62803-0.

Zhao, Z.Q., Zheng, P., Xu, S.T., Wu, X., 2019. Object detection with deep learning: a review. IEEE Trans. Neural Networks Learn. Syst. 30, 3212–3232. https://doi.org/10.1109/TNNLS.2018.2876865.