DOI: 10.15439/2022F58
ISSN 2300-5963 ACSIS, Vol. 30

An Automated Algorithm for Fruit Image Dataset Building

Horea-Bogdan Mureşan
Babeş-Bolyai University
Faculty of Mathematics and Computer Science
No. 1, Mihail Kogălniceanu Street, Cluj-Napoca, Romania
Email: horea.muresan@ubbcluj.ro

Abstract—**This paper introduces a new algorithm that utilises images from the Fruits-360 dataset, superimposes them of various backgrounds and creates associated annotation files with the coordinates of the bounding boxes surrounding the fruits. The main challenge of this task was accounting for the variations in lighting and occlusion associated with outdoor locations. The utility and application of such an algorithm is to reduce the need to collect real world data for training, accelerating the speed at which new models are developed. Using 3000 images generated by this algorithm we train a single shot multibox detector (SSD) to study the feasibility of using generated data during training. We then test the trained model on 70 real world images of apples (65 images of apples on trees and 5 images of apples in bunches) and obtain a mean average precision of 0.750 and we compare our results with those obtained by other state of the art models.**

I. Introduction

THE most frequently mentioned challenge across papers that study the application of artificial intelligence in agriculture [1], [2], [3] was data scarcity [4], [5]. As such, authors have to collect images from orchards/plantations or obtain images from the Internet however, both methods have downsides. Collecting images from an orchard/plantation requires visiting the location during a certain time frame when the fruits are ripe [6]. Data collection from the Internet via scraping cannot guarantee image quality. Reference [7] suggests that multiple visits to the same or to different orchards during different periods of time are required for an ideal dataset. Both methods also require manual annotations, which is time consuming and is susceptible to human error [8]. One way to address the data scarcity problem is the creation of a data generating algorithm that can produce images that simulate real world conditions. We aimed to study the feasibility of using such generated data for training an apple detector.

The Fruits-360 dataset [9], which provides 90483 images of 131 fruits and vegetables, was introduced in 2017. The images in this dataset contain one fruit or vegetable per image with a white background. This makes the set very good for training classifiers, however it cannot be easily used to create detectors capable of locating/counting the number of fruits in an image. In order to extend the usability of the dataset for such tasks as well, we have created an algorithm that generates artificial training images by superimposing fruits taken from the Fruits-360 dataset [10] on various backgrounds containing tree leaves, branches and other fruit. Alongside these artificial images, the software generates one annotation file per image containing all the coordinates of the bounding boxes that surround the fruits as well as the fruit class. Using a training and validation dataset of apple images generated with our algorithm we trained a SSD [11] and tested it on 70 real world images, obtaining an average precision of 0.750.

II. Related work

In paper [8], the authors proposed a novel approach based on convolutional neural networks for tomato fruit counting rather than area calculation. They used a modified version of the Inception-ResNet [12] network for this task. The authors noted that in order to obtain good performance with a deep learning algorithm, a dataset that captures all the variance in the conditions under which the model is expected to operate. Such datasets with annotated images were not available and the authors observe that they are difficult and time consuming to build. One factor that contributes to this is the limited time frame in which fruits are in the desired development stage for image taking. Another factor is human error, to which manual labelling is susceptible to. Thus, the authors created their own synthetic images by filling an image with green and brown circles to simulate background and then red circles to simulate tomatoes. Afterwards, a Gaussian blur filter was applied on the images. Training the model on a dataset consisting of exclusively synthetic images and then testing it on 100 real world images produced a 91% accuracy in estimating the fruit count.

In [13] several models for fruit detection were reviewed, such as multilayered perceptrons, LedNet [4] and InceptionV3 [14]. One issue that was present in all analysed papers was data scarcity. The authors of papers [5], [6] state that building an annotated dataset is a costly process from the perspectives of both time and material resources. For state of the art performance such a dataset should contain images of fruits in multiple lighting and weather conditions, at different distances and at different levels of occlusions, according to the intended practical application of the model. Kang and Chen, in [4], used a multi-scale pyramid and clustering classifier to assist data labelling. Steven Chen et al. in [15] utilised a custom crowd-sourcing platform for quick data labelling to address this issue. As seen in [16], using transfer learning and a MobileNetV2 [17] model the authors achieved a good compromise between accuracy and inference speed. Similarly, Raheelin Siddiqi shows in paper [18] the effectiveness of transfer learning on classification accuracy of fruit images.

