Data Science and Engineering

Object Detection and Weakly Supervised Semantic Segmentation for Grape Detection

Bachelor's Degree Final Thesis · 2021-06-18

Author
Pau Márquez Julbe

Directors
Philippe Salembier Clairon
Signal Theory and Communications Department
Josep Ramon Morros Rubió
Signal Theory and Communications Department

Escola Tècnica Superior d'Enginyeria de les Telecomunicacions

Facultat de Matemàtiques i Estadística

Facultat d'Informàtica de Barcelona

Abstract

Artificial Intelligence research has been exponentially growing during the last decade. Due to this fast growth, many industries have not had time to adapt their traditional methodologies to the advantages that AI algorithms provide. Many sectors, such as construction and agriculture, are slowly moving towards a more technological approach of decision making. In order to help these industries, public projects like *AI4Agriculture* are arising. *AI4Agriculture* is an European project that provides AI solutions to the agriculture sector.

The project described in this document presents modern computer vision techniques to detect grape racemes from vineyard images. The results of the algorithms made in this project will be part of a pipeline, along with meteorologic and multispectral data, to quantify the quality and quantity of the yield. Furthermore, with the help of the algorithms developed in this work, we will provide insights towards the cleaning and annotation of a novel dataset built specifically for this project.

Keywords

Computer Vision, Object Detection, Semantic Segmentation, Weakly Supervised Learning, Agriculture

Contents

1. Introduction

This project is part of the European pilot *AI4Agriculture*, which is part of the *AI4Europe* project. The project is implementing Artificial Intelligence solutions to different contexts. In the case of *AI4Agriculture*, the objective is to implement AI algorithms on vineyards and the main goals are: counting the number of fruits (grapes), estimating the quality of the production and approximating the yield. This thesis is collaborating in both the goal of predicting the total production and counting the number of grapes of the total yield.

In order to achieve these goals, different kinds of data from Ribera del Duero (a *Denominación de Origen Protegida*) have been collected. One of the datasets built by the consortium is a collection of images of grape vines, which have been annotated with bounding boxes surrounding grape racemes. Furthermore, other kinds of information, such as meteorologic and multispectral data have also been collected. All these kinds of data will be used to build a model that combines all kinds of information.

At the same time, this project is the continuation of the final thesis of a previous bachelor's degree [1], which has been used as a starting point. This previous project approached the problem with an object detection algorithm (predicting bounding boxes). Furthermore, a Cascade R-CNN [2] was re-implemented from scratch in order to improve the best algorithm found, even though the implementation was not fully debugged. The reason for this new implementation was that the original implementation was using a library which was not easy to customize.

This project's main objectives are:

1. Cleaning, preprocessing and setting up the novel dataset so that machine learning models can learn from it.

2. Build an algorithm that estimates the total number of pixels covered by grapes in the images of the AI4Agriculture dataset.

To achieve the second goal, we have approached it in two different ways; from an object detection perspective (predicting bounding boxes, Section 2.2) and from a semantic segmentation perspective (classify each pixel, Section 2.3). Ideally, our predictions should be binary masks so that we can exactly count the number of pixels covered by grapes, which is UPC's responsability in the *AI4Agriculture* pilot. However, we don't have ground truth segmentation masks, so we can not train common supervised learning architectures such as Mask R-CNN [3] (for instance segmentation) or U-Net [4] (for semantic segmentation) in a standard way. That's why we have applied weakly supervised learning methods to build a semantic segmentation model from bounding box annotations. On the other hand, for the object detection task, we have used a *Faster R-CNN* [5] architecture.

1.1 Project Timeline

Figure 1: Gantt of this project.

The milestones of this project can be summarized chronologically as follows:

1. **Analyse and clean the dataset:** The dataset built by *AI4Agriculture* was built by and for the project and we were the first technicians to approach it. This means that we had to look closely to the details and make sure both the images and the annotations were right and set up to train a machine learning model.

2. **Debug Cascade R-CNN implementation:** The first approach to create an object detection model was to train the Cascade R-CNN's implementation from the previous thesis, as their results suggested it was the architecture with the highest performance.

3. **Compare baseline with previous project's results:** We decided that the effort to fully debug the implementation of the Cascade R-CNN was excessive so a Faster R-CNN from *torchvision* was trained on the same dataset to build a baseline. The resulting metrics did not match with the previous project reported metrics so we had to dig into the previous project code. We found an error in the previous project implementation so, at this point, we started to build the object detection model from *torchvision*'s implementation of the Faster R-CNN.

4. **Hyperparameter tuning the Faster R-CNN model:** This process included in depth research on the network's architecture and a deeper analysis of the dataset in order to set properly the hyperparameters of the model.

5. **Weakly Supervised Learning research:** Once the object detection model was built, the next goal was to create a semantic segmentation model in order to have a more accurate estimation. Recent research has shown great results on applying semantic segmentation models to weakly labeled images (images labeled with less accurate annotations than masks, such as bounding boxes, image level classes, etc.).

6. **Weakly Supervised Learning model implementation:** Once a suitable weakly supervised model was found, it was implemented* in *PyTorch* by inspiring the key parts on the paper's author public repository.

7. **Weakly Supervised Learning model training:** Once we had implemented the model, we had to set the hyperparameters according to our dataset statistics. After a small amount of iterations, we selected the best model found and we stepped into an analysis of the results.

8. **Results analysis:** Once we had selected the best models of both approaches, we deeply analysed their weaknesses and strengths so that we could decide how to combine both models to achieve the aforementioned goal.

Figure 2: Example of an image of the built dataset with its annotated bounding boxes.

*https://github.com/paumarquez/AI4Agriculture-grape-detection

2. State of the Art

In this section, the theory behind the algorithms used to build the object detection and semantic segmentation algorithms will be explained so that a better understanding of our methodology and decision making is achieved. Also, the metrics we are using to evaluate models and annotators methodology will be explained.

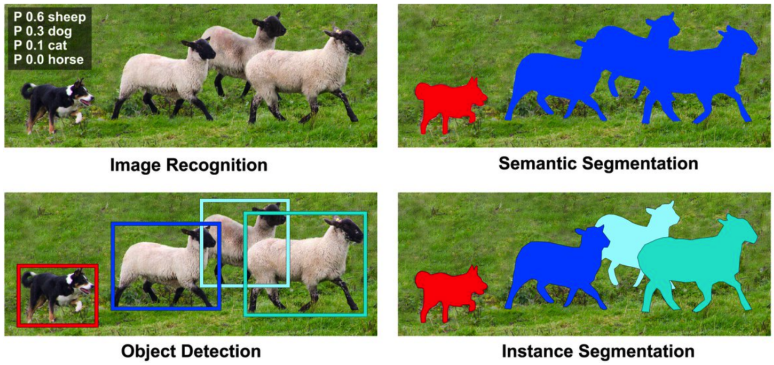


Figure 3: Representation of each task: Note how instances are differentiated in instance segmentation and object detection. However, in semantic segmentation we don't care about instances, just to which class corresponds each pixel. Note that instance segmentation is showed only for educational purposes, we are only working on object detection and semantic segmentation. Figure extracted from [6].

2.1 Metrics

To objectively measure the quality of object detection and semantic segmentation models (see Figure 3), there are many common metrics. Some of the most widely used in the literature are:

- **Recall:** Proportion of ground truth elements correctly predicted by the model.
- **Precision:** Proportion of predictions correctly predicted.
- **F1 Score:** Harmonic mean of the precision and the recall. This metric is used to balance between the recall and precision metrics, as there is a trade-off between them.
- **Average Precision:** Also called AP, a combination of the recall, the precision and the scores of each sample to belong to a class given by the model. AP manages to combine them by computing the precision-recall pairs for each possible score threshold. The area under the curve of this function is the average precision.

Both recall and precision metrics are based on hard labels (labels that belong either to one class or the other). However, AP considers soft labels (score based). Thus, if the model predicts scores for each class, there has to be a **score threshold** (sometimes hard to define) that binarizes them so that we can compute hard label metrics. On the other hand, AP can be computed without setting a score threshold, as it takes advantage of the prediction scores to compute the metric.

2.1.1 Object Detection

The particular case of object detection has the non-trivial problem of labeling each prediction (bounding box) as either true positive (correct) or false positive (incorrect). The standard way to evaluate the quality of a bounding box prediction is by using the **intersection over union** (IoU). This score gives the proportion of total area matched by the prediction, see figure 4. Usually a threshold is set a priori to binarize the score as true or false positive. This bounding box labels are then used to compute metrics, such as *precision*, *recall* and *AP*.

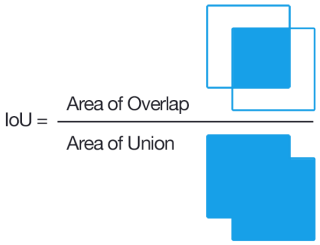


Figure 4: Intersection over Union scheme

Issues arise when object instances are subjective, meaning that there is a lack of definition on what an instance is. It causes that a given image could be labeled in many different correct ways. This fact could damage the learning process of an object detection algorithm, as the predictions can be 100% correct at the same time that the number of true positives are zero. At the same time, the evaluation of the model's performance through IoU metrics is not appropriate as it shows how close the predictions are to the ground truth labels, not how close they are to a right solution. See Figure 12 as a clear example of this issue.

2.1.2 Semantic Segmentation

When evaluating semantic segmentation metrics, recall and precision metrics can be evaluated at a pixel level, as the prediction of the network is the score of a given pixel to belong to a given class. This score is binarized with a threshold and directly compared with the ground truth labels. Average precision can also be used the same way in order to avoid the score threshold step.

These metrics do not take into account instances as object detection metrics do. This fact makes it a good choice in the case that we do not care about detecting different instances independently. In the particular case of this thesis, we are interested in studying the total area covered by grapes, so instance labeling is indifferent. As a consequence of that, we are using semantic segmentation metrics from the bounding box annotations by defining positive pixels as every pixel inside a bounding box. We will call these metrics **mask metrics**.

2.2 Faster R-CNN

Faster R-CNN is an object detection algorithm. Its goal is to detect **instances** of different semantic objects. The way this problem is approached is by surrounding each instance with a **bounding box**. It provides spatial information of each detected instance and it's easy to annotate by a human.

This area of research has been greatly explored during the past 5-10 years because of the boom in computer vision with deep learning methods. Convolutional layers and end-to-end deep neural networks have given excellent results as compared to more traditional methods [7].

Faster R-CNN [5] is the first end-to-end deep convolutional neural network for object detection, achieving

state of the art results at its creation. Figure 5 shows an scheme of the architecture, where each part can be described as follows:

1. **Backbone:** Image feature extractor composed of **convolutional layers**. These feature maps are high dimensional embeddings that contain semantic information about the image.
2. **Region Proposal Network (RPN):** Neural network trained to detect possible objects (proposals). It replaces selective search [8], which was the main algorithm used to detect what could be considered an object, as a function of the relation between familiar pixels. Such relation was quantified with properties like color similarity, texture, etc. RPN gives better and faster results by creating a fully differentiable network that can be optimized through gradient descent by using detector heads.
3. **Region Of Interest Pooling:** Given the proposals of the RPN, it classifies each proposal into a class and adjusts the bounding box limits with a regression method. Detector heads are used in a similar way as in the RPN.

