USING DEEP LEARNING - CONVOLUTIONAL NAURAL NETWORKS (CNNS) FOR REAL-TIME FRUIT DETECTION IN THE TREE

Kushtrim Bresilla, Giulio Demetrio Perulli, Alexandra Boini, Brunella Morandi, Luca Corelli Grappadelli, Luigi Manfrini

Department of Agricultural Sciences, University of Bologna, Bologna, Italy

A paper from the Proceedings of the
14th International Conference on Precision Agriculture
June 24 – June 27, 2018
Montreal, Quebec, Canada

Abstract. *Image/video processing for fruit detection in the tree using hard-coded feature extraction algorithms have shown high accuracy on fruit detection during recent years. While accurate, these approaches even with high-end hardware are still computationally intensive and too slow for real-time systems. This paper details the use of deep convolution neural networks architecture based on single-stage detectors. Using deep-learning techniques eliminates the need for hard-code specific features for specific fruit shapes, color and/or other attributes. This architectures take the input image and divides into AxA grid, where A is a configurable hyper-parameter that defines the fineness of the grid To each grid cell an image detection and localization algorithm is applied. Each of those cells is responsible to predict bounding boxes and confidence score for fruit (apple and pear in the case of this study) detected in that cell. We want this confidence score to be high if a fruit exists in a cell, otherwise to be zero, if no fruit is in the cell. More than 100 images of apple were taken. Each tree image with approximately 50 fruits, that at the end resulted on more than 5000 images of apple and pear fruits each. Labeling images for training consisted on manually specifying the bounding boxes for fruits, where (x, y) are the center coordinates of the box and (w, h) are width and height. This architecture showed an accuracy of more than 90% fruit detection. Based on correlation between number of visible fruits, detected fruits on one frame and the real number of fruits on one tree, a model was created to accommodate this error rate. Processing speed is higher than 10FPS which is fast enough for any grasping/harvesting robotic arm or other real-time applications.*

Keywords. *Computer vision, Deep learning, Fruit recognition, Harvesting robot, Precision agriculture*

Introduction

According to (Schrder, 2014), the agricultural workforce is expected to decline around 30% between 2017 and 2030. This expected decline will be driven by structural changes within the agri-food industry, but also because the opportunities for employment are expected to be better in other sectors. Rural areas are already facing difficulties in creating attractive jobs in general, pushing towards an ongoing migration towards urban centers. Those structural changes in agriculture are expected to continue with higher investments in technology. For example, investing in precision farming and digital agriculture are expected to significantly increase (Colbert et al., 2016). New technologies are set to impact the farm labor dynamic in many ways (Pierpaoli et al., 2013), but two developments stand out. One, the increasing use of data collection tools, such as sensors, and increasing sophistication of farm hardware and software is increasing demand for higher analytical and technical skill sets (Aubert et al., 2012, Mulla, 2013). And two, the advancement of automation and autonomy on farm will decrease the reliance on human resources for low-skill and labor-intensive work while increasing autonomous machinery and robotics presence (Bechar and Vigneault, 2016, Stafford, 2007).

During years, different approaches and techniques have been developed to tackle fruit detection and localization (Jiménez et al., 1999, Song et al., 2014). All of techniques until recently relied on feature extraction, be that, color, shape, reflectance etc... (Rahnemoonfar and Sheppard, 2017, Song et al., 2014). Besides the aforementioned techniques, a new ones which recently are gaining momentum and higher accuracy are deep learning techniques (Rahnemoonfar and Sheppard, 2017, Sa et al., 2016).

Deep learning (DL) is a sub field of machine learning (ML). While both fall under the broad category of artificial intelligence (AI), DL is what powers the most human-like AI applications (LeCun et al., 2015). DL makes it possible to automatically learn the proper features from the photos and exploit them efficiently to construct an accurate detector through supervised training. Usually, the local visual features are extracted by a so called convolutional neural network (CNN), and a successive classifier consists of a fully connected network (FCN). CNN preserve the spatial relationship between pixels by learning internal feature representations using small squares of input data. Feature are learned and used across the whole image, allowing for the objects in the images to be shifted or translated in the scene and still detectable by the network (Krizhevsky et al., 2012).

Background and Related Work

Until recent years, traditional computer vision approaches have been extensively adopted in the agricultural field. In recent years, with the significant increase in computational power, in particular with special purpose processors optimized for matrix-like data processing and large amount of data calculations (eg. Graphical Processing Unit - GPU), a lot of deep learning, CNN models and methodologies specifically have achieved breakthroughs never achieved before. (Sa et al., 2016), developed a model called DeepFruits, for fruit detection. Adopting a Faster R-CNN model, goal was to build an accurate, fast and reliable fruit detection system. Model after training was able to achieve 0.838 precision and recall in the detection of sweet pepper. In addition they used a multi-modal fusion approach that combines the information from RGB and NIR images. The bottle-neck of the model is that in order to deploy on a real robot system, the processing performance required is a GPU of 8GB or more. It is well known that all deep

leaning models, to have a high accuracy they need high number of data (Krizhevsky et al., 2012). In case of CNN, more images of the object of interest, the better the classification/detection performance is. In a model called DeepCount, (Rahnemoonfar and Sheppard, 2017) developed an CNN architecture based on Inception-ResNet for counting fruits. In order to use less training data, (Rahnemoonfar and Sheppard, 2017) used a different approach. They use another model to generate synthetic images/data to feed te main model to train on. Those generated images were simply a brownish and greenish color background with red circles drawn above it to simulate the background and tomato plant with fruit on it. They used twenty four thousand images generated to feed into the model. The model after was tested on real world images and showed an accuracy from 85-80%.

In this paper we present a CNN model for fast and quiet accurate fruit detection based on YOLO model (Redmon et al., 2015). By using those DL techniques, it was eliminated the need for hard-code specific features like specific fruit shapes, color and/or other attributes. The network consist of several convolution and pooling layers, tweaked and changed from the standard model. Those modifications made to the model, make it more accurate to detect objects of the same class on close proximity (eg. only apple fruits, or only pear fruits).

Materials and Methodology

Models

YOLO

YOLO is a state-of-the art convolutional network for detection and localization. There are different versions of YOLO, and in this paper we modified and used YOLO900 (also known as YOLOv2), and as such, in the remaining part of the paper, we refer to YOLO900 as YOLO. Compared to other state-of-the art methods that treat detection, classification and region extraction as different problems, YOLO does all in one pass (hence the name You Only Look Once). To achieve that, YOLO in one hand losses in accuracy but in other hand gains speed. YOLO takes as input an image of max size 608 × 608 and divides into S × S. Each grid cell is responsible for the bounding box whose center is at the location of the grid cell, and predicts B bounding boxes as well as confidence level and class probability. In a dataset with C class labels, the output tensor is S × S × (C +B × 5). In our modified model, the class C is equal to 1 since we train the network each time with one class (only apples to detect).

Modified model

Despite being one of the most popular state-of-the art model, YOLO has problems detecting small objects (Redmon et al., 2015). The main problem with YOLO is that, the model can detect only one object class per cell, making it very difficult to detect two apples at the same cell. And since in apple tree, we are dealing with small fruits in relation to the canopy, we made different changes to the model, resulting in different accuracy scores. The standard YOLO takes input image and divides into 13 × 13. In our case we scaled up to 26 × 26. This improved the detection, as at this division of input image, we approximately have the grid cell size approximately like the apple size (Figure 1). In addition, to increase the detection size, we changed the structure of the model by removing some layers. This results in decreasing of accuracy, however it increases the speed of the model significantly.

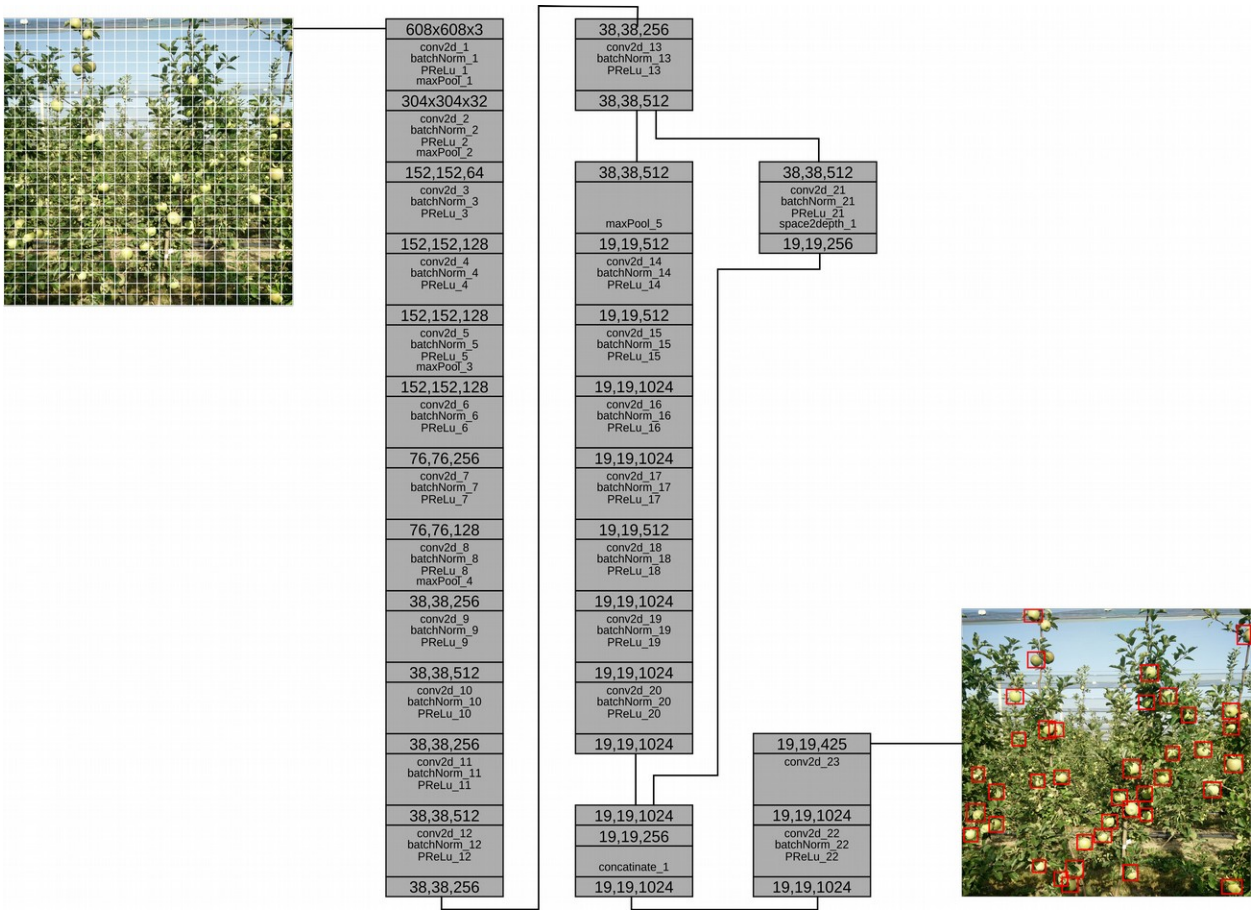

Figure 1. YOLO – Modified model

Dataset

Images were collected in the experimental orchards of University of Bologna - Department of Agriculture (Italy), and in privately owned production orchards in Ferrara (Italy). There have been taken multiple images of same tree with different angles, multiple sources (webcam, DSLR camera, smartphone) and during different time of the year with different weather and light conditions.

In total 100 images of apples and 50 of pears are taken. Each image containing approximately 50 fruits, resulting in more than 5000 images to train the models. Images were taken before the coloring of apples occur, thus all apples were still green.

We used all 100 images to train the model. However, we wanted to know if the number of images influences the accuracy of the model (as is always with DL approaches). We tested the model with 50 images, 100, 200 and with 400. Since the amount of images we had was 100, we used techniques like augmentation to reach the number of 200 and 400 images. Augmentation is a technique, where images are transformed, scaled, randomly modified, color shifted and changed, so they differ from original. This helps in two ways, first, it increases the number of images, and second, it makes the model not over-fit on the same data (Figure 2).

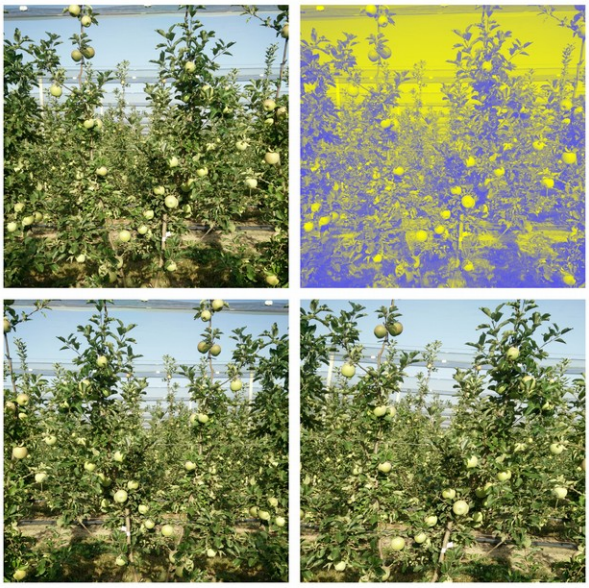

Figure 2. Augmentation of the same image into four others

Data preparation

Images are divided into Training set and Testing set. Testing images are never shown to the training procedure, as it is important for testing the accuracy of the models into images not seen before. This way we can see if the model is over-fit only on those training images, or is still able to generalize and detect fruits on images never seen before.

For labeling, we used a free and open-source labeling tool called BBoxLabelTool where each image went through process of labeling. Most of problems during labeling come because of occlusion and overcrowded image. This happens when one object is either partially or completely occluded by the other, and when a large amount of objects are close or attached to each other. Due to the nature of the fruit trees, this is present on every image taken of that tree. In each image, every apple fruit visible was labeled with a bounding box representing the location of apple fruit. This is done manually and very carefully to avoid mislabeling or occlusion. However, this tool annotates the data into PASCAL Visual Object Classes format, and in our case we needed the DARKNET format. All labeling data were automatically converted for each bounding box with a Python script.

Training

YOLO originally is designed to run in DARKNET, which is an open-source deep learning library written in C. However, in our case we use YOLO in Keras with TensorFlow back-end (another deep learning framework written in C and CUDA with Python bindings). Keras is a very high level abstraction for many deep-learning frameworks that makes very easy for us to make changes in the network and test the changes immediately. In addition, with Keras we can easy use preprocessing techniques like transformation and augmentation before the image hits the first layer.

We trained the model on Amazon E3 cloud instance with NVIDIA Tesla K80 12GB GPU. We use the stochastic gradient descent optimization method with 60 steps and the Adam optimizer with 0.002 learning rate to minimize the cost function. Choosing number of epoch is very tricky,

as the number of epochs is related to the number of rounds of optimization that are applied during training. With more rounds of optimization, the error on training data will be reduce further; however there is a point where the network becomes over-fit to the training images and will start to lose performance of generalization to unseen images. To avoid this, we monitored the error performance on testing images while the number of epochs increases. We ended up with 35 epochs as this resulted in the best accuracy while still maintaining the generalization on other images. All weights were initialized randomly.

Metrics

The results will be will be evaluated using the data (images with labels) from the testing and validation set. The metrics evaluating the accuracy will be according to the well known criteria based on Pascal Visual Object Classes (VOC) that many research on this field use. Pixel-wise accuracy was measured by comparison of ground truth and predicted information. F1 score is used to evaluate the accuracy of the model. While for measuring speed, frames per second (FPS) is used.

Results

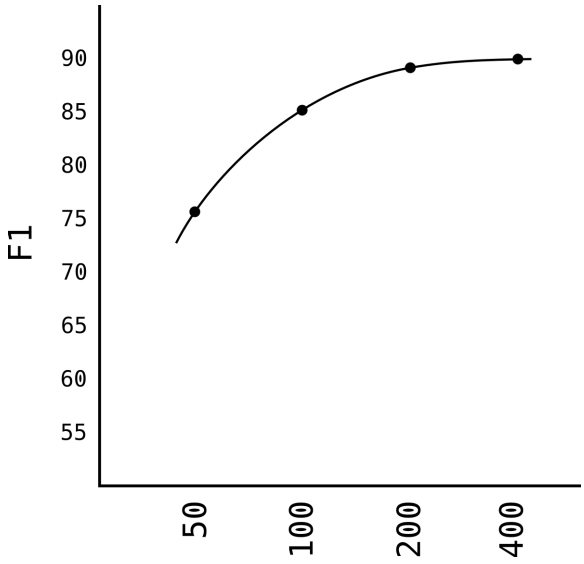

Figure 3. Accuracy of the model related to quantity of images

There are in total 100 images, in first set, we trained the model with 50% of them. Approximately from 2500 apples different features were learned in those 50 images. Threshold of detection was set to 60%. If bounding box predicted for an apple with more than 60% accuracy, then it was refereed as apple, a bounding box was drawn to it and was counted. After 35 iterations, the model was able to have an accuracy F1 score of 0.75. When the number of images was increased to 100, respectively 5000 images of apple fruits, F1 score reached 0.84.

We used random augmentation techniques like flipping, rotation to increase the number to 200. When this was applied to the dataset, and the model was trained in those additional data, F1

score reached 0.88 F1 score. To increase this number further to 400 images, we applied color shifting and random noise as augmentation. After retraining, the model reached the highest F1 score with 0.9 accuracy.

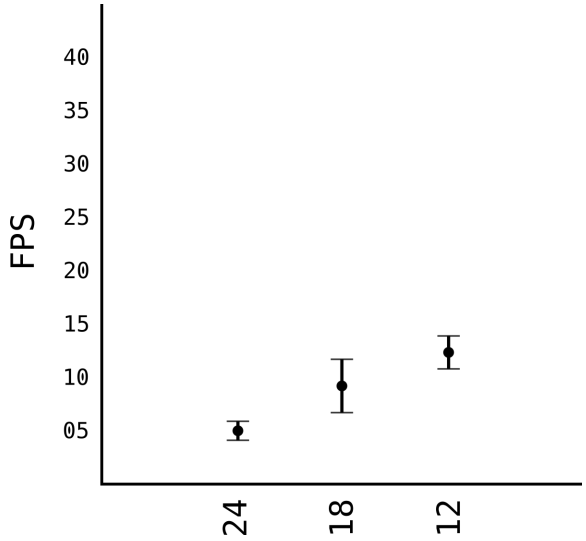

Figure 4. Model speed based on different layer number

When we increased the number of grid from 13 × 13 to 26 × 26, we significantly decreased the speed of the model. This due to the processing power needed to check for objects and assign bounding boxes, four times more than the original mode. In order to accommodate for that, we made the model more shallower. We removed some layers, from the model (Figure 4). This goes against the principle of deep-learning that more deeper the model the better the accuracy. However, to reach a more suitable FPS rate, we tested different FPS that still will give us more than 80% accuracy. While with original 24 layers of the model, the speed was very low at only 4FPS per seconds. However with 12 layers, the speed goes above 10FPS.