To the extent of our knowledge, outside of [8], no other projects have attempted to create a data generating algorithm that creates images of fruits on various background as well as the associated annotations.

III. Methodology

A. Data Generation

The goal of this algorithm is to allow a user to create the entirety or the majority of their training dataset without the need of seeking an orchard, collecting images and manually annotating them. This would reduce dependence on obtaining access to an orchard/plantation and on manual labelling. Furthermore, such an algorithm would allow experiments to be executed even when real world data cannot be collected. The idea behind this algorithm was inspired by paper [8], in which the authors created a dataset of synthetic images (green background with red dots) in order to simulate images of tomato plants. Using this dataset alone for training and then testing on real world images of tomato plants, the model trained by the authors achieved 91% accuracy.

The process relies on images from the Fruits-360 dataset [10] and on real images that will be used as backgrounds for the generated images. The image output size, class and maximum number of fruits contained in each such image can be customised to fit the scope of the project for which they are used.

The main steps of the algorithm that creates the dataset are:

- Randomly select a background image:
  - A folder of RGB background images (JPEG/PNG) of any resolution must be specified.
- Resize the selected background to the specified output width and height. We will refer to this image as a canvas.
  - The output width and height can be customized according to the purpose of the trained model. Similarly, random stretch can be applied to the image. This is done by specifying two intervals of floating point numbers (one for width, the other for height). From these intervals, a random float is selected and the respective dimension is multiplied by it. This simulates images taken using cameras with different aspect ratios (4:3, 16:9, etc.)
- Select fruit image from given labels.
  - Once the canvas is selected and resized, the algorithm randomly chooses a fruit image from the Fruits-360 dataset from one of the classes specified by the user.
  - The fruit image is resized to a randomly generated size within a given interval.
- Create mask and crop image to be centered on the fruit.
  - The images from the Fruits-360 dataset have a simple white background however, some fruits with a shiny texture, such as red apples, reflected the ambient light and contain white spots.
  - To ensure that, when isolating the fruit pixels, we do not remove the aforementioned white spots we apply the following operations:
    1) Firstly we create a mask by converting the fruit image to grayscale, then applying a threshold function such that the white pixels are transformed to black and the non-white pixels become white.
    2) Secondly, the mask is copied and a flood fill algorithm is applied starting from the corners.
    3) The flood filled mask copy is then inverted and a bitwise or is applied between it and the initial mask.
    4) Finally, the mask is eroded with a 3×3 kernel to eliminate border pixels between the fruit and the white background (Fig. 1).
- Augment fruit image.
  - The operations used to augment the fruit image are 90, 180, 270 degree rotations, brightness and contrast alteration and partial cropping.
  - Partial cropping simulates fruit occlusion by removing rows or columns of fruit pixels from the image. Both the probability of applying this

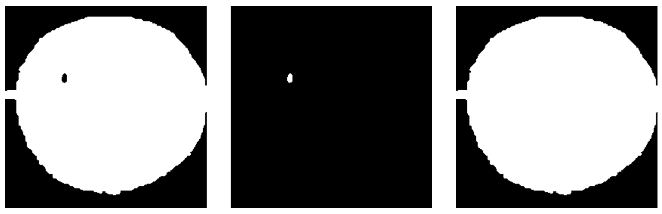

(a) Bright spot (b) After flood fill and inversion (c) After bitwise or with the inverted mask

Fig. 1: Example of the bright spot issue caused by a granny smith apple and the solution to the problem.

TABLE I: PARAMETERS USED FOR GENERATING THE TRAINING AND VALIDATION DATA.