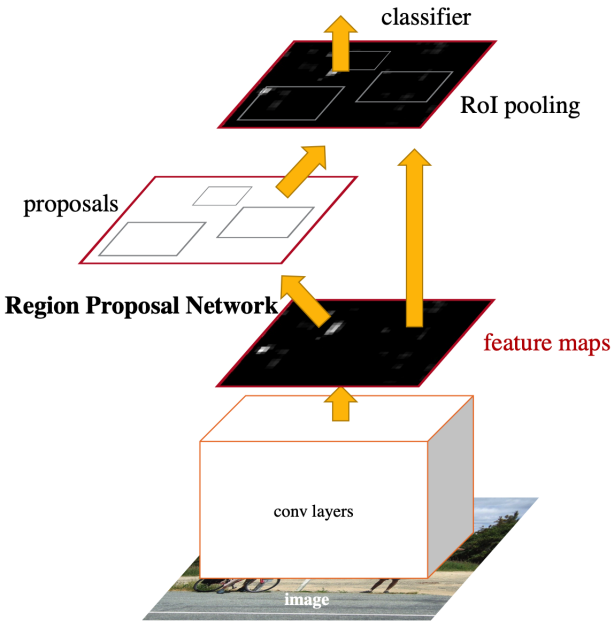


Figure 5: Faster R-CNN architecture scheme. It is remarkable that the same feature maps are fed into both RPN and RoIs. Image extracted from [5]

2.2.1 Backbone and Feature Pyramid Network

A backbone is a convolutional neural network trained on big datasets (usually ImageNet [9]), mostly on classification tasks. These networks are composed by a series of convolutional layers, which then flatten the 3D tensors into 1D tensors to be fed into fully connected layers. Those last layers return a classification score for each label.

Theoretically, the convolutional layers extract features from which a classification model composed of fully connected layers have the capability to classify into semantic objects. Thus, these features can be understood as high dimensional representations of the semantic information that the image contains. However, these

features are created specifically for the classification module. This means that the features built are the ones that maximize the classification capability of the classifier. The furthest convolutional layers (the closest to the classification module) return embeddings that contain information more specific to the concrete task the network is being trained on. On the other hand, the deepest convolutional layers return more general and higher scale information.

In order to combine higher and lower scale information, a **neck** is optionally applied between the backbone and the Region Proposal Network. The neck used in this project is the Feature Pyramid Network (FPN) [10].

The FPN combines the different layers by upsampling finer and spatially smaller feature maps to the shape of the following larger layer, then a 1×1 convolutional layer is applied to both bigger and upsampled layer to standarize the number of channels and they are summed up. This process is applied recursively for each upsampled and combined layer. The architecture scheme is showed in Figure 6.

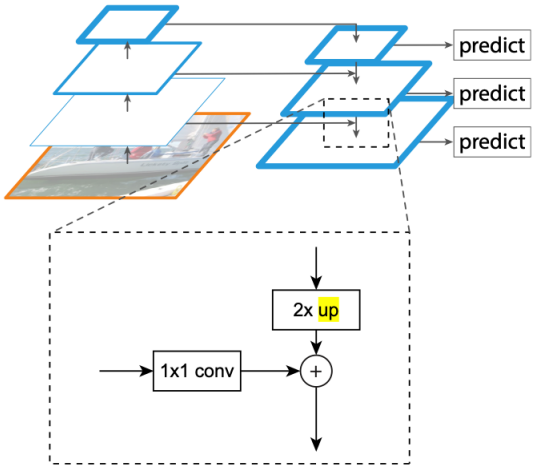


Figure 6: Scheme of the FPN network. Each white square represents a vectorial representation of the image. Note how the left (backbone) and more general features do not have finer information. However, the right hand features (FPN) contain more detailed information extracted by further layers. Image extracted from [10]

2.2.2 Detection Head

In order to understand RPN and RoIs, we need first to understand detection heads. A detection head is a module that receives an image embedding and a set of bounding boxes and returns a set of classification scores and a regression vector for each possible label. Such scores are interpreted as the probability of the related bounding box to belong to the associated label. The regression vectors are the adjustments to be applied on the bounding box coordinates so that it delimits the given object more accurately.

In order to optimize such module, the classification scores are trained with a standard Stochastic Gradient Descent (SGD) method by applying the cross-entropy loss for each class. Take into account that it is taken as a multi label task, which means that the sum of the scores over the different classes is not 1. The regression vector is trained the same way as a usual regression problem. Faster R-CNN concretely uses the smooth L1 loss[11].

2.2.3 Region Proposal Network

This neural network is used to generate rectangular slices (called proposals) of a given image with high probability of containing an object. This network generates such proposals by training a detection head over the image features and a fixed number of bounding boxes called **anchors**. The classification module classifies each anchor in two labels: potential object and background. Finally, **non maximum supression** is applied and the n bounding boxes with top scores are selected as proposals (applying regression inferences). The number of proposals returned is usually set to 2000 and 1000 for train and inference, respectively, even though it is an hyperparameter.

If multiple feature levels (layers) of the backbone/FPN are given to the RPN, this process is applied for each feature layer.

Non maximum supression is a method used to filter overlapped bounding boxes that refer to the same instance. To do so, the bounding boxes of the same class with an intersection over union greater than a given threshold (usually 0.5) are filtered, except for the bounding box with larger score.

2.2.3.1 Anchors

The input bounding boxes of the Region Proposal Network are generated by creating a mesh grid with different strides for each axis (x and y). For each grid intersection, a set of bounding boxes, called anchors, are generated.

These anchors should represent all shapes (size and aspect ratio) of the bounding boxes that the model should detect. This way, the RPN is able to generate proposals of any kind of shape.

The weights are shared for each anchor location of the mesh grid. Furthermore, heads share weight across the different scales of the backbone/FPN features so there must be the same number of anchors for each RPN feature level.

An important implementation issue that has to be taken into account when applying the detection heads to the Region Proposal Networks is that, in detection heads, the memory on inference time grows linearly with the number of bounding boxes to be predicted, since, even though the weights are shared among anchor locations, the number of anchors grows quadratically with the shape of the image, as the mesh grid generates locations of the order of $O(w \cdot h)$ and a classification score and a regression vector is computed for each anchor. It could be implemented so that anchors are partitioned and inferences are computed sequentially. However, the current implementations (such as torchvision) compute each anchor inference in parallel taking advantage of the GPU. This fact causes the GPU memory to increase greatly during that step.

2.2.4 Region of Interest Head

The Region of Interest Head applies a detection head to the proposals. The main difference with this head and the RPN head is the number of possible labels. This head detects the number of classes we want our model to detect, while the RPN detects just whether there is an object or not inside the bounding box.

Finally, non maximum supression is applied and, from the resulting bounding boxes, the ones with a classification score bigger than a score threshold are taken as positives bounding boxes after applying the regression vectors to the proposals bounding boxes.

2.3 Semantic Segmentation

Semantic segmentation is a computer vision task which goal is to label each pixel of an image into a semantic class (i.e. cats, dogs, tables, etc.) without taking into account to which instance each pixel belongs to. See Figure 3 as an example.

The main way this task is learned is by training neural networks in a fully supervised way. This means that matrices containing a label for each pixel (**masks**) must be introduced for the neural networks to learn to segment images.

2.3.1 U-Net

U-Net architectures are one of the most common architectures when it comes to semantic segmentation tasks. They receive an image with usually 3 channels (RGB) for color images (could be any number of channels) and return a tensor of the same width and height but as many channels as number of classes (including the background). The channels give a PDF on what class is assigned for each spatial location.

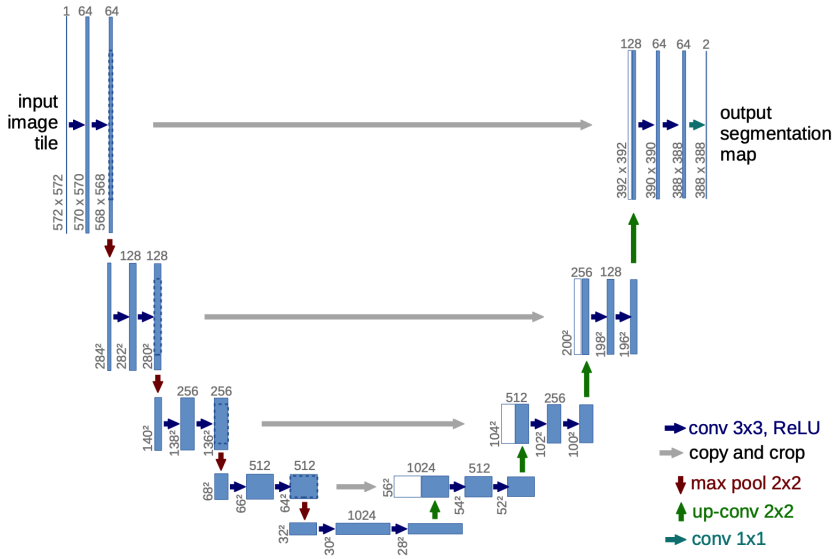


Figure 7: Architecture scheme of a U-Net. Note the concatenation of the downsampled to the upsampled embeddings for each different scale. Image extracted from [4]

They are based on two stages:

1. **Downsampling stage:** A series of convolutional blocks where each block is applied to a different spatial resolution scale. Each scale is downsampled with a max pool layer.

2. **Upsampling stage:** This stage has the same idea as the downscampling module but with up-convolutions instead of max pooling layers, where each scale has the same spatial shape as the downsampling stage. This way, the upsampling module transforms the downscaled image, which can be interpreted as a high dimensional embedding of the image containing semantic information, into a mask of the same spatial shape of the image for each possible label.

The key idea of the U-Net is the link between the downsampling and upsampling modules, since there is a concatenation of both embeddings for each scale of the upsampling stage. The downsampled embedding

gives spatial information in every scale level (the deeper the layer, the finer is the information provided) to the upsampling module, which has higher level embeddings containing semantic information. Furthermore, it helps to propagate gradients with respect to the loss. See Figure 7 for an scheme of the architecture.

The loss is a cross entropy function applied pixel-wise. It can be seen as a pixel-wise classification task that takes advantage of the whole image information to classify each pixel.

2.3.1.1 Residual U-Net

Residual U-Net [12] is an architecture that takes advantage of both the U-Net and the skip connections benefits. It improves the U-Net architecture by substituting the convolutional block in each step for a residual convolutional block. See Figure 8 for a more detailed scheme.

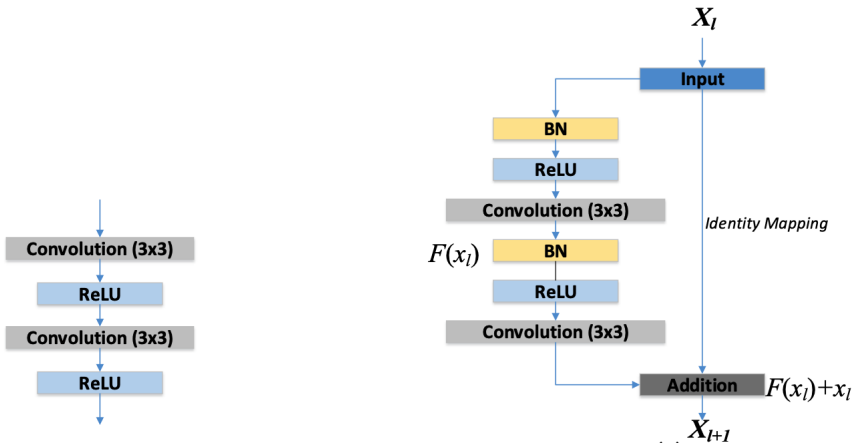


Figure 8: (a) U-Net Convolutional block. (b) Convolutional Block with the residual unit. It's also adding batch normalization before each convolutional layer. Image extracted from [12]

2.3.2 Weakly Supervised Learning

Weakly supervised learning is a branch of machine learning which uses noisy or poorly accurate labels to feed algorithms as ground truth. These techniques have a big impact when accurate labeling is costly or unavailable.

We have applied the approach of [13] to embed bounding boxes information to a semantic segmentation architecture as it showed very close results to supervised learning algorithms.

The way they embed bounding boxes information into the supervised learning U-Net [4] like architecture was by adding constraints to the optimization problem, which include prior information that we can infer from both the bounding box annotations and expertise on the subject of matter.

