Segmentation Refinement by Region Growing - A Post-processing Module for Convolutional Neural Network

Kiprono Elijah Koech (kiprono@aims.ac.za)
African Institute for Mathematical Sciences (AIMS)

Supervised by: Dr. Bubacarr Bah
University of Stellenbosch and AIMS South Africa

Co-supervised by: Mr. Stuart Reid and Mr. Munsanje Mweene
Aerobotics, Cape Town, South Africa

24 October 2019
Submitted in partial fulfillment of a structured masters degree at AIMS South Africa

Abstract

Machine learning algorithms, precisely the convolution neural networks, has undergone tremendous improvements to reach start-of-art in their classification tasks. Despite these advancements, most of these classifiers and segmentation models do not attain a perfect performance in object detection and classification tasks. Some segmentation models fail in object boundary adherence, and others display instances of false detections for multiple classes. This provides an opportunity of refining the output of the models in a post-processing module. In this paper, we investigate a refinement approach based on region growing, namely Refinement by Region Growing (RbRG) algorithm. This method takes as input the confidence scores of the base model and corresponding image. The method refines the output of the base classifier by re-labelling the pixels in the low-confidence regions. To test the algorithm, we take the output of Mask Region-based Convolutional Neural Network (mask R-ConvNet) and use it as the input of the refinement approach. RbRG indeed improved the performance of the base model.

Keywords: Region Growing, Mask R-ConvNet, Instance Segmentation.

Declaration

I, the undersigned, hereby declare that the work contained in this research project is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

[illegible]

Kiprono Elijah Koech, 24 October 2019

Contents

1. Introduction

1.1 Image Understanding

Image understanding is a central theme in computer vision because of its many applications in the real world which contributes immensely to technological advancement. The applications of image understanding include the development of autonomous vehicles, military/security surveillance, traffic control systems, agricultural practices, among others. The primary outcome of image understanding is the determination of objects present in an image and the localization of the determined objects (Dias and Medeiros, 2018). Such determination and localization can be carried out in four steps: first, image classification, second, object detection, third, semantic segmentation, and lastly instance segmentation (Dias and Medeiros, 2018).

Image classification involves extracting information from an image at an image-level (Dias and Medeiros, 2018) whereas object detection localizes the objects in the image by bounding them with boxes (Dias and Medeiros, 2018). Semantic segmentation results in localized pixel-wise object detection, that is, object class label is assigned at a pixel level (Davis et al., 2018). Instance segmentation is an advancement of semantic segmentation so that at this level a pixel is not only identified as an object or not, but it is also assigned the object instance it belongs to (Dias and Medeiros, 2018). To understand these four concepts of image understanding, consider the image in Figure 1.1a with 5 fruits in it: image classification recognizes the presence of the fruits in the image, object detection localizes the fruits, semantic segmentation identifies all the fruit pixels without differentiating the 5 instances and lastly, instance segmentation defines the image as having 5 different fruits at specified locations.

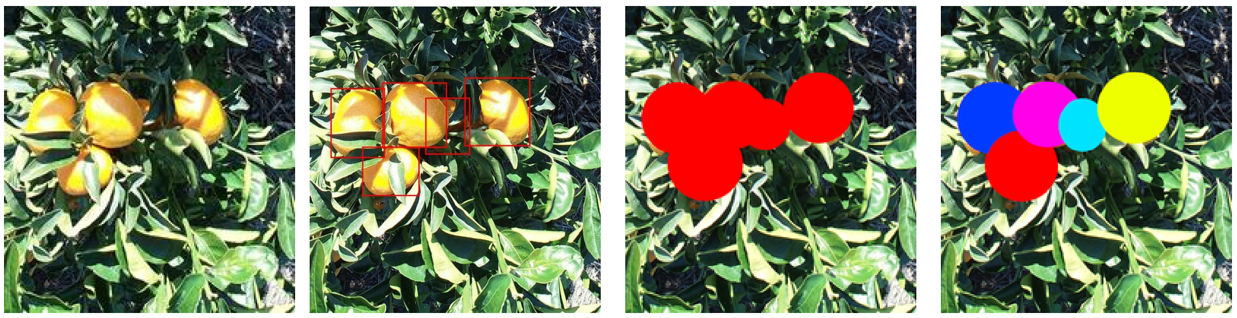

(a) RGB image. (b) Object detection. (c) Semantic segmentation. (d) Instance segmentation.

Figure 1.1: Image understanding. Illustration of the three concepts of image understanding, namely, object detection, semantic segmentation and instance segmentation.

In the past years, Convolutional Neural Network (ConvNet) has undergone tremendous improvements to reach the State-Of-The-Art (SOTA) in the image classification tasks. First on stage was Region-Based ConvNet (R-ConvNet) (Girshick et al., 2014), then Fast R-ConvNet (Girshick, 2015), followed by Faster R-ConvNet (Ren et al., 2015) all of which are frameworks for object detection. Fully Convolution Network (FCN) (Long et al., 2015) is available for semantic segmentation.

Mask R-ConvNet (used here) (He et al., 2017), which is an extension of Faster R-ConvNet, is a robust framework used in object instance segmentation.

1.2 Problem Statement

Machine learning has been thought to be a process executed with the following stages: preprocessing (gathering data, preparing data, and choosing the model), actual model training, evaluation, parameter tuning, and prediction. In image processing, the preprocessing stage entails annotating the images. Annotation is done by hand, which makes it not only tedious but also time-consuming and expensive. For these reasons, ConvNets are often trained on a small dataset making it prone to classification errors, which manifest in the output (Davis et al., 2018). Other factors affecting the model performance are intrinsic. They are deemed as natural model limitations — for example, spatial insensitivity caused by quantization during pooling and image down-sampling.

Despite Fast R-ConvNet (Girshick, 2015), Faster R-ConvNet (Ren et al., 2015) and Mask R-ConvNet (He et al., 2017) achieving SOTA results in the instance segmentation learning task, these models are prone to false positives[1] and other classification errors. In this project, Mask R-ConvNet model is trained, and results generated. These results are then refined using a post-processing module.

The refinement method used in this project is based on region growing, namely, Refinement by Region Growing (RbRG). RbRG is an approach that performs region growing based on appearance. This method takes the confidence scores from the initial classifier and threshold the image into three regions, namely, high confidence background, high confidence foreground, and the uncertainty regions. The pixels in the uncertainty region are often misclassified, and therefore RbRG aims to relabel these pixels. To achieve this, region growing is performed through adaptation of Simple Non-Iterative Clustering (SNIC) algorithm (Achanta and Susstrunk, 2017).

1.3 Related work

There exist some other methods that have been used in post-processing to refine the output. Here are some examples.

LabelBank (Hu et al., 2017). LabelBank is an approached based on ConvNet architecture, that is, a ConvNet is trained on the data to estimate the object classes present in a given image and filter out labels in the output that do not belong to the list. This approach is especially efficient in removing of out-of-context errors [2] but not in-context errors [3].

Davis et al. (2018), in their paper, discussed a post-processing method based on Bayesian framework. The approach uses confusion statistics of the classifier to refine the initial pixel label hypothesis. This method is an improvement of LabelBank (Hu et al., 2017) because it is more general as it removes both in-context and out-of-context errors.

Jing et al. (2019) proposes a novel recursive coarse-to-fine semantic segmentation framework based on image-level category labels. This framework conducts refinement in the following steps: (i) Generate a coarse-mask by a ConvNet, (ii) Enhancement of the coarse-mask using a graph model and lastly, (iii) The enhanced model is fed into a fully convolutional network to be recursively refined.

[1]A **false positive** is a detection of an object which in fact is not an object.

[2]**Out-of-context errors** arises from labelling a pixel in error by giving it a label which is not within the set of actual labels for that particular image.

[3]**In-context errors** are labelling errors that arise form labelling a pixel in error by assigning it a wrong label amongst the set of actual labels for the given image.

1.4 Dissertation Outline

The rest of the dissertation is structured as follows: In Chapter 2, we review mask R-ConvNet model. Chapter 3 describes RbRG method of refinement and state the algorithm. For Chapter 4, we describe the dataset used, results and discussions for mask R-ConvNet and mask R-ConvNet + RbRG. We also compare the performance of the two models. Finally, we conclude by giving the limitations of the study and contingent on these limitations provide suggestions for future work in Chapter 5.

2. Mask R-ConvNet review

In this Chapter we will discuss the literature review of mask R-ConvNet model. In Section 2.1, we give a brief history of the model and explain how it works in Section 2.2.

2.1 Brief History of Mask R-ConvNet

Mask R-ConvNet is a result of the improvement of previous ConvNet algorithms. Below are the algorithms (in order from the oldest) representing the development of ConvNet up to mask R-ConvNet:

2.1.1 R-ConvNet (Girshick et al., 2014).

R-ConvNet is an object detection algorithm whose objective is to generate a bounding box for every object in an image. To obtain this objective, R-ConvNet runs in 4 steps. First, the algorithm generates a set of region proposals ($\sim 2k$) which will be checked through Selective Search (SS)(Sande et al., 2011) to determine whether or not a given proposal contain the object. Second, for each proposal generated, we run a deep learning model and the result of the model is fed into a Support Vector Machine (SVM) classifier which identifies the object in the region. Lastly, a regression model is run on the output of the classifier to improve the bounding boxes by refining their location coordinates.

