Grape Cluster Detection using Faster-RCNN, YOLO and SSD architecture

***Abstract*—Over the past decade, with the advent of convolutional neural networks, research and applications in the field of computer vision has become widespread. Computer vision can be used in the automated monitoring and management in the agriculture industry. Object detection is one of the most important and popular computer vision tasks. In this work, detection of grape clusters is performed which can be useful in monitoring of vineyards, yield estimation and automated harvesting. We use CNN based deep learning architectures like YOLO, Faster R-CNN and SSD to perform the Grape Cluster detection and compare the results obtained.**

I. Introduction

Computer vision can be useful for automation in the agriculture and horticulture industry. It can help in automated monitoring and management of the growth of the plants and fruits with lesser human intervention resulting in reduced cost. One of the preliminary steps for the monitoring and management, is the detection and localization of plant leafs, fruits, etc.

So as to detect and localize, object detection which is a popular computer vision technique can be used. Object detection has been used in multiple applications such as pedestrian detection, video surveillance, face detection, etc. Based on the type of application, there could be a single or multiple objects required to be detected. The standard approach is detect the presence of a object, identify the class of the object and localize the object by a drawing bounding box around it.

Object detection methods could be classified into either machine learning based approaches or deep learning based approaches. In case of machine learning based approaches, the features have to be handcrafted and then have to be extracted from the images. These extracted features can be provided as a input to machine learning classifiers such as SVM for classification and identification of the objects. Some of the features used in this approach are SIFT [1], Histogram of Oriented gradients (HOG) [2] and Haar [3].

In case of deep learning based approaches the features are automatically learnt from the training images by making use of Convolutional neural networks (CNN). The feature extractors (kernels/filters) of the CNNs are randomly initialised and the they are updated through back-propagation while minimising the desired cost function.

CNN based object detection methods have been quite popular because of the automatic feature extraction capability based on the training images. There are two major types of Object detection methods. The first type of methods performs detection in two stages. In the first stage a region where object could be present is proposed and in the next stage the object is identified, classified and localized by a bounding box. Example of two stage methods are R-CNN, Fast R-CNN, Faster R-CNN, etc. The second type of methods perform detection of objects in a single stage. Example of such methods are YOLO [4], EfficientDet [5], etc.

In this project, we perform detection of wine grape clusters which can help in monitoring and management of vineyard. Detection of grape clusters is one of the preliminary steps in yield estimation by counting the grapes, nutrient deficiency monitoring and disease detection. We use two stage detection method as well as single stage detection method for grape cluster detection. Specifically we perform detection using Faster R-CNN, YOLO and SSD compare the results obtained from these methods.

II. Related Work

A robust grape cluster detection method is proposed in [6] using Adaboost classifier and the color components of the image. The RGB images were converted to YCbCr, HSI and LAB format and each of the color components were used. A 7 $\times$ 7 window is considered around each pixel and average color components from the window are provided to the AdaBoost framework for classification. The time required for detection is approximately 0.5 seconds for each image and methods fails in different illumination conditions. In [7] detection , instance segmentation and tracking of grape clusters is performed by using CNNs. Particularly they use Mask R-CNN and YOLO v2 and YOLO v3 for performing the object detection. Yield estimation is performed in [8] by extracting features from LAB color space and then performing feature selection using Mutual Information metric. They detect the grape clusters by performing K-means clustering on the selected features.

III. YOLO

Unlike the two stage methods, YOLO (You Look Only Once) [4] is a single stage unified object detection method which works in real time. A single network is used to predict the probabilities of objects and for determining the bounding boxes (localization) of the detected objects. The prediction of the bounding boxes and class probabilities is modeled as a

regression problem. The first version of YOLO i.e. YOLO v1 [4] was proposed in 2015 and over the past few years several improvements have been made in YOLO and the latest version being YOLO v5 [9]. In the coming sections general framework of YOLO would be described along with the notable changes made in the newer versions.

A. Approach

A single convolutional neural network is used in YOLO to simultaneously predict the bounding boxes and the class probabilities. The methodology used in YOLO is as follows. Input images to YOLO are strictly square and of fixed dimensions. The image is divided into blocks with a $S \times S$ grid as shown in the Figure 1.

Fig. 1. Image with the grid (left) and Image with bounding boxes (right)

Each grid cell is supposed to detect the object, if the object center is present inside it. Each grid cell predicts confidence scores for B bounding boxes in YOLO. The confidence scores determine how accurately YOLO is able to predict that a object of a certain class is present in the box. The confidence score is calculated as $P(Class_i) * IOU$ where $P(Class_i)$ is the probability of $Class_i$ object being in the bounding box and IOU is the ratio of intersection to the union of the area between the predicted bounding box and the actual bounding box. Further, $P(Class_i) = P(Class_i|Object) * P(Object)$ where $P(Object)$ determines the chance of a object being in the box. If the object is not present in the box, confidence score is expected to be zero.

The four parameters to be predicted for the bounding box are its center (x, y) and its width (w) and height (h). The center of the bounding box (x, y) is predicted relative to the top left corner of the the grid cell and the width (w) and height (h) are relative to the image dimensions. If there are C classes of objects, the output tensor predicted by YOLO is $S \times S \times (B * 5 + C)$.

The loss function to be minimised so as to learn the parameters of the deep convolutional neural network of YOLO is shown in Equation 1. In the loss function $\mathbb{1}_i^{obj}$ denotes whether the cell i contains an object or not and $\mathbb{1}_{ij}^{obj}$ denotes the j^{th} bounding box of the i^{th} cell in the grid. So the loss function takes into account the corresponding terms only if the object is present in a grid cell ($\mathbb{1}_i^{obj}$) and the bounding box ($\mathbb{1}_{ij}^{obj}$) is responsible for the prediction. For the centers of the bounding boxes (x, y) mean square error is computed but for width and height $(w,\ h)$ mean square error is computed between the square roots of the width and height. Square root has been used to give equal weightage to the errors from both large and small bounding boxes. In the loss function two parameters λ_{coord} and λ_{noobj} have been introduced. The parameter λ_{coord} ensure higher weightage is given to the localization error than the confidence error and λ_{noobj} ensures that loss due to confidence errors is lesser when no object is present. The values of λ_{coord} and λ_{noobj} have been set to 5 and 0.5 respectively.

$$
\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
\tag{1}
$$

In YOLO v2 the fully connected layers used to predict the four parameters of the bounding box are removed and instead anchor boxes are used. The anchor boxes act as a prior for predicting the bounding boxes. The anchor boxes used as priors are computed by applying K-means clustering on the bounding boxes from the training data. The distance metric used is 1 - IOU(bounding box,centroid). Since predicting offsets from the anchor boxes takes longer time to get sensible offsets, instead in YOLO v2 the coordinates are predicted relative to the grid cell. Hence the ground truth is in the range 0 to 1 and logistic activation is used to get the output in this range. The five output values predicted for each grid are centers (t_x, t_y), width (t_w) and height (t_h). The following equation describing the relationship of these output values with bounding box coordinates is

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h}
\end{aligned}
\tag{2}
$$

where (c_x, c_y) are the coordinates of the grid cell relative to the top left corner of the image, (p_w, p_h) are the width and height of the bounding box prior and σ refers to the sigmoid function. By introducing anchor boxes YOLO v2 is able to improve the recall of the bounding boxes.

In YOLO v3 [10] mostly architectural changes are made and they are described in the next section. In YOLO v4 [11] in addition to some architectural changes over YOLO v3, some new concepts to improve the object detection have been introduced. The two major concepts introduced are *Bag of freebies* and *Bag of specials*.

Bag of freebies refers to improving object detectors by improving training strategies to get better accuracy while keeping the inference time as low as possible. Most methods in Bag of freebies perform different types of data augmentation so as to reduce overfitting. CutMix [12] is a image augmentation method used in YOLO v4. The authors of YOLO v4 also propose a new image augmentation method called as Mosaic augmentation. Figure 2 shows the details of the augmentation techniques.

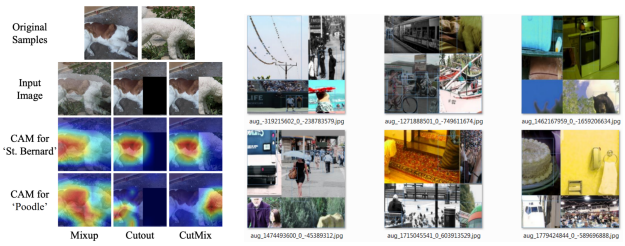

Fig. 2. CutMix augmentation (left) and Mosaic augmentation (right)