The explained U-Net like architectures are fully supervised. However, in weakly supervised learning we are trying to embed other kind of information to the network. In [13], bounding boxes were used for semantic segmentation tasks by adapting the U-Net like architecture's loss. In the following sections we are going to summarise their approach.

2.3.2.1 Preliminary Notations

We are going to use the same notation as in Section 3.1 from the original paper [13]. Let $\Omega \subset \mathbb{R}^{2,3}$ represent the spatial domain of an RGB image. We will denote Ω_O and Ω_I as the area outside and inside any bounding box, respectively. Let $s_\theta \in [0,1]^\Omega$ denote the probabilities predicted by the CNN, where 0 belongs to the negative class (background) and 1 belongs to the positive class (foreground). θ represents the weights of the neural network.

2.3.2.2 Negative Pixels

We have the certainty that pixels outside bounding boxes belong to the background class. This fact is trivial to implement with a classic cross-entropy, where only the negative part is taken into account:

$$\mathcal{L}_{MCE} := -\sum_{p \in \Omega_O} log(1 - s_\theta(p)) \tag{1}$$

In the original paper, they experimented with another kind of loss for the negative class, which they called *emptiness constraint* ($\mathcal{L}_{EMP}$). The loss is based on the knowledge that the sum of s_θ over the negative pixels (Ω_O) must be 0. Thus, implementing it as a constraint and knowing that $s_\theta \in [0,1]^\Omega$:

$$\sum_{p \in \Omega_O} s_\theta(p) \leq 0 \tag{2}$$

Experimenting with both loss functions, they showed that the emptiness constraint greatly surpassed the cross-entropy loss metrics. In Section 2.3.2.5 we will see how they implemented such constrained optimization problem

These equations give two different ways to make sure the network learns to detect the background. Now we are going to show how to detect positive pixels without actually knowing the exact pixels containing positive labels. We are going to do so by exploiting prior information extracted from bounding boxes and expertise knowledge on the subject matter.

2.3.2.3 Global Positive Size Prior

With expertise on the subject of matter and an analysis of the dataset, we can infer from bounding boxes annotations that, at least, a proportion $\epsilon \in [0,1]$ of the pixels inside the bounding boxes are positive. At the same time, we can set an upper bound[†] on the proportion of positive pixels $\delta \in [0,1]$. With both ϵ and δ, we can infer bounds for the total positive size contained in the image, resulting into the following inequality[‡]

$$\epsilon|\Omega_I| \leq \sum_{p \in \Omega} s_\theta(p) \leq \delta|\Omega_I| \tag{3}$$

[†] In the original paper it's set to 1.

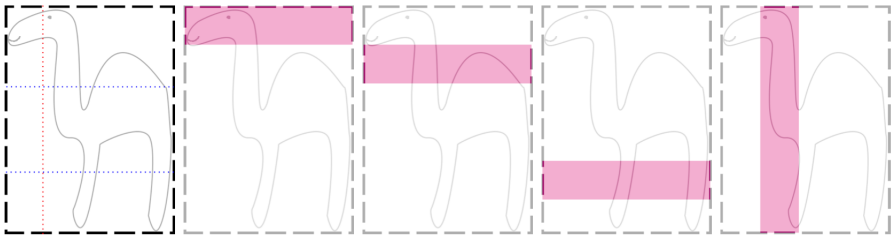
[‡] Our implementation sums over Ω_I, instead of Ω.

2.3.2.4 Tightness Prior

The tightness prior information, introduced in [14], assumes that bounding boxes are tight to the instance each one is surrounding. Thus, we know that any vertical and horizontal line will include, at least, a positive pixel. We can generalize this idea to the fact that any vertical and horizontal line of width ω will include, at least, ω pixels. Figure 9 illustrates the tightness prior assumption.

If we define S_L as the set of segments parallel to the bounding box bounds and $s_l \in S_L$:

$$\sum_{p \in s_l} s_\theta(p) \geq \omega \qquad \forall s_l \in S_L \tag{4}$$



(a) (b)

Figure 9: (a) Dashed horizontal and vertical lines show how every (one pixel of width) drown line should contain at least one pixel. (b) Generalization to any line of width ω, which should contain at least ω pixels. Image extracted from [13]

2.3.2.5 Log-barrier extensions

Optimization theory behind log-barrier extensions [15] is beyond the scope of this project. However, we can grasp the general idea without fully going into the details.

Log-barrier extensions show an efficient and successful way to optimize differentiable constrains, through gradient descent methods.

Given a constrained optimization problem as the following:

$$\begin{aligned} \min_{\theta} \quad & \mathcal{L}(\theta) \\ \text{subject to} \quad & f_i(\theta) \leq 0, i = 1, \dots, P \\ & f_i, \mathcal{L} \subset C^1 \end{aligned} \tag{5}$$

Log-barrier extensions accommodate restrictions into a classic non-restricted differentiable problem, which can be minimized with standard gradient descent methods. They achieve so by reformulating problem 5 into:

$$\min_{\theta} \mathcal{L}(\theta) + \sum_{i=1}^{P} \tilde{\psi}_t(f_i(\theta)) \qquad \text{where} \quad \tilde{\psi}_t(z) = \begin{cases} -\frac{1}{t} log(-z) & \text{if } z \leq -\frac{1}{t^2} \\ tz - \frac{1}{t} log(\frac{1}{t^2}) + \frac{1}{t} & otherwise \end{cases} \tag{6}$$

$\tilde{\psi}_t$ is what is called the log-barrier extension, which is a convex, continuous and twice-differentiable function. It includes the positive parameter t, which is set as a function of time during the minimization process. See Figure 10 for a graphical representation of the function, it shows how it penalizes big positive values of z, while negative values are not penalized. Furthermore, the penalization is stronger as the optimization process goes on.

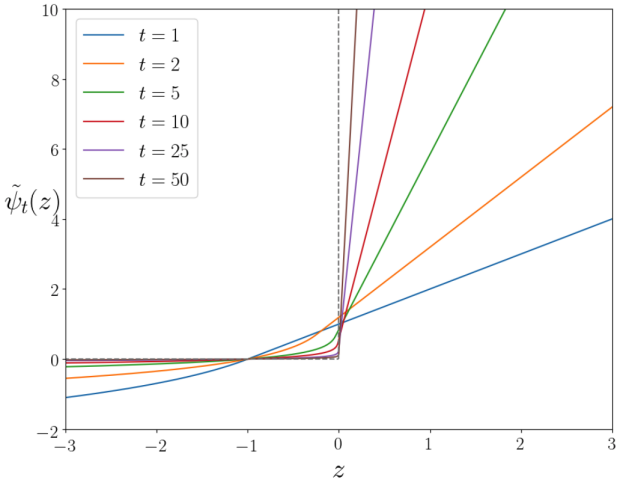


Figure 10: Log-barrier extension plot for different values of time. Note that as the time through the optimization process goes by, the barrier penalizes more abruptly the positive values. Image extracted from [16].

2.3.2.6 Weakly Supervised Learning Loss

If we combine the emptiness constraint, the tightness prior and the global positive size prior into a whole optimization problem:

$$\begin{aligned} \min_{\theta} \quad & \mathcal{L}_{EMP}(\theta) \\ \text{subject to} \quad & \epsilon|\Omega_I| \leq \sum_{p \in \Omega} s_\theta(p) \leq \delta|\Omega_I| \\ & \sum_{p \in s_l} s_\theta(p) \geq \omega \qquad \forall s_l \in S_L \end{aligned} \tag{7}$$

Now inequalities are standardized in the form of Eq. 6 and the log-barrier extensions are applied:

$$\begin{aligned} \min_{\theta} \tilde{\psi}_t \left(\sum_{p \in \Omega_O} s_\theta(p) \right) + \lambda \left[\sum_{s_l \in S_L} \tilde{\psi}_t \left(\omega - \sum_{p \in s_l} s_\theta(p) \right) \right] + \\ \tilde{\psi}_t \left(\epsilon|\Omega_I| - \sum_{p \in \Omega} s_\theta(p) \right) + \tilde{\psi}_t \left(\sum_{p \in \Omega} s_\theta(p) - \delta|\Omega| \right) \end{aligned} \tag{8}$$

Where λ is a real number, which balances the tightness prior loss with respect to other losses.

This minimization problem **can be solved through standard gradient descent methods**, since it's differentiable with respect to θ.

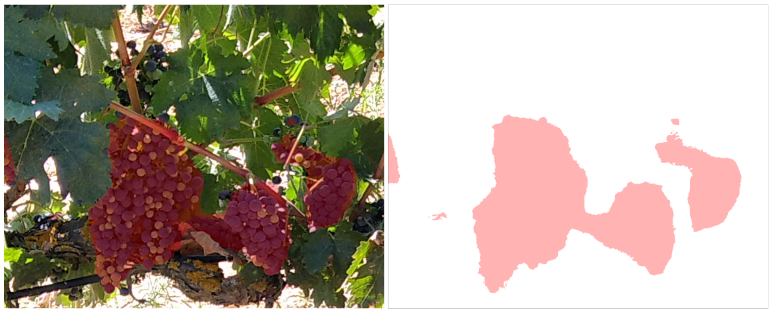
2.3.3 Dense Conditional Random Fields

Dense Conditional Random Field (CRF) [17] is an important algorithm in computer vision. Nonetheless, the mathematical background is behind the scope of the project. Thus, we will give an idea of what we used it for.

One of the Dense CRF applications is to enhance the quality of semantic segmentation outputs from a model that generates posterior probabilities. These probabilities are usually generated independently for each pixel. Thus, raw masks predicted by models, such as U-Net, can be noisy and have pixels with incoherent labeling with respect to familiar correctly labeled pixels.

Dense CRF receives the posterior probabilities of each pixel to belong to every possible class and returns a new probability distribution. This PDF takes into account other RGB pixels from the image from which it extracts and combines pairwise characteristics (color, spatial distance, texture, etc.). Pixel pairs with high similarity are more likely to be matched into the same class. Therefore, the new labels assigned to each pixel are more coherent with the familiar pixels with similar properties.

(a)



(b)

Figure 11: (a) Output mask of a semantic segmentation neural network, concretely a Residual U-Net. (b) Same output as (a) postprocessed by a Dense CRF algorithm. Note how the noise is being removed if similar pixels are detected in the neighborhood of a positive/negative pixel.

3. AI4Agriculture Dataset

One of the goals of the *AI4Agriculture* project is to build a dataset of grape vine images labeled with bounding boxes surrounding grape clusters.

The dataset is made up of 445 images of size 3000×4000 collected in Ribera del Duero during the summer of year 2020. The images have been annotated by 8 different annotators with some guidelines to take into account when deciding exactly what is a positive label; only grape clusters from the first grape tree line are considered, prioritize drawing bigger bounding boxes.

Each annotator has annotated about 96 images, from which 52 images are shared among some of the annotators. Those 52 images have been used to understand each annotator's methodology and find discrepancies between them, as they can gravely affect both the model's performance and the metrics used to measure the prediction capability of the model. Furthermore, it has also been used as a **benchmark** for the object detection approach, as we can evaluate the human's performance with the same metrics by crossing annotator's annotations with themselves. We'll talk about this in Section 4.2.1.

Unfortunately, annotator 3 has not annotated the intersecting sample. For this reason, we have excluded its annotations from the dataset, as we don't know how it behaves with respect to other annotators neither the quality of its annotations. Also, since we have a dataset which is big enough to successfully train our models, we have not delved too much into this annotator's data.

Figure 12: Slice of an image from the *AI4Agriculture* dataset. The annotations coming from individual annotators are shown with different colors. Discrepancies between them can be appreciated, as explained in Section 3.1.1.

3.1 Dataset Analysis

As we were the first researchers dealing with this data set, we had to make sure everything was coherent and well set up for models to be trained on it before stepping further into training them. Furthermore, in order to configure model's hyperparameters, as well as choosing the model architecture, an understanding of the data we are working with is mandatory.