2.1.2 Fast R-ConvNet (Girshick, 2015).

This algorithm is an improvement of R-ConvNet with focus on promoting computational simplicity and increasing the training speed. These objectives are attained in two modules. Firstly, SVM classifiers in R-ConvNet are replaced by a classification layer (to output classification results) and a regression layer (to output the bounding boxes). Secondly, in fast R-ConvNet, SS (Sande et al., 2011) is done on the feature map instead of the entire image input. This saves us the need of sending each proposal through the entire network. The feature map is processed by RoIPool (Region of Interest Pooling) which is usually achieved by max pooling. Clearly, fast R-ConvNet completes its object detection tasks by jointly training a ConvNet, a classifier and a regressor in a single network whereas R-ConvNet runs three different models: ConvNet model to extract features, an SVM classifier and a regressor to improve the bounding boxes.

2.1.3 Faster R-ConvNet (Ren et al., 2015).

SS in Fast R-ConvNet is replaced with Region Proposal Network (RPN) which is placed after the last convolution of the feature map. RPN generates all the proposals required which means that the training of the neural network is done in a single stage.

2.1.4 Mask R-ConvNet (He et al., 2017).

This framework is an improvement of the faster R-ConvNet. Mask R-ConvNet is different from its predecessor in 3 ways: Firstly, it uses Feature Pyramid Networks (FPNs) to improve object detection. RoIPool in faster R-ConvNet quantizes floating number into an integer to match the granularity of the grids on the window. This quantization on RoIPool causes misalignment and data loss which in turn negatively affect masks prediction. To solve this problem, mask R-ConvNet uses RoIAlign instead of RoIPool. The former does not quantize floating numbers. Lastly, mask R-ConvNet generates classification results, bounding boxes and masks around the objects as output.

2.2 Mask R-ConvNet Architecture

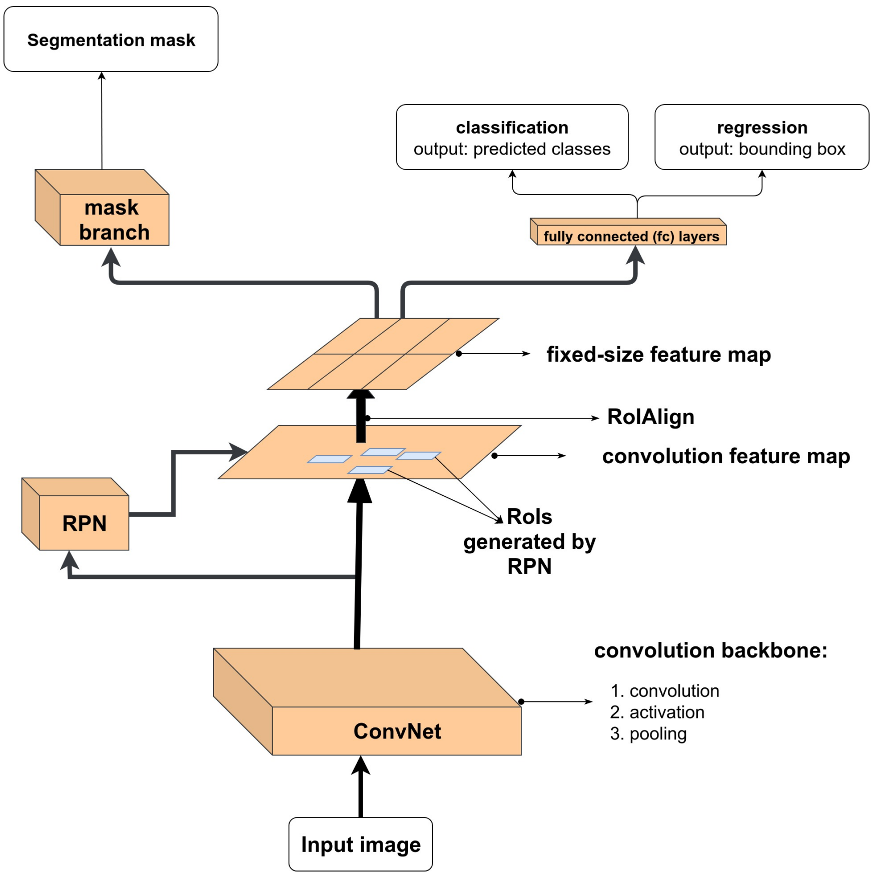


Figure 2.1: Mask R-ConvNet Architecture.

2.2.1 Convolutional backbone. The backbone processes an entire input image by passing it through several layers of convolution, activation and pooling (in that order).

(a). **Convolutional layers** (Khan et al., 2018).

Convolution, in these layers, is achieved by sliding a kernel/filter along the entire input image. Mathematically, convolution is an operation based on dot product. For an input image,I of size $h \times w$, kernel,K of size $K_h \times K_w$ then the of the size of the output,O will be $(h - K_h + 1) \times (h - K_w + 1)$.

Formally, convolution is defined as,

$$O(r, c) = \sum_{p=1}^{K_h} \sum_{q=1}^{K_w} I(r + p - 1, c + q - 1) K(p, q),$$

where,

- $r \in \mathbb{Z}^+ | 1 \leq r \leq (h - K_h + 1)$ and $c \in \mathbb{Z}^+ | 1 \leq c \leq (w - K_w + 1)$,
- $O(r, c)$, $I(r, c)$ and $K(r, c)$ is the entry at the r^{th} row and the c^{th} column of the output, input and the kernel respectively.

Example,

Here, $h = 5$, $w = 5$; $K_h = 3$, $K_w = 3 \implies$ Output dimension $= (5-3+1) \times (5-3+1) = 3 \times 3$.

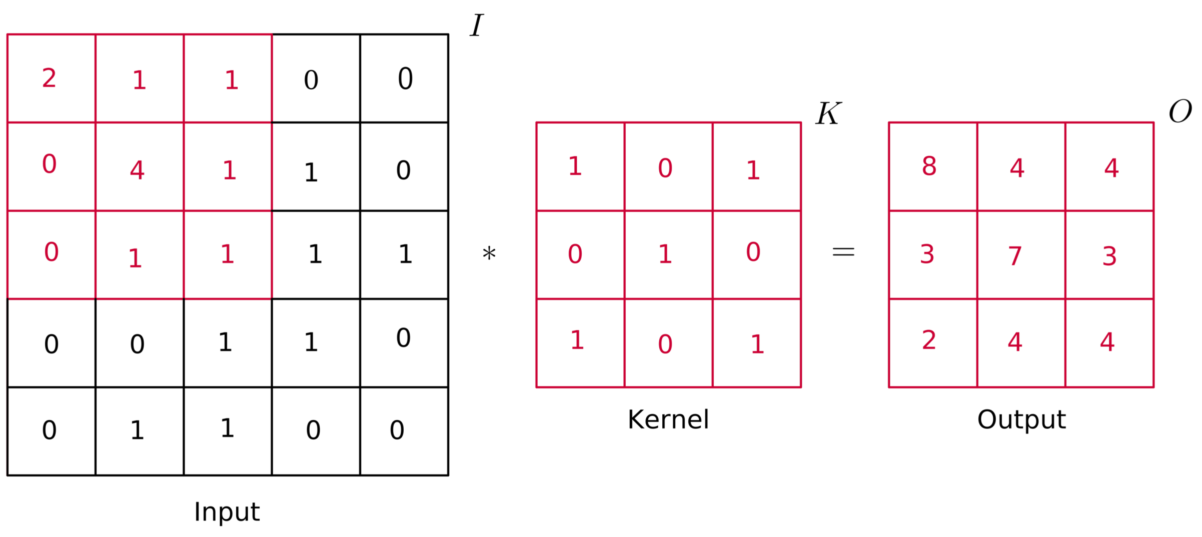


Figure 2.2: **Best Viewed when coloured:**A simple convolution process.

As an illustration, let us calculate the value of $O(1,1)$ in Figure 2.2,

$$\begin{aligned}
O(1,1) &= \sum_{p=1}^{3}\sum_{q=1}^{3} I(r+p-1, c+q-1)K(p,q),\\
&= \sum_{p=1}^{3} [I(1+p-1,1)K(p,1)] + [I(1+p-1,2)K(p,2)] + [I(1+p-1,3)K(p,3)]\,,\\
&= [I(1,1)K(1,1) + I(1,2)K(1,2) + I(1,3)K(1,3)]\\
&\quad + [I(2,1)K(2,1) + I(2,2)K(2,2) + I(2,3)K(2,3)]\\
&\quad + [I(3,1)K(3,1) + I(3,2)K(3,2) + I(3,2)K(3,2)]\,,\\
&= 2(1) + 1(0) + 1(1) + 0(0) + 4(1) + 1(0) + 0(1) + 1(0) + 1(1),\\
&= 2 + 1 + 4 + 1,\\
&= 8.
\end{aligned}$$

Convolution process may invoke the use of terminology such as padding and strides which can be defined as follows:

Definition (Stride(u,v))(Khan et al., 2018)

Stride is the number of pixels with which we slide the kernel horizontally or vertically along the image. Stride(u,v) implies that we slide the kernel over u pixels horizontally and v pixels vertically.

