2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA) | 978-1-6654-4337-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICMLA52953.2021.00249

Size Does Matter: Overcoming Limitations during Training when using a Feature Pyramid Network

Fabian Fallas-Moya
EECS Department
The University of Tennessee
Tennessee, United States
University of Costa Rica Fellow
ffallasm@vols.utk.edu

Manfred Gonzalez-Hernandez
Atlantic Campus
Universidad de Costa Rica
PRIAS Lab: Geomatics laboratory
with an emphasis on Earth Observation
gonzalezhernandez.manfred@gmail.com

Amir Sadovnik
EECS Department
The University of Tennessee
Tennessee, United States
asadovnik@utk.edu

***Abstract*—State-of-the-art object detectors need to be trained with a wide variety of data in order to perform well in real-world problems. Training-data-diversity is very important to achieve good generalization. However, there are scenarios where we have training data with certain limitations. One such scenario is when the objects of the testing set have a different size (discrepancy) from the objects used during training. Another scenario is when we have high-resolution images with a dimension that is not supported by the model. To address these problems, we propose a novel pipeline that is able to handle high-resolution images by cropping the original image into sub-images and put them back in the end. Also, in the case of the discrepancy of object sizes, we propose two different techniques based on scaling the image up and down in order to have an acceptable performance. In addition, we also use the information from the Feature Pyramid Network to remove false-positives. Our proposed methods overcome state-of-the-art data augmentation policies and our models can generalize to different object sizes even though limited data is provided.**

***Index Terms*—data augmentation, object detection, drone imaging, feature pyramid network**

I. Introduction

Training data diversity is extremely important when training an object detection model to be able to generalize better. However, in some specific cases, this could be limited due to the number and quality of the annotated samples or related to the diversity of the data. This problem is usually tackled using data augmentation techniques. Recent work on object detection [1], [2], [3], [4], [5] uses benchmark-datasets such as COCO [6] or PASCAL VOC [7] to show the effectiveness of data augmentation. However, what happens in cases when COCO and PASCAL are not useful? What if the standard data augmentation techniques do not help in generalization? What if the resolution of the image is not supported by the deep learning model? In this work, we address these questions with datasets that have these limitations.

This type of problem arises frequently in the agriculture industry - which is a key aspect for the economy of many countries [8], [9]. Usually, unmanned aerial vehicles (UAVs) - drones - are used to take images of the fields, which allow companies to analyze their crops. One important piece of information which can be gained is the number of plants in the field. For example, when harvest time comes, the number of plants that were first planted may vary due to a variety of factors such as human error during plantation, wild animals eating the plants, and plant diseases that cause plant loss. This variation of the number of plants can cause an important problem for the companies when trying to create a forecast about the number of products they can sell [10]. For example, Upala Agricola[1] must close deals nine months before the harvest, so, accurate forecasting is extremely important for them. In this context, a tool for counting plants is important. In this research, we used object detection in order to detect all the plant instances and thus be able to accurately predict the number of plants in one plantation.

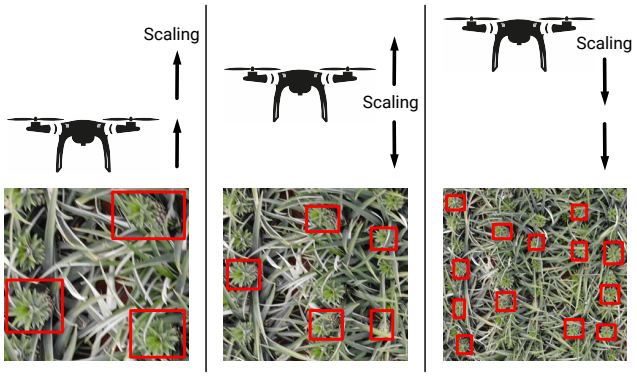


Fig. 1: Drone images at different heights will give us objects at different scales. We address a common problem where there is just one dataset of images at a specific height and we have to be able to generalize to objects at different scales. We propose to scale the images up and down based on the drone height.

We can summarize the contributions of this paper as follows. First, we propose a method to crop images for training, and in the end, we put them back together solving any conflict. Then, during training, we can perform two methods related to scaling transformation in order to improve the performance of the network when applying cross-testing

[1]The largest pineapple company of the world and from which we obtained the datasets we used.

978-1-6654-4337-1/21/$31.00 ©2021 IEEE
DOI 10.1109/ICMLA52953.2021.00249

(testing objects of different sizes as the ones we used during training). In the results section, we show how our proposed pipeline outperforms current state-of-the-art techniques when using deep learning. Our code is publicly available at https://github.com/ManfredGonzalez/scaling-augmentation.

A. Related Work

As indicated by Zhao et al. [5], Object Detection has been an important research topic over the last 20 years. Although early approaches relied on human-engineered features (i.e Histograms of Oriented Gradients [11]), current methods mostly rely on Deep learning (DL). DL techniques can be divided into two main categories: two-stage models such as Faster R-CNN [4] and Mask R-CNN [2], and one stage models such as YOLO [1] and EfficientDet [3], to mention just a few examples.