The analysis will be applied to the intersection sample, where we can compare different annotators with the same dataset subset.

3.1.1 Annotators Discrepancies

One of the main issues of this dataset is that annotators had a different understanding of what an instance was, as different grape clusters are mixed in the image (see Figure 12). This issue has been greatly noticed when overlapping bounding boxes from different annotators. The fact that different annotators use different annotation methodologies might damage the learning process of object detection algorithms, as annotations contradict each other from the model's point of view. We are going to compare their methodologies with the aim of minimizing discrepancies and have a better understanding of the dataset.

In order to compare them, we have used both mask and bounding box metrics the same way as we would compare an algorithm's predictions with a given ground truth. Those metrics along with some simple statistics give a good estimation of how each annotator draws bounding boxes as compared to others.

On one hand, box metrics do not give information about the quality of the labels in terms of the total true positive area covered by boxes (our final goal), but they give a good estimation of the degree of similarity between annotators. On the other hand, mask metrics give a better idea of how every annotator misses or fills area covered by grapes using as a reference other annotators annotations.

First of all, we made the hypothesis that there were no discrepancies between annotators. To test the hypothesis, we built the **box recall table** (see Table 1). This table shows what percentage of bounding boxes of a given annotator (the so called ground truth) is matched by an other one (predictor) with an IoU threshold of 0.7. Thus, it's showing that less than 50% of the bounding boxes are matched between different annotators. The group of annotators $\{0, 4, 5, 6\}$ stands out as a group for having more matches between them than others do. On the other hand, annotators $\{1, 2\}$ have much worse metrics. This fact implies a discrepancy on the labels that could damage the learning process of an object detection model.

Ground Truth Annotator **Predictor Annotator**	**0**	**1**	**2**	**4**	**5**	**6**	**7**
0	1.00	0.02	0.28	0.44	0.42	0.47	0.37
1	0.06	1.00	0.10	0.06	0.06	0.05	0.06
2	0.30	0.04	1.00	0.21	0.23	0.18	0.15
4	0.54	0.02	0.25	1.00	0.38	0.46	0.46
5	0.48	0.01	0.25	0.35	1.00	0.40	0.34
6	0.49	0.00	0.18	0.40	0.38	1.00	0.48
7	0.40	0.01	0.17	0.40	0.33	0.48	1.00

Table 1: **Box recall** of each annotator against other annotators with an IoU threshold of 0.7. Note that the precision is the recall changing the ground truth and the predictor annotators, so this table transposed turns into a **box precision** table.

In order to further understand the annotators methodologies, the **mask metrics** from Table 2 give some valuable insights:

1. Annotator 1 draws small and tight bounding boxes, as the precision with other annotators (column 1) is almost 1.0. However the recall is low (between 0.3 and 0.4). This fact suggests that either the annotator is missing positive area, or the area inside the other annotators boxes covered by grapes is very low (less than 40%), which is unlikely. Table 3 also shows how the positive area per image of annotator 1 is 30% lower than annotators $\{0, 4, 5\}$. Finally, in Figure 12, it's shown this annotator's methodology and how it misses a big proportion of grape area.

2. When comparing annotators 6 and 7, metrics between them are very high, while they also have similar statistics with other annotators. Therefore, the behaviour of both annotators is expected to be very similar. They both have high mask recalls (row 6 and 7) and lower mask precisions (columns 6 and 7). The recall suggests that their bounding boxes include other annotators' bounding boxes. Low precision shows that bounding boxes are greater (not as tight) and/or the number of boxes is greater and those annotators detected more grapes than others did. Table 3 shows that the total positive area is much larger. This fact suggests that both annotators are either noticing more grapes than others do or drawing bounding boxes that surround others. As the precision (column) suggests that 30% of the drawn area is not intersected, it is more likely that they are finding more grapes than others do.

   Nonetheless, the discrepancies between these two annotators are found when comparing their box metrics with annotators 0 and 5. Annotator 6 has much better metrics, which means that, even though the area covered is very similar, among the boxes annotated by 0 and 5, the annotation methodology of 6 is much more similar to the annotators 0 and 5 than annotator 7's methodology.

3. Annotator 2 covers 90% of 1's area (see column 1 of Table 2), while other annotators have about 100%. This fact gives the idea that positive area is being missed. At the same time, the total area of annotator 2 that is overlapped with annotator 1 is about 10% greater than others (row 1). It means that the total area covered by this annotator is lower, as can be checked in Table 3. These facts suggest that this annotator misses positive area and, as a consequence, the mask metrics of this annotator with other annotators are much worse than the average.

Ground Truth Annotator **Predictor Annotator**	**0**	**1**	**2**	**4**	**5**	**6**	**7**
0	1.00	0.99	0.86	0.85	0.83	0.72	0.72
1	0.36	1.00	0.45	0.36	0.36	0.28	0.29
2	0.64	0.90	1.00	0.62	0.63	0.51	0.52
4	0.87	0.99	0.85	1.00	0.83	0.74	0.75
5	0.83	0.98	0.85	0.82	1.00	0.70	0.71
6	0.95	1.00	0.91	0.96	0.93	1.00	0.89
7	0.92	0.99	0.88	0.94	0.90	0.85	1.00

Table 2: **Mask recall** matrix. Note that the precision is the recall on the opposite direction, so transposing this table it turns into a **mask precision** table.

From the information extracted above, we concluded that annotators $\{1, 2, 3, 7\}$ had to be dropped from the dataset used to train the models. The quality of the annotations was prioritized against the dataset size, as we considered that the dataset was big enough. Table 5 shows the dimensions of the selected dataset. For a more in depth analysis the interested reader is suggested to take a look at the published python notebooks[§]

3.1.2 Grapes Colors Imbalance

There are two kind of grapes in this dataset: dark and light grapes. An important issue related to these two classes is caused by the big imbalance, as the number of dark grapes is much bigger than the number of light grapes. There were no labels referring to the color of each grape cluster so we don't have exact statistics of the imbalance between the two kind of grapes. However, it is easily recognized when visually analysing the images and the false negatives of the trained models.

Algorithms trained with this kind of imbalance are (usually) greatly impacted since, if they are labeled with the same class label, the algorithm has to learn the properties (such as color, texture, volume, shape...) of both classes with the same weights, instead of having exclusive sections of the algorithm (could be a neural network) dedicated to learn the concrete properties of each class. Sharing weights within two different classes causes the network to either prioritize in favor of the dominant class or, if possible, learn the properties shared (or not, but much harder) by both classes, which might turn into less prediction capability, depending on the homogeneity of properties among both classes.

A simple clustering and a classification model (with a manually labeled sample), both color space based (concretely in the *YCrCb* space, so that luminance could be removed), were attempted in order to discriminate between both classes. However, it was a hard problem, as the background color was too similar to both green (leaves) and dark (shadows) grapes. We considered it a hard problem after some attempts, so we skipped this step.

Annotator	Mean number of boxes per image	Mean area per box	Mean positive area per image (%)
0	18.1	17728	2.51
1	20.2	6023	0.95
2	18.3	13139	1.84
4	25.6	13353	2.54
5	19.9	16051	2.49
6	24.0	18093	3.22
7	24.1	17567	3.12

Table 3: Annotators statistics. The discrepancies between annotator's total positive area per image give insights about how each annotator has found more or less grapes in images.

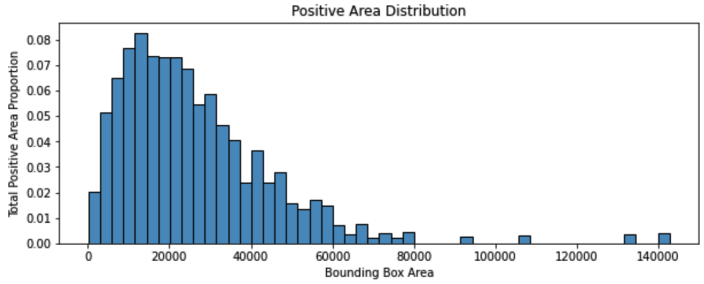
3.1.3 Bounding Box Analysis and Relation to the Covered Area

We noticed that bounding box annotations disagreed between different annotators, not only in the way bounding boxes were drawn, but in the fact that they were noticing different grape clusters, since some annotators missed small clusters. Furthermore, smaller bounding boxes are the ones with highest probability of being missed by annotators.

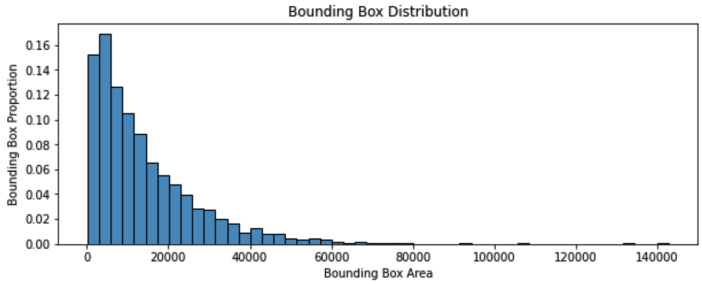
Figure 13 shows how smaller bounding boxes are the most frequent and the least important when it comes to the area covered. See Section 21 of the Appendix for examples of small bounding boxes.

[§]https://github.com/paumarquez/AI4Agriculture-grape-detection/blob/main/notebooks/annotators_exploration.ipynb

Table 4 shows a detailed analysis of the joint distribution focused on the smaller bounding boxes. 24% of bounding boxes cover 3.7% of the total area. This fact might be an issue if models trained with this dataset weight the loss function the same way for bounding boxes of different area.



(a)



(b)

Figure 13: (a) Output mask of a semantic segmentation neural network, concretely a Residual U-Net. (b) Same output as (a) postprocessed by a Dense CRF algorithm. Note how the noise is being removed if similar pixels are detected in the neighborhood of a positive/negative pixel.

Area	Area PDF	Area CDF	Bounding Box PDF	Bounding Box CDF
0	0.000196	0.000196	0.009088	0.009088
500	0.000993	0.001188	0.018452	0.027541
1000	0.002113	0.003301	0.023685	0.051226
1500	0.003870	0.007171	0.031396	0.082622
2000	0.005550	0.012720	0.034977	0.117598
2500	0.006893	0.019613	0.035803	0.153401
3000	0.007810	0.027423	0.034426	0.187827
3500	0.008833	0.036256	0.033600	0.221427
4000	0.008942	0.045198	0.030019	0.251446
4500	0.009910	0.055108	0.029744	0.281190
5000	0.009519	0.064627	0.025888	0.307078
5500	0.009208	0.073834	0.022859	0.329937

Table 4: Joint distribution of the number of bounding boxes along with the proportion of the total area covered by these bounding boxes. Both probability density functions (PDF) and cumulative density functions (CDF) are displayed. Statistics extracted from the training set for annotators 0, 4, 5, 6.

4. Experiments

In this section we are going to describe the experiments carried out through this project. As we have experimented with two different architectures, we are going to divide this section in two parts: Object detection and weakly supervised semantic segmentation. A deeper analysis of the best models found during the experimentation will be performed later in Section 5.

4.1 Dataset

The dataset used to train and evaluate both models is the dataset mentioned in section 3.1. It has been split in train, validation and test as it is described in table 5. The validation set has been used during the experiments to compare models performances and the test set has been used to evaluate the definitive model so that a real estimation of the expected errors in inference time can be given. Also, it was used to make sure that no validation overfitting was made.

	Train	**Validation**	**Test**	
0	66	13	15	94
4	34	7	8	49
5	37	7	8	52
6	35	7	9	51
Total	172	34	40	246

	Train	**Validation**	**Test**	**Total**
0	527	115	112	754
4	405	86	40	531
5	361	74	57	492
6	482	57	100	639
Total	1775	332	309	2416

Table 5: For each annotator and dataset split, the left table contains the number of images and the right table shows the number of bounding boxes. Note that annotator 0's annotations have been used in the intersection set.

