Seeds Germination Detection Using Convolutional SSD-like Architectures

Juan Valencia

***Abstract*—In this article SSD-like models are implemented to identify seeds in germination using a methodology based on [7] work.**

I. Introduction

II. Materials and Methods

A. Dataset

The dataset used was taken from the image acquisition process made by a project with the same goal [7]. It consists of 3 folders which contain the annotation of three seeds species (ZeaMays, SecaleCereale and PennisetumGlaucum) during its germination, figure 1 shows one example of the image with its annotations. The images of the dataset are in jpeg format

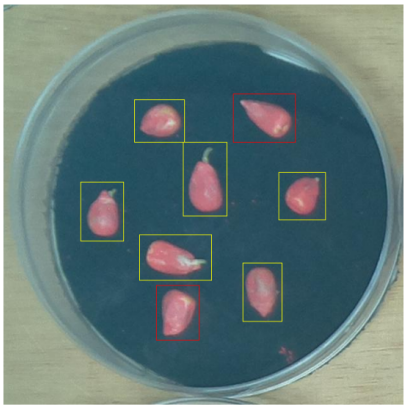

Fig. 1. ZeaMays Seeds with their respective ground truth bounding boxes

and its respective annotations are xml files generated by https://cvats.org

B. Multiple Objects Detection Problem

Let $\{I_n \in \mathbb{R}^{R\times C}, B_n \in \mathbb{R}^{M_n\times 4}, L_n \in \{l_1, l_2\}^{M_n}\}_n^N$ be an 1 input - 2 outputs set holding N labeled images, where I_n is the n-th image with R rows and C columns. B_n and L_n contains the bounding boxes and the classes of the M_n objects of interest from the image I_n respectively.

C. Model Architecture

There are two approaches to perform object detection over multiple elements. The first is based on two stage processing which is present in architectures like R-CNN [2] and Faster R-CNN [9] where is necessary to process the images twice to generate proposal regions an then classify them, while in architectures like YOLOv3 [8] or SSD [6] the image is passed once through the network. Models like R-CNN and Faster R-CNN tend to be more accurate than YOLO or SSD architectures but are slower [3]. Thus we decided to use SSD framework due to its improvements in speed/accuracy made in architectures like RetinaNet [4].

SSD architectures are made of three parts (see the Figure 2):

- Backbone
- Bottleneck
- Head

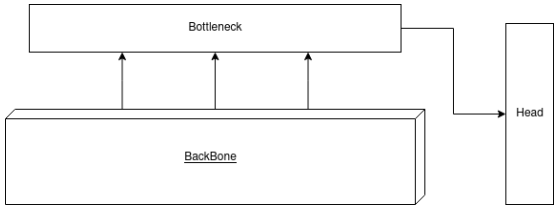

Fig. 2. SSD Architecture

The backbone is responsible of generate the feature maps which will be extracted to the bottleneck, it consist of a set of convolutional layers $\{W_l \in \mathbb{R}^{P_l\times P_l\times D_l}\}_{l=1}^L$ where P_l denote the l-th layer size and number of filters respectively. The backbone will generate three features maps $\hat{Z}_1 \in \mathbb{R}^{R_{L_1}\times C_{L_1}\times D_{L_1}}$, $\hat{Z}_2 \in \mathbb{R}^{R_{L_2}\times C_{L_2}\times D_{L_2}}$ and $\hat{Z}_3 \in \mathbb{R}^{R_{L_3}\times C_{L_3}\times D_{L_3}}$ as follows:

$$\begin{aligned}\hat{Z}_1 &= (\varphi_{L_1}\circ\cdots\circ\varphi_1)(I)\\ \hat{Z}_2 &= (\varphi_{L_2}\circ\cdots\circ\varphi_1)(I)\\ \hat{Z}_3 &= (\varphi_{L_3}\circ\cdots\circ\varphi_1)(I)\end{aligned} \tag{1}$$

where $L_3 > L_2 > L_1$ correspond to the layer position in the backbone, $F_l = \varphi_l(F_{l-1}) = \nu_l(W_l \otimes F_{l-1} + B_l) \in \mathbb{R}^{R_l\times C_l\times D_l}$ is a tensor holding D_l feature maps at the l-th layer, $\varphi_l : \mathbb{R}^{R_{l-1}\times C_{l-1}\times D_{l-1}} \mapsto \mathbb{R}^{R_l\times C_l\times D_l}$ is a function among the backbone layers, $B_l \in \mathbb{R}^{R_l\times C_l\times D_l}$ is a bias tensor and $\nu_l(\cdot)$ is a non-linear function like $ReLu(x) = \max(0, x)$.

The bottleneck mix the feature maps Z_1, Z_2, Z_3 from the backbone to share the features at different heights of the backbone module, it generates another three feature maps Z_1, Z_2, Z_3 with the same dimensions as $\hat{Z}_1, \hat{Z}_2, \hat{Z}_3$ respectively:

$$\begin{aligned}Z_1 &= (\varphi_{K_1}\circ\cdots\circ\varphi_1)(Z_1, Z_2, Z_3)\\ Z_2 &= (\varphi_{K_2}\circ\cdots\circ\varphi_1)(Z_1, Z_2, Z_3)\\ Z_3 &= (\varphi_{K_3}\circ\cdots\circ\varphi_1)(Z_1, Z_2, Z_3)\end{aligned} \tag{2}$$

As before the feature maps generated over the network is represented by $F_k = \varphi_k(F_{k-1}) = \nu_k(W_k \otimes F_{k-1} + B_k) \in \mathbb{R}^{R_k \times C_k \times D_k}$

The head of the network consists of a set of convolutional and reshape layers that will transform the output of the bottleneck in a matrix $\hat{A}$ of predictions which are encoded *anchor boxes*, for simplicity we will denote it as a simple function β where $\hat{A} = \beta(Z_1, Z_2, Z_3) \in \mathbb{R}^{\hat{M} \times (4+|\{l_1,l_2\}|}$ and $\hat{M} = 3\sum_{i=1}^{3} R_{L_i} C_{L_i}$.