Definition (Padding)(Khan et al., 2018)

Padding an image is adding pixels on the border of the image pixel matrix. This is done when the kernel does not fit perfectly on the feature map during convolution. Padding process can be done in two ways:

i. **Same padding** (Khan et al., 2018) - This involves padding the image borders with zeros so that the kernel fits perfectly. This form of padding is also called zero padding.

ii. **Valid padding** (Khan et al., 2018) - Padding in this case involves dropping the part of the image that is not covered by the kernel.

Dimensions in convolution

In this section, we are going to see how to determine the dimensions of the output component given dimensions of the input, dimensions of the kernel, the stride length and the padding size.

Consider the following variables:

$$\begin{aligned}
h &- \text{height of the input image,}\\
w &- \text{width of the input image,}\\
d &- \text{colour channel e.g. RGB colour channel has d=3 and GRAYSCLALE has d=1,}\\
K_h &- \text{kernel height,}\\
K_w &- \text{kernel width,}\\
m &- \text{number of kernels used,}\\
u, v &- \text{ stride length on the horizontal direction and vertical direction respectively,}\\
p &- \text{Padding size } (p = 0 \text{ implies no padding}).
\end{aligned}$$

Given the values of the above variables, dimensions of the output can be determined as follows:
Input: $h \times w \times d$,

Padding size: p,

Stride length: (u, v),

Kernel: $K_h \times K_w \times d$, m,

Output: $\left(\frac{h + 2p - K_h}{v} + 1\right) \times \left(\frac{w + 2p - K_w}{u} + 1\right) \times m.$

Example,

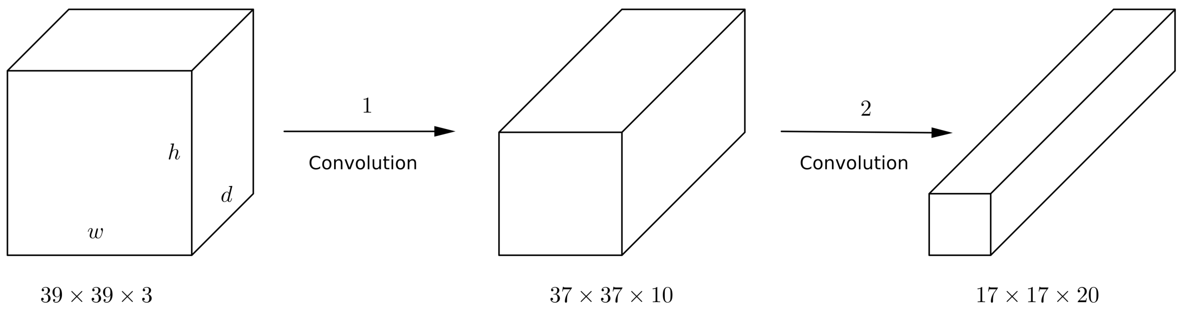


Figure 2.3: A simple convolution process

For the convolution process, 1 in Figure 2.3, we can calculate the output dimensions as follows,

Input: $h \times w \times d = 39 \times 39 \times 3$,

Padding size: $p = 0$,

Stride: $(u, v) = (1, 1)$,

Kernel: $K_h \times K_w \times d,\ m = 3 \times 3 \times 3,\ 10$,

$$\begin{aligned}\textbf{Output} &= \left(\frac{h + 2p - K_h}{v} + 1\right) \times \left(\frac{w + 2p - K_w}{u} + 1\right) \times m,\\ &= \left(\frac{39 + 2(0) - 3}{1} + 1\right) \times \left(\frac{39 + 2(0) - 3}{1} + 1\right) \times 10,\\ &= 37 \times 37 \times 10.\end{aligned}$$

For the convolution process, 2 in Figure 2.3 we can determine the output dimensions as follows,

Input: $h \times w \times d = 37 \times 37 \times 10$,

Padding size: $p = 0$,

Stride: $(u, v) = (2, 2)$,

Kernel: $K_h \times K_w \times d, m = 5 \times 5 \times 10, 20$,

$$\begin{aligned}\textbf{Output} &= \left(\frac{h + 2p - K_h}{v} + 1\right) \times \left(\frac{w + 2p - K_w}{u} + 1\right) \times m,\\ &= \left(\frac{37 + 2(0) - 5}{2} + 1\right) \times \left(\frac{37 + 2(0) - 5}{2} + 1\right) \times 20,\\ &= 17 \times 17 \times 20.\end{aligned}$$

(b). **Activation layers** (Khan et al., 2018)

At this point, the convolved map is passed through an activation function with the aim of introducing non-linearity which enables the ConvNet to learn complex data from the input map. The commonly used activation function is Rectified Linear Unit (ReLU) for non-linear operation. The output of ReLU function is defined as:

$$f(x) = \max(0, x), \text{where} \quad 0 \leq f(x) < \infty,$$

where x is the input entry and $f(x)$ is the output entry. Other activation functions includes:

i. Tanh function;

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \quad \text{where} - 1 \leq f(x) \leq 1.$$

ii. Logistic/Sigmoid function;

$$f(x) = \frac{1}{1 + e^{-x}}, \quad \text{where} 0 \leq f(x) \leq 1.$$

It is important to note that activation operation is an element-wise operation implying that the dimension of the input and output of the activation layer is the same.

(c). **Pooling layers** (Khan et al., 2018)

After convolution and activation, the resulting map is pooled. Just like convolution, pooling reduces the dimensionality of the feature map while retaining important details about the image. Reducing the dimension of the feature map is an important concept because it enhances computational simplicity. While pooling, dominant features about the input map are extracted and by so doing the output map will turn out to be positional and rotational invariant. The two commonly pooling methods include:

i. **Max pooling**- It returns the maximum value of all the entries covered by the sliding kernel.

ii. **Average pooling** - It returns the average value of all the entries covered by the kernel.

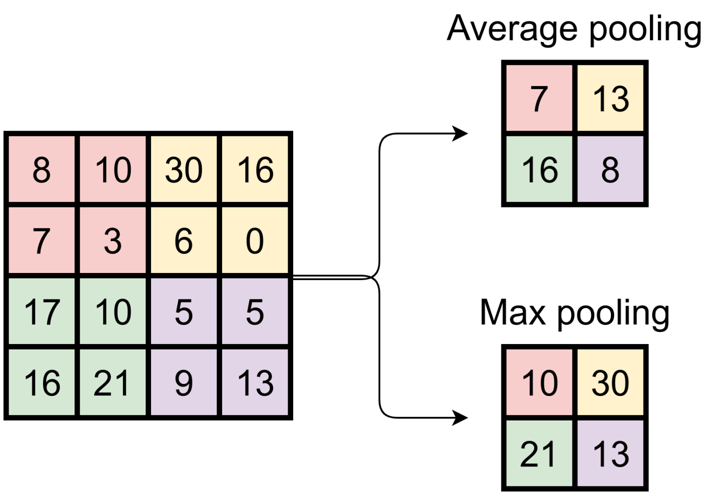


Figure 2.4: Pooling process with Kernel of size 2×2 and stride$(2, 2)$.

2.2.2 Region Proposal Network (RPN). (Ren et al., 2015)

This network takes as input a ConvNet feature map from the last layer of convolution and outputs rectangular object proposal called anchor boxes. Each proposal has an objectness score [1]. It is important to note that RPN's proposals have only two classes: foreground (object) and background. To define a region proposal, we need to state the objectness score and its position. For these reasons, RPN has both a classifier (cls) layer and a regressor (reg) layer. cls generates the probability (objectness) of a region proposal to contain the target object and reg defines the position of the proposal using coordinates.

Anchors (Ren et al., 2015)

An anchor is defined as the centre of a sliding-window in RPN. At each sliding-window location, k region proposals are made. The value of k is determined by the scale[2] and aspect-ratio[3] of the image.

[1]**Definition** (Objectness score): This is the likelihood of belonging to a given class: object or background.
[2]Scale of an image is simply the size of an image.
[3]Aspect ration=$\frac{\text{Width of the image}}{\text{Height of the image}}$

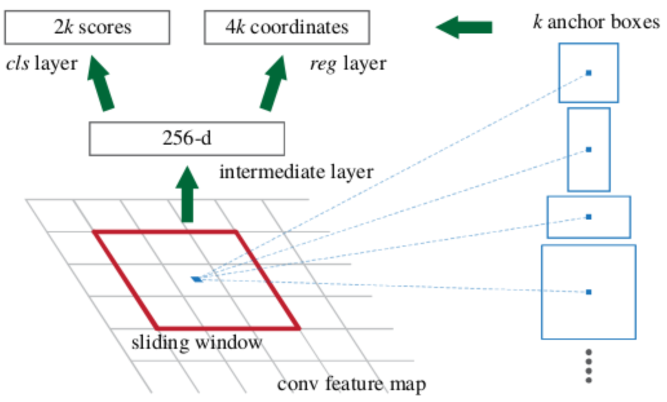


Figure 2.5: Anchor. Credits :(Ren et al., 2015)

Example, suppose we choose 3 scales (say 128-d, 256-d, 512-d) and 3 aspect ratios (say 1:1, 1:2, 2:1) then for every sliding-window location k=9 region proposals are generated.

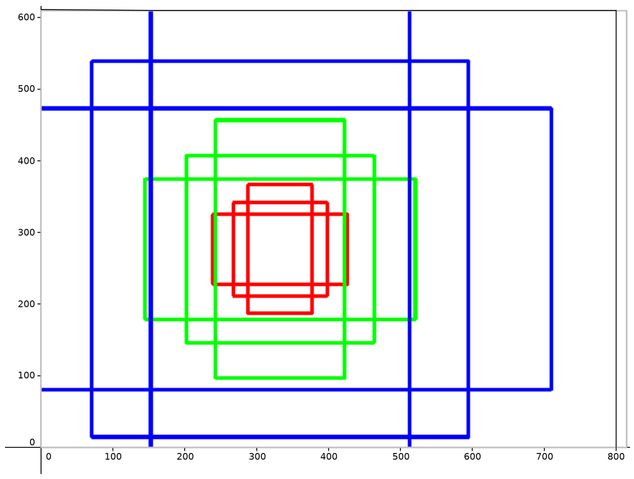


Figure 2.6: **Best Viewed when coloured:** Anchors at (320,290)

Figure 2.6 shows 9 boxes anchored at (320,290) for image of size 800 $\times$ 610. The three colours represent the three scales, and for every colour, three boxes are representing the three aspect rations 1:1, 2:1 and 1:2.

reg layer of RPN

For each sliding window location, reg layer outputs $4k$ values defining the coordinates of the anchor box (proposal), that is, every proposal box is defined by a 4-tuple coordinates, (x, y, w, h) where (x, y) is the box center, w the box width and h the box height.

cls layer of RPN

cls layer outputs $2k$ probability values for every region proposal, that is, every proposal has two probability values: the probability of the object being present in the proposal and otherwise. These probabilities are generated based on Intersection over Union (IoU) metric, which is defined as follows:

$$IoU = \frac{\text{Anchor box} \cap \text{Ground-truth box}}{\text{Anchor box} \cup \text{Ground-truth box}}$$

.

The final anchor label is defined as an object if IoU> 0.7, not object if IoU< 0.3 and undetermined otherwise (Ren et al., 2015). The anchor boxes with undetermined label do not contribute to the training of the RPN. In contrast, proposal boxes which are labelled to contain the target object make valid Region of Interest (RoI).

2.2.3 RoIPool and RoIAlign.

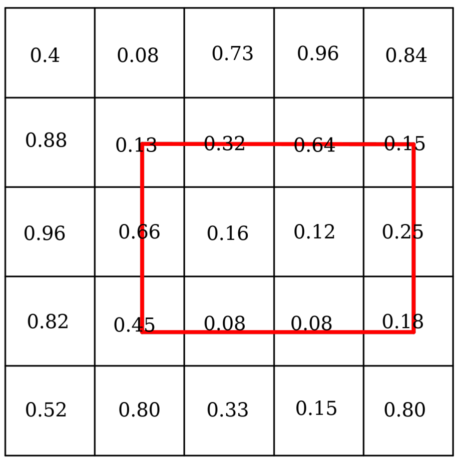
RoIPool (Girshick, 2015) uses max pooling operation to convert the features inside a valid RoI (RoIs are not uniform in size) into a smaller predefined fixed-size feature map of $H \times W$(say 7×7). Each RoI is a rectangle defined by a 4-tuple (l, c, h, w) where (l, c) is the top-left co-ordinates and h, w the height and width respectively. RoI pooling works by dividing $h \times w$ RoI window into sub-window of approximately h/H and w/W and max-pooling each sub-window into a corresponding output buffer. In most cases, h/H and w/W are floating numbers and we need to quantize them so that the resulting bins and RoIs fit the granularity of the ConvNet feature map. Quantization is done by either rounding-up or rounding-down the floating number into the nearest positive integer. Rounding-up leads to the problem of misalignment and rounding-down causes data loss. Misalignment and data loss negatively affect the accuracy of the mask prediction in mask R-ConvNet. To address these problems, we use RoIAlign instead of RoIPool.

In RoIAlign (Ren et al., 2015) we conduct max pooling on continuous bins, that is, we avoid quantization by using $x/7$ instead of $[x/7]$ where $[\cdot]$ is a rounding operation. Using bilinear interpolation (Hurtik and Madrid, 2015), we compute the values of 4 regular sampled points in each RoI window and aggregate the result by taking the maximum or the average of the four values.

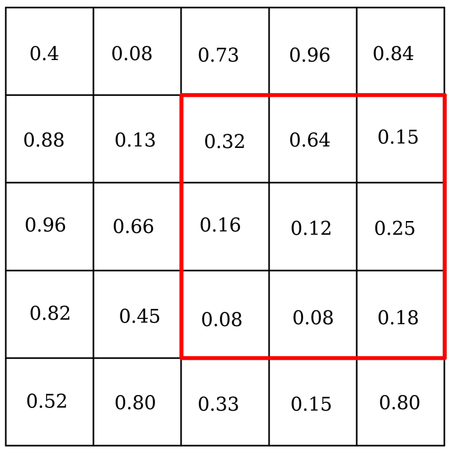
To understand the concept of RoIPool and RoIAlign lets consider the following example:

(i) *RoIPool*

Let us consider a 5×5 feature map with RoI plotted in it as shown in Figure 2.7a. Notice that the boundaries of RoI do not coincide with the granularity of the feature map in the said Figure.



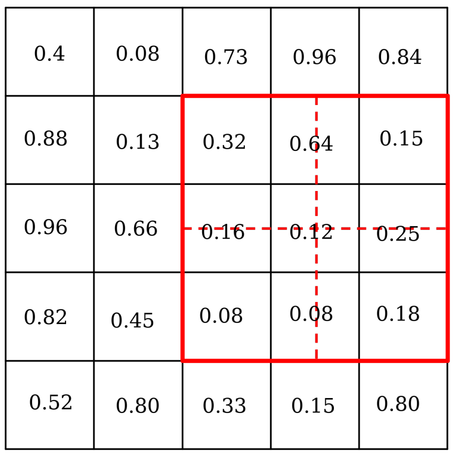
(a) 5× 5 feature map with RoI plotted.



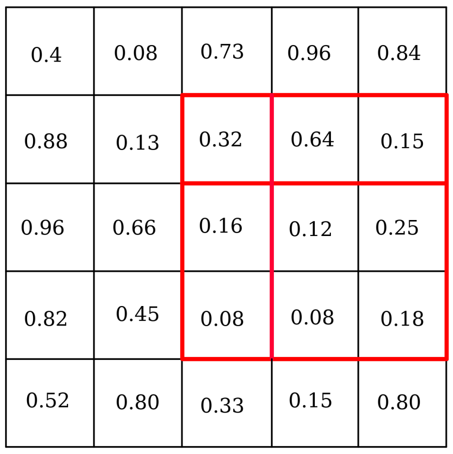
(b) 5× 5 feature map with new RoI boundaries plotted.

Figure 2.7: **Best Viewed when coloured:** Delimitation of RoI boundaries in a 5×5 feature map - RoIPool process.

In RoIPool, we solve this by rounding-off the boundaries of the RoI to match the granularity of the map which leads to new RoI boundaries as shown in Figure 2.7b and a new RoI of size 3×3. With the new RoI we divide the region into bins of pre-determined size,m. In our case we consider $m = 2$.



(a) Sub-windows (bins) in the RoI.



(b) Quantized bins.

Figure 2.8: **Best viewed when coloured:** Quantizations of bin boundaries

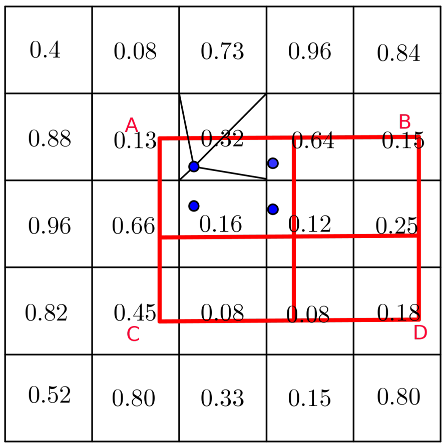
We perform quantization operation again to get the bins matching the granularity of the feature map as shown in Figure 2.8b. We then perform max pooling operation in each of the bin resulting to a $[m] \times [m]$ volume shown below.