4.2 Object Detection

The object detection algorithm we have worked on is the Faster R-CNN (see Section 2.2). It was the first architecture we built as a baseline and, after some configuration adjustments, we considered that it had a big enough prediction capability for us to stay with this architecture.

4.2.1 Evaluation

As explained in Section 2.1, the metrics we are interested in are the **mask metrics**, since we are willing to detect the amount of area in the images covered by grapes. However, the goal of object detection models is to maximise the box metrics. Both kind of metrics are related so, in general, an increase of box metrics will increase the mask metrics (it is not a real implication, see annotator 1 in Section 3.1.1). Concretely, we used **mask AP** and **mask F-score** to discriminate between different configurations. F-score has been used to compare models with the benchmark and the AP has been used to update the learning rate during training, since there is no need to set a score threshold and it gives a good indication of the prediction capability of the model. As F-score needs a score threshold to be set and our models F-scores vary only between 0.001 and 0.005 for different score thresholds, we decided to set it arbitrarily to 0.5 for all models, since we don't have a validation set sufficiently big to test whether the optimum score threshold generalizes well.

Ground Truth Annotator **Predictor Annotator**	**0**	**1**	**2**	**4**	**5**	**6**	**7**
0	1.00	0.52	0.73	**0.85**	**0.83**	**0.81**	0.80
1		1.00	0.59	0.52	0.52	0.43	0.44
2			1.00	0.71	0.72	0.64	0.64
4				1.00	**0.83**	**0.83**	0.84
5					1.00	**0.79**	0.79
6						1.00	0.86
7							1.00

Table 6: **Mask F-score** of annotators against each other.

The model was trained with different kind of labels (each annotator had its own labeling method), which also had an implicit subjectivity and a high probability of missing annotations. Thus, we didn't expect to get metrics close to 1.

In order to have a reference, a **benchmark** was built. We computed the F-score of each annotator against each other (see Table 6) the same way it was done to analyse the annotators discrepancies in Section 3.1.1. By computing the average F-score of the annotators selected to run the experiments (in bold), we can set an upper bound on the validation metrics and consider the resulting F-score a **human level F-Score**. Such average gives an F-score of **0.823**, so the models metrics should get as close as possible to this number, but knowing that we **can't surpass it**.

4.2.2 Implementation Details

We used the torchvision implementation of both the Faster R-CNN and the backbone. The backbone was the ResNet 50 [18], pretrained with the ImageNet dataset [9]. The optimizer settings were the same as the original Faster R-CNN paper [5], a Stochastic Gradient Descent (SGD) with a learning rate of 0.001 and a decrease factor of 0.1 updated as the AP metric stopped increasing. A weight decay of 0.0005 and a momentum of 0.9. Early stopping was applied on the best AP, as the model overfitted the training set. However, the models used to converge in the validation set after between 20 and 70 epochs. Non maximum supression was not applied, since we only took into account mask metrics. We trained the neural networks with an NVIDIA GTX 2080 Ti. The source code is available online[¶]

4.2.3 Hyperparameters

There were two hyperparameters that had a notable impact on the models metrics; the re-scaling of the images before being fed into the neural network and the anchors shapes. However, we experimented with other configurations that didn't succeed as much; in particular, we tried to remove small bounding boxes or to use the Faster R-CNN pretrained in the *COCO* [19] dataset as a start point to fine-tune it into our dataset.

[¶]https://github.com/paumarquez/AI4Agriculture-grape-detection

4.2.3.1 Downscale Factor

To start off, a **baseline** was built with the same hyperparameters as the model was trained in the *COCO* dataset [19] with the Torchvision implementation. This configuration had a high downsampling with respect to the shape of the images in our dataset and the size of the bounding boxes. Thus, the first thing we tested was to downscale the images by a smaller factor. The original downscaling factor of the images was 3.75 and it was manually set to 1.5. See Table 7 with the exact metrics.

Scale Factor	F-Score	AP
3.75	0.7290	0.6111
2.25	0.7584	0.6426
1.50	**0.7977**	**0.6972**

Table 7: Scale factor experiments: The lowest downscale gives the best results.

4.2.3.2 Pretrained on the COCO Dataset

We used the public checkpoint of the Faster R-CNN pretrained on the COCO dataset to fine-tune it in our data. Both the RPN and ROIs, as well as the last two layers of the backbone, were trained. However, even though it converged much faster than training the model from scratch, it did not get the same results as other configurations. The main reason being (we theorize) that the dataset is big enough for a model to be trained without requiring fine-tuning techniques. The F-Score was **0.8056** and the AP was **0.7081**.

4.2.3.3 Bounding Box Filter

As explained in Section 3.1.3, smaller bounding boxes were predominant. This fact could damage the Faster R-CNN model, as the area of each bounding box is not taken into account when weighting the loss. We theorized that removing small bounding boxes it would be easier for the model to learn to detect bigger boxes. Since our objective is to maximise the mask metrics, detecting a bigger proportion of big bounding boxes could increase the mask metrics even at the cost to miss some smaller boxes.

We tested our hypothesis by running multiple configurations, where each configuration filtered the bounding boxes with an area smaller than a given threshold. The results are shown in Table 8. Metrics were computed with the same filter used during the training process, as we were already taking into account the amount of area removed by the filter shown in table 4 and it could have been added as a random noise to the estimation if the results turned out to be better. Note that, even though the number of filtered bounding boxes was high, the relative area was low.

The results were not as expected and the hypothesis we claimed was rejected. Thus, we concluded that small bounding boxes were not damaging the training process of the model.

Minimum Area Threshold	F-Score	AP
0	**0.7977**	**0.6972**
840	0.7893	0.7065
1700	0.7924	0.6863
2500	0.7963	0.6887
4000	0.7971	0.7037
6700	0.7745	0.6696

Table 8: Minimum Area Experiment: The base configuration was the baseline with the adapted resize. For each trained model, the bounding boxes with area smaller than the threshold were filtered.

4.2.3.4 Anchor Shapes

An important section of the Faster R-CNN architecture that is set as a function of the dataset statistics is the anchors from where the Region Proposal Network builds the proposals. If we want to detect objects of an area that is too different to the anchor shape, it will be harder for the network to detect (objectness loss) and adjust (regression loss) the bounding box to the object's optimal shape. For this reason, we analysed the bounding boxes shapes and tried to find the anchor size and aspect ratios that best matched our dataset statistics.

The analysis consisted in studying the joint distribution of the width and the height of the bounding boxes, since in such space both the area and the aspect ratios can be studied at the same time. As explained in Section 2.2, for each layer extracted of the Feature Pyramid Network, we have to determine a number of anchors, where each anchor is set as a combination of an area and an aspect ratio.

To decide which anchor sizes and aspect ratios best matched our dataset, we ran a *k-means* algorithm where the distance function was the negated Intersection over Union. The reason for that metric was that euclidean distance (the usual distance for a *k-means* algorithm) does not take into account the magnitude of the points (boxes in this case) so when comparing two big bounding boxes, the euclidean distance is much bigger than when comparing two small bounding boxes, even when the difference between them is the same in relative terms. IoU normalizes the distance and, consequently, compare big and small boxes in the same way.

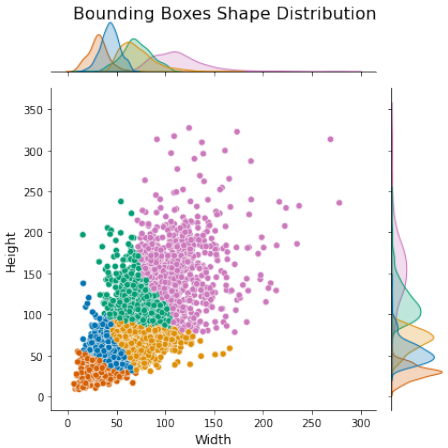


Figure 14: Joint distribution of the Bounding Boxes width and height. Each color is a different cluster assigned by *k-means* using the negated IoU metric.

Max Scale	FPN Layers	Inverse Aspect Ratios	F-Score	AP
400	2	No	0.3258	0.3183
400	4	Yes	0.7936	0.7190
400	4	No	0.8069	0.7222
400	5	No	0.7910	0.6852
500	**4**	**Yes**	**0.8186**	**0.7293**
500	4	No	0.8142	0.7237
600	5	Yes	0.8149	0.7310

Table 9: Anchors experiment results: The scales from the k-means clustering were used for every configuration, but the biggest scales were added, since bigger bounding boxes were not represented by the extracted centroids and, even though there weren't so many, they were the most important when it comes to the area. The inverse aspect ratios columns refers to using aspect ratios which give bigger width than height (inverse of what the centroids are). Surprisingly they gave better results.

Furthermore, since the *k-means* centroids are not representative of the biggest boxes (there are few boxes of such shapes), we added bigger anchors. It helped the network to fit better the biggest bounding boxes. Figure 14 shows the distribution of the bounding boxes shapes, along with the cluster assigned to each box. The k parameter is set to 5 clusters (after a short analysis based on the within-cluster sum-of-squares), which will have to be distributed through the different layers extracted to the Feature Pyramid Network.

Our baseline sets one anchor size and three aspect ratios (a total of 3 anchors) for each FPN layer. We have distributed the 5 clusters with different aspect ratios through the FPN layers in different ways. However, the best metrics were noticed after adding bigger shape anchors. At the same time we have tested the usage of a different amount of FPN layers, as the usage of too much information can be related to overfitting. When skipping FPN layers, we skipped the most general layers (biggest).

Table 9 shows the metrics of some of the models trained with different configurations of anchors sizes and number of FPN layers. They show that modifying the anchors sizes improves the models performances, by comparing them with the baseline, which had an F-score of 0.7977.

By comparing the best F-score found of **0.8186** with the human benchmark of **0.8230** (see Section 4.2.1), we concluded that the model had arrived to a **human level performance**. Therefore, because of the annotations quality (human mistakes, discrepancies, etc.), we can't expect to get better results than 0.823 in this dataset. Therefore, the metrics we achieved are close enough to be considered successful.

4.3 Weakly Supervised Semantic Segmentation

Once we developed an object detection model, an algorithm from a semantic segmentation point of view was approached. However, our annotations were bounding boxes instead of masks, which is the usual supervised learning setup to build semantic segmentation models. For this reason, weakly supervised techniques were applied.

The lack of time did not let us experiment with this architecture. Nonetheless, the network showed promising results, which can be in a par with supervised learning models if more experimentation and more accurate annotations were possible, as we will see in Section 5.

4.3.1 Dataset and Evaluation

Ideal evaluation of mask predictions was not achievable, as the ground truth were bounding boxes instead of masks. For this reason, a first visual and subjective evaluation was applied by overlapping the masks inferred by the trained models with the images and the ground truth bounding boxes. This visual evaluation was correlated with the evaluation metrics that we used to objectively assess the prediction capability of the model.

In order to objectively evaluate the model, we used two different metrics; the **mask metrics** explained in section 2.1.2 (taking as positive ground truth the background included in bounding boxes) and a variation of the **box metrics** that we used for the Faster R-CNN evaluation, explained in Section 2.1.1. The variation will be explained in the following section.

It has to be taken into account that **mask precision** should be 1 if a perfect ground truth (even being bounding boxes) was available. However, our annotations are not as accurate, since the annotators miss grape clusters (predominantly smaller boxes). This implies that the precision could be different from 1 and still be right. Biggest racemes are usually not missed so it should not impact the metrics too much.

On the other hand **mask recall** should be (ideally) exactly the proportion of the bounding boxes area covered by grapes. However, we don't know this factor so it is hard to estimate accurately. Through visual inspection, we can roughly estimate this fraction to be between 0.3 and 0.8 so we expect to get mask recall values of around these boundaries.

