

## Article

# FruitDet: Attentive Feature Aggregation for Real-Time Fruit Detection in Orchards

Faris A. Kateb <sup>1</sup>, Muhammad Mostafa Monowar <sup>1,\*</sup>, Md. Abdul Hamid <sup>1</sup>, Abu Quwsar Ohi <sup>2</sup>  
and Muhammad Firoz Mridha <sup>2</sup>

- <sup>1</sup> Department of Information Technology, Faculty of Computing & Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia ; fakateb@kau.edu.sa (F.A.K.); mabdulhamid1@kau.edu.sa (M.A.H.)  
<sup>2</sup> Department of Computer Science & Engineering, Bangladesh University of Business & Technology, Dhaka 1216, Bangladesh; quwsarohi@bubt.edu.bd (A.Q.O.); firoz@bubt.edu.bd (M.F.M.)  
\* Correspondence: mmonowar@kau.edu.sa

**Abstract:** Computer vision is currently experiencing success in various domains due to the harnessing of deep learning strategies. In the case of precision agriculture, computer vision is being investigated for detecting fruits from orchards. However, such strategies limit too-high complexity computation that is impossible to embed in an automated device. Nevertheless, most investigation of fruit detection is limited to a single fruit, resulting in the necessity of a one-to-many object detection system. This paper introduces a generic detection mechanism named FruitDet, designed to be prominent for detecting fruits. The FruitDet architecture is designed on the YOLO pipeline and achieves better performance in detecting fruits than any other detection model. The backbone of the detection model is implemented using DenseNet architecture. Further, the FruitDet is packed with newer concepts: attentive pooling, bottleneck spatial pyramid pooling, and blackout mechanism. The detection mechanism is benchmarked using five datasets, which combines a total of eight different fruit classes. The FruitDet architecture acquires better performance than any other recognized detection methods in fruit detection.

**Keywords:** deep learning; object detection; agriculture; convolutional neural network



**Citation:** Kateb, F.A.; Monowar M.M.; Hamid, M.A.; Ohi, A.Q.; Mridha M.F. FruitDet: Attentive Feature Aggregation for Real-Time Fruit Detection in Orchards. *Agronomy* **2021**, *11*, 2440. <https://doi.org/10.3390/agronomy11122440>

Academic Editors: Thomas Scholten, Karsten Schmidt and Ruhollah Taghizadeh-Mehrjardi

Received: 6 October 2021  
Accepted: 26 November 2021  
Published: 29 November 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Precision agriculture [1] implements the knowledge gained through data processed by machinery and software to deal with uncertainties of agricultural systems. Currently, the uncertainties of agricultural systems are dealt with various approaches, such as weather data, field sensor data, vision data, and so forth. Precision agriculture is required to achieve sustainability to fulfill the requirements of the current population growth and effectively handle the food-to-land demand. Therefore, massive harvesting is often conducted by farmers to fulfill the current agricultural needs. Moreover, modern agriculture seeks to manage crops in controlled environments, which maintains specific environmental properties enabling fast growth of agricultural products.

Robotic harvesting is being implemented to reduce labor costs that also offer a faster and autonomous harvesting process. Autonomous harvesting systems often require vision systems to pinpoint the targeted objects, plants, or crops. Further, some agricultural perspectives require specializations, such as identifying diseases, identifying crops, estimating crops in orchards, etc. Moreover, developing an autonomous harvesting system requires solving various vision-related problems: (a) is the fruit seen or not, (b) pinpointing fruit location, (c) is the fruit ready to be picked or not, (d) estimating the fruit load, (e) diseased or not, and so on.

Due to the accomplishments of deep learning, computer vision is in a period of rapid advancement. Hence, deep learning-based computer vision is being implemented to solve numerous challenges in agricultures including leaf disease segmentation [2,3],

weed detection [4], three-dimensional reconstruction of fruit [5,6], and fruit detection [7]. Moreover, computer vision is being drastically applied in agriculture to detect fruits from orchards, solving the challenges mentioned earlier. Detection systems are observed to be implemented for detecting individual fruits such as mango [8], tomato [9], apple [10], kiwifruit [11], strawberry [12], and so on. Apart from the particular fruit detection systems, efforts have been made to implement a general fruit detection system that can detect fruits from the general environment [13] as well as the orchard environment [14].

Although there are general object detection systems [15,16] which can detect a wide range of objects, agricultural detection systems face some specific challenges. Often, fruits that appear in agricultural environments are small, causing the available detection systems to struggle. Further, agricultural detection systems often have to detect more objects per image than general object detection systems. Moreover, the detection system has to be lightweight, resulting in a real-time detection system. Being such a challenging task, fruit detection systems in orchards generally target specific fruits for achieving better performance [9–11]. Therefore, implementing a system that works as a generic fruit detection is required.

This paper introduces an architecture based on deep learning that is specifically designed for agricultural large-scale fruit detection. The proposed model is dubbed as FruitDet, defining “fruit detection”. The model is built upon one-stage detection based upon the YOLO pipeline. The proposed architecture is faster and robust. Therefore, the proposed method is a suitable fruit detection system for automated machinery.

The overall contribution of the paper includes:

- The paper introduces a faster and robust fruit detection system, FruitDet, capable of detecting single and multiple fruits from a single model.
- The paper proposes a modified attention mechanism for better stability of the detection model.
- The paper introduces an attentive pooling block, which combines the attention model and a max-pooling layer. The attentive pooling block can flow prioritize and down-sample the essential features.
- The paper proposes a blackout regularization, which provides better detection capability by neglecting the object size to head detection mapping.

The rest of the paper is segmented as follows: Section 2 represents the relevant works carried out on the fruit detection domain. Section 3 defines the methodology of the proposed FruitDet. Section 4 evaluates the FruitDet architecture by benchmarking. Finally, Section 5 concludes the paper.

## 2. Related Work

Deep learning strategies are overflowing the object detection domain of computer vision due to their robustness. The current implementation of object detection systems is segmented into two different scenarios: two-stage and one-stage object detectors. Two-stage detectors [17] implement detection and prediction in a different module, which often causes a time complexity bottleneck in implementation perspectives. In contrast, one-stage detectors (also called single-shot detection [18]) perform both detection and classification in a single module.

However, from a user perspective, a one-stage detection system may not achieve real-time inference speed due to hardware limitations. Thus, a one-stage detection system often has to balance the exactness and time-complexity tradeoff from architectural perspectives. EfficientDet [19] architecture achieves better performance in detection yet is slow in inference time. On the other hand, SDD [18] and RetinaNet [16] are insignificantly fast. Nevertheless, they lag due to low accuracy depending on the detection speed. Comparatively, YOLO [15,20] achieves better speed by slightly compensating the detection performance. Object detection systems need to be lightweight and robust. Lightweight systems cause real-time inference and require cost-efficient devices. Amongst the numerous object detection pipelines, YOLO

architectures are currently preferred for faster and comparatively accurate fruit detection systems [21].

Detecting fruits from the orchard environment is critical due to various aspects. The quantity of detection can be massive compared to the general object detection systems. The objects/fruits can be overlapped by environmental factors causing a critical detection situation. Further, objects in orchard situations can be much smaller than the general object detection systems target. Therefore, detecting fruits from orchards requires more specialized engineering concepts than the available detection systems.

The early stage of deep learning-based object detection systems in agriculture was implemented using a two-stage object detection system. Two-stage object detection systems are typically robust. Therefore, researches were conducted on multifruit detection systems using two-stage detectors. Sa et al. [13] used faster RNN to detect multiple fruits using faster region-based CNN (Faster R-CNN) [17] from two different modalities: RGB and Near-Infrared (NIR). The authors inherited a transfer learning strategy to train the classifier and gain better detection results in their dataset. Bargoti et al. [14] introduced their dataset and implemented Faster R-CNN to detect three fruits (apple, almond, and mango) simultaneously. However, although the Faster-RCNN produced better results on detecting mango and apple, it generated average accuracy on detecting almonds. Although the Faster-RCNN methods achieved marginal accuracy, they suffer from higher inference time. Therefore, the systems are not perfect for a cost-friendly and real-time robotic harvesting system.

One stage detectors solve the drawback for lightweight and real-time fruit detection systems by negotiating robustness. However, in fruit detection systems, the robustness of one-stage detection methods is competitive to the two-stage detection strategies. Koirala et al. [22] implemented a mango detection system based on the YOLO backend and named it MangoYOLO. The authors designed their YOLO detection mechanism specifically for mango detection and tested it in their produced dataset. Lawla et al. [9] implemented a tomato detection system which is a modified version of YOLOv3 [20]. The authors named the architecture YOLO-Tomato model. The authors benchmarked their proposed architecture in their generated dataset and achieved a marginal detection accuracy. However, the benchmark showed that the current YOLO, YOLOv4 achieves more competitive performance than the YOLO-Tomato model.

Kang et al. [10] introduced the apple detection model LedNet, which is a refined version of RetinaNet [16]. The authors compared their proposed architecture with YOLO family detectors and demonstrated that their proposed architecture performs better in detecting apples. Further, Santos et al. [23] introduced a grape dataset and implemented YOLOv2 [24] and YOLOv3 [20] as a grape detector system. Surprisingly, the older version of the YOLO family, YOLOv2, performed superior in detecting grapes than the improved version YOLOv3. Therefore, the enigma remains. What is the optimal architecture for detecting fruits?

The paper introduces a fruit detection system, FruitDet, that is built upon the YOLO pipeline. The FruitDet architecture is faster, lighter, and can recognize multiple fruits at once. Further, we perform benchmarks in five different datasets to validate the FruitDet architecture to be efficient in the most complex scenarios, solving the enigma for finding one robust detection model.

### 3. FruitDet

A general end-to-end detection system consists of three major segments: (a) backbone, (b) neck, (c) head. The backbone of a detection mechanism performs feature extraction. The extracted features are passed to the neck of the object detector. The neck performs a fusion of the features gathered from the different layers of the backbone model. Finally, the neck passes the feature maps to the head of the detector model. The head finally predicts the classes and bounding box regions which is the final output produced by the object detection model. The head may produce a stack of outputs, mainly designed to recognize

different sizes of objects from an image. Figure 1 illustrates the standard concept of an object detection system.

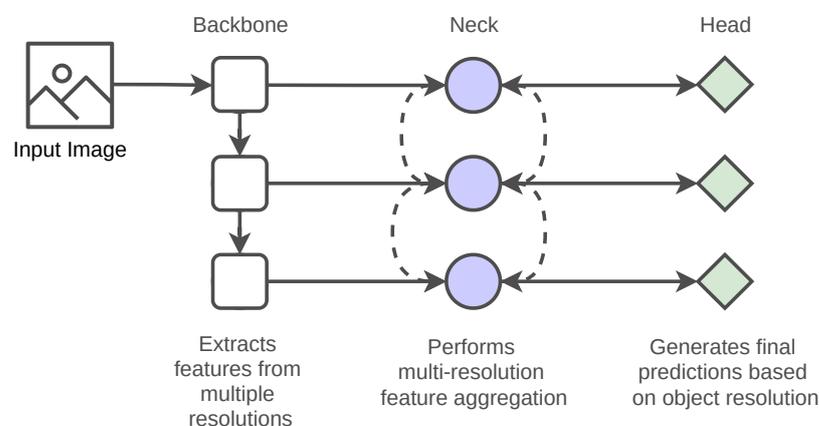
FruitDet architecture follows the general input–output pipeline of the YOLO family. However, the proposed FruitDet architecture contains a modification in the backbone, neck, and head model. Figure 2 illustrates the architectural schema of the FruitDet model. The FruitDet architecture consists of the three general segments of the YOLO family (as illustrated in Figure 1). A lightweight modification DenseNet architecture (fused with attentive pooling mechanism) is implemented as a backbone of the network, discussed in Section 3.1.1. Consequently, feature pyramid network (FPN) along with spatial pyramid pooling (SPP) is used in the neck of the FruitDet architecture, discussed in Section 3.2. Finally, the head of the FruitDet network is implemented using a modified attention module, discussed in Section 3.3.1. The training process of the network consists of a head dropout regularization scheme discussed in Section 3.3.2.

### 3.1. Backbone

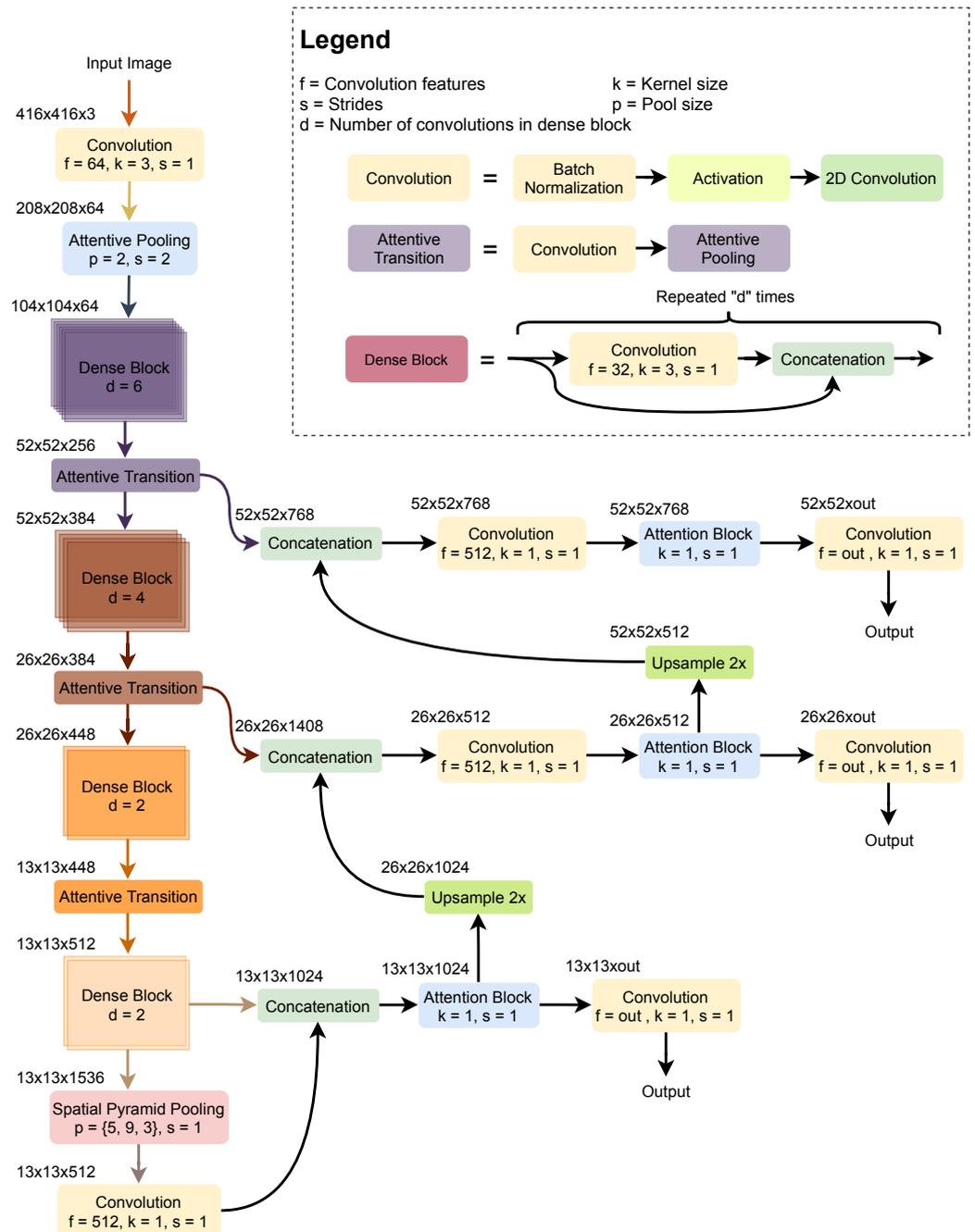
The backbone of a detection model extracts essential features in different resolutions and provides adequate information related to position and object structure. A backbone architecture is generally implemented using a robust classifier model, excluding the final classification layers of that model. However, while selecting a backbone architecture for a real-time detection system, the inference time must also be considered. The current proceedings of the YOLO family, both YOLOv3 and YOLOv4, implements the Darknet53 architecture as a backbone. The Darknet53 was initially proposed in YOLOv3. The DarkNet53 consists of general convolutions with residual blocks. The backbone was faster yet heavily parameterized. Hence, in YOLOv4, the parameter of DarkNet53 architecture is reduced by half using cross-stage spatial blocks (CSP).

FruitDet implements a modified version of DenseNet [25] as a backbone. DenseNet architecture has a general pattern of aggregating the previous features and extracting the spatial information from a set of the previous features. Therefore, DenseNet architecture offers a better propagation of features to the detection mechanism. Besides, as DenseNet architecture can fuse the features of multiple blocks, it alleviates the vanishing-gradient problem.

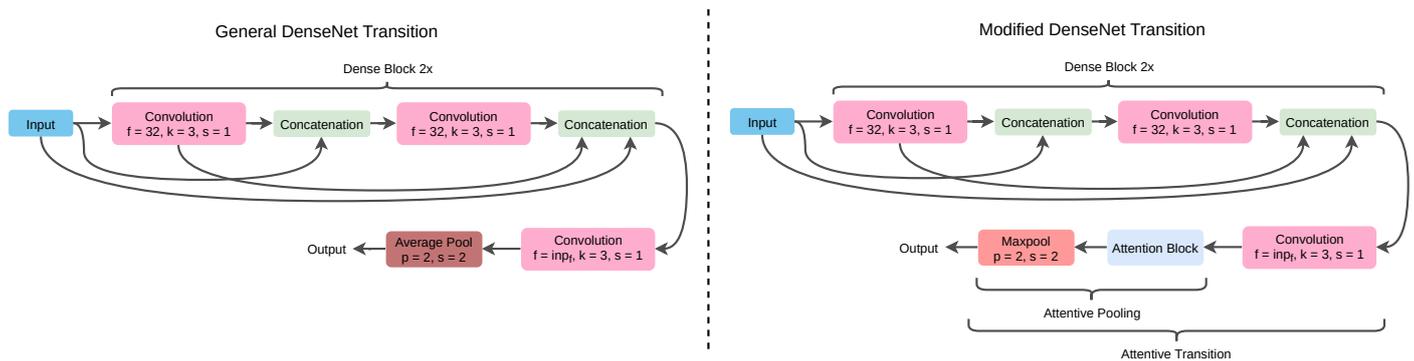
The DenseNet proposed for FruitDet is re-engineered to be smaller and lighter. The traditional DenseNet architecture implements a transition block. In FruitDet, the general transition block is replaced with an attentive transition block (depicted in Figure 3). The DenseNet backbone and attentive transition is discussed in Sections 3.1.1 and 3.1.2, respectively.