0.32	0.64
0.16	0.25

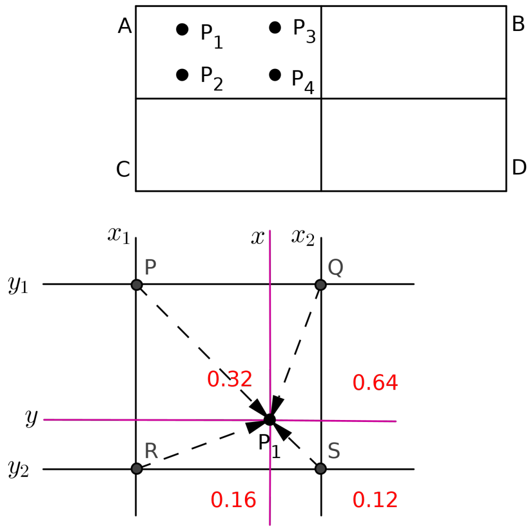
Figure 2.9: RoIPool: Results of max pooling on the bins.

(ii) RoIAlign

We do not perform quantization in RoIAlign. We instead make use of the coordinates of the RoI with respect to the feature map to calculate the values of the four sampled point within respective bins. Consider the feature maps shown in Figure 2.10a with RoI and bins plotted.



(a) Sub-windows (bins) in the RoI.



(b) **Top**: Sub-windows not quantized. **Bottom**: Focusing on finding $P_1 = f(x, y)$.

Figure 2.10: **Best Viewed when coloured:** RoIAlign. Four points sampled from the first bin.

The RoI coordinates are $A(1.7, 1.4)$, $B(4.7, 1.4)$, $C(1.7, 3.6)$ and $D(4.7, 3.6)$. From every bin, we take four points (as shown for one bin in Figure 2.10a and zoomed in 2.10b top). The coordinates of the four sampled points are $P_1(2.075, 1.675)$,$P_2(2.825, 1.675)$, $P_3(2.075, 2.225)$ and $P_4(2.825, 2.225)$. The number and the position of the sampled points do not matter so much.

Lets concentrate in finding the value of $P_1 = f(x, y)$. First, we need to identify 4 nearest points on the feature map that matches the resolution/granularity of the feature map (that is, points whose coordinates will be integers). The 4 nearest points are $P(2, 1)$, $Q(3, 1)$, $R(2, 2)$ and $S(3, 2)$ (see Figure 2.10a and the zoomed version in 2.10b bottom). Secondly, we need the identify the 4 feature map values corresponding to the 4 points. In this case we have the values 0.32, 0.64, 0.16 and 0.16 corresponding to P, Q, R and S respectively. Lastly, we calculate the value of $P_1 = f(x, y)$ using bilinear interpolation technique (Hurtik and Madrid, 2015) as follows:

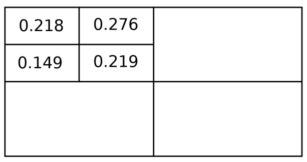
Performing interpolation to the x-direction yields

$$\begin{aligned}
f(x, y_1) &\approx \left(\frac{x_2 - x}{x_2 - x_1}\right) f(P) + \left(\frac{x - x_1}{x_2 - x_1}\right) f(Q) && (2.2.1)\\
&\approx \left(\frac{3 - 2.075}{3 - 2}\right) (0.32) + \left(\frac{2.075 - 2}{3 - 2}\right) (0.64) && (2.2.2)\\
&= 0.344 && (2.2.3)\\
f(x, y_2) &\approx \left(\frac{x_2 - x}{x_2 - x_1}\right) f(R) + \left(\frac{x - x_1}{x_2 - x_1}\right) f(S) && (2.2.4)\\
&\approx \left(\frac{3 - 2.075}{3 - 2}\right) (0.16) + \left(\frac{2.075 - 2}{3 - 2}\right) (0.12) && (2.2.5)\\
&= 0.157 && (2.2.6)
\end{aligned}$$

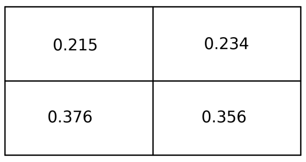
We then perform interpolation on y-direction which results to

$$\begin{aligned}
f(x, y) &\approx \left(\frac{y_2 - y}{y_2 - y_1}\right) f(x, y_1) + \left(\frac{y - y_1}{y_2 - y_1}\right) f(x, y_2) && (2.2.7)\\
&\approx \left(\frac{2 - 1.675}{2 - 1}\right) (0.344) + \left(\frac{1.675 - 1}{2 - 1}\right) (0.157) && (2.2.8)\\
&= 0.218 && (2.2.9)
\end{aligned}$$

This process is repeated for the other three sampled points in the first bin and also the sample points on the other three bins.



(a) Average pooling the values on the bins.



(b) Results of average pooling.

Figure 2.11: RoIAlign: Average pooling on bins.

2.2.4 Fully connected layers. (Khan et al., 2018)

Fully connected (fc) layers takes as input fixed-sized volumes from RoIAlign. From these volumes, fc generates RoI feature vector by flattening the input volumes. The RoI feature vector is then used to predict the object class (classification) and to generate bounding boxes (regression). Classification is often done using softmax classifier. Unlike the cls layer in RPN which outputs only two classes for the RoI: foreground and background, softmax classifier is deeper than that because it outputs the probabilities of K object classes plus a background class.

2.2.5 Segmentation masks. (He et al., 2017)

In the mask branch, a mask is generated for the object at the valid RoI using FCN(Long et al., 2015). The generated masks are of low resolution, 28×28 pixels, and can be scaled. During training, the ground-truth masks are scaled down to 28×28 so that training loss can be determined. The final mask is obtained by scaling up of the predicted masks to the size of the bounding box for inferencing and visualization. The generated mask is small in size so that the mask branch is kept light.

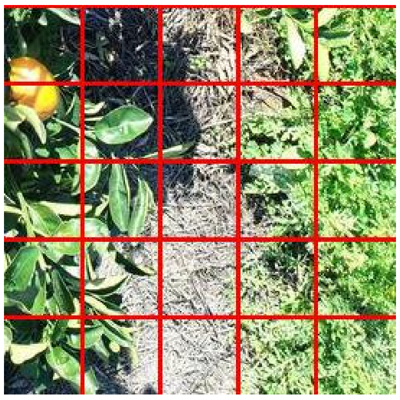
3. Refinement by Region Growing (RbRG)

RbRG is a post-processing module that can be coupled with the output of any ConvNet or a similar semantic segmentation model to improve the performance of the base classifier. In this paper, RbRG is used to refine the predictions of mask R-ConvNet model. This method is going to take as input an RGB image and the corresponding confidence scores from the mask R-ConvNet and refine the segmentation boundaries.

3.1 Key Concepts

3.1.1 Confidence scores.

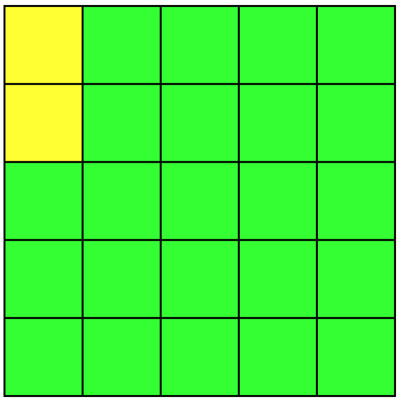
Confidence scores are per-pixel probabilities generated by a typical ConvNet model at the softmax layer. The number of these probabilities per pixel are as many as the number of object classes.



(a) 5× 5 RGB image

F:0.8 B:0.2	F:0.4 B:0.6	F:0.4 B:0.6	F:0.3 B:0.7	F:0.1 B:0.9
F:0.9 B:0.1	F:0.3 B:0.7	F:0.4 B:0.6	F:0.2 B:0.8	F:0.2 B:0.8
F:0.4 B:0.6	F:0.3 B:0.7	F:0.2 B:0.8	F:0.3 B:0.7	F:0.2 B:0.8
F:0.3 B:0.7	F:0.4 B:0.6	F:0.2 B:0.8	F:0.3 B:0.7	F:0.0 B:1.0
F:0.2 B:0.8	F:0.3 B:0.7	F:0.4 B:0.6	F:0.1 B:0.9	F:0.3 B:0.7

(b) Confidence scores for 2 classes



(c) Confidence map for 2 classes

Figure 3.1: (a) RGB image with pixels drawn. (b) confidence scores generated by softmax in mask ConvNet with two classes: Fruit (F for foreground) and not fruit (B for background). (c) confidence map for the two classes.

From Figure 3.1b, it is notable that the confidence scores makes a $5 \times 5 \times 2$ volume for 5×5 image with 2 classes. In fact, for a $H \times W$ image and k classes, the confidence scores are a $H \times W \times k$ volume. It is also important to note that, for a classification task with only two classes, confidence scores may mean providing just one probability, p per pixel because the second value will be $1 - p$.

3.1.2 Region growing. (Kamdi and Krishna, 2012) This is a region-based image segmentation approach whose major aim is to identify and group all homogeneous (similar) pixels in an image. The implementation of region-growing is in the following steps:

- *Picking initial seeds.* These seeds act as the initial regions for the growing process. Choosing the initial seeds can be done in different ways which include, picking the pixels within a certain grayscale range, pixels spaced regularly on the image grid, randomly et cetera.
- *Growing the regions.* The initial regions are grown by adding the pixels in the neighbourhood into the region based on some similarity criterion. This process of growing is iterative and will continue until all the pixels belong to some region.

3.1.3 The neighborhood and order of the neighborhood. (Boskovitz and Guterman, 2002)

A d-th order neighborhood over a $H \times W$ image L is defined as

$$N^{(d)} = \{N^{(d)}(x,y) \subset L : (x,y) \in L\}$$

where $N^{(d)}(x,y)$ is such that

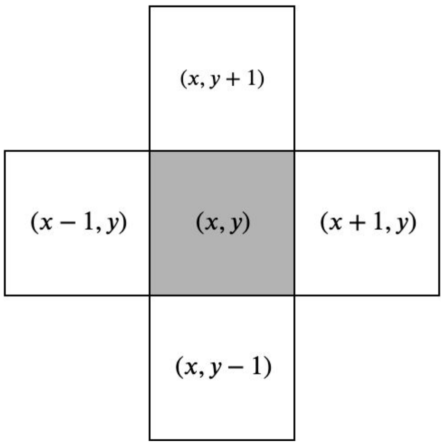
- pixel $(x,y) \notin N^{(d)}(x,y)$,
- pixel $(t,s) \in N^{(d)}(x,y) \implies$ pixel $(x,y) \in N^{(d)}(t,s)$,
- $\{x \in \mathbb{Z}^+ : 1 \leq x \leq W\}$ and $\{y \in \mathbb{Z}^+ : 1 \leq y \leq H\}$,
- pixel (x,y) is termed as the central pixel (centroid) over the neighborhood $N^{(d)}(x,y)$.

(i) *First-order neighborhood* (Figure 3.2a). A first-order neighborhood of pixel (x,y) is a set of pixels,

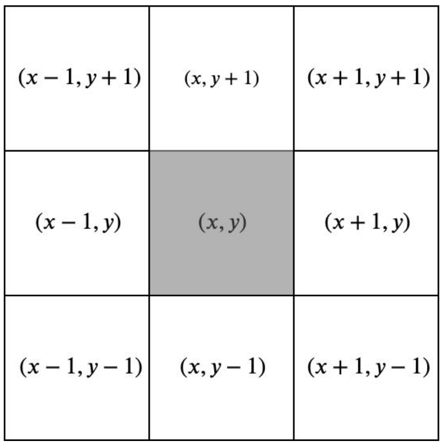
$$N^{(1)} := \{(x \pm 1, y), (x, y \pm 1)\}.$$

(ii) *Second-order neighborhood* (Figure 3.2b). This neighborhood consists of a set of pixels,

$$N^{(2)} := \{(x \pm 1, y), (x, y \pm 1), (x \pm 1, y \pm 1)\}.$$



(a) first-order neighborhood



(b) second-order neighborhood.

Figure 3.2: The neighborhood of pixel (x,y).

3.2 Implementation of RbRG

RbRG method can be implemented in the following steps: i) Relabelling pixels by thresholding; ii) Region growing; iii) Dealing with 'orphan' pixels and iv) Final classification.

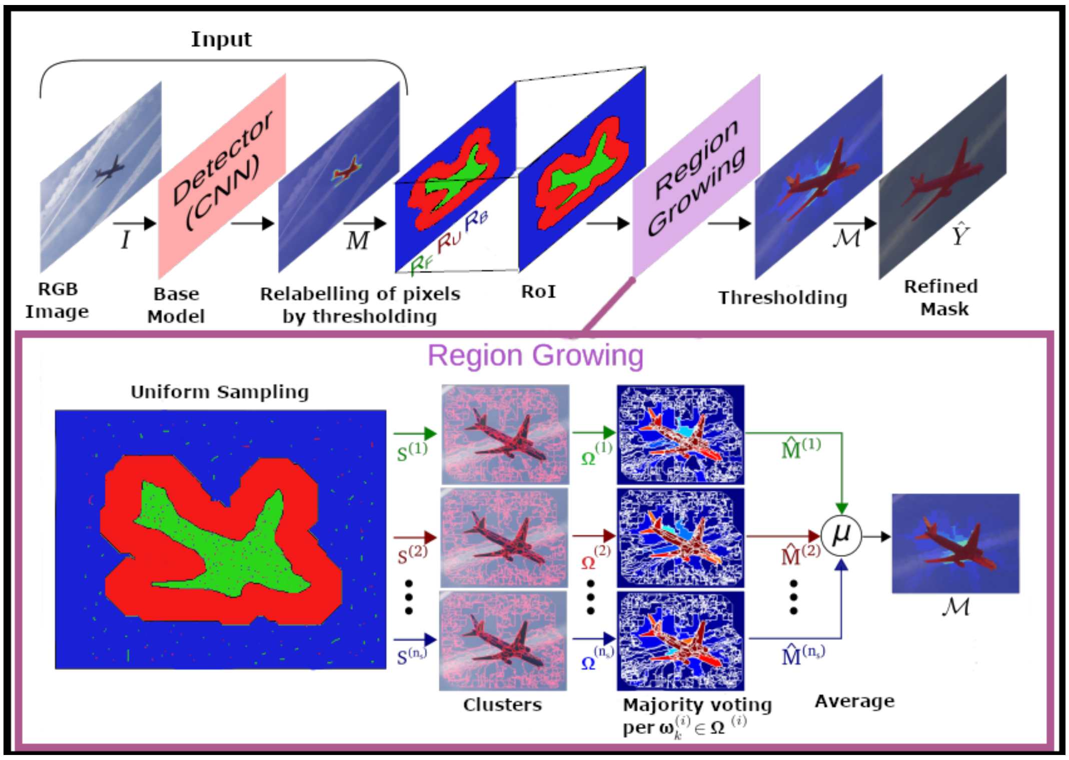


Figure 3.3: RbRG architecture. The input is confidence scores (M) from the base model and the corresponding RGB image, I. Relabelling of pixels is done to divide the image into three regions. RoI is then defined. Region growing is performed and lastly final classification is conducted by thresholding ·The output is the refined mask $\hat{Y}$. Credits: (Dias and Medeiros, 2018)

(i) *Relabelling of pixels by thresholding* (Dias and Medeiros, 2018). Typical ConvNet models infer the final classification by pixel-wise thresholding the confidence scores obtained for each class. The ConvNet threshold value (η_0) is an hyperparameter. In this step we will exploit the confidence scores and threshold the image into three regions: (i) High confidence object (R_F); (ii) Uncertainity region (R_U); (iii) High confidence background (R_B). These regions are identified using two threshold values: η_F and η_B such that the three regions can be defined by the equation

$$\begin{aligned} R_F &= \{p_i | D(p_i) > \eta_F\}, \\ R_U &= \{p_i | \eta_B \le D(p_i) \le \eta_F\}, \\ R_B &= \{p_i | D(p_i) < \eta_B\}, \end{aligned} \tag{3.2.1}$$

where p_i is the i-th pixel on the input image, $D(p_i)$ is the confidence score at the i-th pixel. The choice of η_F and η_B represents a trade-off between false positive detections and false negative detections. Values lower than η_B identify pixels on high confidence background and the values larger than η_F indicates pixels on high confidence foreground region. η_B is choosen close to 0.0 and η_F close to 1.0.

To understand this step, let us consider the confidence scores we introduced in Figure 3.1b. In every pixel, we will consider the probability of being an object (the probability F). We will also choose $\eta_B = 0.1$ and $\eta_F = 0.8$ and continue to relabel the pixels as shown in the Figure 3.4 below:

0.8	0.4	0.4	0.3	0.1
0.9	0.3	0.4	0.2	0.2
0.4	0.3	0.2	0.3	0.2
0.3	0.4	0.2	0.3	0.0
0.2	0.3	0.4	0.1	0.3

(a) Confidence scores.

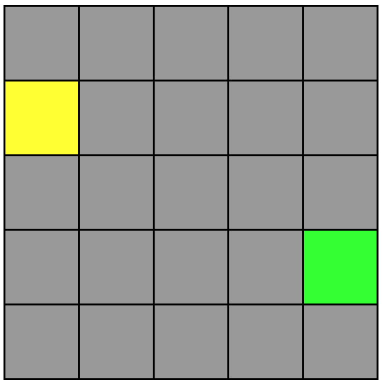

(b) **Best viewed when coloured:** Confidence map for relabelled pixels.