D. Prediction decoding

To get the bounding boxes B_n, is necessary to transform the output $\hat{A}$ knowing that each row $\hat{a}$ is represented by:

$$\hat{a} = \left(\frac{x_b - x_a}{w_a}, \frac{y_b - y_a}{h_a}, \log \frac{w_b}{w_a}, \log \frac{h_b}{h_a}, p(l_1), p(l_2) \right)$$

where (x_a, y_a) and (w_a, h_a) are the positions and dimensions of one anchor box and x_b, y_b are the positions of the associated bounding box with dimensions w_b, h_b, thus the respective bounding box $\hat{b}$ associated with $\hat{a}$ is:

$$\hat{b} = \left(w_a \hat{a}_1 + x_a, h_a \hat{a}_2 + y_a, e^{\hat{a}_3} w_a, e^{\hat{a}_4} h_a, l_i \right)$$

where $i = \arg\max\left(p(l_1), p(l_2)\right)$. Therefore the matrix $\hat{A}$ can be decoded to get an associated bounding box matrix $\hat{B} \in \hat{M} \times 5$.

To get the final detection is necessary to use *Non Maximum Suppression* to filter overlapped detections.

E. Evaluation Metrics

To compare box detections against ground truth boxes is necessary to use the *Intersection over Union* IoU (3), a metric that measures the overlap between to boxes, for example if a box perfectly match with another it's IoU will be 1 on the other hand when there's not overlap the metric will output 0, the figure 3 illustrates it's operation.

$$IoU = \frac{b_1 \cap b_2}{b_1 \cup b_2} \tag{3}$$

With this in mind is possible to determine if a bounding box is a *True Positive (TP)*, *False Positive (FP)* or *False Negative* (FN) detection setting a threshold for the IoU.

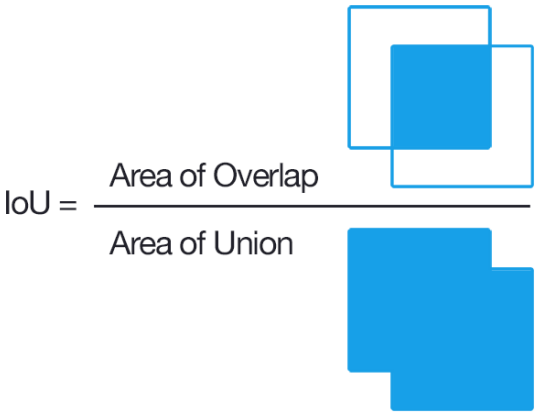

Fig. 3. Intersection Over Union

In object detection challenges [5, 1], measures like *PR-curve, Average Precision (AP)* and *mean Average Precision (AP)* are commonly used. *PR-curve* is obtained calculating the Precision (4) and Recall (5) using different IoU thresholds, the plotted values will show a monotonically decreasing function (Figure 4) which shows the precision recall trade-off.

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = \frac{TP}{TP + FP} \tag{5}$$

The AP is the Average of all calculated precisions for each class and finally mAP is the AP mean over the classes.

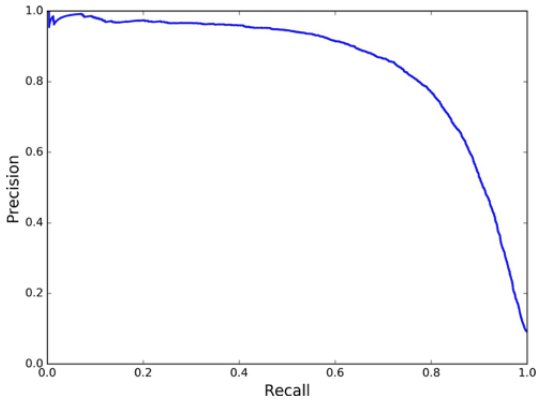

Fig. 4. PR-curve example for on class

III. Experimental set-up

References

[1] Mark Everingham et al. "The Pascal Visual Object Classes (VOC) Challenge". In: *International Journal of Computer Vision* 88 (2009), pp. 303–338.

[2] Ross Girshick et al. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587. DOI: 10.1109/CVPR.2014.81.

[3] Jonathan Huang et al. *Speed/accuracy trade-offs for modern convolutional object detectors*. 2017. arXiv: 1611.10012 [cs.CV].

[4] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. 2018. arXiv: 1708.02002 [cs.CV].

[5] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 740–755. ISBN: 978-3-319-10602-1.

[6] Wei Liu et al. "SSD: Single Shot MultiBox Detector". In: *Lecture Notes in Computer Science* (2016), pp. 21–37. ISSN: 1611-3349. DOI: 10.1007/978-3-319-46448-0_2. URL: http://dx.doi.org/10.1007/978-3-319-46448-0_2.

[7] Genze Nikita. “Accurate machine learning-based germination detection, prediction and quality assessment of three grain crops”. In: *Plant Methods* (). ISSN: 1746-4811. DOI: 10.1186/s13007-020-00699-x. URL: https://doi.org/10.1186/s13007-020-00699-x.

[8] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. 2018. arXiv: 1804.02767 [cs.CV].

[9] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV].