2019 International Conference on Cutting-edge Technologies in Engineering (ICon-CuTE)
Shri Ramswaroop Memorial University, Barabanki, India, Nov. 14-16, 2019

Vegetable Classification Using You Only Look Once Algorithm

Sachin C
Dept of Computer Science and Engineering
Amrita School of Engineering, Bengaluru
Amrita Vishwa Vidyapeetham, India
sachinc97@gmail.com

N Manasa
Dept of Computer Science and Engineering
Amrita School of Engineering, Bengaluru
Amrita Vishwa Vidyapeetham, India
manasa.sivaram@gmail.com

Vicky Sharma
Dept of Computer Science and Engineering
Amrita School of Engineering, Bengaluru
Amrita Vishwa Vidyapeetham, India
vickysharma1711@gmail.com

Nippun Kumaar A. A.
Dept of Computer Science and Engineering
Amrita School of Engineering, Bengaluru
Amrita Vishwa Vidyapeetham, India
nippun05@gmail.com

***Abstract*—Vegetable detection and classification is a challenging objective in daily production and use, and the complexity increases when other parameters such as shape, size and color are taken into consideration. In this paper, we define a methodology that will detect and classify three different green vegetables of different sizes. This method involves the use of Tensor Flow, Dark flow which is a Tensor flow version of You only look once (YOLO) algorithm, OpenCV. To train the desired network, several various vegetable images were fed into the network. The images were pre-processed before training by drawing bounding boxes around the vegetable manually using OpenCV. The main algorithm responsible for the detection and classification is YOLO. This method provides a faster and smarter way to identify an object in the given image or video. Once the network is trained, the test input is passed in to the network. Once the input is provided to the network, the output will display bounding boxes around the recognized vegetable and label it with its predicted class category with accuracy of 61.6 percent.**

Keywords—Fruits, Vegetables, YOLO, Bounding boxes, Detection, Complex background, Classification.

I. INTRODUCTION

Fruits and vegetables are a constant and main source of nutrition for humans, animals and all other living creatures. Now, with the ever-increasing population of all living beings, the demand for food also increases at twice the previous requirements. Such a massive demand requires extreme and intensive labor work by farmers with day and night monitoring of the farms. In addition to the increasing population, the weather change also affects the produce. Untimely rain and scorching heat damages all the manual effort put in by the farmers. Also, in addition if certain produce is not harvested at the right time, the value of the produce depreciates and is not worthy of being harvested. So, there rises a need for harvesting in the right time as well.

The application of technology in the field of agriculture has been implemented for a very long time now. From having bullock carts whose purpose is to plough to planting by making use of automatic tractors do the same. From making use of human effort for harvesting to usage of machines for more efficient and less human-intervention process. Technology has not only helped in reducing human effort but also meeting the ever-increasing needs of produce. For such a vast and expanding field, having an algorithm that would automatically detect and classify the right produce will be of immense help to everyone, not only the farmers.

Vegetable detection and classification involves the development of two important applications merged as one. The various algorithms being provided for this field are Convolutional neural network (CNN), R-CNN where R stands for region, Fast RCNN, YOLO, etc. A detailed research led us to the development using YOLO algorithm. YOLO, You only look once, as the name suggests provides efficient detection of objects based on the training data provided to the network. Scanning of the input is done once per frame update and within that span of time, the required objects within that frame are identified.

There have been numerous methods for the detection and classification of vegetables. Methods like CNN, RCNN and Fast RCNN involve only specific regions of interest, and these regions are used to localize the object within the image. The above-mentioned networks do not look at the complete image. Those regions of interest are provided as input the required model's network and the training is done only for those objects within the region. YOLO is much different from the region-based algorithms explained above. In YOLO, the class probabilities and the bounding box details can be achieved by using just one convolution network,

Section II of the paper briefs about the motivation and related works, section III of the paper explains the architecture of You Only Look Once (YOLO) algorithm, section IV of the paper explains our approach to detect and classify the vegetables correctly and accurately, section V of the paper talks about how well our algorithm was successful in identifying and classifying vegetables and finally section VI talks about the future scope of this solution.