4.3.1.1 Box Metrics Variation

In order to approximate how many of the grape clusters were detected (recall), we understood as a match every bounding box that had at least a positive pixel inside. To approximate how many of the predictions were accurate (precision), each connected component of the predicted mask was a positive instance which, if it had at least one pixel inside a bounding box, it was considered as true positive. Otherwise it was considered as false positive.

These metrics would be quite accurate if we assumed that the precision inside bounding boxes is 1, meaning that the false positives of the model are not close to positives (grapes).On the other hand, we should assume that recall is also 1, meaning that it never detects only partially a cluster, so it either matches the whole grape cluster or it does not detect any grape.

It also has to be considered the analysis in Section 3.1.3, which shows how the distribution of the number of bounding boxes along with its covered area is very different. It also shows how smaller (and much more frequent) bounding boxes have lower confidence of being correctly annotated. Consequently, we expect the recall not to be as high.

4.3.2 Implementation Details

We have implemented the algorithm using the *PyTorch* library by extracting and adapting the key parts (the loss constraints) of the original implementation.[‖] Our implementation has isolated the restrictions with the log-barrier extensions in totally separated modules so that it is easy to reuse in any other project. The code is available online.[**]

The implementation of the postprocessing algorithm *Dense CRF* used was the Python wrapper PyDenseCRF.[††] The hyperparameters are the same as in the implementation we are based on [13].

[‖] https://github.com/LIVIAETS/boxes_tightness_prior

[**] https://github.com/paumarquez/AI4Agriculture-grape-detection

[††] https://github.com/lucasb-eyer/pydensecrf

	Initial Learning Rate	Patch Downscaling	Tightness Prior Width	Global Positive Size Lower Bound	Global Positive Size Upper Bound
v1	0.0010	Yes	15	0.25	0.7
v2	0.0010	Yes	15	0.50	0.7
v3	**0.0005**	**No**	**20**	**0.40**	**0.9**
v4	0.0010	Yes	15	0.40	0.8

Table 10: Hyperparameters of each configuration we have trained.

As in the original implementation, we set the λ parameter from Equation 8 to 0.0001. The optimization methodology also followed the original paper's methods. An Adam [20] optimizer decreasing the learning rate after 15, 38 and 55 epochs by a factor of 0.1. The models were trained during 80 epochs. The neural network used is the Residual U-Net [12].

We were having memory issues even after downscaling the images. Thus, as too much downscale would hurt the model, images were divided in squared patches of 1024 pixels which were fed to the network one by one. We tried to either avoiding any downscaling or downscaling the patches to 800×800 pixels. Furthermore, the dataset was highly unbalanced between the negative (background) and the positive (grapes) class. Thus, we skipped the patches that did not contain any positive pixel. Another option was to weight the negative function by a factor to be tuned. However, we opted to skip the patches without positive pixels, as the original paper did not use this hyperparameter. It turned out to extrapolate well to the negative patches during inference. The time parameter from the log-barrier extensions was updated every 10 epochs by a factor of 1.1 and initialized to 5.

4.3.3 Hyperparameters

Table 10 shows the different configurations we trained. As can be appreciated in Figure 1, we had very few time to experiment with this architecture so a small amount of experiments were carried out.

An important fact about some other failed experiments was the number of epochs that the model was trained, as it learned quite fast to detect grapes but it lasted a long time to learn to distinguish background, since it was labeling background as grapes (low precision).

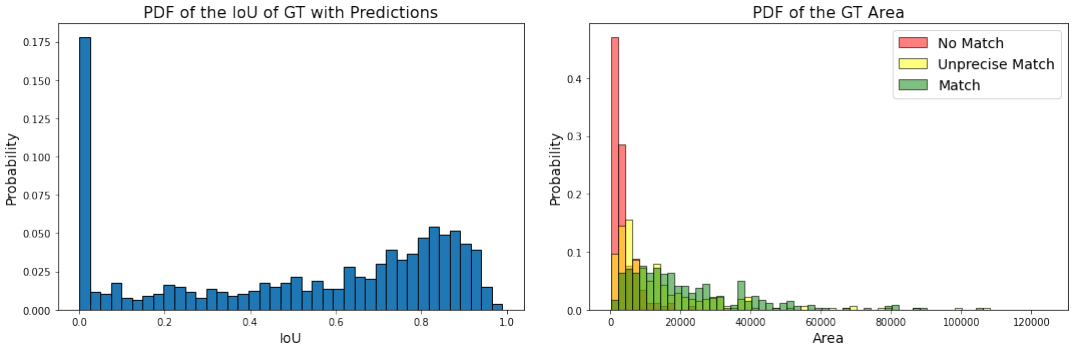
The definitive model v3 had no downsampling and the global positive size bounds were set to $[0.4, 0.9]$. Tables 11 and 12 show how the selected model has better metrics than other configurations in every aspect; Both recall and precision for box and mask metrics, lower standard deviations across different images, which means that the model is more consistent inferring in different scenarios. In the next section we are going to do a more in detail analysis of the models behaviour.

5. Algorithms Analysis

In order to understand both models behaviour, we are going to explain and show the trained models inferences along with the ground truth. These insights will be useful to researchers who take this thesis as a starting point as well as farmers who take pictures to be evaluated by the trained algorithms. We are also going to understand a bit more the metrics showed in Section 4 and the models weak points. Furthermore, **test metrics** for the definitive models will be shown. The analysis performed from now on is made on the **test dataset**.

5.1 Object Detection

We have shown in Section 4.2 how the Faster R-CNN metrics can be compared to human's annotations by using annotators as benchmarks. Nonetheless, the algorithm is still missing bounding boxes, (which are probably wrong or very hard to detect, as humans are missing them).



(a) (b)

Figure 15: (a) Probability Density Function of the IoU for each ground truth bounding box. Note how most inaccurate matched ground truth have skewness towards greater IoUs. (b) Probability Density Function of ground truth bounding boxes stratified by Faster R-CNN's matching predictions. *No Match* boxes have an IoU lower than 0.01, *Unprecise Match* boxes have an IoU $\in [0.01, 0.70]$ and *Match* boxes have an IoU $\in [0.70, 1.00]$. Take into account that unprecise match boxes might have a mask recall of 1, since the box can be inside a predicted bounding box.

Figure 15 shows the area distribution for missed bounding boxes, as well as detected boxes. It is clear how both differ in terms of the size of the boxes, as the missed ones have mostly less than an area of 5000. By taking a look at Table 4, boxes with an area lower than 5000 pixels include 6.4% of the area. Thus, we can conclude that almost 6.4% of the missed area (mask recall) is attributed to small bounding boxes, which have poor quality (remember Section 3.1.3.

Another unknown percentage of the mask recall can be attributed to the mismatch between annotators drawing methodologies, since even drawing to the same instance, there is a small difference in the area covered by both. The rest of the missed area should be associated to either wrongly labeled bounding boxes or real grapes racemes missed by the Faster R-CNN.

Figure 16 shows the low variance of the models across images. Different pictures of the dataset have

	Box Average			**Box Standard Deviation**		
	Precision	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
v1	0.827	0.564	0.670	0.248	0.254	0.251
v2	0.891	0.518	0.655	0.162	0.291	0.208
v3	**0.975**	**0.584**	**0.730**	**0.062**	**0.239**	**0.098**
v4	0.905	0.440	0.592	0.207	0.257	0.229

Table 11: Weakly supervised learning adapted box metrics (see Section 4.3.1.1). Standard deviation is taken into account, as there is a considerable variance on the metrics through different images.

	Mask Average			Mask Standard Deviation		
	Precision	**Recall**	**F-Score**	**Precision**	**Recall**	**F-Score**
v1	0.823	0.290	0.429	0.286	0.166	0.210
v2	0.832	0.2840	0.423	0.314	0.201	0.245
v3	**0.903**	**0.330**	**0.484**	**0.238**	**0.185**	**0.208**
v4	0.857	0.222	0.352	0.320	0.160	0.214

Table 12: Mask metrics for each weakly supervised model. Aggregation of the metrics has been applied through the different images in the validation set. Note that there is a huge variance on the performance of the models across different images. Even though the high variance, the mean is still high, which means that it has polarized metrics. A deeper analysis has been carried out in Section 5.

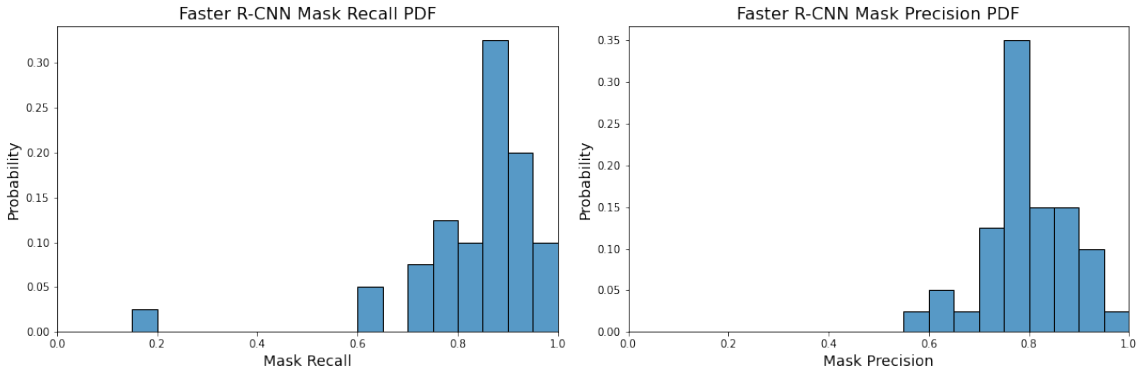


Figure 16: PDF of the Faster R-CNN mask recall and mask precision. It shows the low variance among different images of both recall and precision metrics. Thus, proving the robustness of the model.

different characteristics. Therefore, the fact that the model has learnt to perform well on different kind of scenarios proves the robustness and confidence of the model. Furthermore, we can see that the mean of the recall and the precision is close to the annotators metrics in Table 2, which is coherent with our hypothesis that the model's behaviour is the same as annotators behaviour.

Table 13 shows the metrics of the model evaluated in the **test set**. They give a more realistic estimation

Given the variance on the images annotations and images quality (see Section 5.3), the decrease in the metrics from the validation set to the test set is considered to be reasonable. Thus, we can conclude that the model has really learnt to detect grapes and generalized the knowledge learnt in the train set and validated in the validation set.

Mask Precision	**Mask Recall**	**Mask F-Score**	**Mask AP**
0.7968	0.8330	0.8022	0.7134

Table 13: Mask metrics extracted from the **test set**.

5.2 Weakly Supervised Semantic Segmentation

We selected the version **v3** from the experiments. Again, metrics from the **test set** (shown in Tables 14 and 15) do not diverge from the validation set metrics. Therefore, the model built shows to **generalize the knowledge** to the test set and we can conclude that we can expect the model to have the same performance in future images.

Box Average			Box Standard Deviation		
F-Score	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**
0.6849	0.9687	0.5297	0.118	0.0728	0.311

Table 14

Mask Average			Mask Standard Deviation		
F-Score	**Precision**	**Recall**	**F-Score**	**Precision**	**Recall**
0.4574	0.9291	0.3034	0.2115	0.216	0.2072

Table 15

Figure 17b shows a high precision with a low variance while Figure 17a shows a lower recall with a much higher variance. The box recall is not surprising, since the inconsistency between annotators and the high frequency of very small bounding boxes, which are much more likely to be wrongly labeled or undetected (see Section 3.1.1 and 3.1.3).

Figure 17 shows how high precision with low variance is achieved at the same time that the mask recall has a higher (and comprehensive) variance with lower values. See Section 4.3.1.1 for a better understanding of the possible reasons. It stands out the great number of images with mask recall 0. It gives the intuition that there are some image level characteristics that penalize the model so that the model is not able to detect not even a pixel from the whole set of grapes contained in the image.

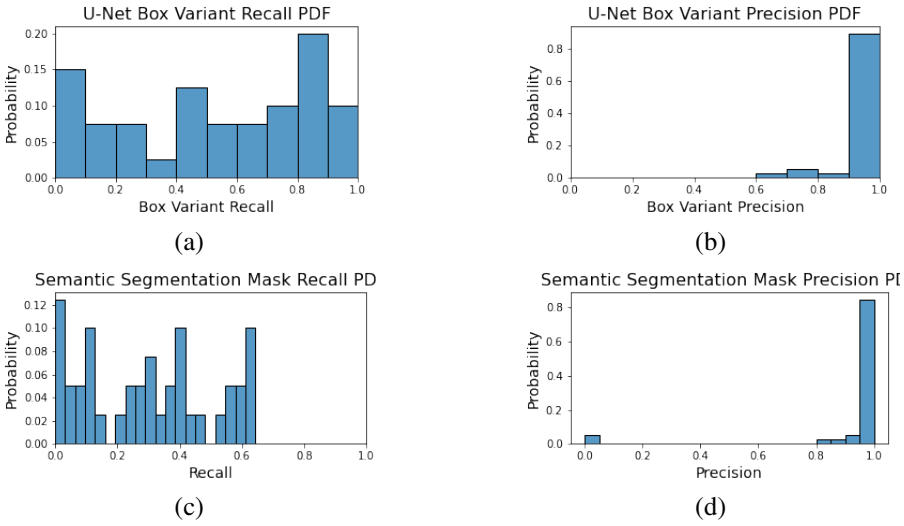


Figure 17: PDF of U-Net images precision and recall. (a) Box variant metrics. (b) Mask metrics

5.3 Demonstration and Combination of Models

The main goal of this project was to give an approximation of the number of pixels in an image covered by grapes. In order to achieve so, we will combine both semantic segmentation and object detection models. In this section we will visually understand their behavior and see how the object detection model can give an accurate and tight upper bound, while the semantic segmentation might help to squeeze the bound even more towards a better estimation. To understand their behavior we will see crops of some of the images with the worst metrics. This way, we will see the worst we can expect from each model.

In Figure 11 and 18, we can see crops of some of the images where the Faster R-CNN has the worst mask recall. Visually, Faster R-CNN is missing some ground truth bounding boxes, which make the mask recall go down. However, the missed boxes are hard to detect because of the shadow. Some of them can be considered as wrong labels, as it is not clear whether there are grapes inside or not. The easier ones are correctly detected by the Faster R-CNN.

On the other hand, **U-Net's performance** has a higher variance (as shown in Table 14 and Figure 17c). As said in the previous section, there is an image level variable that confuses the U-Net. Figure 19 shows crops of images where U-Net has not detected any positive pixel. Something they all have in common is the illumination, as sun points towards the lens of the camera and it reflects to the green leaves (see Figure 23 from Appendix for another representation of the illumination issue). Furthermore, in Figure 20 we can see how an image with low

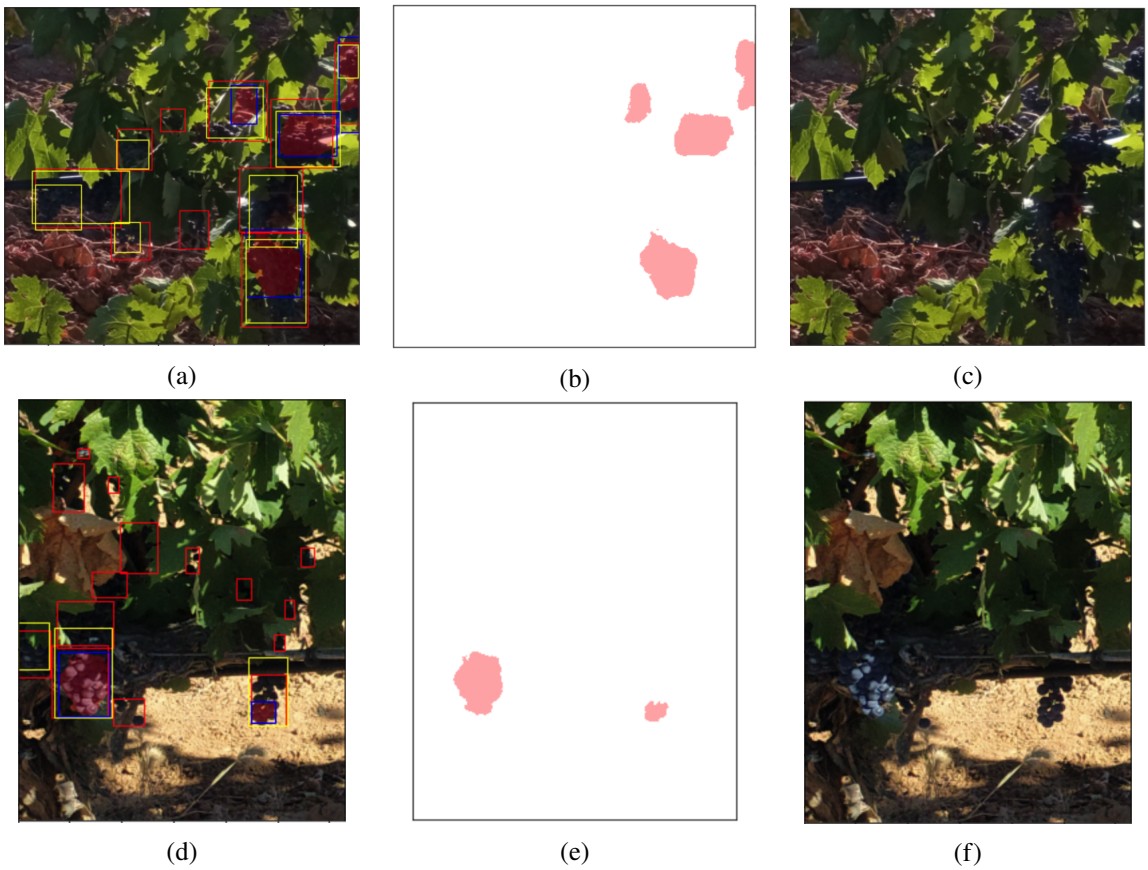


Figure 18: (a) Faster R-CNN (yellow), ground truth (red) and U-Net (red masks and blue bounding boxes) inferences. (b) Residual U-Net predicted mask. (c) Raw image.

(a) (b)

(c) 3c (d) 4c

Figure 19: Yellow and red bounding boxes are Faster R-CNN predictions and ground truth annotations, respectively. U-Net has not detected any positive pixel in any crop.

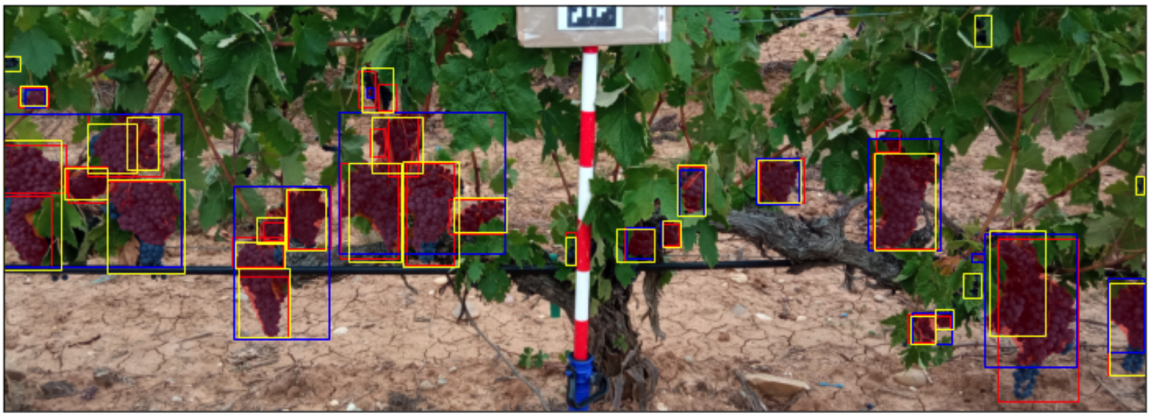
and constant illumination has much higher metrics.

In general, the precision metric is high and has low variance for both models, as showed in Tables 14 and 15 and Figures 16 and 17. Consequently, we can use their predictions as a reliable lower bound of what they should predict (in case of bounding boxes, including the background pixels inside the boxes). This fact is useful for the semantic segmentation model, which has a lower recall. Furthermore, because of the high precision and recall of the Faster R-CNN, the bound should be as tight as the average proportion of negative pixels per bounding box: $y + \delta a_t = a_t$ where y is the real number of negative pixels, a_t is the total sum over the bounding boxes area and δ is the real proportion of background pixels inside the bounding boxes. Hence, as $\delta \geq 0$ and $a_t \geq 0$, we can state that $y \leq a_t$, where δa_t is the tightness of the bound.

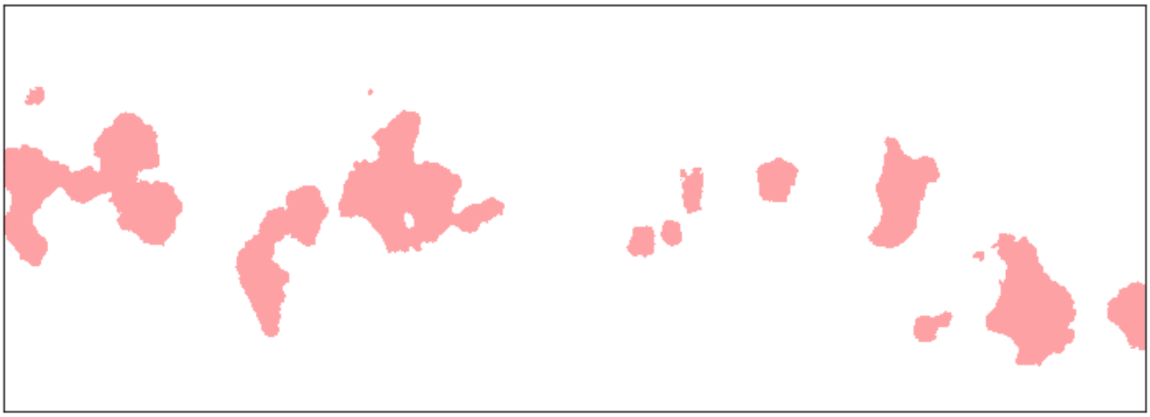
On the other hand, regarding the Faster R-CNN, the visual analysis of the images shows that even in images with the lowest mask recall, there are very few and not as relevant misses, since annotations are not accurate and the metrics we are getting have an implicit noise.

Regarding the U-Net, the recall is not as high. Box metrics did already tell us how 50% of the bounding boxes are missing (from which many small unreliable bounding boxes are included). In the visual analysis, we have checked how there is a considerable number of important undetected boxes.

Our solution is to take advantage of Faster R-CNN's great performance and use it as an upper bound estimation from which we can tighten up towards the real amount of positive pixels with the help of the U-Net masks. Therefore, we will filter U-Net predictions outside Faster R-CNN's boxes and use the ones inside them to filter out the possible background pixels included in the bounding box prediction. However, we have seen examples where the U-Net masks detect grape racemes only partially. Therefore, we should find a way to filter which bounding boxes we should accept U-Net predictions as reliable. As far as this study has gone, we propose to assume that if less than ϵ proportion of the bounding box area is positive, the prediction will miss positive pixels. We can set ϵ arbitrarily to 0.3 but a study should be made on how this parameter affects the estimation. Taking into account that many bounding boxes will be used without the U-Net masks, the included background pixels will balance the U-Net's missed positive pixels.