Max fruits per image	Min fruit size (px)	Max fruit size (px)	Training images	Validation images
50	30×30	250×250	500	100
30	150×150	400×400	1000	200
10	250×250	300×300	1000	200
25	50×50	500×500	500	100

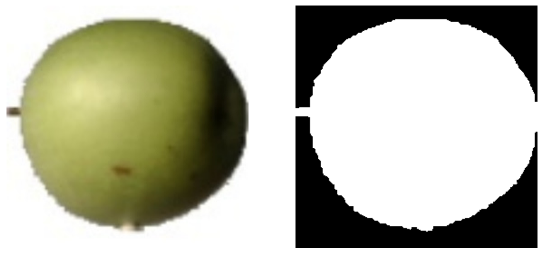

(a) Original fruit image (b) The resulted mask

Fig. 2: An example of the mask-creating algorithm that selects only the fruit pixels while dropping the white background

operation and the maximum amount of pixel rows or columns that can be removed can be customized.

- Attempt to add the fruit image to the canvas.
  - The algorithm keeps a set of coordinates for each fruit added onto the canvas.
  - When a new fruit image can be added to the canvas, its coordinates are randomly generated. This is done to allow a more uniform distribution of fruits on a canvas, as well as to speed up the adding phase.
  - If a set of generated coordinates allows the image to be placed with an acceptable level of overlap, then the image is added onto the canvas. This is done by copying the fruit pixels from the fruit image with the help of the generated mask.
  - If, after a number of attempts, the image was not added, then the algorithm retries by resizing the image to 80% of its initial size, but not smaller than the user defined minimum size.
  - If the image is not successfully added even after resizing, then it is discarded and a new fruit image is selected.
- Once the fruits have been added, the canvas is saved in the PNG format with an index number as a name. The bounding boxes associated with the fruits in the image are stored in a separate file, either an xml or csv.
- The process is repeated until the requested number of images have been generated.

For the experiments done in this paper we generated a dataset of 3000 training images and 600 validation images of size 768×1024 using all the apple classes from the Fruits-360 dataset and 30 background images. The background images were scraped from the Internet and contained foliage and trees. The images were generated according to the parameters presented in Table I. This distribution was chosen to simulate both a scenario in which fruits are close to the camera, which would produce an image with a few large fruits as well as the scenario in which a photo is taken from further away, in which case there would be numerous small fruits in the image.

B. Model Evaluation

In order to evaluate the feasibility of using a synthetic dataset to train a model that can then be used in real world scenarios, we selected the SSD512 model [11] as it has shown promising results in the area of intelligent agriculture, such as fruit detection [5] and leaf disease detection [19]. The SSD network was implemented in Keras [20] and was adapted for images of size 768×1024. Training was done on the 3000 training images using an Adam optimizer with a learning rate of 0.0001 for the first 5 epochs and with a learning rate of 0.00005 for 5 more epochs. At the end of each epoch the model was evaluated on the 600 validation images and it was saved if it improved from the previous evaluation. The experiment was repeated 3 times, each time with a new set of generated images. The machine on which the training was done was equipped with an nVidia 2070 RTX GPU, 16 GB RAM and an Intel i7-8750H CPU. The implementation was done using TensorFlow 2.4 [21] and Python 3.7.7.

IV. RESULTS

For testing the model, a set of 70 real world images was created by scraping freely reusable images from the Internet and by taking photos of apples on trees or in bunches. The images contain multiple species of apples at medium to close distance, with some fruits being partially occluded by foliage or by other apples. The quality and resolution of the test images was varied, containing even some blurred apples, further increasing the difficulty of detecting them. The images were manually annotated using the **labelme** tool [22]. In order for an apple to be included in the annotations, at least half of it had to be visible in the image and had to be larger than 30×30 if the image is resized to 768×1024.