II. RELATED WORKS

[1] Develops a system capable of differentiating between the vegetables and other components such as leaves, stem, etc. Preliminary study included, a hyperspectral study which would determine the wavelengths to differentiate between vegetable and other components. To identify the different initial regions of interest a SVM classifier is used which is trained on multiple color information. This is further used to generate a mask around the result for the vegetables. The necessary "masked-regions" detected were classified as

vegetable or leaf or stem and cropped out based on a cropping window of a particular size. Next, a logical AND operation is conducted between the edited images detected as cucumbers and the mask of blobs obtained from the pixel-based classification map. In this way, only cucumber pixels are retained. Also, segmentation is used for pixels with multiple cucumbers. The overall processing allows the classification of vegetables and other components separately. The final result involves the drawing of an ellipsoid around the detected vegetable.

[2] Proposed a solution for detection of plant diseases using image processing. The process is divided into the following steps. RGB images of the leaf are acquired. Color space transformation is later applied to them. Image pre-processing such as noise removal, cropping, smoothing, enhancement are then applied to the transformed image. After this, segmentation, partitioning the image based on similarity is applied such as 'otsu method, k-means clustering, converting RGB to HIS model. Then, feature extraction helps in extracting only the necessary information from the image which would act as an input to the neural network, and for this color co-occurrence method or using the H and B components are can be used. Once the necessary features are extracted, the data is fed into the network and classification the input occurs, based on the weights updated in the network. BPNN can be used to update and refine the weights even more based on the accuracy required. After the model is trained, it is tested with certain image samples for diseases detection and classification. This paper was solely based on the application of image processing in this field and yield's favorable results.

[3] Proposed a vegetable detection solution in a complex background. Dataset included manually clicked 5000+ tomato images from the farm. Implementation involved usage of the RCNN with TensorFlow and feature extraction involved the ResNet 50 which provided excellent results. K-Means allows it to cluster based on the necessary features extracted. Model is trained using this data with a Linux machine and NVidia 1080Ti. Implementing the ResNet 50 with RCNN provided improved results and an accuracy of over 92%.

[4] Proposes an efficient solution for classification and detection of vegetables and fruits. Image dataset is cleaned using image saliency to remove noise and unwanted features. Using this data with traditional RCNN resulted in 85% efficiency. Later VGG model was used to detect and classify and results improved up to 96% true positive results.

[5] Presents a traditional solution to preprocess an image by enhancing the contrast between important and unnecessary features of an image. The next step involves feature extraction to segregate regions of interest from the image. This further involved to train the CNN network with the feature extracted images to generate bounding boxes around regions of interest. The PASCAL VOC dataset was implemented to train. Drawbacks were grouping of objects, complex background, low light images went unnoticed with average accuracy.

[6] Proposed a solution to classify vegetables which involves pre-processing, training, verification and testing processes. Pre-processing involves the labelling of the data. Next, training of the labelled data was implemented using the transfer learning of the Inception-v3 model of TensorFlow. This was later bundled into an android application to classify various vegetables. The data set was manually developed using hand clicked images from source and labelling them. Each vegetable consisted of a minimum 150 images for better efficiency and accuracy. The images were labelled if the data set of that species consisted of very few images.

[7] Proposes an object detection algorithm using YOLO on the PASCAL and Picasso Artwork dataset. 2/3rd of PASCAL VOC dataset was used as training and rest as validation dataset. Picasso Artwork was set as testing dataset because artwork is different from real world images. The implementation of YOLO was done with TensorFlow in Python. For testing, 3 fold cross validation was done and also exponential moving average which improved the performance and prevented overfitting. Testing on CPU was implemented which provided better than average results. Results included some false positive outcomes too.

[8] Present a paper which explains the working of the YOLO model by providing a complete implementation and detailed comparison between YOLO and other Neural Network models for object detection.

[9] Provides a research study of the comparison of the various models available for object detection. The study includes cnn, rcnn, fast rcnn, faster rcnn, and all versions of yolo. The results show the best performance with yolo and also it yields more accurate results. Yolo also takes the least processing time to compute the results and has a better generalization of representation of objects.