(a)



(b)

(c)

Figure 20: Annotations follow the same methodology as in Figure 11. Note how binary masks give a much more accurate estimation of the positive pixels than bounding boxes. Nonetheless, Faster R-CNN is still detecting more grape clusters than U-Net.

6. Conclusions and Future Work

In this project we have developed two AI algorithms with the purpose of detecting grape racemes. The dataset from which the algorithms have been trained is composed of vineyard images annotated with bounding boxes. Furthermore, since this dataset was created specifically for this project and we were the first researchers to put our hands on it, we also analysed the images along with their annotations.

Each of the built algorithms detects grapes in a different way; the object detection algorithm detects bounding boxes (rectangles that surround racemes) with the so called Faster R-CNN neural network. The semantic segmentation algorithm detects pixel-wise labels, delimiting more accurately the grape regions of the image. Concretely, the semantic segmentation model is a Residual U-Net convolutional neural network trained in a weakly supervised way.

A human level benchmark has been built in order to compare the prediction capability of the Faster R-CNN. With the help of this benchmark, we have concluded that the model's performance is on a par with humans. Therefore, predictions are confident with respect to both recall and precision.

Since bounding boxes contain background pixels, we can only give an (accurate) upper bound of the positive area. Because of this, we have also approached the problem from a semantic segmentation point of view. The model has shown great results when the images are taken under certain conditions. However, there are some images where the model does not work well (we theorized exposure was the main reason). The variance of the performance within different images is too high to consider it an independent and consistent model. Even so, we can use it along with the Faster R-CNN predictions to tighten up the upper bound towards a more accurate estimation.

In order to infer the total area of an image covered by grapes, **our proposed solution** is to sum the area inside Faster R-CNN's bounding boxes only where the number of positive pixels inside the bounding box predicted by the U-Net is lower than 30%. For the bounding boxes where the proportion of positive pixels is higher than 30%, use the U-Net prediction to filter pixels that are more likely to be background than actual grapes. This filter is necessary to reduce the chances that U-Net is only detecting a fraction of the grape raceme inside the bounding box.

The fact that the U-Net network worked so well in some images indicates that it is possible to build a network that distinguishes grapes pixel-wise even with only bounding box annotations. Since this network has been explored during a very short amount of time in this project, there is much performance left to get out of it. There are different possible next steps to improve this network; at first, it would be reasonable to analyse images stratifying by U-Net's worst metrics so that we could understand why it's failing and making sure illumination is really deteriorating the performance of the U-Net, as we stated. In order to solve the illumination issue, an option would be to apply illumination normalization preprocessing to the images before feeding them into the network. Another option would be to apply fine-tuning techniques to this project's model by sampling the current dataset in a way that illuminated images are balanced with normal ones, since it will help the model to learn how to distinguish grapes in both situations. Also, more hyperparameter configurations of the network should be tested, since we didn't experiment enough to get the highest possible results. Nonetheless, the current model can be used as a good baseline to start from.

The aforementioned dataset has been analysed and we have concluded that it needs a second iteration to refine its annotations. The main reasons being the discrepancies between annotator's methodologies and the huge quantity of wrong (mostly small) annotations. Furthermore, the dataset has a huge imbalance between light and dark grapes towards the dark ones. This fact can cause models to miss the lighter (and underrepresented) class of grapes. Our proposal to solve this problem is to annotate light and dark grape racemes into different

labels.

In addition, the way pictures are taken might affect future machine learning models built with this dataset. In order to train robust models, more data that represents the different kind of scenarios where images are taken might be needed. The example we have seen to affect the semantic segmentation model is the difference on the illumination, as sun might reflect directly towards the lens of the camera, turning camera's exposure up. Basic photography lessons can be given to the photographers to adjust the exposure settings while taking the pictures. They could also specify an hour of the day where there is no sun (early in the morning or in the evening) to take the pictures. It will have a huge effect, not only to collect images to enlarge the dataset, but to run models on them.

With respect to the reannotation process required, we propose our models as a starting point from which delete false positives and adjust true positives if needed. It will ease the annotation process and help standardizing annotators methodology. However, if annotators do not correct them, it will bias annotations towards the model's weaknesses.[‡‡]

We have seen the relation between the real amount of positive area and the total amount of area inside the predicted bounding boxes of the Faster R-CNN. The relation stands for $y + \delta a_t = a_t$, where y is the real number of positive pixels we want to estimate, a_t the total amount of area covered by bounding boxes and δ is the real proportion of background pixels inside the bounding boxes. Thus, the better we estimate δ, the better we will estimate y. For a future work, we can use the fact that U-Net works very well when the image is taken under certain conditions and run both models on a set of images to extract a (big enough) number of samples where the U-Net works well (by selecting the images with the highest mask recall). With this sample we could compute the proportion of positive pixels per bounding box and study its distribution (probably the joint distribution with the bounding box area). If the distribution turns to be i.i.d, applying the law of large numbers, we can assume that if we compute the proportion of positive number of pixels over a big enough number of bounding boxes using the proportion extracted from the aforesaid study, the resulting estimation of the total number of positive pixels will tend to the real value. Besides, it is a consistent estimator; the larger the number of bounding boxes we use to compute the estimation, the lower will be the variance of the estimation.

With some of the ideas we mentioned to do as the next steps, we think that accurate metrics could be achieved for the U-Net network, so that an accurate estimation of the area is given. Furthermore, the fact that we can detect grapes pixel-wise might lead to other studies which, as a function of grape color, texture, etc., we also give an estimation of the quality of the fruit.

[‡‡] Actually, we sent images annotated by our model where the discrepancies with the ground truth were greatest so that they could be corrected in the future, as many of the inconsistencies were caused by human errors.

References

[1] Ignasi Nogueiras Marco. *Análisis de imágenes de viñedos*. PhD thesis, UPC, Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona, Departament de Teoria del Senyal i Comunicacions, Jan 2021.

[2] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: Delving into High Quality Object Detection. *arXiv e-prints*, page arXiv:1712.00726, December 2017.

[3] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. *arXiv e-prints*, page arXiv:1703.06870, March 2017.

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv e-prints*, page arXiv:1505.04597, May 2015.

[5] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv e-prints*, page arXiv:1506.01497, June 2015.

[6] John Wilson. Instance segmentation, ai-pool. 2019.

[7] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object Detection in 20 Years: A Survey. *arXiv e-prints*, page arXiv:1905.05055, May 2019.

[8] Jasper Uijlings, K. Sande, T. Gevers, and A.W.M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104:154–171, 09 2013.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[10] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. *arXiv e-prints*, page arXiv:1612.03144, December 2016.

[11] Ross Girshick. Fast R-CNN. *arXiv e-prints*, page arXiv:1504.08083, April 2015.

[12] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. Road Extraction by Deep Residual U-Net. *IEEE Geoscience and Remote Sensing Letters*, 15(5):749–753, May 2018.

[13] Hoel Kervadec, Jose Dolz, Shanshan Wang, Eric Granger, and Ismail Ben Ayed. Bounding boxes for weakly supervised segmentation: Global constraints get close to full supervision. In Tal Arbel, Ismail Ben Ayed, Marleen de Bruijne, Maxime Descoteaux, Herve Lombaert, and Christopher Pal, editors, *Proceedings of the Third Conference on Medical Imaging with Deep Learning*, volume 121 of *Proceedings of Machine Learning Research*, pages 365–381. PMLR, 06–08 Jul 2020.

[14] Victor Lempitsky, Pushmeet Kohli, Carsten Rother, and Toby Sharp. Image segmentation with a bounding box prior. pages 277 – 284, 11 2009.

[15] Hoel Kervadec, Jose Dolz, Jing Yuan, Christian Desrosiers, Eric Granger, and Ismail Ben Ayed. Constrained Deep Networks: Lagrangian Optimization via Log-Barrier Extensions. *arXiv e-prints*, page arXiv:1904.04205, April 2019.

[16] Hoel Kervadec, Jose Dolz, Jing Yuan, Christian Desrosiers, Eric Granger, and Ismail Ben Ayed. Log-barrier constrained cnns. 04 2019.

[17] Philipp Krähenbühl and Vladlen Koltun. Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. *arXiv e-prints*, page arXiv:1210.5644, October 2012.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv e-prints*, page arXiv:1512.03385, December 2015.

[19] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context. *arXiv e-prints*, page arXiv:1405.0312, May 2014.

[20] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, page arXiv:1412.6980, December 2014.

A. Small Bounding Box Samples

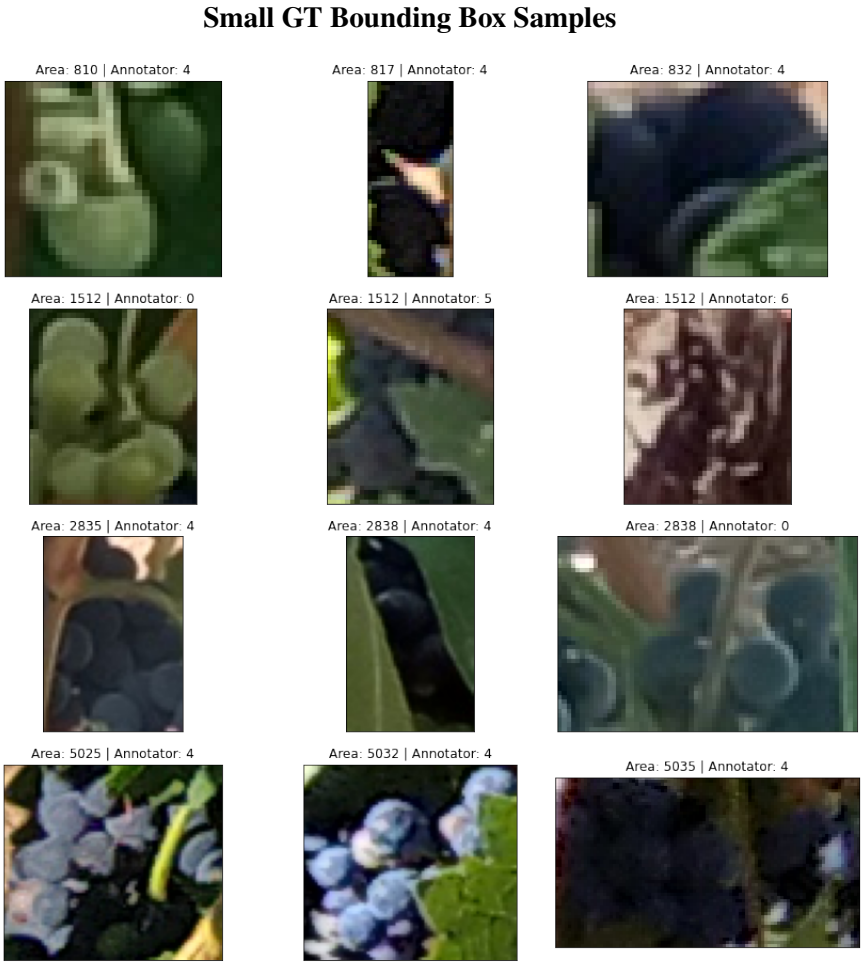


Figure 21: Samples of small ground truth bounding boxes. Note the low image definition and quality of the smaller ones, at the same time that some of these annotations are questionable.

B. Model's Inferences

(a)

(b)

(c)

Figure 22: (a) Faster R-CNN (yellow), ground truth (red) and U-Net (red masks and blue bounding boxes) inferences. (b) Residual U-Net predicted mask. (c) Raw image. Both models give accurate results. Note how some of the U-Net missed bounding boxes are green.

(a)

(b)

(c)

Figure 23: (a) Faster R-CNN (yellow), ground truth (red) and U-Net (red masks with blue bounding boxes) inferences. (b) Residual U-Net predicted mask. (c) Raw image. Illumination greatly affects the U-Net estimations while Faster R-CNN is consistent.