The authors also introduce the concept of Self Adversarial Training (SAT) in YOLO v4, which operates in two stages. In the first stage, the input image is transformed using the back-propagated loss signal rather than updating the weights. In second stage the transformed image has to be classified by the model by learning the weights. They also perform class label smoothing where the ground truth probability of a class is set to 0.9 instead of 1.0 so that the model does not overfit. YOLO v4 also uses a new loss function called as Complete-IOU Loss [13]. The Complete-IOU loss takes into account overlap area (IOU), normalized central point distance and aspect ratio of the bounding boxes. Equation 3 describes the Complete-IOU loss function

$$\mathcal{L}_{CIoU} = 1 - IoU + \frac{\rho^2(\boldsymbol{p}, \boldsymbol{p}^{gt})}{c^2} + \alpha V. \tag{3}$$

where V encodes the aspect ratio of the bounding boxes and is given by Equation 4, α is a parameter given by Equation 5, ρ^2 is the euclidean distance between the ground truth bounding box center $\mathbf{p}^{gt}$ and predicted bounding box center $\mathbf{p}$ and c^2 is the diagonal length of the smallest box enclosing both the boxes. Similarly the width w and height h are also defined.

$$V = \frac{4}{\pi^2}(\texttt{arctan}\frac{w^{gt}}{h^{gt}} - \texttt{arctan}\frac{w}{h})^2. \tag{4}$$

$$\alpha = \begin{cases} 0, & \text{if } IoU < 0.5, \\ \frac{V}{(1-IoU)+V}, & \text{if } IoU \geq 0.5. \end{cases} \tag{5}$$

Bag of specials refers to the post processing methods that increase the inference time slightly but improve the object detection significantly. For Bag of specials new blocks are introduced into the architecture which will be discussed in the next section.

B. Architecture

Figure 3 shows the architecture proposed in YOLO v1 [4]. The proposed architecture has 24 convolutional layers and 2 fully connected layers as shown in the figure. The final output predicted by the network has the dimension 7 × 7 × 30. The first 20 layers of the network are pretained using the ImageNet 1000-class dataset. For training the network and inference Darknet framework is used. Input image resolution of to the network is 448 × 448 and Rectified Linear unit activation function is used for all the layers except the final where linear activation is used.

In YOLO v2 [14] Batch Normalization [15] was introduced instead of Dropout [16] which has helped in better convergence and has also improved the Mean Average Precision (mAP). YOLO v2 uses Multi Scale training where the input image resolution of the network is randomly changed every 10 batches so the network is able to learn across different image dimensions. So as to improve the localization of the small objects, low level features are concatenated with high level features resulting in increase in depth of the final features.

In YOLO v3 [10] the existing YOLO v2 architecture is extended by adding convolutional layers and introducing skip connections like the ResNet. The network has 53 convolutional layers consisting of 3 × 3 and 1 × 1 kernels with skip connections and the authors name it as DarkNet-53. Figure 4 shows the details of the architecture.

In YOLO v4 [11] the architecture proposed is divided into three major components namely Backbone, Neck and Head. The backbone of YOLO v4 is CSPDarkNet-53 which is based on CSPNet [17]. CSPDarkNet-53 is pretrained on ImagNet data and its main purpose is optimize the flow of the gradients through the network while keeping the computation cost low. The neck of YOLO v4 consists of Spatial Pyramid pooling (SPP) [18] and Path aggregation network (PAN) [19]. The Spatial Pyramid pooling (SPP) helps in increasing the receptive field while also keeping the network speed high. The Path aggregation network helps in boosting the flow of information in the network by aggregating the features from different levels of the backbone. For the head YOLO v3 architecture is employed in YOLO v4. YOLO v4 also make use of Mish activation [20] which gives better results as compared to ReLu but is computationally expensive as compared to it. Spatial Attention Module (SAM) [21] makes use of attention to find the most important features extracted by the convolutional layers and remove the ones that are

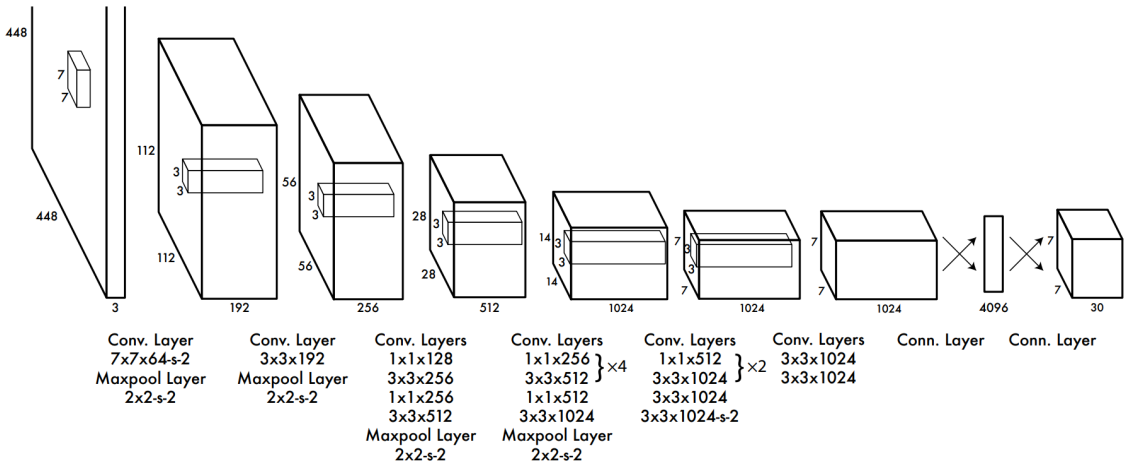

Fig. 3. YOLO v1 architecture

not necessary. DropBlock [22] regularization which is the equivalent of Dropout for fully connected layers is used as a regularizer in YOLO v4. In Dropblock regularization features in a block of feature map are dropped which works better in case of convolutional neural networks. YOLO v4 also makes use of genetic algorithms for the selection of the best set of hyper-parameters for the model.

YOLO v5 [9] is the latest version of the YOLO architecture for object detection. The architecture is very similar to YOLO v4 except for a few changes like Leaky ReLU and sigmoid activation is used instead of Mish activation. For the class probability and object score binary cross entropy loss function is used instead of Mean Square Error used in the previous versions. The authors propose four different versions with different number of parameters in YOLO v5.

C. Advantges of YOLO

As compared to other object detection approaches, YOLO has relatively simpler pipeline which results in extremely fast detection of objects. The predictions performed by YOLO are by taking into account the entire image as opposed to sliding window approach in DPM [23] and region proposal method in Fast R-CNN [24] which do not incorporate the global context of the image. Since Fast R-CNN cannot capture the global context, the background patches are mistaken for objects. As compared to Fast R-CNN YOLO makes fewer than half background errors. YOLO has shown better generalization capability when tested on images of different types as compared to the training images. Hence it can be used for object detection in new domains.

D. Limitations of YOLO

Although YOLO can quickly detect the objects, it struggles in localization of the objects specially in case of small objects. This is due to the fact that entire image is divided into equal sized grid cells and if there is a large variation in the size

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fig. 4. YOLO v3 architecture (DarkNet - 53)

of the objects in the image, YOLO struggles to localize the objects.

IV. Faster R-CNN

Faster R-CNN [25] is the third model in family of R-CNN [26] for the object detection task. Mainly it consists four building blocks. The first part would be Region Proposal Network(RPN) which generates the bounding boxes from the images. In the next part features would be extracted from these bounding boxes. These features would further used for the two class classification and regression. The classification part useful for detecting the object and regression layer predict the location in the bounding box.

The major pitfall of R-CNN & its successor Fast R-CNN [24] would be Region Proposal algorithm which was the Selective-Search [27]. It takes significant time for single image object detection as its runs on CPU. The Faster R-CNN has solved this issue by replacing Selective-Search algorithm by Convolution Neural Network(CNN). The major advantage of CNN is that it can run on GPU which reduces training and inference(testing) time by higher margin.

A. Architecture

Figure 5 shows the Faster R-CNN model. On the upper side of figure we can see the Region Propsoal Network(RPN) which consist pre-train VGG-16 [28] network which takes image as an input. Another important aspect of pre-trained model is that it also share its weight in detection part of model which further increases the efficiency. Further Faster R-CNN introduces "anchor" based object detection which are bounding boxes with fixed size and aspect ratio. Now at each pixel in output feature map this anchors checks for the presence of object or not. Specifically if k anchors and (H x W) feature map chosen than (H x W x 2*k) features generated for the classification and (H x W x 4*k) features generated for the bounding box regression. The loss function for RPN network show in equation(6).

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{6}$$

Here 'i' represents index of the anchor in the mini-batch. The classification loss is log loss between probability of anchor having object(p_i) and ground truth(p_i^*). Now if anchor actually contains the object then only regression loss activated. The t_i term represents the prediction by the regression layer. It further consist four variables $[t_x, t_y, t_w, t_h]$. This terms are calculated as following.

$$t_x^* = \frac{x^* - x_a}{w_a}, t_y^* = \frac{y^* - y_a}{h_a}, t_w^* = \log(\frac{w^*}{w_a}), t_h^* = \log(\frac{h^*}{h_a}), \tag{7}$$

Where x,y,w,h represents the cluster center and its corresponding width and height.