[10] Provides a robot to detect and differentiate between colored balls and avoid obstacles. The main algorithm to detect objects is based on hough transform.

[11] Presents a solution for detection of hidden objects for classification of threat. Dataset used was stereo thermal dataset. Precision rate for segmenting and detection was 88.89%.

The various work in the literature here have their pros and cons. Most of the papers focus on object detection. [1], [2], [3] and [4] specifically help in identifying plant features. [5] accompanies by providing a solution for detection in complex background but, provided average accuracy. [6] and [7] present efficient solution for identifying and mainly classifying objects into their categories. [8] and [9] provide a comparative study of various algorithms that could be implemented to provide this solution. [10] and [11] implement object detection into different domains other than vegetable detection and classification to provide efficient results. Our main focus is on accurate and faster object

III. ARCHITECTURE OVERVIEW

YOLO is an object detection system that is targeted mostly for real time processing. The basic concept of YOLO is that You Only Look Once. With this motive, it scans around once as it moves and is still able to detect and classify images based on the previously trained set that it was fed with.

This algorithm requires the basic setup for training and then testing these models with the required versions (as it differs version to version). YOLO has overtaken its otherwise traditional CNN. CNN is like YOLO except that it isn't all that efficient for real time object detection. YOLO uses a single CNN which predicts both the bounding boxes

and the class probabilities for those boxes at the same time. YOLO trains on full image sets and can directly optimize detection performance. YOLO is chosen over CNN because YOLO is extremely fast thanks to its frame detection. Secondly, while making predictions, it reasons globally unlike the sliding window and region proposal-based techniques of CNN. Thirdly it is due to the quick learning of YOLO about generalizable representations of objects.

YOLO's implementation involves the splitting of an image into a P x P grid where, within each grid, 'n' number of bounding boxes are created. The characteristic features of each box i.e., the dimensions of the bounding box and the different class probabilities is found by the network. A threshold is set and only the bounding boxes whose class probability is more than the threshold is used to detect the object within the box.

Coming to the Architecture of YOLO, the importance should be given to how YOLO encodes its output. The image which has been given as an input is divided into an N x N grid. There might be many objects in the image and only one cell of the grid, the cell that is present at the center of object helps predicting the presence of the object. Each cell is part of 5 bounding boxes and each bounding box has 5 different characteristic components which is presented as (x, y, w, h, c).

The (x, y) are the coordinates of the center cell of the box. The (w, h) are the width and height of the bounding box.

The last component c is called the confidence score which predicts if the object is present in the box or not. If the object is not present in the box then the confidence score should ideally be zero and if the object is present then the confidence is calculated by using something called the intersection over union of the predicted box and the truth. In addition to these characteristics YOLO also calculates the class probabilities. Class probability means the probability that that the object in the box belongs to that class. So, each cell is responsible for predicting one class probability and one ends up getting N x N x C probabilities, where C is the no of classes.

Pre-processing of the image is done to eliminate the noise and presence of outliers and for better enhancement of the image to get a better run with the algorithm. In this algorithm, although various pre-processing techniques can be used, the main focus here is on Non- Max Suppression (where one selects the best bounding box out of the five predicted bounding boxes) as shown in Fig.1.

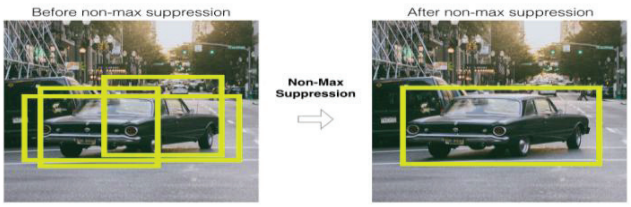

Fig. 1. Non-max suppression used to eliminate the unwanted bounding boxes

The network structure looks like a normal CNN, it contains 24 convolutional layers followed by 2 fully connected layers at the end. The network architecture of YOLO is inspired by the GoogLeNet model. There is also a fast version of YOLO called Fast YOLO which uses 9 convolutional layers rather than 24 and the remaining parameters except the size of the network remains the same. The Yolo architecture is shown in Fig. 2.

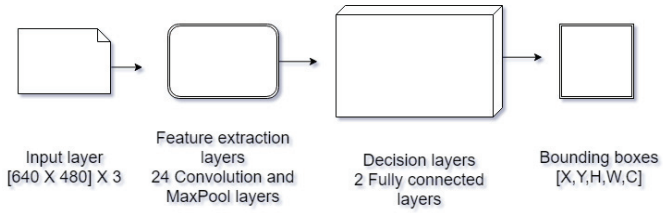

Fig. 2. Complete YOLO network Architecture

There are few limitations of YOLO. YOLO is not so efficient in detecting small objects in large groups due to the limitation of spatial arrangement of grid cells that contribute to form the bounding box. YOLO majorly learns from the data so the algorithm fails to detect in new or modified configurations and aspect ratios.

IV. IMPLEMENTATION

There are quite a few different implementations of YOLO algorithm and **Darknet**, which is an open source neural network framework is used in our system. Open source computer vision library (OpenCV) is mainly aimed around real-time computer vision. Originally written in C++, it has been extended to other programming languages as well. The Python library of OpenCV, which is called cv2 will be used. And the four major functions of cv2 are mainly being used here where, one is to draw the bounding box depicting the top left corner and bottom right coordinates. This function is used to draw rectangles based off the given coordinates. Second function to add the predicted class labels and the confidence scores on each bounding box. Third function to read the image for which one has to detect class labels for. Last function to read a video from the local storage and predict the class labels to it. It can also be used obtain real time camera data using a webcam or any other computer cameras

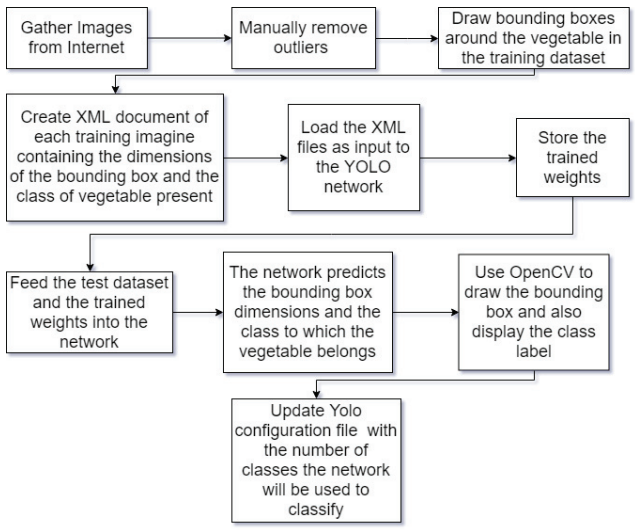

Fig. 3. Complete implementation flowchart

Tensor flow runs on both GPU and CPU. The implementation of YOLO on the GPU version is faster than that of the CPU version. Due to the hardware constraints, YOLO has been implemented on an Intel i5- 6th gen

processor with 8GB RAM and also, ran the tensor flow CPU version. For a small comparison, YOLO running on GPU can process a video at 40-100 FPS based on the GPU specifications whereas YOLO running on a CPU version processes a video at 3-8 FPS based on the CPU specifications. Moreover, the training of custom dataset is 10 times faster on a GPU version. The complete implementation flowchart is shown in Fig. 3. Tensor Flow, Darkflow, Python are a few software technologies that have been used in this work. Along with these, pre-trained weights from internet sources (like Pascal VOC dataset) have also been used to validate the algorithm.

A. Training

The first step in training is to gather the related images which is basically images of different types of cucumber, green apple and green capsicum images from internet sources. In order to get an efficient and high accuracy classification, train as many images. A total of one hundred images of each vegetable were taken from various internet sources, of which 60 images were used for training and the other 40 used for testing.

YOLO configuration file contains many variables which determine the characteristic features of the network, these variables must be changed to suit our inputs and outputs accordingly. For example, the number of classes the network has to identify would be different in different situations. In this case, there are three classes i.e., cucumber, green apple and green capsicum. The number of output channels of a layer is given by filters variable. Once the images are available, the YOLO configuration file must be edited. The number of filters must also be updated and is given by the formula:

$$\text{Number of filters} = 5*(5+\text{number of classes}) \qquad (1)$$

The number of filters for the model would be 40 and the filter variable should be altered accordingly. A gist of the changes in the configuration file is shown in Fig. 4.

```
107 batch_normalize=1
108 size=3
109 stride=1
110 pad=1
111 filters=1024
112 activation=leaky
113
114 [convolutional]
115 size=1
116 stride=1
117 pad=1
118 filters=40
119 activation=linear
120
121 [region]
122 anchors = 1.08,1.19,  3.42,4.41,  6.63,11.38,  9.42,5.11,  16.62,10.52
123 bias_match=1
124 classes=3
125 coords=4
126 num=5
127 softmax=1
128 jitter=.2
129 rescore=1
130
131 object_scale=5
132 noobject_scale=1
133 class_scale=1
134 coord_scale=1
135
136 absolute=1
137 thresh = .6
138 random=1
```

Fig. 4. Editing the configuration file to change the number of classes and the corresponding filter calculated using the formula

An xml document is then created that records the top-left and bottom-right coordinates of each image in the training dataset. Then, there is a rectangular selector that performs this operation which runs on a python script that is written. This rectangular selector is used as shown in Fig. 5, to point out where the vegetable is present in each of the training images and a corresponding xml file is created which stores the top left coordinates and the bottom ride coordinates and the class to which the vegetable belongs. This xml files will be later fed inside the network to train the network.

In order to train the network two sets of information are needed, one being the image and the other being the xml document to be fed into the network. In addition to the information one provides to the network, a pre-trained dataset like yolo-tiny should be loaded. The learning rate for training is set to 0.001, and the epochs as 300, overwriting the default 1000. The entire dataset is divided into batches of size 16 images each and in each step of training the entire batch will go through the hidden layers and the weights will be updated correspondingly. Since the dataset contains 180 images, there are 11 batches and each epoch has 11 steps. The number of steps in an epoch is equal to the number of batches we have. At the end of each step all the images in that batch would have gone through all the hidden layers, the weights (error) would be back-propagated and the mean error is calculated after each step. And so, at the end of each epoch, all the images would have gone through the hidden layer once and the mean error would be calculated. If the average error does not change between consecutive epochs, the training is stopped or the learning rate is changed. The training was stopped after 137 epochs where the average error was found to be between 4 to 6 and did not lower further. At the end of every 125 steps yolo will store the weights file in the local directory. These weight files are used to test our model.

Fig. 5. Drawing the bounding box around the training images to record the coordinates where the object is present.

B. Testing

To correctly detect and classify the presence of the vegetable in an image, a script is run that loads the trained YOLO model which in-turn loads the dataset that needs to be classified. The script also identifies the coordinates and the class of the vegetable. A bounding box is drawn using OpenCV that displays the class label along with the confidence score. A threshold of 0.15 is set which means that the model classifies only those bounding boxes that have confidence score of 0.15 or greater.

The same is done for classifying vegetables in a video. Each frame is taken from the video which classifies each image present on the frame same as any other image with a threshold of 0.15. The video processing requires high computing power or else the object detection and classification on a video will be very slow.

In order to use this algorithm to correctly classify one of the three classes in real time, a script is written which loads the trained model and captures video from the webcam. The script draws bounding boxes for each image in each frame by passing the frames to the algorithm. Here as well, the same threshold of 0.15 is maintained. The webcam video processing happens at different speeds based on the computing power of the system for a small comparison, the webcam processing happens at 4-6 FPS on a CPU version of tensor flow and runs at 40-80 FPS on a GPU version.

V. RESULTS AND ANALYSIS

The dataset that has been used consists of 180 test images that comprises of mixed settings of all the three vegetables viz. it being horizontally placed or vertical or under complex backgrounds or as a bunch.