**Figure 1.** A detection model contains a backbone, neck, head module. The backbone module exploits the essential features of different resolutions, and the neck module fuses the features of different resolutions. Finally, multiple head modules perform the detection of objects in different resolutions.



**Figure 2.** The figure illustrates the overall mechanism of the FruitDet detection model. The backbone of the FruitDet model contains a lighter DenseNet architecture with the proposed attentive transition block. The neck of the FruitDet model contains a feature pyramid network (FPN) with a bottleneck spatial pyramid pooling (SPP) method. The head consists of a modified attention module.



**Figure 3.** The figure depicts the construction of a dense block (with two convolutions) followed by different transition modules. The left figure illustrates the transition module of DenseNet. The right figure depicts the proposed attentive transition module. At first, the attentive transition module performs a convolution operation. Further, there exists an attention block, followed by a max pool layer.

### 3.1.1. Densely Connected CNN

Apart from the general implementation of the YOLO family, FruitDet is designed with a DenseNet [25] backbone. The general implementation of DenseNet (as a backbone) is evaluated to be slower than Darknet-53 [20]. Hence, the FruitNet architecture contains handcrafted DenseNet blocks with lesser parameters, resulting in a lightweight detection mechanism.

A dense block consists of multiple convolutions. In general, a dense block with  $n$  convolution performs the following operation:

$$x_n = H_n([x_0, x_1, x_2, \dots, x_{n-1}]) \quad (1)$$

Here,  $x_i$  are the feature maps produced by a non-linear function  $H(\cdot)$ . A dense block contains a growth rate, referring to the number of channels produced by a non-linear function  $H_i(\cdot)$ . The general DenseNet architecture contains a growth rate of 32. The implemented dense block of FruitDet is the same as the general dense block introduced in DenseNet. Nevertheless, the number of dense blocks in FruitDet is significantly reduced. Each of the convolution blocks of DenseNet consists of activation, batch normalization, and convolution block in sequence.

### 3.1.2. Attentive Pooling: Feature Selective Pooling

In the general implementation of DenseNet, each dense block is followed by a transition block that produces a simple non-linear feature map containing a pointwise convolution. Afterward, a transition block also performs  $2 \times 2$  average-pooling to reduce the resolution of the feature map.

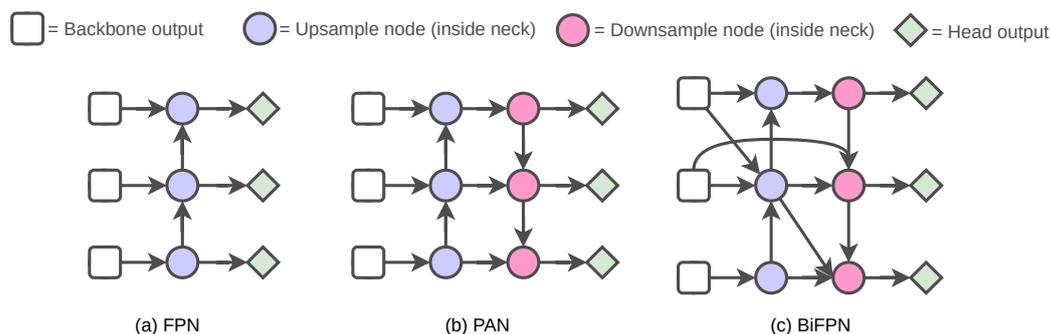
In contrast, FruitDet architecture is packed with an attentive transition module, fusing attention with the transition block. In general, an attention block can prioritize a feature by increasing the activation value and vice-versa. Before performing the pooling operation, we found that adding an attention block improves the robustness of the FruitDet architecture. As the attention layer can prioritize features, it focuses on passing important features via the pooling layers. Moreover, we perform max-pooling instead of performing average-pooling suggested in DenseNet. To validate such an implementation, we theorize an example. Let  $x_n$  be a set of feature map values  $x_n = \{x_0, x_1, \dots, x_{n-1}\}$ . Performing a max-pool to choose  $\frac{n}{2}$  values would select the high confidence features. Further passing  $x_n$  through an attention layer would guarantee the max-pool layer choosing the best feature maps. In contrast, performing an average-pooling would cause mixing the high-priority feature maps with low priority feature maps. Hence, average-pooling is avoided in the attentive transition block.

Figure 3 illustrates the schema of a dense block followed by the attentive transition. The attentive transition first performs a convolution, followed by the attentive pooling mechanism. The attentive pooling mechanism contains an attention block and a max-pooling layer. The attention mechanism implemented in FruitDet is discussed in Section 3.3.1.

### 3.2. Neck

The neck of object detection systems is observed to be implemented using different pyramid network [26] approaches. Pyramid networks are observed to be better at performing feature fusion from different layers of the backbone model. In general, there exist three types of hand-crafted pyramid networks: feature pyramid network (FPN) [26] implemented in YOLOv3, path-augmented network (PAN) [27] implemented in YOLOv4, and bi-directional feature pyramid network (BiFPN) [19] introduced in EfficientDet. The internal workings of the pyramid networks are illustrated in Figure 4. The neck of FruitDet is implemented using the FPN. Adding FPN in FruitDet resulted in better performance, being lightweight and faster than the rest of the pyramid networks. Besides the FPN, adding other pyramid networks would cause an extra computation burden, resulting in a comparatively slower detection system in low-cost devices.

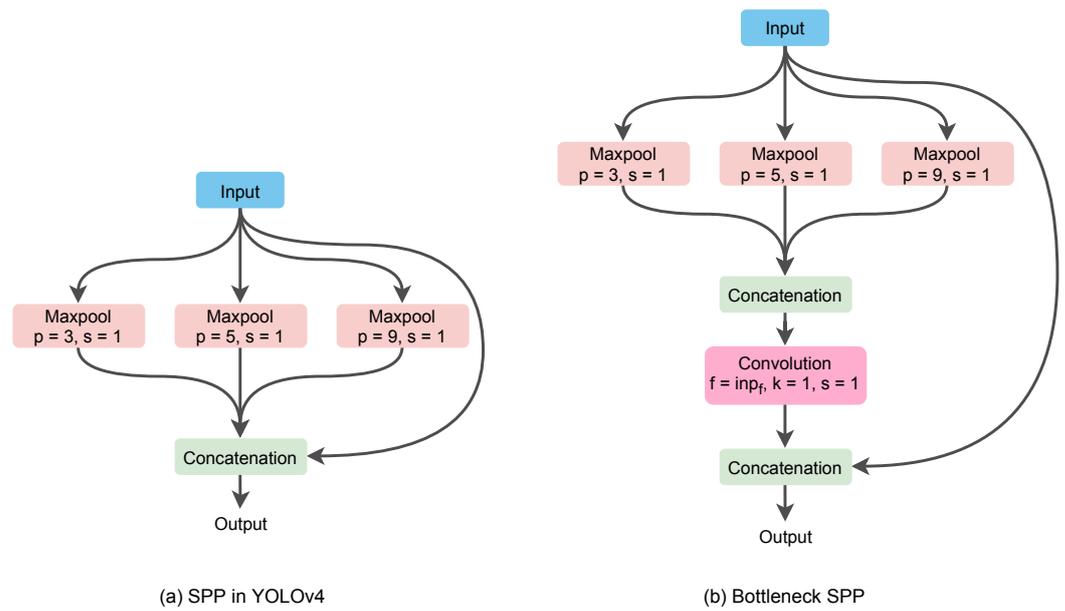
Apart from the pyramid network in the neck of the detection system, YOLOv4 adds Spatial Pyramid Pooling (SPP) [28]. SPP has been demonstrated to be better at recognizing numerous features, ignoring the size of the appearance. Inspired by the analogy of SPP in object detection, FruitDet architecture implements a scarce modification of SPP block, which is discussed below.



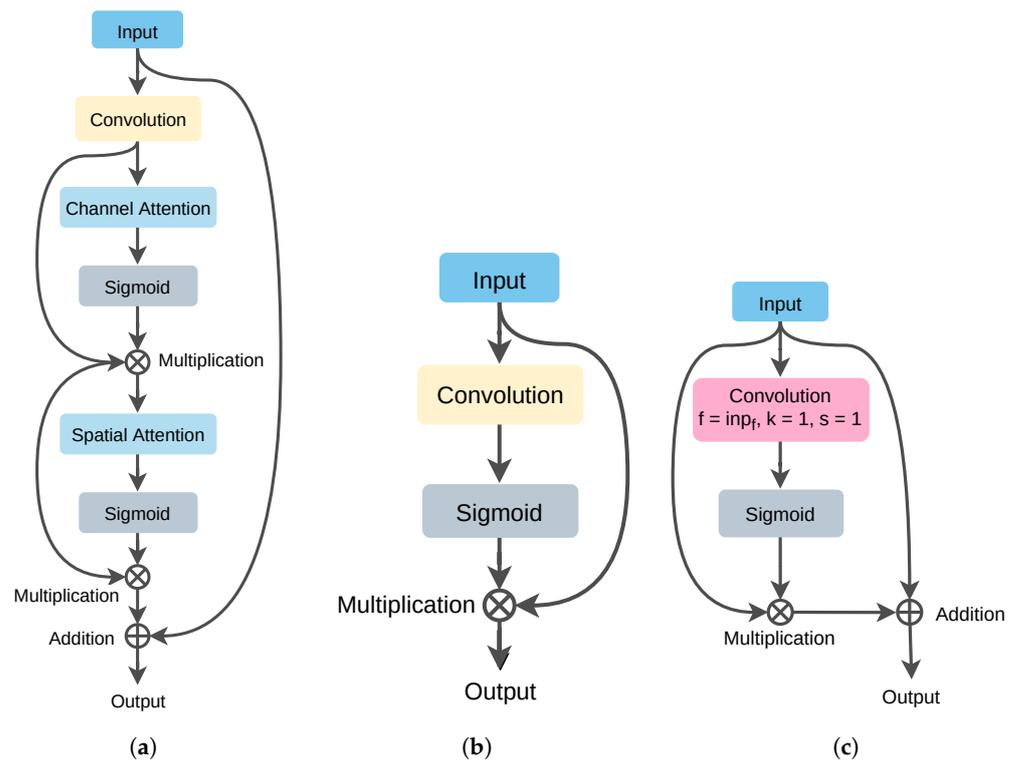
**Figure 4.** The figure illustrates three different implementations of neck architectures in detection models. FPN is implemented in YOLOv3, PAN is implemented in YOLOv4, and BiFPN is introduced in EfficientDet. The neck of FruitDet is implemented using FPN.

#### Bottleneck Spatial Pyramid Pooling

SPP performs a set of max-pool operations comprising different pooling sizes. Different pooling sizes help to identify similar feature maps neglecting the different resolution of feature patterns. Adding SPP in YOLOv4 causes an increase in parameters and complexity. Hence, FruitDet is implemented with a bottleneck SPP block. The proposed bottleneck SPP block concatenates the set of pooling outputs and performs a pointwise convolution operation. The pointwise convolution operation reduced the number of features, and the output result is again concatenated with the actual input of the SPP block. The bottleneck reduces the required parameters after the SPP block and provides a slight performance improvement. Figure 5 illustrates a visual difference between the implementation of SPP (in YOLOv4) and bottleneck SPP implemented in FruitDet.



**Figure 5.** The figure illustrates the comparison of the SPP block (implemented in the YOLO family) and the proposed bottleneck SPP block. The bottleneck SPP reduces the feature dimension resulting in a reduction in trainable parameters and complexity.



**Figure 6.** The figure depicts a comparison of attention modules. CBAM [29] is an attention module combining channel and spatial information. SAM [15] is used as an attention module in YOLOv4. Finally, the proposed attention module for FruitDet is illustrated. (a) CBAM [29] attention module. (b) SAM attention module. (c) Attention module in FruitDet.

### 3.3. Head

The head of the object detection model performs the final prediction of the bounding boxes and class scores. End-to-end object detection systems contain multiple heads to detect objects of different resolutions accurately. In general, the YOLO family contains three heads, whereas EfficientDet contains three to five heads. Based on mapping object detection to different heads based on the resolution, a new object detection to head mapping policy is proposed. The policy is introduced in Section 3.3.2.

The head construction contains slight differences between YOLOv3 and YOLOv4. YOLOv4 contains a Spatial Attention Module (SAM) [15] block. The SAM block is a generalized version of the Convolutional Block Attention Module (CBAM) [29], which is less parameterized and requires less computational cost. Apart from the general implementation of the attention module, FruitDet is introduced with a modified version of the attention block discussed below. Moreover, a comparison of the attention modules is given in Figure 6.

#### 3.3.1. Modified Attention

The general implementation of the SAM block contains a non-linear function  $H_l$ , which is used for extracting spatial relationships. The output of the non-linear function is passed through a sigmoid function, resulting in the output to be in scale  $[0, 1]$ . The sigmoidal output is further multiplied with the actual input of the non-linear function. The process can be mathematically illustrated as:

$$x_{out} = \sigma(H_l(x_{in})) \times x_{in} \quad (2)$$

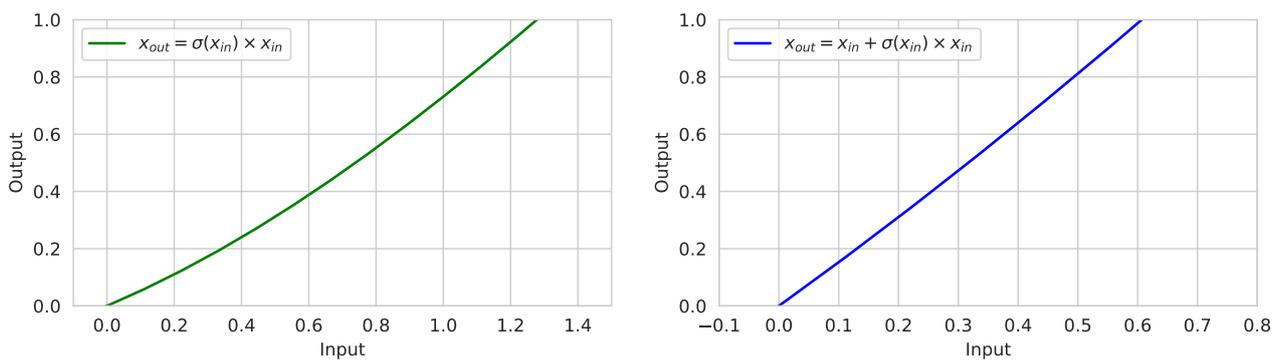
Here,  $x_{in}$  is the input feature map and  $x_{out}$  is the output feature map after applying the attention module. The function  $\sigma(H_l(\cdot))$  is the attention function that prioritizes spatial features of the input. Higher priority features receive values close to 1, and low priority features receive values close to 0.

The flaw of the SAM attention module is that if it fails to prioritize a vital feature map, the overall output of the detection model can be erroneous. Furthermore, the SAM attention module can hardly produce the same results as the input. The sigmoid activation function requires a higher value as an input to achieve the output value of 1. Therefore, SAM blocks might struggle to prioritize vital feature maps.

To solve such a problem, FruitDet is implemented with a modified attention module that can be mathematically represented as follows:

$$x_{out} = x_{in} + \sigma(H_l(x_{in})) \times x_{in} \quad (3)$$

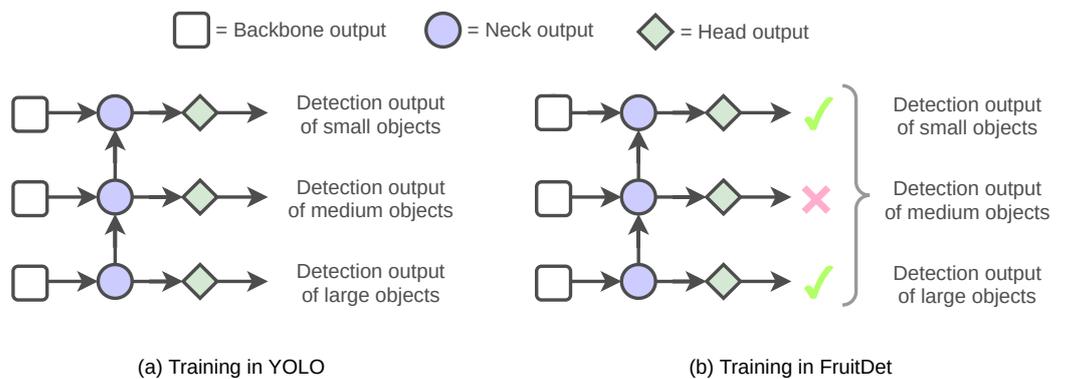
The proposed attention module performs an addition operation with the input of the attention module. Unlike the SAM module, the proposed attention module cannot directly vanish vital spatial feature maps, and even the attention module fails to identify them. Figure 7 depicts a difference between the SAM module (Equation (2)) and the proposed attention module (Equation (3)). The proposed attention module can achieve the value of one faster than the SAM module. Therefore, the proposed attention model does not push the detection architecture to produce high-valued feature maps for activating the attention function. Hence, the attention block achieves a better flow of information from the neck to the head of the FruitDet model.



**Figure 7.** The left and right figures visualize the SAM module and the proposed attention module, respectively. The x-axis is the input of the functions, whereas the y-axis indicates the output values. The SAM module requires a higher input value to produce an output of 1. In contrast, the proposed attention module can achieve an output of 1 by getting almost half of the input of the SAM module.

### 3.3.2. Blackout: A Head Dropout Mechanism

In the training process of FruitDet, a head dropout policy is introduced. The head dropout policy randomly drops one or two head outputs by multiplying the outputs by zero. The head dropout policy forces the detection model to detect objects from the rest of the head models, which results in a more confident detection output. Let  $h_1, h_2,$  and  $h_3$  be the heads, where an object  $p$  is only detected by  $h_2$ . Using the head dropout policy, if the output of the  $h_2$  is multiplied by zero, the object would not be detected by the detection model. Therefore, during backpropagation, the detection model would update the weights of head  $h_1$  and  $h_3$  to correctly detect object  $p$  via the other heads (excluding  $h_2$ ). Therefore, the head dropout policy improves the probability of detecting an object from multiple heads. Hence, it improves the overall performance of the FruitDet model. We name the overall policy of dropping out detection heads as *blackout*. The overall process of blackout is illustrated in Figure 8.



**Figure 8.** The figure illustrates the detection process of multiple heads concerning the object size (illustrated on the left). FruitDet applies a blackout strategy that eliminates the output of some detection heads, forcing other detection heads to detect any size of objects.

### 3.4. Training and Inference

The FruitDet is trained using the loss criteria as same as YOLOv4. The loss function aggregates three distinct features: positional loss, confidence loss, and class loss. The positional loss defines the quality of overlap of the detected object by the FruitDet concerning the actual object. The confidence loss defines the existence of an object for a specific bounding box. The class loss defines the correctness of the detected class in the case of multi-fruit classification. The positional loss is implemented using Ciou loss [30]. The overall loss function is derived below:

$$\begin{aligned}
Loss &= L_{position} + L_{confidence} + L_{class} \\
L_{position} &= 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v \\
v &= \frac{4}{\pi^2} \left( \arctan \frac{w^{gt}}{h^{gt}} - \arctan \frac{w}{h} \right)^2 \\
\alpha &= \frac{v}{1 - IoU + v} \\
L_{confidence} &= - \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{obj} CE_{binary}(\hat{C}_{ij}, C_{ij}) - \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{noobj} CE_{binary}(\hat{C}_{ij}, C_{ij}) \\
L_{class} &= \sum_{i=0}^{S^2} I_{ij}^{obj} CE_{categorical}(\hat{P}_{ij}, P_{ij})
\end{aligned} \tag{4}$$

Here,

$S^2$  = 2D detection grid of FruitDet head

$B$  = Number of candidate boxes in each head ( $B = 3$ )

$I_{ij}^{obj}$  = Object mask, take 1 if object exists for a grid position  $(i, j)$ , 0 otherwise

$I_{ij}^{noobj}$  = Object mask, take 1 if object does not exist for a grid position  $(i, j)$ , 0 otherwise

$\rho(\cdot)$  = Euclidian distance

$\hat{C}_{ij}, C_{ij}$  = Confidence of object in ground and predicted output, respectively

$\hat{P}_{i,j}, P_{i,j}$  = Category probability of ground and predicted output, respectively

$b^{gt}, w^{gt}, h^{gt}$  = Center coordinates, width, and height of actual object

$b, w, h$  = Center coordinates, width, and height of the predicted object

The YOLOv4 architecture uses CIOU loss for detecting large-scale objects. CIOU loss is a refined version of the DIOU loss that adds an extra penalty based on the aspect ratio difference between true/actual and predicted objects. The additional aspect ratio penalty is added by the term  $\alpha v$  while detecting  $L_{position}$ .

The training of FruitDet is conducted implementing the blackout policy. Further, general augmentation strategies including zooming, flipping, rotating, random crop, brightness, and contrast shifting were conducted. In the inference, the blackout policy is removed.

#### 4. Experimental Analysis

An object detection system not only classifies an object but also pinpoints the object's location using a bounding box (rectangle), which covers the area of an object. A detection system is trained using a training dataset containing ground truth values. The ground truth values comprise bounding boxes for a given image. Moreover, each bounding box contains the class labels indicating the object name that a particular bounding box covers.

##### 4.1. Datasets

Five datasets have been used to evaluate the robustness of FruitDet. In the experiments, both single-class and multi-class datasets have been implemented for better evaluation. Each dataset contains a predefined, train-test split by the dataset producers, which was directly used in the evaluation. Figure 9 illustrates some of the image examples of the five datasets. The figure explains that the number of detectable objects in an orchard situation is substantially higher than in general environments.

Table 1 illustrates the distribution of fruit counts in the train and test portion of each dataset. The datasets have unique characteristics which help the evaluation to be more

extensive and proper. The DeepFruit [13] dataset contains an insufficient number of images available for training. Therefore, the dataset is suitable for evaluating multi-class object detection considering the scarcity of data. In contrast, the MultiFruit [14] dataset contains a large number of objects. However, the image and color quality of MultiFruit is challenging, and it contains images in both daylight and night-time. The WGISD [23] dataset contains grape cluster images, which are distributed into four classes. The four classes of grapes are mostly similar, which makes the classification task critical.

Both MineApple [31] and MangoYOLO [22] are single-class object detection datasets. The MangoYOLO dataset contains images in critical low-light environments. The MineApple dataset contains a large number of detectable objects in a single image frame (Figure 9a). WGISD contains different grape classes, which are easy to localize yet harder to classify the correct grape type (Figure 9c). The DeepFruit dataset contains data in different light conditions and also represents a scenario of data scarcity (Figure 9(e.1–e.3)). The MultiFruit dataset contains diverse lighting (day/night) with low-quality images (Figure 9(b.1,b.2)). Overall, the five datasets target numerous challenges of the object detection platform, which are observed explicitly in fruit detection from orchards.



**Figure 9.** The figure illustrates snapshots of five different datasets: (a) MineApple [31], (b) MultiFruit [14], (c) WGISD [23], (d) MangoYOLO [22], and (e) DeepFruit [13]. Some datasets contain numerous objects, complex classification scenarios, diverse lighting, and day/night-time photography.

**Table 1.** The table illustrates the number of objects present in the training and testing portion of the datasets. The number of objects indicates the total number of objects found in the overall subset of the dataset.

Dataset	Classes	Train		Test	
		Counts	Total	Counts	Total
DeepFruit [13]	Apple	294		65	
	Avocado	138		40	
	Mango	905	1952	240	489
	Orange	221		53	
	Rockmelon	123		14	
	Strawberry	271		77	
MangoYOLO [22]	Mango	12,681	12,681	2600	2600
MultiFruit [14]	Almond	3980		797	
	Apple	5211	15,309	554	2298
	Mango	6118		947	
MinneApple [31]	Apple	24,539	24,539	3643	3643
WGISD [23]	Chardonnay	660		180	
	Cabernet Franc	910		159	
	Cabernet Sauvignon	532	3582	111	850
	Sauvignon Blanc	1034		283	
	Syrah	446			117

#### 4.2. Evaluation Metrics

To identify if a detection system correctly determines an object, a two-step process is followed. Firstly, the ratio of overlap of the predicted bounding box (predicted by the detection system) and the actual/ground bounding box is calculated. Intersection over union (IOU) is used to measure the ratio of overlap between the predicted and ground bounding box. If the IOU score is above a certain threshold, it is assumed that the detection system has pinpointed an object correctly. Secondly, the class/label of the corresponding predicted and the ground bounding box is matched. An object is assumed to be correctly detected by the detection system if both constraints are fulfilled. In the experiments, a default IOU threshold of 0.2 is used. IOU = 0.2 is selected through a grid search approach over all datasets and baseline models (YOLOv3, YOLOv4, MangoYOLO, and FruitDet). For IOU = 0.2, most of the model performs to its best.

An object detection system can be evaluated using numerous metrics. Among the various metrics, the mean average precision (mAP) is widely used in benchmarking famous object detection pipelines [32]. mAP is the mean of the average precision (AP) metric. Average precision is the numeric representation of the area under the curve of the precision-recall graph of an object detection system, for a given set of queries, with a particular IOU threshold. We refer to [32] for the clear mathematical concept of AP and mAP metrics.

Further, in the case of agricultural detection systems, the precision score is also preferred [22]. Precision evaluates the ability to identify relevant objects by a detection system. Therefore, mAP and precision are used to evaluate and compare FruitDet with respective detection models.

#### 4.3. Experimental Setup

The introduced and compared detection mechanisms are implemented in Python. Tensorflow [33] and Keras [34] frameworks are used to implement deep learning models. The training of deep learning models included general augmentation techniques such as flip, rotate, brightness, and contrast manipulations. The augmentations are performed using Albumentation [35]. In the overall training, a default batch size of 8 is used. The

YOLO detection models are observed to work better using cosine decay [36] that resets the learning rate after a particular epoch. Instead, while training, the learning rate was manually reset for better convergence. The learning rate was decayed by a factor of 0.5 if the loss does not reduce by the previous three epochs. The input image for FruitDet and the rest of the image models are set to be  $416 \times 416$ , which is the default input image shape for YOLO detection models. The training was halted when the mAP score of the model did not improve within 50 epochs.

YOLO architectures depend on anchor points, an offset value (height and width) assigned to the head model's output. In general, each head of the YOLO architecture produces three outputs per grid pixel. Therefore, there are total  $3 \times 3$  anchor positions for the YOLO architectures. The anchor offset values are dataset-dependent and may provide poor results if not properly calibrated. Hence, in the overall experiment, the anchor values are set to  $\{(30, 32), (33, 51), (34, 39), (39, 47), (44, 41), (46, 50), (48, 58), (55, 58), (61, 69)\}$ . The values are generated using k-means clustering, performed on the five datasets (mentioned in Section 4.1). The updated anchor values are also used for the other baseline architectures introduced in the comparison.

#### 4.4. Result Analysis

In the analysis, six versions of the FruitDet architecture are used to determine the necessity of some modules. By FruitDet, the general architecture of FruitDet, proposed in the paper, is indicated. The experiment contains versions of FruitDet architecture by excluding blackout, attention, and attentive pooling system. In addition, to identify the integrity of FruitDet architecture with FPN, FruitDet architecture is also implemented with a PAN neck structure. Further, to test the robustness of the DenseNet backbone, two other backbone networks, EfficientNet [37] and DarkNet53 [20], are attached with FruitDet, presented in the benchmarks. Apart from FruitDet versions, two pioneers of the YOLO family, YOLOv3 and YOLOv4, are used for comparison. MangoYOLO architecture is a detection model based on YOLO, explicitly engineered for detecting mangoes on the MangoYOLO dataset. MangoYOLO architecture is also implemented and presented in the comparison. Table 2 explains the acronyms for different models used in the benchmarks.

**Table 2.** The table explicates the different model acronyms used in the comparison.

Model	Construction
FruitDet	The actual FruitDet model with attentive pooling, FPN, attention, and blackout.
FruitDet-Blackout	The FruitDet architecture without blackout.
FruitDet-Att	FruitDet architecture excluding attention module.
FruitDet-AttPool	FruitDet architecture excluding attentive pooling module.
FruitDet+SAM	FruitDet architecture with SAM attention module.
FruitDet+PAN	FruitDet architecture with PAN network.
DarkNet+FPN	FruitDet architecture with DarkNet53 as backbone and FPN in the neck.
DarkNet+PAN	FruitDet architecture with DarkNet53 as backbone and PAN in the neck.
EfficientNet+PAN	FruitDet architecture with EfficientNetB0 as backbone and PAN in the neck.
EfficientNet+FPN	FruitDet architecture with EfficientNetB0 as backbone and FPN in the neck.
MangoYOLO	The default MangoYOLO architecture.
YOLOv4	The default YOLOv4 architecture.
YOLOv3	The default YOLOv3 architecture.

Detection models require a vast amount of data, which is often costly and time-consuming. In the case of fruit detection models, data scarcity can also be a challenge, as the orchard scenario tends to have more objects than usual. The DeepFruit dataset represents the scenario mentioned above. Table 3 represents a benchmark on the DeepFruit dataset, presenting the average precision and precision of each of the six classes present in the dataset. Moreover, mAP and average precision are presented for a better comparison. Amid data scarcity, FruitDet architecture significantly outperforms any other detection mechanisms in per-class average precision and precision. Apart from the other implementations, FruitDet architecture is also compared with its variants: FruitDet without the

attention module and FruitDet without the attentive pooling modules. It can be observed that the precision of FruitDet architecture is worst if the attention module is left off. Further, if the attentive pooling layer is replaced by an average pooling module, the FruitDet architecture gives a better mAP and precision value. The improvement by adding attention and attentive pooling module validates both implementations' outstanding contributions in the FruitDet architecture.

**Table 3.** The table represents the benchmark conducted on the DeepFruit dataset. Average Precision and Precision are presented for each class of the DeepFruit dataset. Further, mean average precision (mAP) and mean precision are reported for better comparison. Best scores are marked bold.

Dataset	Apple		Avocado		Mango		Orange		Rockmelon		Strawberry		mAP	Mean Prec.
	AP	Prec.												
FruitDet	73.53	61.96	<b>68.78</b>	<b>59.26</b>	76.38	80.61	52.15	<b>77.50</b>	53.05	<b>47.37</b>	69.22	64.49	<b>65.52</b>	<b>71.30</b>
FruitDet+SAM	<b>80.78</b>	56.19	55.06	49.15	<b>83.50</b>	<b>81.95</b>	51.64	68.18	43.61	34.78	72.89	<b>71.88</b>	64.58	69.65
EfficientNet+PAN	75.65	60.23	53.70	48.15	73.90	68.28	<b>61.76</b>	64.71	54.86	45.00	71.28	60.18	65.19	62.82
EfficientNet+FPN	68.78	52.94	52.81	53.06	67.69	69.61	58.18	66.67	<b>55.97</b>	60.00	67.48	57.26	61.82	62.72
FruitDet-Blackout	73.10	59.18	64.24	51.67	68.53	74.21	51.80	75.12	48.40	46.72	70.42	64.28	62.75	61.86
DarkNet+PAN	75.29	60.47	35.27	45.24	62.56	64.49	48.37	50.00	11.11	11.11	64.93	71.25	49.59	60.22
FruitDet-AttPool	80.58	61.54	40.66	31.67	70.76	63.32	56.46	46.67	21.59	28.57	78.36	67.01	58.07	58.08
FruitDet+FPN	79.25	42.86	56.54	40.51	79.16	65.45	56.05	58.33	56.91	28.57	73.26	51.39	66.86	54.09
DarkNet+FPN	65.89	49.02	32.48	33.33	54.81	57.88	46.34	49.09	20.86	18.52	46.82	63.01	44.53	52.45
FruitDet-Att	73.16	48.18	41.12	39.62	59.53	58.82	40.28	22.43	9.09	11.11	<b>78.42</b>	69.15	50.27	50.86
MangoYOLO	46.90	39.17	18.99	47.06	37.66	50.59	17.68	20.93	1.14	12.50	16.77	34.18	23.19	40.67
YOLOv4	41.27	20.54	0	0	25.64	30.26	11.98	13.62	0	0	15.39	22.91	15.71	23.11
YOLOv3	60.76	<b>69.49</b>	27.27	1.00	15.58	52.70	22.53	40.74	0	0	27.27	66.67	25.57	59.80

Apart from FruitDet, the MultiFruit and WGISD dataset contain a greater quantity of trainable objects, sufficient for training a fruit detection model. Tables 4 and 5 represent the benchmark conducted on MultiFruit and WGISD datasets, respectively. For both datasets, FruitDet architecture performs superior in the benchmark. In case of MultiFruit dataset, apart from the FruitDet architecture, the rest of the detection mechanism performs competitive mean precision results. However, considering mAP, FruitDet architecture achieves a quality detection score on a large margin. In the WGISD dataset, the FruitDet family performs superior to any other detection model presented in the benchmark. Considering both of the comparisons (MultiFruit and WGISD), it can be validated that FruitDet also performs better if trained on sufficient data.

**Table 4.** Benchmark conducted on MultiFruit dataset is presented in the table. Best scores are marked bold.

Dataset	Almond		Apple		Mango		mAP	Mean Prec.
	AP	Prec.	AP	Prec.	AP	Prec.		
FruitDet	58.87	<b>83.02</b>	<b>84.02</b>	<b>93.10</b>	<b>81.08</b>	88.31	<b>74.66</b>	<b>88.14</b>
YOLOv3	55.84	82.12	77.01	92.51	74.57	88.43	69.14	87.40
MangoYOLO	53.68	80.88	77.81	91.30	72.92	85.21	68.14	85.39
YOLOv4	58.26	76.42	77.82	92.94	73.95	88.52	70.01	85.27
FruitDet-Blackout	58.67	81.40	83.18	91.33	80.41	88.10	74.09	86.94
FruitDet-Att	58.54	77.12	76.70	91.15	72.36	<b>86.88</b>	69.20	84.53
FruitDet-AttPool	<b>59.17</b>	78.59	76.06	91.22	78.43	84.74	71.22	84.27
DarkNet+PAN	57.09	75.57	76.69	91.20	68.67	81.28	67.49	81.57
FruitDet+SAM	58.61	74.34	81.93	87.24	77.63	83.77	72.72	81.28
EfficientNet+FPN	57.27	75.21	74.47	88.08	68.46	82.42	66.73	81.18
FruitDet+PAN	58.42	70.94	82.44	88.65	79.75	84.15	73.54	80.30
EfficientNet+PAN	54.78	70.57	71.99	82.62	70.56	82.27	65.78	78.14
DarkNet+FPN	53.67	73.05	76.60	88.95	71.81	75.26	67.36	77.81

Apart from the one-to-many object detection mechanism, Tables 6 and 7 present benchmarks on single-class object detection datasets. Table 6 presents a benchmark on the MangoYOLO dataset and Table 7 presents a benchmark on the MineApple dataset. FruitDet architecture slightly outperforms MangoYOLO architecture, which is specifically designed for the MangoYOLO dataset. Further, in the MangoYOLO dataset, YOLOv3 and YOLOv4 produce a competitive score but not better than the score of FruitDet. In contrast, on the MineApple dataset, FruitDet architecture outperforms the existing detection models in a larger margin, validating a superiority in single-class object detection.

**Table 5.** Benchmark conducted on the WGISD dataset is presented in the table. Best scores are marked bold.

Dataset	Chardonnay		Cabernet Franc		Cabernet Sauvignon		Sauvignon Blanc		Syrah		mAP	Mean Prec.
	AP	Prec.	AP	Prec.	AP	Prec.	AP	Prec.	AP	Prec.		
FruitDet	75.35	60.00	<b>70.15</b>	<b>74.07</b>	<b>60.61</b>	58.06	<b>75.35</b>	<b>80.00</b>	<b>70.35</b>	77.68	<b>70.36</b>	<b>75.63</b>
FruitDet-AttPool	<b>77.25</b>	73.63	66.57	71.08	59.67	68.52	74.33	77.15	68.71	81.82	69.31	74.75
FruitDet-Att	67.16	<b>74.07</b>	68.95	70.06	59.25	71.28	63.17	71.34	70.03	<b>85.86</b>	65.71	73.35
EfficientNet+PAN	67.19	75.28	57.48	64.33	50.66	57.14	67.00	80.40	59.67	82.11	60.40	72.56
FruitDet+SAM	77.24	83.24	67.45	74.10	60.18	59.15	73.13	78.69	62.36	64.07	68.07	71.85
EfficientNet+FPN	63.03	61.95	65.78	64.57	42.25	63.53	65.56	73.40	49.73	66.99	57.27	67.06
FruitDet+PAN	54.68	55.29	51.13	62.59	43.36	67.61	55.07	65.35	35.15	68.85	47.88	62.66
DarkNet+PAN	66.87	63.64	67.59	55.14	47.18	43.06	64.83	70.77	60.21	50.94	61.34	58.96
MangoYOLO	48.56	46.43	48.94	60.29	27.27	83.78	48.22	58.66	35.89	78.00	41.78	58.25
DarkNet+FPN	64.86	48.54	66.28	51.75	44.24	65.06	65.54	65.55	62.52	63.72	60.69	57.73
YOLOv4	39.17	39.82	45.63	53.05	26.65	<b>75.00</b>	55.64	49.47	35.59	43.85	40.53	48.08
YOLOv3	12.41	22.64	1.30	9.09	0	0	1.54	16.98	9.09	61.54	4.87	18.21

**Table 6.** Benchmark conducted on the MangoYOLO dataset is presented in the table. The reported average precision and precision are utilized on a single class (mango). Best scores are marked bold.

Dataset	AP	Prec.
FruitDet	<b>81.50</b>	<b>99.18</b>
FruitDet-noBlackout	81.28	99.14
MangoYOLO	81.15	98.96
FruitDet-Att	81.14	98.94
YOLOv4	81.20	98.62
YOLOv3	80.90	98.60
EfficientNet+FPN	80.88	98.57
FruitDet-AttPool	80.97	98.52
DarkNet+FPN	80.73	98.33
FruitDet+SAM	80.75	98.22
DarkNet+PAN	80.79	98.04
FruitDet+PAN	80.54	97.88
EfficientNet+PAN	79.51	96.81

Aggregating the benchmarks presented in Tables 3–7, FruitDet achieves superior performance in all of the datasets. FruitDet architecture performs robustly independent of the data quantity and the domain of data. Additionally, the elimination of certain modules causes FruitDet’s performance degradation. Adding SAM instead of the proposed attention module also degrades the performance of FruitDet. The comparison of PAN and FPN in different datasets explains that both methods are competitive in different cases. However, FruitDet maintains a balanced performance using FPN. Finally, implementing FruitDet architecture with EfficientNet and Darknet53 architectures greatly degrades the model’s performance. Hence, it can be concluded that FruitDet is a mixture of certain modules and ideas that aggregately enhance its performance.

**Table 7.** Benchmark conducted on the MineApple dataset is presented in the table. The reported average precision and precision are utilized on a single class (apple). Best scores are marked bold.

Dataset	AP	Prec.
FruitDet	<b>79.34</b>	<b>85.07</b>
FruitDet-noBlackout	79.01	84.60
FruitDet-AttPool	71.01	84.41
FruitDet-Att	70.50	84.00
FruitDet+SAM	78.28	83.83
FruitDet+PAN	78.00	83.55
YOLOv4	68.82	80.04
YOLOv3	68.59	80.34
MangoYOLO	68.25	79.36
DarkNet+PAN	66.35	77.68
EfficientNet+PAN	65.65	77.62
EfficientNet+FPN	64.20	75.22
DarkNet+FPN	78.20	73.47

Object detection architectures are often designed to perform better for a particular situation, environment, and data quantity. YOLOv3, YOLOv4, are specifically designed and benchmarked to perform better on general-purpose object detection. However, the detection performance usually degrades due to data quality, quantity, and domain variation. Therefore, numerous enhancements are made on such object detection models in multiple platforms [21,38,39]. The phenomenon is also true for the fruit detection environment because YOLO family architectures are tested to work better on large datasets, comparatively bigger objects, with sufficient information available. In contrast, fruit detection datasets are often smaller. Therefore, YOLO architectures struggle to learn proper annotations from the given scarce dataset. Due to such domain variation, YOLO family architectures become more unproductive than the usual performance on large-scale datasets. Furthermore, the number of parameters of YOLO is not suitable for limited data in fruit detection. Therefore, it often becomes hard for the model to learn due to over-parameterization while training on a small dataset [40]. Thus, YOLO family architectures may produce zero performance on scarce datasets, as observed in Tables 3 and 5.

Comparatively, in the case of fruit detection in agriculture, data-dependency is a big challenge. Architectures developed on fruit detection platforms are mostly data-centric. For instance, MangoYOLO [21], YOLOmuskmelon [41], and a tomato detection model [9] are designed to perform better on the self-developed proprietary dataset. Similar to the YOLO family, the data-centric development of fruit detection models may cause unsatisfactory results in a diverse dataset of comparable or different fruits. Such a scenario can be observed for the MangoYOLO detection model. Although MangoYOLO performs excellently in the actual dataset (shown in Table 6), it delivered contradictory results in the rest of the benchmarks in Tables 3 and 5 due to domain variation and data scarcity. Therefore, it is required to benchmark on various datasets while developing detection architectures. FruitDet architecture is a generic model suitable for most diverse data domains in fruit detection schemes compared to most detection models.

Apart from the robustness, Table 8 illustrates a comparison of the number of parameters and computational complexity of the architectures. Floating-point operation (FLOPs) is used to measure the computational complexity of the detection models. From the comparison, it can be validated that FruitDet architecture requires lower memory than the existing YOLO baselines. Further, the model is computationally less expensive than any other detection model in the comparison. From the overall perspective, it can be validated that FruitDet architecture is faster, lighter, and more robust than any other detection model, specifically for detecting fruits from orchards.

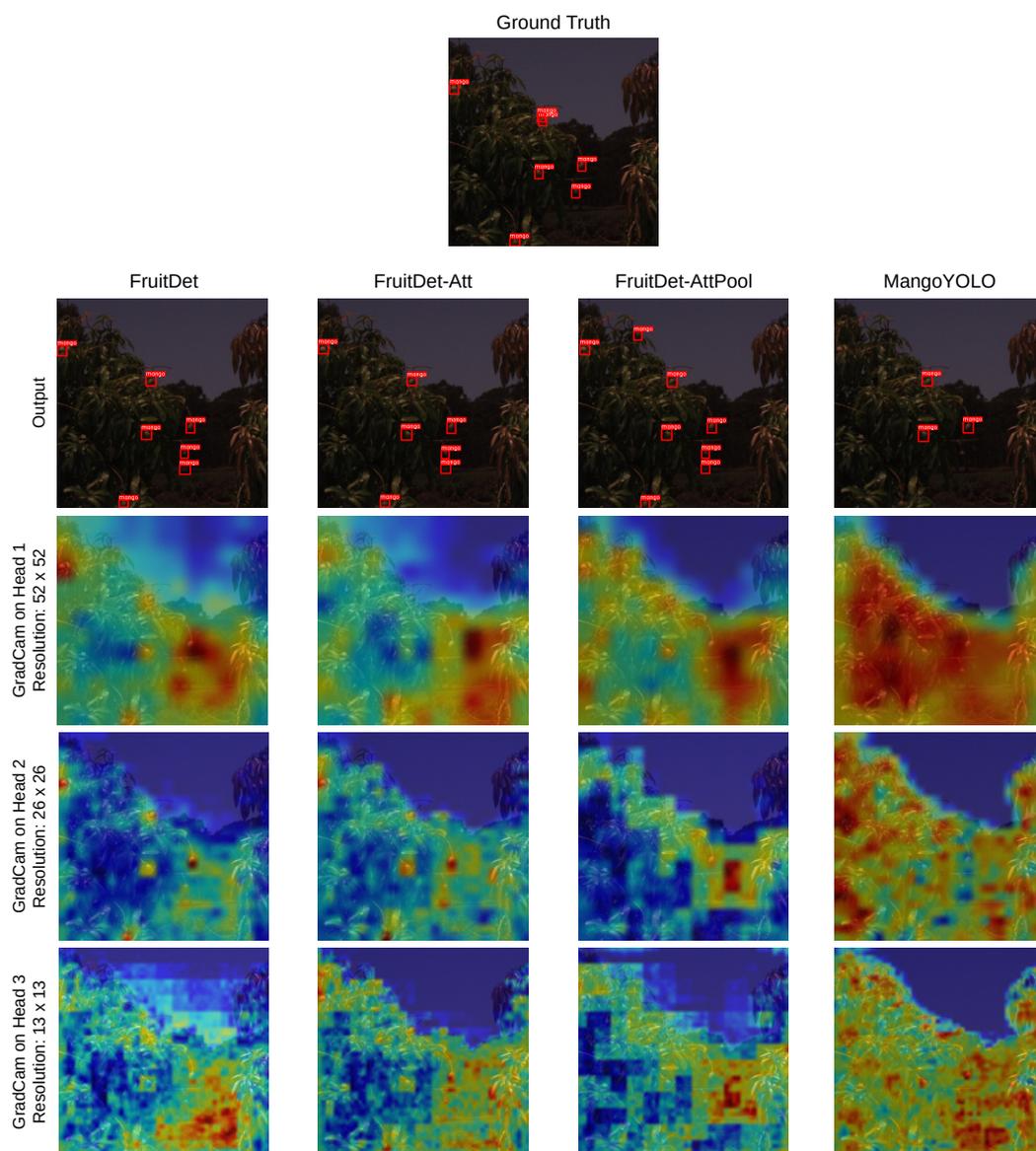
**Table 8.** Complexity benchmark conducted on the different detection models are presented in the table. Best scores are marked bold.

Model	Parameters (in Million)	FLOPs	Time (Milliseconds)
FruitDet	<b>6.79</b>	<b><math>1.45 \times 10^9</math></b>	<b>211.0</b>
YOLOv3	61.53	$3.26 \times 10^9$	214.67
YOLOv4	60.28	$26.36 \times 10^9$	221.33
MangoYOLO	13.78	$5.16 \times 10^9$	217.67

#### 4.5. Inference

The difference in the produced results of FruitDet and other detection architectures can be reasoned via the inference example depicted in Figure 10. The figure illustrates the final inference output of the models. Further, GradCam [42] is used to show the activation of the final outputs, produced by the three heads of the detection mechanisms in three different resolutions:  $52 \times 52$ ,  $26 \times 26$ ,  $13 \times 13$ . In comparison to FruitDet architecture, the MangoYOLO architecture misses some of the detectable objects. The MangoYOLO architecture produces a strong probability of objectness in almost the entire region of the tree. Hence, the MangoYOLO architecture fails to pinpoint the objects due to an imbalance of the final prediction probabilities.