Figure 3.4: (a) Confidence scores for probability of object being present. (b) Confidence map for relabelled pixels (yellow represents R_F, gray represents R_U and green represents R_B).

(ii) *Region growing.* Based on the notion of pixel affinity employed in most superpixel segmentation algorithms, RbRG applies region growing to reclassify the pixels at the uncertainty region (R_U). Specifically, we leverage the Simple Non-Iterative Clustering (SNIC)(Achanta and Susstrunk, 2017) method for superpixel segmentation.

Unlike SNIC which samples the initial seeds on a regular grid over the whole image, RbRG will uniformly sample the initial seeds on high confidence area (R_H), where

$$R_H = R_F \cup R_B.$$

Let p_h be the pixels within R_H where $h = 1, 2, \ldots, |R_H|$, then, the seeds are uniformly sampled in an index-wise manner such that $h \sim U(1, |R_H|)$ defines the sampling process. There should also exist spacing between the samples so that the centroids can propagate over the uncertainty zone.

The region growing process evolves iteratively. In a given iteration, i, a set , $S^{(i)}$, of initial seeds are generated such that for n_s iterations we have the sets $S^{(1)}, S^{(2)}, \ldots, S^{(n_s)}$ of initial seeds, making the initial regions. Each subsequent step of region growing process involves the addition of unlabelled pixels, one by one, into the regions based on some similarity criterion, that is, a candidate pixel is added into the region if it is similar to the pixels already in the region.

3.2.1 Definition (Unlabelled pixels)**.** Unlabelled pixels (Adams and Bischof, 1994) refers to those pixels which have not been admitted into a given region. Consider a state of set S_i after n iterations then τ will be the unlabelled pixels defined as

$$\tau = \left\{ (x, y) \notin \bigcup_{i=1}^{n_s} S_i | N^{(p)}(x, y) \cap \bigcup_{i=1}^{n_s} S_i \neq \emptyset \right\}, \tag{3.2.2}$$

where $N^{(p)}(x, y)$ is the set of p-th order neighborhood of (x, y).

3.2.2 Definition (Similarity criterion)**.** In RbRG method, the similarity of pixels is defined based on the colour and spatial location just like in SNIC (Achanta and Susstrunk, 2017). The similarity between a given candidate pixel and the centroid is defined as a distance in 5-dimensional space of colour and spatial co-ordinates. Let the spatial location of a pixel be a vector $\mathbf{x} = [x, y]^T$ and the colour-space be represented by the vector $\mathbf{c} = [l, a, b]^T$ in the CIE-lab colour channel. The distance $\delta(j, k)$ is then defined as

$$\delta(j,k) = \sqrt{\frac{\|\mathbf{x}_j - \mathbf{x}_k\|_2^2}{s} + \frac{\|\mathbf{c}_j - \mathbf{c}_k\|_2^2}{m}}, \tag{3.2.3}$$

where $\delta(j, k)$ is the distance between a candidate pixel j and the centroid k, s and m are normalizing factors for the spatial and colour distance respectively. For an image of size $H \times W$, each of the K superpixels is expected to contain $(H \times W)/K$ pixels. Assuming a square superpixel, the value of s in 3.2.3 is set to $\sqrt{(H \times W)/K}$. m is termed as the *compactness factor* and it is user-defined. A higher value of m results in more compact superpixel at the cost of boundary adherence, and vice-versa. In our study, we relax the assumptions that every superpixel is square-shaped and that the superpixels are of the same size and allow for unequal clusters with different shapes.

A candidate pixel is said to be similar with the centroid if the distance defined in Equation 3.2.3 is minimum,that is,

$$\min\{\delta(s,t)\}, \tag{3.2.4}$$

such that $s \in \tau^c$ and $t \in \tau$.

Now that we have defined the initial seeds/regions we proceed to add the unallocated pixels into the regions. To do this, we will first stage the unallocated pixels to a priority queue. The unallocated pixels considered relative to the centroid are the pixels on the first and second-order neighbourhood, as shown in Figure 3.2a and 3.2b respectively. The pixels that have been added into the priority queue are ordered according to the level of similarity with the region being grown according to the metric defined in 3.2.3. While the queue is non-empty as stated in 3.2.2 every iteration in the region growing process entails the following steps (Achanta and Susstrunk, 2017) (i) Popping the first element (pixel) on the queue, which corresponds to an unlabelled pixel in τ that is mostly similar to the centroid. (ii) Annexing this pixel into the respective cluster/region, ω_k^i. (iii) Updating the region/centroid. (iv) Populating the queue with the neighbours of this pixel that are still in τ.

Region growing process occurs in a Region of Interest (RoI), and we will define RoI as the whole image such that:

$$RoI = R_B \cup R_U \cup R_F. \tag{3.2.5}$$

When we define the RoI as the whole image, we make RbRG to be not only efficient for images with one object but also those with multiple objects.

(iii) *Dealing with 'orphan' pixels.*

To reduce the amount of false positive detections, we define another constraint on the metric 3.2.3. We require the distance between the centroid and the pixel to be added into the queue to be less than some value $\delta_{\max}$.

By introducing this constraint, some pixel will not be admitted by any cluster and therefore remains as 'orphan' pixels at the end of region growing process. Such pixels are surmised to belong to the background class.

The region growing process comes to an end with the output that for each set of initial seeds S_i a cluster map $\Omega^{(i)}$ that associates each pixel to the corresponding cluster $\omega_k^{(i)}$ is generated.

(iv) *Final classification.* (Dias and Medeiros, 2018)

Final classification is based on majority voting. A pixel, (p_i), is regarded as a positive vote on the foreground class if its original confidence score $D(p_i) > \tau_0$ for some threshold value τ_0. A positive score is defined by the equation

$$Y = \{p_i | D(p_i) > \tau_0\}. \tag{3.2.6}$$

We then compute the ratio of positive votes for each set of clusters and generate a refined likelihood map as follows

$$\hat{M}^{(i)}(p_i) = \frac{|Y|}{|\omega_k^{(i)}|}. \tag{3.2.7}$$

The likelihood maps $\hat{M}^{(i)}$ obtained for n_s clusters are averaged to generate the refined score map

$$\mu = \frac{1}{n_s} \sum_{i=1}^{n_s} M^{(i)}. \tag{3.2.8}$$

Finally, a pixel is labelled as foreground if more than 50% of the average votes are positives, else, the pixel is labelled as background.

$$\hat{Y}(p_i) = \mathbb{1}_{\mu(p_i) > 0.5}. \tag{3.2.9}$$

3.3 RbRG Algorithm

Input: RGB image, I, and confidence scroes in the format shown in Figure 3.1b.
Output: Refined semantic segmentation $\hat{Y}$ for I.

```
Relabel pixels in $M$ to obtain 3 regions: high background ($R_B$), uncertainty region ($R_U$) and high
 foreground ($R_F$) ;
Define RoI as in Equation 3.2.5;
for (i=0 to i=$n_s - 1$) do
    Sample initial seeds to a set $S^{(i)}$;
    Generate a set of clusters $\Omega^{(i)}$ by region growing $S^{(i)}$;
    For the generated cluster $\omega_k^{(i)} \in \Omega^{(i)}$, compute a confidence map $\hat{M}^{(i)}$ as in Equation 3.2.7;
end
Compute pixel-wise average of $\hat{M}^{(i)}$, $i = 1, 2, \dots n_s$ to obtain $\mu$ ;
Generate $\hat{Y}$ by pixel-wise thresholding $\mu$ with a threshold value of 0.5 ;
```

Algorithm 1: RbRG Algorithm.

4. Results and Evaluation

In this chapter, we give a description of the dataset used (see Section 4.1), discuss various model evaluation metrics (in Section 4.2) and give the results of mask R-CNN (4.3) and mask R-ConvNet+RbRG (that is, RbRG coupled with mask R-ConvNet) (4.4). In the last Section (4.6), we will give a conclusion on the efficiency of RbRG in the refinement process.

4.1 Dataset Description

The dataset used in this project consist of natural outdoor images of farm trees and the aim is to detect all the fruits in every tree. The images are sourced by flying the drone $\sim$ 2m above the tree canopy which generated $2.7k$ video imagery. These short videos are then segmented to generate images which are then used to train the Mask R-ConvNet model. The model was trained on 1000 images, and the output of the model is used as the input to RbRG. We make use of 42 images to test the efficiency of RbRG.

4.2 Evaluation Metrics

4.2.1 Intersection over Union (IoU)(Everingham et al., 2015).

IoU, also called Jaccard index, is a metric that evaluates the overlap between the ground-truth mask (M_{gt}) and the predicted mask (M_{pd}). In object detection, we can use IoU to determine if a given detection is valid or not.