There have been specific agricultural DL methods that have been developed in recent years. Zhang and Wu [12] and Muresan and Oltean [13] focus on fruit classification. Their experiments were applied in controlled environments (indoors) without noisy backgrounds. Rahnemoonfar and Sheppard [14] use an interesting approach, training their models (Inception-ResNet [15]) by using only synthetic data (but no aerial images).

Specific to pineapple counting, Rahutomo et al. [16] use RetinaNet [17] to detect pineapples. However, they use same-scale images, and the lack of crop details makes it hard to adapt to our problem. Also, Sa et al. [18] use a Faster R-CNN [4] model and include a Near-infrared (NIR) channel into the RGB image to improve performance. However, their experiments were done in controlled environments (using a camera with a NIR channel) and they do not use aerial images.

Finally, Singh et al. [19] created SNIP which is a training scheme to include scaling during training in a two-stage detector. They apply a scaling augmentation and then a filtering step to keep only bounding boxes which corresponds to specific sizes. In a way, this method tries to balance the number of bounding boxes of different sizes. Our scaling methods differ in a way that we analyze the Feature Pyramid Network (with the Pyramid Mask) and use that along with metadata to decide the scaling factor.

1) High-resolution vs. Deep Learning: High-resolution images are usually difficult to process for Deep Learning because the memory requirements grow as the size of an image increases [20]. Recently, Lee et al. [21] showed how we can manipulate high-resolution images with general-purpose DL models. However, when performing object detection the task is still challenging because we have to consider the size of the bounding boxes. Figure 2 shows that the size of the bounding boxes is larger when using just a section of the original image. Although it is possible to deal with high-resolution images using multiple GPU's [22], [20] these methods remain unstable.

The best object detectors that are available right now, use resolutions of 800, 1024, or 2048 pixels, and if the images are larger, the detector performs a resize step to fit their input constraints [23] as most of the deep learning models do. We proposed a method to crop the original image into 8 different sub-figures with a resolution of 1014×1140 each, and we use these splits for training. Figure 2 shows the idea we implemented where we crop the original high-resolution image into smaller sub-images.

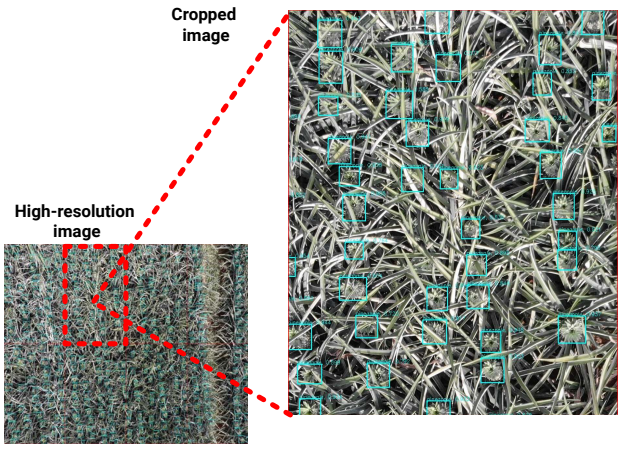


Fig. 2: One example of the images we use in our experiments with a resolution of 4056×2280. On the left, we have the original high-resolution image and on the right is the image we used to train our object detector.

2) Scaling: Sometimes drone images of the crops are taken at different heights (such as 4, 10, or even 25 meters) during the year, producing images with objects of different sizes. Although farmers prefer to take images at 4 meters because the visibility of the pineapples is better, sometimes they need images at 15 meters to analyze the health of the crop at a certain section. Farmers may have labeled images from a limited number of flights. This might mean that we have a dataset of images taken at one height for training, and we need a model that can generalize to images taken at a different height, we refer to this problem as cross-testing.

We use EfficientDet [1] which has a Feature Pyramid Network [24] that deals with different object sizes. We use the metadata related to the images -basically, the height of the images- in order to do the following tasks: (i) to use a scaling technique as a pre-processing step, or (ii) to apply a scaling in the data-augmentation framework to improve the performance of the model. Figure 1 shows images taken at different heights and the idea of scale augmentation. Our proposed methods improve the performance of the model and also show that we beat the policies [25], [26] proposed by the state-of-the-art research related to data-augmentation in Object Detection.

B. Datasets

We used drone images taken from real pineapple crops located in the tropics, especifically, we used the images taken from Upala Agricola[2] which is the largest producer company of pineapples in the world, located in Costa Rica. Although

[2]https://upalagricola.com/

they have a large number of drone images of their crops, the quality itself varies greatly because of the resolution of the camera, weather conditions, and speed of the drone. To test our methods for the cross-testing problem, we have 6 sets of drone images taken from different heights (4m, 8m, 15m) and at different times of the year (first harvest, second harvest) annotated with bounding boxes. We have two datasets at 4m (88 and 115 images each), two at 8m (215 and 205 images each), and two at 15m of height (135 and 146 images each).

II. Methods

Figure 3 shows our main pipeline. First, we use the cropping module in order to slice the image. We have images with a resolution of 4056×2280, we slice them into 8 sub-images with a resolution of 1014×1140. Then, we train EfficientDet using the (i) pre-processing or the (ii) scale-augment method. Finally, we resemble the original image to solve conflicts - where the same object is present partially in two sub-images - to have an accurate counting. As seen in this figure, there is an optional module called Pyramid Mask which helps to filter out false-positive detections from the model.

A. Cropping Module

We pre-processed the original high-resolution images with a resolution of 4056×2280 into 8 sub-images of size 1014×1140 in the first module. 1024×1024 corresponds to the default resize applied to the input image of EfficientDet in the D4 architecture. For this to be accurate, the sizes of the sub-images should be as close as possible to this new resolution applied by the detector model of objects. Here the order of the split matters since we have to reintegrate them into the original image afterward. Therefore, in this module, we also generate metadata that will allow the reintegration in the conflict-solving module.

B. Scaling methods

In order to figure out the correct scaling needed for both of our methods we use the metadata provided by the drone (drone height), and then use the linear magnification equation. Equation 1 shows the underlying idea of this equation.

$$M = \frac{d_i}{d_o} = -\frac{h_i}{h_o} \tag{1}$$

where M is the magnification factor, d is the distance from the camera, h is the height of the object, and subscripts i, o refer to the image plane and object respectively.

Assuming that we are using the same drone (that is d_i is constant) and attempting to detect the same object (h_o is constant), we can derive the scaling factor needed when changing the drone height. That is:

$$scaling\_factor = \frac{h_{i_1}}{h_{i_2}} = \frac{h_o d_i}{d_{o_1}} \Big/ \frac{h_o d_i}{d_{o_2}} = \frac{d_{o_2}}{d_{o_1}} \tag{2}$$

where subscripts 1 and 2 simply refer to images taken from different heights. Since we can assume that the distance to the camera is equal to the height of the drone, we can simply use the ratio between the heights as our scaling factor. For example, if we have images taken at a height of 4m and want to scale those images to 8m - by using equation 2 - we can re-scale the image by a factor of 0.5.

We use the calculated scaling factors in two different ways:

(i) Given that we have trained our network at a specific height, we re-scale/re-size the images of our testing set to fit the size of the pineapples we used during training.

(ii) A data augmentation step for our training set. We take the single height images we receive and apply data augmentation using the scaling factor in order to have a model that can recognize objects at a different scale.

Our resulting methods need the current height from which the training set images were taken and a list of goal heights. We apply either task (i) or (ii) by using the scaling factors obtained from the goal heights and show that we outperform standard methods of data augmentation [25], [27], [26].

C. Conflict-solving Module

As indicated in the previous section, when the image is reintegrated (one image is composed of 8 sub-images) there are many duplicated detections and thus this introduces an extra error when trying to predict the real number of pineapples. In order to solve, this we propose a conflict-solving algorithm (Algorithm 1).

Algorithm 1 Conflict-solving algorithm to indicate if two detections belong to the same object.

Require: $image$ is the reintegration of the 8 sub-images.

```
1:  function CONFLICTS_SOLVING(image)
2:      pairs ← all pairs with IoU > 50% from image
3:      for bbox1, bbox2 in pairs do
4:          new_image ← bbox1 + bbox2
5:          number ← EfficientDet(new_image)
6:          if number == 1 then
7:              merge(bbox1, bbox2)
8:              image ← update(image)
9:          end if
10:     end for
11:     return image
12: end function
```

Our method first reintegrates the 8 different sub-images into a single image. Then, line 2 of algorithm 1, we match all the bounding boxes that lie over the boundaries of the sub-images with those bounding boxes that are *close enough* to be considered for conflict solving. Our criteria for *close enough* is as follows, the bounding boxes in a border with an Intersection over union (IoU) greater than 50% by using one of the axis, either X-axis or Y-axis (see figure 4). With 50% of IoU over 1D, we allowed the algorithm to find half of the pineapples that need to be matched. For example, in figure 4 the blue bounding boxes that are on the left (labeled as 1) of the figure, have an IoU of almost 100% considering the X-axis, so, this conflict

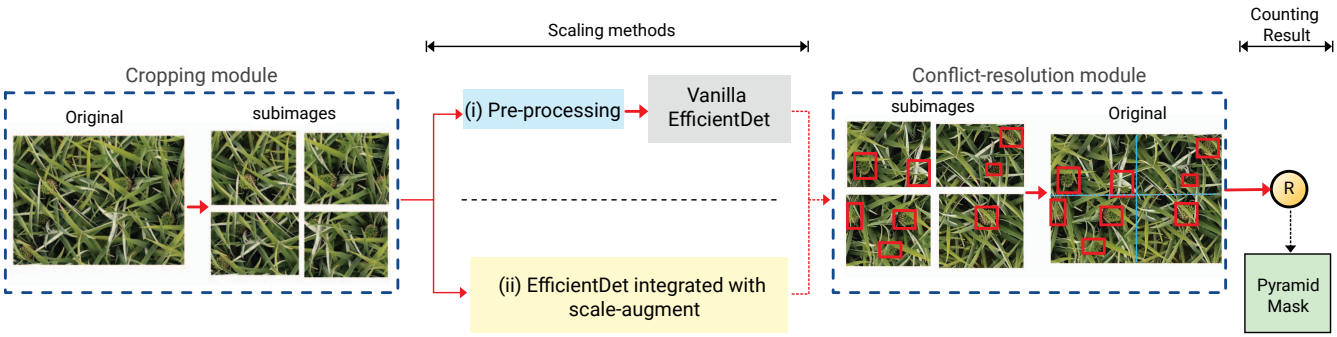
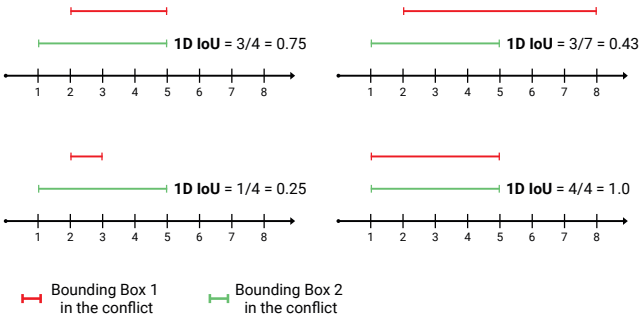
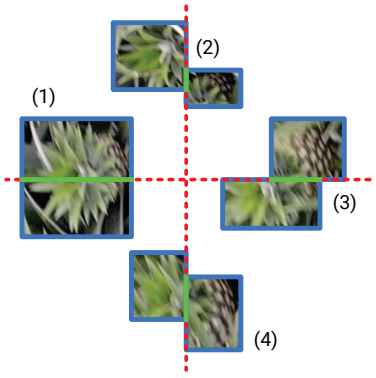


Fig. 3: The pipeline of our proposed method. First, we pre-process all the original high-resolution drone images to create the splits. Then, we train an EfficientDet model using these splits with either method, (i) pre-processing or (ii) scale-augment. Finally, the conflict-solving module solves all the conflicts to generate the counting results from our model.



(a) Calculation of the Intersection over Union (IoU). This is the same IoU used by most Object Detectors but we modified it to be able to use it for only one dimension.



(b) Examples of IoU over 1D as we proposed in this research. (1) has a 1.0 IoU. (2) has a 0.31 IoU. (3) has a 0.44 IoU. (4) has a 0.55 IoU.

Fig. 4: Intersection over Union in the conflict-solving module. It is the ratio between the intersection of the overlapping and the union of the overlapping in the X-axis or the Y-axis.

will be analyzed to see if the bounding boxes in conflict have two different pineapples or only one pineapple.

Once we have filtered all the bounding boxes on the edges to keep the potential conflicts, we take every pair and create an image with only the two bounding boxes which are in conflict, and the rest of the image is filled with zero paddings as if this was a mask. Figure 4b shows an example of this idea, but, instead of taking all the conflicts at once -as the figure suggests- we take one pair with zero paddings. This is then passed to the EfficientDet model again. If the model detects one pineapple, this means that the conflict was correct and then we merge the two bounding boxes into a single one. If the model predicts two bounding boxes we just keep both bounding boxes for the final count.

D. Pyramid Mask

We utilize the fact that we expect all bounding boxes to be approximately the same size for this task. The relation between bounding boxes sizes and the pyramid levels is very important when using a Feature Pyramid Network because every level handles bounding boxes at a specific size [24]. We can derive a pyramid mask from the training data which indicates what levels were used/activated during training.

Level	Stand/img_sc(i)			Aug_sc(ii)			STAC		
	4	8	15	4	8	15	4	8	15
P_7	x	x	x	x	x	x	x	x	x
P_6	x	x	x	x	x	x	x	x	x
P_5	1	x	x	1	1	1	1	x	x
P_4	1	1	x	1	1	1	1	1	1
P_3	1	1	1	1	1	1	1	1	1

TABLE I: Pyramid masks examples at the three heights (4m, 8m, and 15m).

The method works as follows. After training, we perform object detection on our training set and observe which levels of the pyramid were used to make these detections. The level with the most detections is considered the main level. We then create a binary mask that includes the main level and all other neighboring levels which have enough detections (above a certain threshold).

Table I shows example of masks. In column 'Stand/ img_sc(i)' (no-augmentation) we show that when we train

using images at 4m, the active levels are P_3, P_4, and P_5 because at 4m we have the largest bounding boxes (we can see this in figure 1-left image). For level names we used the notation of EfficientDet [1] (from P_3 to P_7). At 8m the activation occurs with only P_3 and P_4 because of the smaller bounding boxes, and finally, 15m activates only P_3 as at this height the bounding boxes are the smallest. Note that P_3 represents the lowest level of the pyramid, that is, the feature map at the largest resolution, and, in terms of objects, the smallest bounding boxes.

We use this mask to filter false-positive bounding boxes during the test phase. Each detected bounding box is marked with the pyramid level it was detected on. If the level is not in the mask we got from the training we remove that detection. This helps to remove detections of significantly different sizes and to ensure that our test detections come from the same levels.

E. F_1 score over mAP

The use of mean average precision (mAP) is almost ubiquitous when it comes to Object Detection, and this metric is great when comparing models in terms of Intersection over Union (IoU) percentages, bounding boxes sizes, and confidence thresholds, to mention a few. However, we realized that for counting problems it makes more sense to use the F_1 score. The reason is simple: mAP hurts the metric when depending on the confidence threshold of the bounding box, that is, for one mistake mAP will penalize heavily if the confidence was high. For counting, however, we really do not care if the confidence was high or low, the mistake with high or low confidence should penalize the model equally - the same for a success detection -, so, we found that F_1 (equation 3) score is more suitable for counting problems. The F_1 score formula is defined as

$$F_1 = 2 \times \frac{precision \cdot recall}{precision + recall} \tag{3}$$

Sa et al. [18] also use F_1 score when detecting fruits, but they do not provide any insights of the reasons.

III. Experiments and Results

An important aspect of EfficientDet is to select the appropriate compound coefficient. Then, we started our experiments by looking if the cropping module was successful. Finally, we focused on the cross-testing experiments.

A. Compound Coefficient Selection

Given that we are using EfficientDet it is important to choose the optimal architecture size of the model. EfficientDet has an extra parameter known as the compound scaling coefficient [1] to control the scale of the model. Xie et al. [28] scale just the backbone of the object detector, but what makes EfficientDet different is the fact that the compound scaling value, increases not only the backbone but also the feature pyramid network (FPN), the resolution of the input, and the number of channels in the layers. As shown by Zagoruyko and Komodakis [29], the scaling factor of the network impacts the performance greatly, table II compares the results for different compound scaling values.

Coefficient	Detected	F1_Score
D1	3190	0.57
D2	4875	0.77
D3	5918	0.85
D4	**6637**	**0.88**
D5	6674	0.879
D6	6827	0.77
D7	7039	0.72

TABLE II: Results of EfficientDet for models D1 to D7. Using a mixed dataset of 487 images at 3 different heights.

Padilla et al. [30] provide a useful implementation to calculate mean average precision (among other metrics). We modified its code to calculate precision and recall and consequently, the F1 score. Table II shows which compound scaling coefficient has the best f1 score as the EfficientDet model was trained and tested from D1 to D7 compound scaling coefficient. As shown, in that table, D4 turns out to be the best coefficient for this dataset and we used D4 for all our experiments.

B. High-resolution Images

First, we perform an experiment using EfficientDet with the original images with no cropping. We split our datasets into training, validation, and testing sets (60/20/20). Table III shows these results (column 'Original'), as compared to the results using our proposed method (column 'Our method'). Column "Case" indicates the images used during training (labeled with 'T') and the images used for testing (labeled with 'H'). As seen, the best results are obtained when the dataset has more diversity of heights in the training set.

When we use the original images the results are close to zero in most cases. This can be explained due to the fact that EfficientDet cannot handle the original resolution of 4056×2280 pixels and it performs a resize of the images to 1024×1024. Also, the bounding boxes are resized and this directly impacts the performance.

When we use the compound scaling coefficient for D4, the maximum size of the images is 1024×1024. Now, the most important factor here is the minimum size of bounding boxes that EfficientDet can detect. According to Tan et al. [1], the level P3 - of the FPN - is the level that detects the smallest objects in an image. When analyzing the levels of the FPN, the pyramid has nine different sizes of bounding boxes that can be detected. The smallest three sizes of the anchor boxes (in pixels) are:

- Anchor 1: [width: 32, height: 32]
- Anchor 2: [width: 45, height: 23]
- Anchor 3: [width: 23, height: 45]

Then, we analyzed the bounding boxes obtained by EfficientDet after the automatic image resizing and found that the on average the bounding boxes have the following sizes:

Case (T = training, H = testing)	Ground Truth Objects	Detections - Original	Detections - Our method	Original	Our method
T:5,8,15 / H:5	1627	6	1666	0.0	0.94
T:5,8,15 / H:8	1452	106	1414	0.13	0.93
T:5,8,15 / H:15	5532	0	5417	0.0	0.96
T:5 / H:5	1627	0	1614	0.0	0.87
T:5 / H:8	1452	1	1245	0.0	0.83
T:5 / H:15	5532	0	1333	0.0	0.35
T:8 / H:5	1627	9	1055	0.0	0.71
T:8 / H:8	1452	232	1301	0.28	0.87
T:8 / H:15	5532	3	2736	0.0	0.64
T:15 / H:5	1627	0	1466	0.0	0.47
T:15 / H:8	1452	8	1843	0.0	0.14
T:15 / H:15	5532	18	5763	0.0	0.91

TABLE III: This table shows the results of the EfficientDet model trained with high-resolution images and with our proposed pipeline. Column 'Case' indicates the images used during training (T) and the images used during testing (H). We wanted to see the results of cross-height so, sometimes we trained with some images at different heights from the images used during testing. We show the ground truth objects (i.e. number of pineapples), the number of detections using the original high-resolution images, the number of detections using our pipeline, and the last two columns show the resulting f1 scores.

- Images at 15 mts: [width: 12, height: 12]
- Images at 8 mts: [width: 25, height: 25]
- Images at 5 mts: [width: 30, height: 30]

Consequently, EfficientDet is not able to perform well because the bounding boxes are smaller than the minimum size that it can detect, and this explains the terrible results achieved by using the original resolution. This reinforces the idea that when dealing with high-resolution images, our proposed method provides a good solution to this problem.

By looking table III we can see that the results in the column "Our method" are very good. When using the three heights during training the results are above 0.9 in the f1 score and tend to decrease when experimenting with cross-height, that is, using weights trained with one height and test using images at another height. The cross-height is not as good as when using all heights during training, so, this is an indicator that data-augmentation may help for these cases.

C. Conflict-solving Module

Since cropping the image is extremely important, we perform an experiment to test the importance of our conflict-solving module. Within our pipeline, we used the crops but we turn on and off the conflict-solving module to see if our proposed module was improving the results. Table IV shows the results for this experiment.

For this experiment, we trained using images at different heights, as usually is performed, but also we used cross-height testing. An example of cross-height is shown in row 5 where we trained using images at 5m and tested using images at 8m. Usually, cross-height does not have as good results as when using the full dataset for training, but it does show how flexible the model is using different heights for training. We show precision instead of the f1 score because the improvement is almost imperceptible using the f1 score and the recall stays approximately the same for all cases, also, we show the number of pineapples to see the number of conflicts solved.

Case	CS deactivated		CS activated	
	Precision	Counting	Precision	Counting
T:5,8,15 / H:5	0.92	1686	**0.93**	1666
T:5,8,15 / H:8	0.92	1444	**0.94**	1414
T:5,8,15 / H:15	0.96	5441	**0.97**	5417
T:5 / H:5	0.87	1639	**0.88**	1614
T:5 / H:8	0.88	1268	**0.9**	1245
T:5 / H:15	0.89	1334	0.89	1333
T:8 / H:5	0.9	1062	0.9	1055
T:8 / H:8	0.91	1321	**0.92**	1301
T:8 / H:15	0.97	2742	0.97	2736
T:15 / H:5	0.5	1470	0.5	1466
T:15 / H:8	0.13	1847	0.13	1843
T:15 / H:15	0.89	5778	**0.9**	5763

TABLE IV: The effectiveness of the Conflict-solving (CS) module. As predicted, when the module is activated the precision is improved in several cases. Furthermore, we show that the counting result (the number of detected pineapples) decreases in all cases when the module is activated.

Table IV rows 1-2 show the improvement in precision when the conflict-solving module is activated. This result is meaningful since we want to increase the precision of the counting. As seen, we go from 1686 detections to 1666, so, here we could solve 20 conflicts, showing that our method approaches the ground truth of 1627.

Also, in cross-height testing, there is always an improvement when training height (T) is the same as testing height (H) indicating a good influence of the conflict solving approach when the model is tested on equal terms. An important aspect is shown in the rows where the precision keeps the same because the number of detections is always decreasing (i.e. approaching the ground truth), however, the number of conflicts solved is small and that is why the precision keeps the same. This experiment shows that the conflict-solving module helps to increase the accuracy of the results and this is important in order to have a better estimation of objects.

Methods	H: 4 / T: 8	H: 4 / T:15	H: 8 / T: 4	H: 8 / T: 15	H: 15 / T: 4	H: 15 / T: 8
Standard† [1]	0.35 ± 0.29	0.09 ± 0.1	0.53 ± 0.21	0.43 ± 0.04	0.14 ± 0.16	0.44 ± 0.04
STAC [25]	0.38 ± 0.29	0.07 ± 0.07	0.52 ± 0.17	0.42 ± 0.03	0.09 ± 0.11	0.48 ± 0.09
Ours (i) pre-processing†	0.49 ± 0.21	0.52 ± 0.13	0.77 ± 0.06	0.73 ± 0.09	$\mathbf{0.74 \pm 0.08}$	0.60 ± 0.07
Ours (ii) scale-augment	$\mathbf{0.72 \pm 0.05}$	$\mathbf{0.77 \pm 0.02}$	$\mathbf{0.79 \pm 0.02}$	$\mathbf{0.80 \pm 0.03}$	0.47 ± 0.06	$\mathbf{0.60 \pm 0.06}$

TABLE V: Comparison of F1 scores for different methods on the cross-testing. We report the f1 score mean and standard deviation for the four replicas of our experiment. Every column indicates the height of the testing images (H) and the height of the images used during training (T). Methods with † do not have any data augmentation.

D. Cross-testing

Given that we have a single object class with fixed object sizes, it is difficult to find a proper benchmark to compare to. Consequently, we tested our methods against the policy proposed by STAC [25] for object detection and a standard EfficientDet model which does not have any augmentation or pre-processing. Also, we note that our goal is to train on one dataset taken at a specific height and to test on a dataset taken at a different height, as done by Tzanetis [31].

We conduct an experiment where we train using the four methods as independent runs. We perform four replicas to ensure statistical significance and for these experiments, we used the conflict-solving module, otherwise, the results are zero.

Table V shows the performance of the methods we used. Each column shows a different train/test split. For example, the first column "H: 4 / T: 8" represents the case where we use images taken at 4m for testing but the model was trained over images taken at 8m (H stands for the height of the testing images and T stands for the height of images used for training).

We can see that STAC does not work properly, and its results are close to the standard method with almost no significant difference. For example, in the first column STAC performs better with 0.38, compared to the standard method with 0.35, but both with a standard deviation of 0.29. These two methods consistently have the lowest results. Surprisingly, sometimes it seems that STAC augmentation produces worse results, as in column "H: 15 / T: 4" that goes from 0.14 (standard method) to 0.09 using STAC, showing that the augmentations proposed by STAC do not apply in this problem. On the other hand, our two proposed methods improve the results over STAC and standard. The mean and standard deviation are better for scaling-augmented.

We applied one-way ANOVA in order to have more solid evidence. First, the Pr(>F) value gives us 99.9% confidence that using different methods in our experiments is statistically significant, as we can confirm by looking at table V. Second, the Tuckey test, at 95% of confidence, consistently shows that our two methods outperformed STAC and standard with a p-adjusted value of ≈ 0.0. Finally, our (i) pre-processing and (ii) scaling-augment method have a p-adjusted value of 0.78, meaning that there is no statistical evidence that indicates one method is better than the other. However, given that method (i) only requires training once on the original data, and can be used to detect objects from any height, it is preferable.

E. Cross-testing: Pyramid Mask

Lastly, we use the pyramid mask in order to filter out false positives. Table VI shows some improvements over the original results. Since we are getting rid of false-positives, we are improving the precision and not the recall. Example 3, where the mask is equal to $[1, 0, 0, 0, 0]$ (only P_3 is active), shows a meaningful improvement from 0.73 to 0.93. Example 2 also shows a meaningful improvement, but, there are examples where the improvement is very small like in 1 and 4. In the others not shown in the table, the improvement is 0.0. Even though the results are not statistically significant this feature is very important at a production level since every detected object should be consistent in size.

Example	Method	Case	Old-Prec	New-Prec
1	Standard	H:8 / T:4	0.49	0.51
2	STAC	H: 5 / T:15	0.68	0.72
3	(i) img_sc	H:8 / T:15	0.73	0.93
4	(i) img_sc	H:8 / T:15	0.86	0.87

TABLE VI: Some results of using the pyramid mask to filter out false-positive detections. We compare the old precision (no filtering) and the new precision (with filtering) and omit recall since it stays the same.

IV. Conclusions and Future Work

Cropping high-resolution images into sub-images is crucial when using object detectors. Here we show that training EfficientDet using high-resolution images will not work as expected due to limitations in the input size of the model. We have shown that if we crop the image from 4056×2280 pixel to get 8 different 1014×1140 images, we will be closer to the input expected by the EfficientDet D4 setting (which is 1024×1024). In this case, the performance will be better because our objects in the sub-images almost match the expected object size by D4.

Scaling is very important when dealing with limited data such as objects of a single scale. We showed that standard object detection models and standard data augmentation techniques are not enough for generalization in this case. Scaling images - and as a result, the objects themselves - is a very important task since the size of the objects during training plays an important role in generalization. We proposed (i) a pre-processing step to scale the input images to fit the trained model and (ii) a data augmentation scaling to learn objects at different sizes. Both methods use the linear magnification

equation for scaling and we showed, with statistical evidence, that they outperformed the standard methods. In addition, as the objects' size has strong implications when using a Feature Pyramid Network, we showed that using a mask based on the filter levels can lead to an additional improvement.

As the next steps, we consider using semi-supervised learning due to the difficulty of annotating images. We spent many hours in labeling and when this task was assigned to other people some annotations were of poor quality. In addition, we want to study the impact of scaling in a semi-supervised setting. Finally, we note that although we present a large improvement, there is still more work to be done in order to have results that can be used in the industry regarding counting. We are close to an f1 score of 0.8 but believe that we can improve these results by using hyperspectral images.

ACKNOWLEDGMENT

The authors would like to thank to the University of Costa Rica (the team under the project Pry01-1431-2019) and to Upala Agricola for the datasets and their guidence.

REFERENCES

[1] Mingxing Tan, Ruoming Pang, and Quoc V. Le, "EfficientDet: Scalable and efficient object detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10778–10787.

[2] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, "Mask R-CNN," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 386–397, 2020.

[3] Alexey Bochkovskiy, Chien Yao Wang, and Hong Yuan Mark Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020.

[4] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[5] Zhong Qiu Zhao, Peng Zheng, Shou Tao Xu, and Xindong Wu, "Object Detection with Deep Learning: A Review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[6] Tsung Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick, "Microsoft COCO: Common objects in context," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8693 LNCS, no. PART 5, pp. 740–755, 2014.

[7] Everingham M., Van-Gool L., Williams C K I., Winn J., and Zisserman A., "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[8] Ana Luísa Soares Peres and Letícia De Souza Daibert, "Negotiating agriculture in the world trade organization: Food security as a non-Trade concern," *Brazilian Journal of International Law*, vol. 14, no. 1, pp. 55–67, 2017.

[9] Jakub Piecuch and Łukasz Paluch, "Importance of Agriculture Within the Structure of Employment and Production in the Mediterranean Countries," *Journal of Agribusiness and Rural Development*, vol. 10, no. 1, 2016.

[10] Thomas van Klompenburg, Ayalew Kassahun, and Cagatay Catal, "Crop yield prediction using machine learning: A systematic literature review," *Computers and Electronics in Agriculture*, vol. 177, pp. 105709, 2020.

[11] Carlo Tomasi, "Histograms of Oriented Gradients," *Computer Vision Sampler*, pp. 1–6, 2012.

[12] Yudong Zhang and Lenan Wu, "Classification of fruits using computer vision and a multiclass support vector machine," *Sensors (Switzerland)*, vol. 12, no. 9, pp. 12489–12505, 2012.

[13] Horea Mureşan and Mihai Oltean, "Fruit recognition from images using deep learning," *Acta Universitatis Sapientiae, Informatica*, vol. 10, no. 1, pp. 26–42, 2018.

[14] Maryam Rahnemoonfar and Clay Sheppard, "Deep count: Fruit counting based on deep simulated learning," *Sensors (Switzerland)*, vol. 17, no. 4, pp. 1–12, 2017.

[15] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," in *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, 2017, pp. 4278–4284.

[16] Reza Rahutomo, Anzaludin Samsinga Perbangsa, Yulius Lie, Tjeng Wawan Cenggoro, and Bens Pardamean, "Artificial Intelligence Model Implementation in Web-Based Application for Pineapple Object Counting," *Proceedings of 2019 International Conference on Information Management and Technology, ICIMTech 2019*, , no. August, pp. 525–530, 2019.

[17] Tsung Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar, "Focal Loss for Dense Object Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 2020.

[18] Inkyu Sa, Zongyuan Ge, Feras Dayoub, Ben Upcroft, Tristan Perez, and Chris McCool, "Deepfruits: A fruit detection system using deep neural networks," *Sensors (Switzerland)*, vol. 16, no. 8, 2016.

[19] Bharat Singh and Larry S. Davis, "An Analysis of Scale Invariance in Object Detection - SNIP," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3578–3587, 2018.

[20] Sai Wu, Mengdan Zhang, Gang Chen, and Ke Chen, "A new approach to compute CNNs for extremely large images," *International Conference on Information and Knowledge Management, Proceedings*, vol. Part F1318, pp. 39–48, 2017.

[21] Seho Lee, Ohsung Oh, Youngju Kim, Daeseung Kim, Daniel S. Hussey, Ge Wang, and Seung Wook Lee, "Deep learning for high-resolution and high-sensitivity interferometric phase contrast imaging," *Scientific Reports*, vol. 10, no. 1, pp. 9891, Jun 2020.

[22] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc' aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Quoc Le, and Andrew Ng, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. 2012, vol. 25, Curran Associates, Inc.

[23] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye, "Object Detection in 20 Years: A Survey," pp. 1–39, 2019.

[24] Xiaohan Li, Taotao Lai, Shuaiyu Wang, Quan Chen, Changcai Yang, and Riqing Chen, "Feature Pyramid Network," *Proceedings - 2019 IEEE Intl Conf on Parallel and Distributed Processing with Applications, Big Data and Cloud Computing, Sustainable Computing and Communications, Social Computing and Networking, ISPA/BDCloud/SustainCom/SocialCom 2019*, pp. 1500–1504, 2019.

[25] Kihyuk Sohn, Zizhao Zhang, Chun Liang Li, Han Zhang, Chen Yu Lee, and Tomas Pfister, "A Simple Semi-Supervised Learning Framework for Object Detection," in *arXiv*, 2020.

[26] Barret Zoph, Ekin D. Cubuk, Golnaz Ghiasi, Tsung Yi Lin, Jonathon Shlens, and Quoc V. Le, "Learning data augmentation strategies for object detection," *Computer Vision – ECCV 2020*, 2020.

[27] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2020-June, pp. 3008–3017, 2020.

[28] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He, "Aggregated residual transformations for deep neural networks," 2017.

[29] Sergey Zagoruyko and Nikos Komodakis, "Wide residual networks," in *Proceedings of the British Machine Vision Conference (BMVC)*, Edwin R. Hancock Richard C. Wilson and William A. P. Smith, Eds. September 2016, pp. 87.1–87.12, BMVA Press.

[30] Rafael Padilla, Wesley L. Passos, Thadeu L. B. Dias, Sergio L. Netto, and Eduardo A. B. da Silva, "A comparative analysis of object detection metrics with a companion open-source toolkit," *Electronics*, vol. 10, no. 3, 2021.

[31] D. E. Tzanetis and P. M. Vlamos, "Cross-Domain Weakly-Supervised Object Detection through Progressive Domain Adaptation," *Proceedings of the Edinburgh Mathematical Society*, vol. 44, no. 3, pp. 585–595, 2001.