In contrast, FruitDet architecture misses the minimum number of objects. In addition, the probability heatmaps of the three heads are relevant and to the point compared to the MangoYOLO architecture's GradCam output. The FruitDet without attention module (FruitDet-noAtt) also generates a similar detection output during inference. However, considering the GradCam output, the FruitDet-noAtt architecture confuses leaves as objects. Comparatively, adding FruitDet architecture with the attentive module better prioritizes leaves instead of objects. FruitDet-noAttPool resembles a grid of gradient maps, considering the output of heads 2 and 3. The mosaic output is caused due to using an average pooling mechanism inside DenseNet's transition block. The average pooling of the default DenseNet prioritizes only a specific region, causing a separation of detectable features. Instead, the FruitDet architecture's attentive pooling mostly avoids the separation of features. Instead, FruitDet architecture provides better feature maps in both low and high resolutions, resulting in better accuracy.



**Figure 10.** The figure illustrates an inference example of the FruitDet, FruitDet-Att, FruitDet-AttPool, and MangoYOLO architectures, along with an example of the ground truth. Each of the models contains four images (contained in an individual column). The first in each column is the detection output, and the latter are the GradCam [42] visualizations of the three heads of the architectures. The warmer tones of the GradCam images indicate the stronger activation of the final head models. Zoom in for a better view.

## 5. Conclusions

Fruit detection from orchards contains numerous challenges in real-time object detection systems, including object size, number of objects per image, memory consumption, speed, and robustness. Therefore, most fruit detection models are designed for detecting only a specific fruit. This paper introduces a fruit detection model named FruitDet to recognize multiple fruits in a single pipeline. The FruitDet model contains architectural exploration, including attentive feature pooling, modification of attention mechanism, bottleneck spatial pyramid pooling layer, and blackout regularization. The detection model is further evaluated using five datasets, containing numerous challenges: data scarcity, low-quality images, a high number of detectable objects, and so on. The comparison evaluates that the FruitDet architecture outperforms the present YOLO family detection models, including YOLOv3 and YOLOv4. Apart from the robustness, FruitDet architecture

is lighter and faster than the existing detection models. However, extensive study is required for dealing with data scarcity in fruit detection systems. Additionally, more rigorous investigations are needed to determine the best methods for fruit detection systems. We believe that the architectural contribution of the paper would encourage the development of lighter and more robust detection models and improve the overall capability of end-to-end detection models.

**Author Contributions:** Conceptualization, F.A.K., M.F.M. and A.Q.O.; methodology, A.Q.O.; software, A.Q.O.; validation, M.M.M. and M.F.M.; formal analysis, M.A.H. and A.Q.O.; investigation, A.Q.O.; resources, M.F.M.; data curation, A.Q.O.; writing—original draft preparation, A.Q.O.; writing—review and editing, F.A.K. and M.M.M.; visualization, A.Q.O.; supervision, M.M.M. and M.A.H.; project administration, F.A.K. and M.M.M.; funding acquisition, F.A.K. and M.M.M. All authors have read and agreed to the published version of the manuscript.

**Funding:** The Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah, Saudi Arabia has funded this project, under grant no. (KEP-7-611-42).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

- Gebbers, R.; Adamchuk, V.I. Precision agriculture and food security. *Science* **2010**, *327*, 828–831. [[CrossRef](#)] [[PubMed](#)]
- Chen, S.; Zhang, K.; Zhao, Y.; Sun, Y.; Ban, W.; Chen, Y.; Zhuang, H.; Zhang, X.; Liu, J.; Yang, T. An Approach for Rice Bacterial Leaf Streak Disease Segmentation and Disease Severity Estimation. *Agriculture* **2021**, *11*, 420. [[CrossRef](#)]
- Shah, D.; Trivedi, V.; Sheth, V.; Shah, A.; Chauhan, U. ResTS: Residual deep interpretable architecture for plant disease detection. *Inf. Process. Agric.* **2021**, in press. [[CrossRef](#)]
- A dos Santos Ferreira, A.; Freitas, D.M.; da Silva, G.G.; Pistori, H.; Folhes, M.T. Weed detection in soybean crops using ConvNets. *Comput. Electron. Agric.* **2017**, *143*, 314–324. [[CrossRef](#)]
- Lin, G.; Tang, Y.; Zou, X.; Wang, C. Three-dimensional reconstruction of guava fruits and branches using instance segmentation and geometry analysis. *Comput. Electron. Agric.* **2021**, *184*, 106107. [[CrossRef](#)]
- Chen, M.; Tang, Y.; Zou, X.; Huang, Z.; Zhou, H.; Chen, S. 3D global mapping of large-scale unstructured orchard integrating eye-in-hand stereo vision and SLAM. *Comput. Electron. Agric.* **2021**, *187*, 106237. [[CrossRef](#)]
- Kang, H.; Chen, C. Fruit detection, segmentation and 3D visualisation of environments in apple orchards. *Comput. Electron. Agric.* **2020**, *171*, 105302. [[CrossRef](#)]
- Shi, R.; Li, T.; Yamaguchi, Y. An attribution-based pruning method for real-time mango detection with YOLO network. *Comput. Electron. Agric.* **2020**, *169*, 105214. [[CrossRef](#)]
- Lawal, M.O. Tomato detection based on modified YOLOv3 framework. *Sci. Rep.* **2021**, *11*, 1447. [[CrossRef](#)]
- Kang, H.; Chen, C. Fast implementation of real-time fruit detection in apple orchards using deep learning. *Comput. Electron. Agric.* **2020**, *168*, 105108. [[CrossRef](#)]
- Zhou, Z.; Song, Z.; Fu, L.; Gao, F.; Li, R.; Cui, Y. Real-time kiwifruit detection in orchard using deep learning on Android™ smartphones for yield estimation. *Comput. Electron. Agric.* **2020**, *179*, 105856. [[CrossRef](#)]
- Yu, Y.; Zhang, K.; Yang, L.; Zhang, D. Fruit detection for strawberry harvesting robot in non-structural environment based on Mask-RCNN. *Comput. Electron. Agric.* **2019**, *163*, 104846. [[CrossRef](#)]
- Sa, I.; Ge, Z.; Dayoub, F.; Upcroft, B.; Perez, T.; McCool, C. Deepfruits: A fruit detection system using deep neural networks. *Sensors* **2016**, *16*, 1222. [[CrossRef](#)]
- Bargoti, S.; Underwood, J. Deep fruit detection in orchards. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 3626–3633.
- Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
- Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
- Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 91–99. [[CrossRef](#)] [[PubMed](#)]
- Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*; Springer: Cham, Switzerland, 2016; pp. 21–37.
- Tan, M.; Pang, R.; Le, Q.V. Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Virtual, 14–19 June 2020; pp. 10781–10790.

20. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
21. Koirala, A.; Walsh, K.; Wang, Z.; McCarthy, C. Deep learning for real-time fruit detection and orchard fruit load estimation: Benchmarking of ‘MangoYOLO’. *Precis. Agric.* **2019**, *20*, 1107–1135. [[CrossRef](#)]
22. Koirala, A.; Walsh, K.B.; Wang, Z.; McCarthy, C. Deep learning—Method overview and review of use for fruit detection and yield estimation. *Comput. Electron. Agric.* **2019**, *162*, 219–234. [[CrossRef](#)]
23. Santos, T.T.; de Souza, L.L.; dos Santos, A.A.; Avila, S. Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association. *Comput. Electron. Agric.* **2020**, *170*, 105247. [[CrossRef](#)]
24. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, USA, 21–26 July 2017; pp. 7263–7271.
25. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, USA, 21–26 July 2017; pp. 4700–4708.
26. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, USA, 21–26 July 2017; pp. 2117–2125.
27. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path aggregation network for instance segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8759–8768.
28. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1904–1916. [[CrossRef](#)] [[PubMed](#)]
29. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. Cbam: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
30. Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; Ren, D. Distance-IoU loss: Faster and better learning for bounding box regression. *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 12993–13000. [[CrossRef](#)]
31. Häni, N.; Roy, P.; Isler, V. MinneApple: A benchmark dataset for apple detection and segmentation. *IEEE Robot. Autom. Lett.* **2020**, *5*, 852–858. [[CrossRef](#)]
32. Padilla, R.; Passos, W.L.; Dias, T.L.; Netto, S.L.; da Silva, E.A. A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics* **2021**, *10*, 279. [[CrossRef](#)]
33. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv* **2016**, arXiv:1603.04467.
34. Chollet, F. keras. 2015. Available online: <https://github.com/fchollet/keras> (accessed on 20 November 2021).
35. Buslaev, A.; Igloukov, V.I.; Khvedchenya, E.; Parinov, A.; Druzhinin, M.; Kalinin, A.A. Albumentations: Fast and flexible image augmentations. *Information* **2020**, *11*, 125. [[CrossRef](#)]
36. Loshchilov, I.; Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. *arXiv* **2016**, arXiv:1608.03983.
37. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
38. Han, X.; Zhao, L.; Ning, Y.; Hu, J. ShipYolo: An enhanced model for ship detection. *J. Adv. Transp.* **2021**, *2021*, 1060182. [[CrossRef](#)]
39. Abdurahman, F.; Fante, K.A.; Aliy, M. Malaria parasite detection in thick blood smear microscopic images using modified YOLOV3 and YOLOV4 models. *BMC Bioinform.* **2021**, *22*, 112. [[CrossRef](#)]
40. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
41. Lawal, O.M. YOLOMuskmelon: Quest for fruit detection speed and accuracy using deep learning. *IEEE Access* **2021**, *9*, 15221–15227. [[CrossRef](#)]
42. Selvaraju, R.R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 618–626.