It was decided to set the learning rate to 0.001 to speed up the training process. The batch size was set to 16 and the model was set to run for 300 epochs, while each epoch consists of 11 steps since the batch size was set to 16. At the end of 100 epochs the average error rate was lowered down to 4-6. On further training the dataset at about 137th epoch the average error didn't change much between successive steps so the training was stopped at that point and the trained weights were stored to test the accuracy of the model.

This model was later tested with sample testing images, and also few sample videos. For the images as test set, more than 50% of the images were detected appropriately and the vegetables were classified correctly and when a video was provided as an input to the model, 70% of the vegetables were properly detected and more than 70% of time the vegetables were correctly classified. The model was successful in detecting vegetables and classifying them in varying conditions and orientations which included vertical, vegetables in a bunch, and some of complex backgrounds were successfully detected. The model also gives a confidence score of each of its prediction and that prediction score was greater than 50% for almost all the images.

A few things that were inferred from this model-

TABLE I. GENERAL STATISTICS OF THE PERFORMANCE OF THE ALGORITHM

Number of images tested per vegetable	40
Average confidence score	57.6%
Percentage of images with true positive	60%
Number of images in which multiple vegetables have been detected	70%
Percentage of images with false positive	50%
Percentage of images with confidence score greater than 50%	55%
Number of images with confidence greater than 80%	10%

A. Cucumber Detection

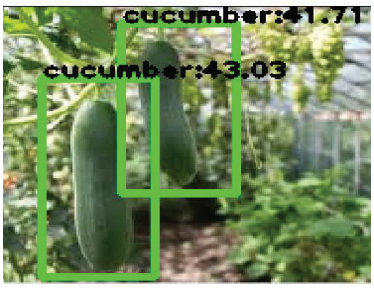

Fig. 6. The model detecting multiple cucumbers correctly and with a high confidence score

Fig. 7. The model fails to detect few cucumbers in few bad orientations and complex backgrounds

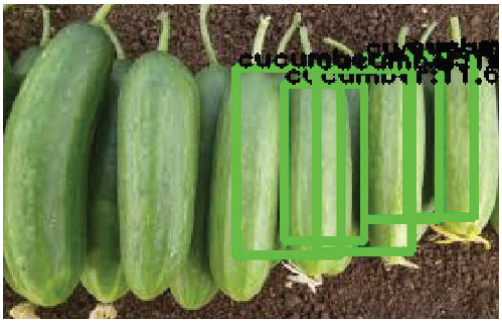

Fig. 8. The model gives a lot of false positives because of the lack of enough training samples

B. Green Apple Detection

Fig. 9. The model detecting green apple correctly

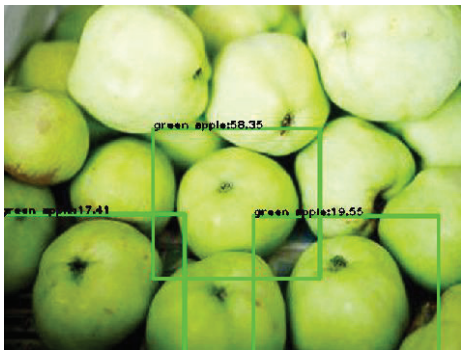

Fig. 10. The model detecting multiple green apples in an image

C. Green Capsicum Detection

Fig. 11. The model detecting and classifying green capsicum correctly and with a high confidence score

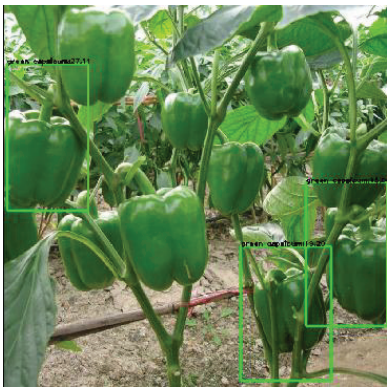

Fig. 12. The model detecting multiple green capsicums in an image but also fails to detect a few.