Following, we will present the results of the trained SSD model on the 70 test images and compare them to

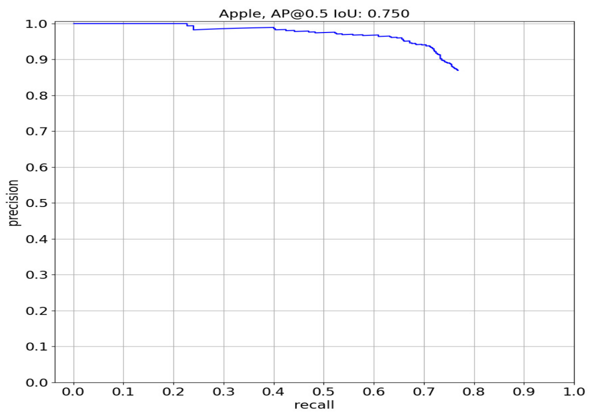


Fig. 3: Precision-Recall curve at 50% IoU.

existing fruit detectors. Table II shows the results of the three experiments conducted.

In Fig. 3 the precision-recall curve is presented for model number 2. It can be noted that the high precision value of this model means that the network produces very few false positives. However, the recall value does not exceed 80% for any of the three models. This indicates that the model fails to detect all apples from the test images, producing false negatives.

After visually inspecting the predictions on the test images, we identified several reasons behind the false negatives. Fig. 4 captures several examples:

- a large difference in lighting between a fruit and the others; as an example, Fig. 4d shows an apple that is underneath a cluster of leaves such that it does not receive the same amount of light as the other fruits in the image
- blurry images can impact the detection of fruits, as seen in Fig. 4a where a fruit located in the background is blurred compared to the apples in the foreground
- fruits that have a similar or identical colour with the background (green apples in this case) paired with partial occlusion produces situations where the fruits are not detected, shown in Fig. 4b and Fig. 4c

Table III presents the results of the models studied in paper [4], in which the authors tested two augmentation pipelines aiming to improve apple detection. The models from [4] were trained on real world images exclusively while our proposed model was trained on 3000 generated images. It can be noted that the performance of our proposed model is in the same vicinity, indicating the viability of training a model on generated data.

In paper [5] a Faster R-CNN and an SSD model were used for counting fruits from five classes: apple, orange, mandarin, lemon and tomato. Compared to our SSD model, the one proposed in [5] performs slightly better on fruit counting on the apple subset, achieving 81% accuracy, while ours achieved 76%. As mentioned previously, the model produces false negatives, explained by partially occluded fruits of the same color as the background or by blurred fruits, which can explain the difference between the predicted count and the actual count.

TABLE II: The results of the three trained models.

Model	AP_{50}	F1 Score	Recall	Precision
SSD#1	0.748	0.770	0.740	0.802
SSD#2	0.750	0.816	0.769	0.869
SSD#3	0.751	0.778	0.795	0.761

V. Conclusion and future work

In this paper we have studied the feasibility of using a dataset of generated images as training data for an object detector and then applying it on real world images. We introduced an algorithm that uses the fruit images from the Fruits-360 dataset and custom background images to create new images alongside annotation files. We trained a SSD on an apple dataset of 3000 images generated with this algorithm and tested it using 70 real world apple images. Then we compared our results with other state of the art works and concluded that using generated images for training produces a model capable of handling real world data.

In the future, we plan to study if using generated data for training and using real world images for fine tuning leads to improved detection performance. Another development direction is the addition of more augmentation operations to the data generating algorithm to account for more data variance. A better way to simulate fruit occlusion, for example overlapping images of potential obstructions (eg. leaves, branches) over the fruit image could improve a model's capacity of detecting such fruits.

Overall, the presented study shows potential in using generated images instead of images collected from orchards or plantations as the principal source of training data for a fruit detector. We think that this paper serves as a starting point for other, more complex approaches.

References

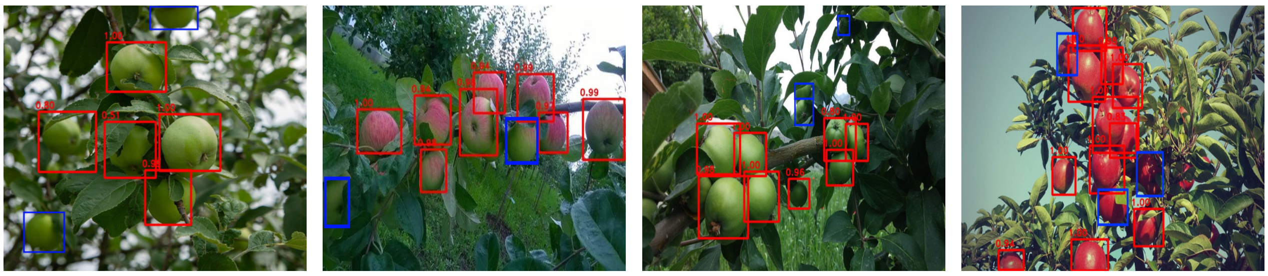
[1] N. C. Eli-Chukwu, "Applications of artificial intelligence in agriculture: A review," *Engineering, Technology & Applied Science Research*, vol. 9, no. 4, pp. 4377–4383, 2019. doi: 10.48084/etasr.2756

[2] O. Apolo-Apolo, J. Martínez-Guanter, G. Egea, P. Raja, and M. Pérez-Ruiz, "Deep learning techniques for estimation of the yield and size of citrus fruits using a uav," *European Journal of Agronomy*, vol. 115, p. 126030, 2020. doi: 10.1016/j.eja.2020.126030

[3] S. Bargoti and J. P. Underwood, "Image segmentation for fruit detection and yield estimation in apple orchards," *Journal of Field Robotics*, vol. 34, no. 6, pp. 1039–1060, 2017. doi: 10.48550/arXiv.1610.08120

TABLE III: THE RESULTS OF OUR PROPOSED METHOD COMPARED WITH THOSE OBTAINED IN [4].

Method	AP_{50}	F1 Score	Recall	Precision
LedNet (LW-Net)	0.826	0.834	0.821	0.853
LedNet (ResNet-101)	0.843	0.849	0.841	0.864
YOLOv3	0.803	0.803	0.801	0.82
YOLOv3 (Tiny)	0.782	0.783	0.776	0.796
Faster-RCNN (VGG)	0.814	0.818	0.814	0.835
Proposed method (SSD#2)	**0.750**	**0.816**	**0.769**	**0.869**



(a) Image Source: Maksym Kozlenko, CC BY-SA 4.0, Wikimedia Commons [23] (b) Image Taken by the Authors (c) Image Source: Neeloapple, CC BY-SA 4.0, Wikimedia Commons [23] (d) Image Source: ElbeObst, CC BY-SA 4.0, Wikimedia Commons [23]

Fig. 4: Cases when the model did not correctly detect all apples. Shown with red are the model's predictions, while blue denotes missed detections.

[4] H. Kang and C. Chen, "Fast implementation of real-time fruit detection in apple orchards using deep learning," *Computers and Electronics in Agriculture*, vol. 168, p. 105108, 2020. doi: 10.1016/j.compag.2019.105108. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169919314395

[5] H. Yu, S. Song, S. Ma, and R. O. Sinnott, "Estimating fruit crop yield through deep learning," in *Proceedings of the 6th IEEE/ACM International Conference on Big Data Computing, Applications and Technologies*, 2019. doi: 10.1145/3365109.3368766 pp. 145–148.