The detector part of Faster R-CNN is same as of Fast R-CNN. The input image to the detector pass through the backbone CNN network. It used as shared parameter with RPN network. Then ROI pooling layer used to extract the features from backbone CNN based on bounding box proposal from RPN. Further on this features specific window based max pooling applied to refine them. This features then applied to classification layer consist of C+1 (Number of classes + Background) for the classification of object proposed by RPN. Here also features passed through regression which further increases the accuracy of predicted bounding box parameters. Here total number of regressor would be 4*C that means each class have its own specific regressor.

B. Advantages of Faster R-CNN

Faster R-CNN gives response quickly as compare to R-CNN & Fast R-CNN because it uses the CNN based RPN. Additionally use of shared parameter between RPN and detector model further reduces the inference time. Faster R-CNN uses single network for regression and classification makes its training process very convenient.

C. Limitation of Faster R-CNN

While Faster R-CNN give better accuracy but it lags in terms of inference time compared to state of the art model. So it is not possible to use it in real life application. Further it comprises two neural network so it still computationally expensive.

V. SSD

R-CNN [26] and its subsequent improvements having two parts for object detection & recognition. On the contrary Single Shot Detection(SSD) [29] consist only one part so its very likely that SSD would be more faster. YOLO [4] is also single shot detection model but numbers of region proposed by SSD would be quite large.

A. Architecture

The initial layers of SSD consist pre trained VGG-16 [28] model which further extrapolated to get more deeper features as shown in figure(6). When SSD model runs on forward propagation it extracts the multiple bounding boxes. This is possible from feature generates through multiple convolution layers rather than only last layer. So if feature map has the size of pxq then from the each location k bounding boxes being generated. So each layer generates pxqxk numbers of bounding boxes. Further on each bounding box 'c' class score and 4 relatively offsets value for bounding box coordinate prediction being generated. The last two layer of the model uses the Atrous(Dilated) convolution which is useful for covering larger receptive field.

SSD at output generates total 8732 bounding boxes which is quite large compare to YOLO with 98 bounding boxes. The scales and aspect ratio calculated at each layer using equation (8) & (9) respectively.

$$\begin{aligned} s_i &= s_{min} + \frac{s_{max} - s_{min}}{n-1}(i-1) \\ &where, i \in [1, n] \end{aligned} \tag{8}$$

In equation(8) 'n' represents number of feature map with scale value of s_i. Now for each scale, s_i, there would be 5 non-square aspect ratios as shown in equation 9.

$$\begin{aligned} w_i^a &= s_i \sqrt{a_r} \\ h_i^a &= s_i / \sqrt{a_r} \end{aligned} \tag{9}$$

$$where, a_r \in 1, 2, 3, 1/2, 1/3$$

The loss function(10) is divided in two terms. The first term represents confidence of the classified object. Second term is

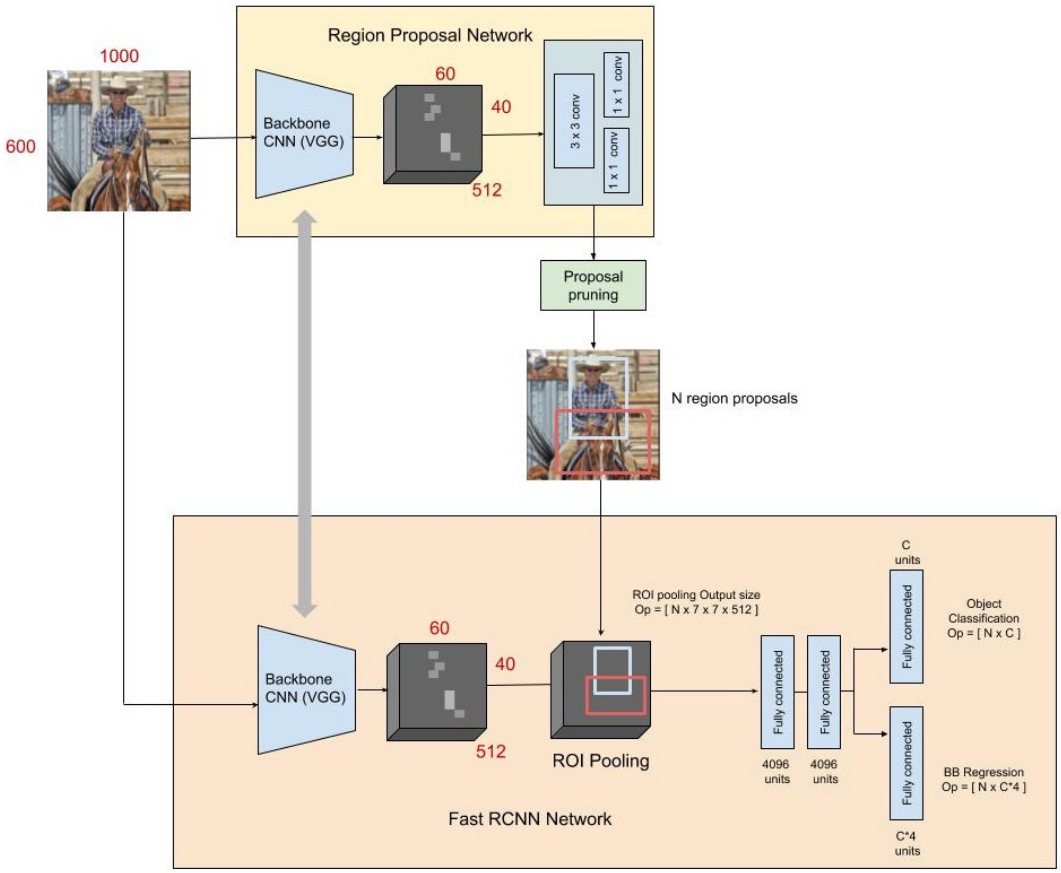

Fig. 5. Faster-RCNN Network, adopted from Medium

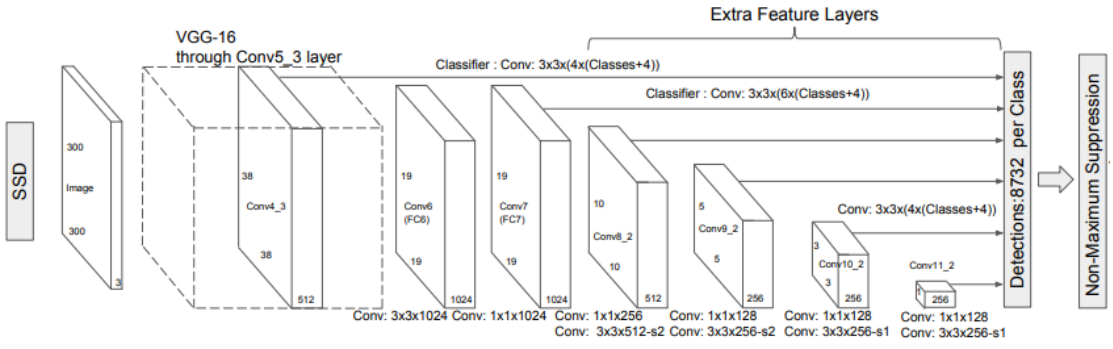

Fig. 6. Single Shot Detection

Variety	Images	Bpxed Clusters	Masked-Clusters
Train\Validation			
CDY	50	660	227(18)
CFR	55	910	418(27)
CSY	48	532	241(23)
SVB	38	446	224(18)
CDY	50	660	227(18)
Total	242	3582	1612 (110)
Test			
CDY	15	180	81(6)
CFR	10	159	95(6)
CSY	9	111	65(5)
SVB	14	283	106(5)
CDY	10	117	61(5)
Total	58	850	408 (27)

TABLE I
DATASET DETAILS

localisation loss(L_{loc}) which is the smooth L1 loss. It indicates the how far the predicted bounding box parameter shifted from the original one. This loss is quite similar with Faster R-CNN loss.

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \qquad (10)$$

B. Advantages of SSD

SSD uses different aspect ratio from different feature maps as compare to YOLO v1 which uses grid of fixed size. This output from multiple stage feature maps helps in getting better accuracy. SSD can detect the object through different scale as features from multiple feature map being used. The use of Atrous convolution in deeper layers gives faster performance as compared to traditional convolution.

C. Limitations of SSD

The authors of SSD suggest the need for complex data augmentation for better results. Additionally SSD doesn't perform well on smaller objects as shallow layers in model may not produce high level features from the image.

VI. EXPERIMENTS

A. Dataset

We will use the Embrapa Wine Grape Instance Segmentation Dataset [7] in this work. All the images were collected from a single vineyard located in Brazil and were resized to 2048 × 2048 resolution. The bounding boxes for the grape clusters were annotated using `labelImg` [1] tool. Dataset includes total of 300 images having 4432 grape clusters from 5 different categories. The camera captures the frontal pose of the grape clusters, with camera axis being perpendicular to the plant rows. Further train/test data split is shown in Table-I.

[1]https://github.com/tzutalin/labelImg

B. Experiment Settings

The training and testing of the object detection methods was performed using on Tesla T4 GPU with 16 GB of RAM. Since all the models works on square shaped images, training and test images have been resized to 416 × 416. All the YOLO models were trained for 100 iterations with SGD momentum optimizer with the learning rate equal to 0.001 and batch size of 16. For YOLO v5, two different models YOLO v5s (smaller) and YOLO v5x (extra large) were trained and tested. While YOLO v5s has 232 layers, YOLO v5x has 484 layers. YOLO v5s is optimized for faster object detection tasks while YOLO v5x optimized for higher accuracy. The Faster R-CNN and SSD both have trained for 200 iterations while optimizer and learning rate kept same as in case of YOLO.

C. Evaluation

We evaluate the results obtained from YOLO, Faster R-CNN and SSD using Mean Average Precision (mAP) and FPS (frames per second). Mean Average Precision helps in understanding how good the method is in terms of detection and localization of objects. Frames per second is calculated using the time required to detect the objects in the image and it evaluates the speed of detection of the method.

VII. RESULTS AND ANALYSIS

Summary of results in shown in Table II. The table describes the Mean Average Precision (mAP@0.5), Number of parameters in each of the model and Frames per second (FPS) calculated from the average inference time for a image.

A. YOLO

We compare the results of YOLO v5 and YOLO v3 which have been proposed recently with the other methods. Since YOLO v5 has minor changes over YOLO v4, we do not perform experiments with YOLO v4. From the table we can observe that YOLO v5x has the best mAP and YOLOv5s has the best FPS. The difference between the mAP of YOLO v5s and YOLO v5x is very small but there huge increase in FPS in case of YOLO v5s. YOLOv5s also has 10 times lesser parameters than YOLO v5x. This suggests that YOLO v5s which is a light weight model with fewer number of parameters could be the right candidate for deploying in real world applications. We also reduce the parameters of YOLO v5s and observe that the decrease in mAP@0.5 is marginal.

Figure 7 shows one of the failure cases of YOLO architecture for both YOLO v5x and YOLO v5s model. In case of YOLO v5x model, one of the grape clusters in the left center of the image is completely missed and both the grape clusters are grouped together as one grape cluster. This could be due to the fact that the two clusters are very close to each other. In case of YOLO v5s model the smaller cluster is detected but the bounding box overlaps the neighboring cluster. The fact that YOLO v5s is able to detect the smaller cluster and YOLO v5x is not able to could also mean that due to large number of parameters and fewer training images

Method	Number of parameters	mAP@0.5	FPS
YOLO v5x	88.43M	**0.809**	27.02
YOLO v5s	7.246M	0.805	90.90
YOLO v5s (reduced layers)	3.171M	0.796	**111.11**
YOLO v3	62.99M	0.657	29.41
Faster R-CNN	12.84M	0.44	0.9
SSD	23.74M	0.39	80

TABLE II
SUMMARY OF RESULTS

YOLO v5x model was overfit.

Figure 8 shows the results of YOLO v5x detection on random images of Grape clusters collected from the internet. It can be seen that YOLO v5x is performing reasonably good although some grape clusters are totally missed. This shows the generalization capability of YOLO architecture.

Fig. 7. YOLO v5x (Left) and YOLO v5s (Right) Failure Case

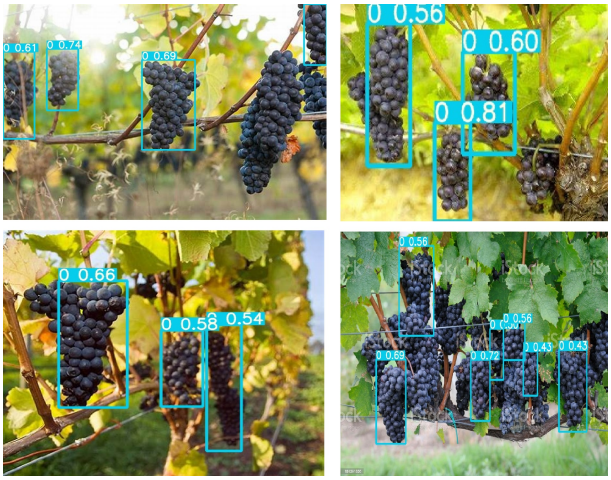

Fig. 8. YOLO v5x detection on random images from internet

B. Faster R-CNN

Figure 9 shows the result of Faster R-CNN model on some random internet images. It is observed that Faster R-CNN fails for the cases when grapes clusters are close to each other. In particular for green colored grapes in bottom left image we can observed that bounding box is barely able to fit any green grape. As the number of samples in training may not be

Fig. 9. Faster R-CNN detection on random images from internet

Fig. 10. SSD detection on random images from internet

sufficient so model may not able to capture actual bounding box correctly.

As Faster R-CNN consists two neural networks for the object recognition so it remains comparatively slower in response as we can see from Table-(II). It is also found that performance of Faster R-CNN is about to half as compared to YOLO v5x while slightly better compared to SSD in terms of mAP@0.5.

C. SSD

From both figures (10) & (11) we can observe that SSD performs very poorly in cases when either object is too large or too small. Specifically in the bottom section of figure (11) one can observe that none of the larger green grapes are detected by SSD. Another observation is that SSD combines

Fig. 11. SSD detection on Test image(Top) & random image from internet(bottom)

multiple objects in single bounding box which could increase false negative cases. These observation also proven by the performance observed from the SSD model which is 0.39 mAP@0.5. Even though model generates responses at faster rate but given the poor accuracy, the SSD model is not good enough as performance wise. In the implementation we didn't incorporate any image augmentation but it might overcome the difficulties faced by SSD.

VIII. Conclusion and Future Work

Grape cluster detection combined with other techniques can be used for yield estimation, harvesting, and monitoring of the grape vineyards. In this work we perform grape cluster detection using both one stage and two stage methods. Specifically, we use YOLO, SSD and Faster R-CNN and compare the results obtained from them. From our experiments we conclude that grape cluster detection can be achieved with very good precision and quickly by using YOLO v5s architecture.

In future, for better evaluation of detection methods larger dataset with varied settings needs to be created. Further, tracking of the grape clusters could be performed in a video for counting grape clusters for yield estimation.

References

[1] G. Lowe, "Distinctive image features from scale invariant keypoints," 2004.

[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," 2005.

[3] P. Viola and M. Jones, "Robust real-time object detection," in *International Journal of Computer Vision*, 2001.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[5] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," 2020.

[6] L. Luo, Y. Tang, X. Zou, C. Wang, P. Zhang, and W. Feng, "Robust grape cluster detection in a vineyard by combining the adaboost framework and multiple color components," *MDPI Sensors*, 2016.

[7] T. T. Santos, L. L. de Souza, A. A. dos Santos, and S. Avila, "Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association," *Computers and Electronics in Agriculture*, vol. 170, p. 105247, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169919315765

[8] S. Liu, S. Cossell, J. Tang, G. Dunn, and M. Whitty, "A computer vision system for early stage grape yield estimation based on shoot detection," *Elsevier*, 2017.

[9] G. Jocher, A. Stoken, and J. Borovec, "Yolo v5," 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[10] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018.

[11] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020.

[12] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe1, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," 2019.

[13] Z. Zheng, P. Wang, D. Ren, W. Liu, R. Ye, Q. Hu, and W. Zuo, "Enhancing geometric factors in model learning and inference for object detection and instance segmentation," *arXiv:2005.03572*, 2020.

[14] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," 2016.

[15] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," 2005.

[17] C.-Y. Wang, H.-Y. M. Liao1, Y.-H. Wu1, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "Cspnet: A new backbone that can enhance learning capability of cnn," *arXiv:2005.03572*, 2020.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *arXiv:2005.03572*, 2020.

[19] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," *arXiv:2005.03572*, 2020.

[20] D. Misra, "Mish: A self regularized non monotonic neural activation function."

[21] S. Woo, J. Park, J.-Y. Lee, and I. Kweon, "Cbam: Convolutional block attention module." *ECCV*, pp. 3–19.

[22] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Dropblock: A regularization method for convolutional networks," *NIPS*, 2018.

[23] D. M. P. F. Felzenszwalb, R. B. Girshick and D. Ramanan, "Object detection with discriminatively trained part based models," p. 1627–1645, 2010.

[24] R. Girshick, "Fast r-cnn," 2015.

[25] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.

[26] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014.

[27] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013. [Online]. Available: https://ivi.fnwi.uva.nl/isis/publications/2013/UijlingsIJCV2013

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[29] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *Lecture Notes in Computer Science*, p. 21–37, 2016. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_2