Therefore, the model was successful in detecting and classifying the three vegetables. The model successfully detects cucumber with a high confidence score of over 40% and it is also successful in detecting multiple cucumbers correctly as shown in Fig.6. There are a few places where the model gives false positives by detecting half cucumber as a full cucumber, this happens when there are lot of cucumbers in a picture as shown in Fig. 8. There are also a few places where the model fails to detect a vegetable due to unknown orientation of the vegetable as shown in Fig. 7. The detection and classification of green apples was also successful with a high confidence score as shown in Fig. 9. With the limited number of images that this model has been trained with, the model detects and classifies vegetables with an accuracy of about 61.6%. As shown in Fig. 10, the model is also successful in detecting multiple green apples in an image and that with a high confidence score. The green capsicum detection results were also great, as shown in Fig. 11 the model correctly detects and classifies green capsicum and is successful in detecting multiple capsicums under complex backgrounds as shown in Fig.12. A few statistics that were observed with the detection and classification of the three vegetables are shown in Table II.

TABLE II. VEGETABLE DETECTION AND CLASSIFICATION STATISTICS

Parameters	Cucumber	Green Apple	Green Capsicum
Number of training images	60	60	60
Number of test images	40	40	40
Successful detection and classification	70%	55%	60%
Multiple vegetables in an image detected	50%	50%	55%
False positive percentage	40%	50%	60%
Undetected vegetable	40%	40%	50%

VI. CONCLUSION AND FUTURE SCOPE

A model to detect and classify three different vegetables is built and the proposed algorithm has been implemented, trained, and tested. This model efficiently detects and classifies 60-70% of the harvest under various constraints and is also successful in detecting multiple vegetables in an image. With a threshold set to 0.15, almost all harvest ready vegetables are classified. Given that the images that were used are mostly obtained from internet sources, 70% of it were successfully classified accurately. With off field images fed in as our training data, there would have been a more efficient training set and much more accuracy in classifying each of the images.

As a further extension of this model, for autonomous harvesting system, depth information is essential to distinguish between foreground and background produce. This can be achieved by using 3D images and altering the system to use 3D image instead of a 2D image for training parameters like size, texture, colour, etc.

REFERENCES

[1] Roemi Fernandez, Hector Montes, Jelena Surdilovic, Dragojlub Surdilovic and Manuel Armada, “Automatic Detection of Field-Grown Cucumbers for Robotic Harvesting”, FP7 Project ECHORD++, Vol 6 - 2018, June 28 2018

[2] Sachin D Khirade and A B Patil, “Plant Disease detection using image-processing”, 2015 International Conference on Computing Communication Control and Automation, 26-27 Feb. 2015

[3] Jun Sun, Xiaofei He, Xiao Ge, Xiaohong Wu, Jifeng Shen, Yingying Song, “Detection of Key Organs in Tomato Based on Deep Migration Learning in a Complex Background”,

[4] Guoxiang Zeng, “Fruit and vegetables classification system using image saliency and convolutional neural network”, 2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC), 3-5 October 2017.

[5] Sandeep Kumar, Aman Balyan, Manvi Chawla, “Object Detection and Recognition in Images”, International Journal of Engineering Development and Research, 2017

[6] Om Patil, Prof. (Dr.) Vijay Gaikwad, “Classification of Vegetables using TensorFlow”, International Journal for Research in Applied Science & Engineering Technology, April 2018

[7] Yihui He, “Object Detection with YOLO on Artwork Dataset” Advanced Computer Vision at Jiaotong University, 2016

[8] Joseph Redmon, Santosh Divvala, Ross Girshick and Ali Farhadi, “You Only Look Once: Unified, Real-Time Object Detection”, ONR N00014-13-1-0720, NSF IIS-1338054, 9th May 2016

[9] Juan Du, “Understanding of Object Detection Based on CNN Family and YOLO”, Journal of Physics: Conference Series, 2018

[10] Karthi Balasubramanian, Arunkumar R, Jinu Jayachandran, Vishnu Jayapal, Bibin A Chundatt, Joshua D Freeman, “Object Recognition

and Obstacle Avoidance Robot", Chinese Control and Decision Conference, 2009

[11] K S Gautham, Senhtil Kumar Thangavel, "Hidden Object Detection for Classification of Threat", International Conference of Advanced Computing and Communications Systems, 2017