[6] R. Kestur, A. Meduri, and O. Narasipura, "Mangonet: A deep semantic segmentation architecture for a method to detect and count mangoes in an open orchard," *Engineering Applications of Artificial Intelligence*, vol. 77, pp. 59 – 69, 2019. doi: 10.1016/j.engappai.2018.09.011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197618301970

[7] A. Koirala, K. B. Walsh, Z. Wang, and C. McCarthy, "Deep learning–method overview and review of use for fruit detection and yield estimation," *Computers and Electronics in Agriculture*, vol. 162, pp. 219–234, 2019. doi: 10.1016/j.compag.2019.04.017

[8] M. Rahnemoonfar and C. Sheppard, "Real-time yield estimation based on deep learning," in *Autonomous Air and Ground Sensing Systems for Agricultural Optimization and Phenotyping II*, J. A. Thomasson, M. McKee, and R. J. Moorhead, Eds., vol. 10218, International Society for Optics and Photonics. SPIE, 2017. doi: 10.1117/12.2263097 pp. 59 – 65.

[9] H. Mureşan and M. Oltean, "Fruit recognition from images using deep learning," *Acta Universitatis Sapientiae, Informatica*, vol. 10, no. 1, pp. 26 – 42, 2018. doi: 10.48550/arXiv.1712.00580. [Online]. Available: https://content.sciendo.com/view/journals/ausi/10/1/article-p26.xml

[10] M. Oltean and H. Muresan, "Fruits 360 dataset on github," 2017, [Online; accessed 16.09.2021]. [Online]. Available: https://github.com/Horea94/Fruit-Images-Dataset

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015. doi: 10.1007/978-3-319-46448-0_2. [Online]. Available: http://arxiv.org/abs/1512.02325

[12] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning," *CoRR*, vol. abs/1602.07261, 2016. doi: 10.48550/arXiv.1602.07261. [Online]. Available: http://arxiv.org/abs/1602.07261

[13] H. Mureşan, A. Călin, and A. Coroiu, "Overview of recent deep learning methods applied in fruit counting for yield estimation," *Studia Universitatis Babeş-Bolyai Informatica*, vol. 65, no. 2, pp. 50–65, 2020. doi: 10.24193/subbi.2020.2.04. [Online]. Available: http://www.cs.ubbcluj.ro/~studia-i/journal/journal/article/view/58

[14] J. Fourie, J. Hsaio, and A. Werner, "Crop yield estimation using deep learning," in *7th Asian-Australasian Conference on Precision Agriculture*, 2017. doi: 10.5281/zenodo.893710 pp. 1–10.

[15] S. W. Chen, S. S. Shivakumar, S. Dcunha, J. Das, E. Okon, C. Qu, C. J. Taylor, and V. Kumar, "Counting apples and oranges with deep learning: A data-driven approach," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 781–788, 2017. doi: 10.1109/LRA.2017.2651944

[16] Q. Xiang, X. Wang, R. Li, G. Zhang, J. Lai, and Q. Hu, "Fruit image classification based on mobilenetv2 with transfer learning technique," in *Proceedings of the 3rd International Conference on Computer Science and Application Engineering*, 2019. doi: 10.1145/3331453.3361658 pp. 1–7.

[17] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018. doi: 10.48550/arXiv.1801.04381 pp. 4510–4520.

[18] R. Siddiqi, "Effectiveness of transfer learning and fine tuning in automated fruit image classification," in *Proceedings of the 2019 3rd International Conference on Deep Learning Technologies*, 2019. doi: 10.1145/3342999.3343002 pp. 91–100.

[19] P. Jiang, Y. Chen, B. Liu, D. He, and C. Liang, "Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks," *IEEE Access*, vol. 7, pp. 59 069–59 080, 2019. doi: 10.1109/ACCESS.2019.2914929

[20] P. Ferrari, "Ssd keras implementation," 2018, [Online; accessed 16.09.2021]. [Online]. Available: https://github.com/pierluigiferrari/ssd_keras

[21] TensorFlow, "Tensorflow," 2015, [Online; accessed 16.09.2021]. [Online]. Available: https://www.tensorflow.org

[22] K. Wada, "labelme," 2010, [Online; accessed 16.09.2021]. [Online]. Available: https://github.com/wkentaro/labelme

[23] WikimediaCommons, "Wikimedia commons - freely usable media files," 2004, [Online; accessed 16.09.2021]. [Online]. Available: https://commons.wikimedia.org/wiki/Main_Page