Cropload

Estimating fruit number based just on detected fruit from the model is not very accurate, despite the model being more than 90%accurate. This not due to the model detection pipelin e or architecture, but because of the nature of the tree itself. When an image is taken of the tree, there are still apples that are not visible even by human looking at the image. Even looking at the tree from two meters far (the same distance the images were taken) is difficult to see fruits inside canopy. To solve this, we correlated the number of visible apples and hidden ones. We counted every fruit in a tree outside in the orchard where the images were taken, and then we counted again each fruit in the image (manually). We found out that, depending on the training system this number varies from 85% visible to 95% visible (Figure 5). Thus we calculate the tree cropload as below:

$$cropload = d_i + \left(d_i \times (1 - F_1)\right) + \left(t_i \times \left(d_i + \left(d_i \times (1 - F_1)\right)\right)\right)$$

where d_i is detection number by the model, F1 accuracy score, and t_i is percentage of hidden in that training system, in our case was 0.05 in one type of training system and 0.1 in another one.

Figure 5. Detection of apple fruits in different training system

Conclusions

In this paper we presented an approach for fruit detection based on state-of-the art deep neural networks techniques using single shot detectors (YOLO) as an convolutional neural network to detect fruits of apple and pears in the tree canopy. This study demonstrates that modifications like the input grid on the standard model of YOLO yields better results. Image quantity is very important in all deep-learning approaches. We demonstrated that with increase of number of mages, the model is able to detect apple fruits with higher accuracy. In the lack of image quantity, there are techniques like augmentation that modify the original image into similar one, but with enough changes that the model learns and generalizes to avoid over-fitting.

Current limitation of our platform is the compute power needed for the system to run. Because most neural networks have many layers, especially convolutional neural networks, the most suitable to run the models are the CUDA capable devices. Our continuation of this work will include more images of different fruit species, at different growth stages, with different training systems, and using a mobile platform for capturing those images.

References

Aubert, B. A., Schroeder, A., and Grimaudo, J. (2012). It as enabler of sustainable farming: An empirical analysis of farmers adoption decision of precision agriculture technology. Decision Support Systems, 54(1):510–520.

Bac, C. W., van Henten, E. J., Hemming, J., and Edan, Y. (2014). Harvesting robots for high-value crops: State-of-the-art review and challenges ahead. Journal of Field Robotics, 31(6):888–911.

Colbert, A., Yee, N., and George, G. (2016). The digital workforce and the workplace of the future. Academy of Management Journal, 59(3):731–739.

Jiménez, A., Jain, A., Ceres, R., and Pons, J. (1999). Automatic fruit recognition: a survey and new results using range/attenuation images. Pattern Recognition, 32(10):1719–1736.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet clas sification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, NIPS'12, pages 1097–1105, USA. Curran Associates Inc.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. Nature, 521(7553):436–444.

Mulla, D. J. (2013). Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps. Biosystems Engineering, 114(4):358–371.

Pierpaoli, E., Carli, G., Pignatti, E., and Canavari, M. (2013). Drivers of precision agriculture technologies adoption: A literature review. Procedia Technology, 8:61–69.

Rahnemoonfar, M. and Sheppard, C. (2017). Deep count: Fruit counting based on deep simulated learning. Sensors, 17(4):905.

Redmon, J., Divvala, S. K., Girshick, R. B., and Farhadi, A. (2015). You only look once: Unified, real-time object detection. CoRR, abs/1506.02640.

Sa, I., Ge, Z., Dayoub, F., Upcroft, B., Perez, T., and McCool, C. (2016). Deep-Fruits: A fruit detection system using deep neural networks. Sensors, 16(8):1222.

Schrder, C. (2014). Employment in european agriculture: Labour costs, flexibility and contractual aspects.

Song, Y., Glasbey, C., Horgan, G., Polder, G., Dieleman, J., and van der Heijden, G. (2014). Automatic fruit recognition and counting from multiple images. Biosystems Engineering, 118:203–215.

Stafford, J., (2007). Precision agriculture '07. Wageningen Academic Pub-lishers.