IoU is calculated as the area of overlap/intersection between M_{gt} and M_{pd} divided by the area of the union between the two, that is,

$$\mathrm{IoU} = \frac{\mathrm{area}(\mathrm{M}_{gt} \cap \mathrm{M}_{pd})}{\mathrm{area}(\mathrm{M}_{gt} \cup \mathrm{M}_{pd})} \tag{4.2.1}$$

IoU metric ranges from 0 and 1 with 0 signifying no overlap and 1 implying a perfect overlap between M_{gt} and M_{pd}.

4.2.2 Confusion Matrix(Fawcett, 2006).

This is a table showing the performance of a classifier given some truth values/instances. A confusion matrix is made up of 4 components, namely, True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN).To define all the components, we need to define some threshold (say α) based on IoU.

		Prediction	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

Figure 4.1: Confusion matrix over 2 classes: positive and negative

- True Positive (TP) - This is an instance in which the classifier predicted positive when the truth is indeed positive, that is, a detection for which IoU $\geq \alpha$.
- False Positive (FP) - This is a wrong positive detection, that is, a detection for which IoU $< \alpha$.
- False Negative (FN) - This is an actual instance that is not detected by the classifier.
- True Negative (TN) - This metric implies a negative detection given that the actual instance is also negative. In object detection, this metric does not apply because there exist many possible predictions that should not be detected in an image. Thus, TN includes all possible wrong detections that were not detected.

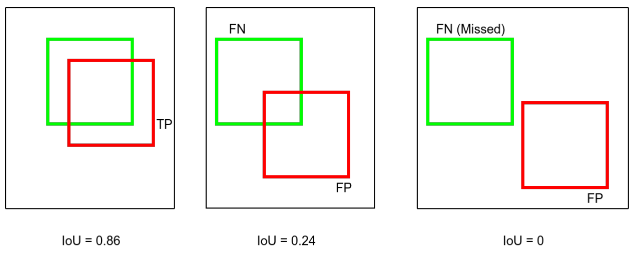


Figure 4.2: **Best Viewed when coloured:** Illustration of TP, FP and FN with an IoU threshold of 0.5. Green bounding boxes represents the ground truths and the red ones represents the predictions.

Notation: We will indicate IoU threshold as a subscript of the confusion matrix components; for example, TP$_\alpha$ implies a true positive detection for an IoU threshold of α.

The components of the confusion matrix can be used to calculate other metrics which includes precision, recall and F_1 score.

(i) Precision is the ability of a classifier to identify only relevant objects. It is the proportion of correct positive predictions and is given by

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{TP}{\text{all detections}} \tag{4.2.2}$$

(ii) Recall is a metric which measures the ability of a classifier to find all the relevant cases (that is, all the ground-truths). It is the proportion of true positive detected among all ground-truths and is defined as

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{TP}{\text{all truths}} \tag{4.2.3}$$

.

(iii) F_1 score is harmonic mean of precision and recall.

$$F_1 \text{ score} = 2\left(\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}\right) \tag{4.2.4}$$

4.3 Results of Mask R-ConvNet

The output of mask R-ConvNet includes segmentation masks, bounding-boxes around the objects and confidence scores. We present this output by plotting the masks on the images (as shown in Figure 4.3) and from this segmentation masks and the ground-truth labels determine whether a given predicted mask is a TP, FP or FN.

Mask R-ConvNet, in some cases, yielded multiple detections for a single fruit. To merge these detections, we consider the detection with IoU greater than the threshold as TP and disregard the other detections. If all the detections have IoU less than the threshold, we report one FP detection and ignore the other detections.

The output of mask R-ConvNet also suggests that the model could not differentiate between fruits and background due to too much occlusion and visual complexity of the tree canopy. Mask R-ConvNet model could not detect all the fruits in a given image and there were false positive detections, false negative detections and missed fruits. Some leaves have the same features (e.g. colour and shape) as the fruits, and therefore, the model marked them as fruits - false positive cases. Similarly, some fruit surfaces are highly similar to the colour of the leaves. Due to these similarities, the model marked these fruits as non-fruits - missed cases. By inspection, green fruits matched the green leaves, causing the misclassification.

Another problem affecting the performance of the model is the illumination change. The images used in this study are captured under natural outdoor conditions making it prone to visual complexities. These complexities negatively affected the performance of mask R-ConvNet.

Remark. We will consider two IoU thresholds throughout the study, that is, $\alpha = 0.3$ and $\alpha = 0.5$.

The results of mask R-ConvNet on 42 images are as follows:

4.3.1 IoU threshold=0.3.

	Fruit count	$\text{TP}_{0.3}$	$\text{FP}_{0.3}$	$\text{Missed}_{0.3}$
Number	4313	2954	1959	970
Percent	-	68.49	39.87	22.49

Table 4.1: Results of mask R-ConvNet on the whole dataset of 42 images. TP(%)=$\frac{2954}{4313} \times 100 = 68.49$, FP(%)=$\frac{1959}{4913} \times 100 = 39.87$, Missed(%)=$\frac{970}{4313} \times 100 = 22.49$

Remark. $\text{TP}_\alpha(\%)$ and $\text{Missed}_\alpha(\%)$ are calculated over the fruit count while $\text{FP}_\alpha(\%)$ is calculated over the number of predictions ($\text{TP}_\alpha + \text{FP}_\alpha$).

For the whole dataset of 42 images with IoU threshold of 0.3, 68.49% of actual fruits were successfully detected. The false positive detections were 39.87%. 22.49% of the actual fruits were missed by the model (see Table 4.1).

4.3.2 IoU threshold=0.5.

	Fruit count	$TP_{0.5}$	$FP_{0.5}$	$Missed_{0.5}$
Number	4313	2411	2491	970
Percent	-	55.90	50.82	22.49

Table 4.2: Results of mask R-ConvNet on the whole dataset of 42 images. $TP_{0.5}$(%)=$\frac{2411}{4313} \times 100 = 55.90$, $FP_{0.5}$(%)=$\frac{2491}{4902} \times 100 = 50.82$, $Missed_{0.5}$(%)=$\frac{970}{4313} \times 100 = 22.49$

When we consider an IoU threshold of 0.5, 55.90% of actual fruits were successfully detected. The false positive detections were 50.82%, and 22.49% of the actual fruits were missed by the model, as shown in Table 4.2.

4.4 Results of Mask R-ConvNet + RbRG

When implementing RbRG, we need to define some hyperparameters. For our experiments, we define different thresholds as follows. $\eta_0 = 0.5$ is the optimal threshold for mask R-ConvNet, $\eta_B = 0.10$ is the threshold for high confidence background, and $\eta_F = 0.90$ is the threshold for high confidence foreground. We choose $m = 10$ as the compactness factor because we want to be strict on a compact boundary adherence.

The results of mask R-ConvNet + RbRG for different thresholds are given below:

4.4.1 IoU threshold=0.3.

	Fruit count	$TP_{0.3}$	$FP_{0.3}$	$Missed_{0.3}$
Number	4313	3104	1000	1229
Percent	-	71.97	24.37	28.50

Table 4.3: Results of mask R-ConvNet + RbRG on the whole dataset of 42 images.TP(%)=$\frac{3104}{4313} \times 100 = 71.97$, FP(%)=$\frac{1000}{4104} \times 100 = 24.37$, Missed(%)=$\frac{1229}{4313} \times 100 = 28.50$

For the whole dataset of 42 images and an IoU threshold of 0.3, RbRG correctly detected 71.97% of the actual fruits. 24.37 were the false positives and 28.5% as the actual fruits that were missed by the algorithm (see Table 4.3).

4.4.2 IoU threshold=0.5.

	Fruit count	$TP_{0.5}$	$FP_{0.5}$	$Missed_{0.5}$
Number	4313	2646	1388	1229
Percent	-	61.35	34.41	28.50

Table 4.4: Results of mask R-ConvNet + RbRG on the whole dataset of 42 images.TP(%)=$\frac{2646}{4313} \times 100 = 61.35$, FP(%)=$\frac{1388}{4034} \times 100 = 34.41$, Missed(%)=$\frac{1229}{4313} \times 100 = 28.50$

If we consider a threshold of IoU=0.5, RbRG correctly detected 61.35% of the actual fruits. 34.41% were the false positive detections and 28.5% as the actual fruits that were missed by the algorithm (see Table 4.4).

False positive detections and missed fruits are attributed to some factors - some intrinsic to the model and some extrinsic. Since RbRG is based on region growing, it is logical that it is more efficient for objects with some level of similarity in appearance. This makes RbRG vulnerable to classification errors in cases where the fruits and the leaves are of nearly the same colours.

RbRG algorithm is also affected by illumination change, just like mask R-ConvNet. This is because RbRG takes as input RGB image without any preprocessing. The hyperparameters used in the RbRG also contributes to false detections and missed fruits. This is because parameters, like η_F and η_B, form the basis for a trade-off between false positives and false negative detections respectively.

Unlike mask R-ConvNet, RbRG is not affected by the classification errors relating to multiple detections.

4.5 Comparing Results of Mask R-ConvNet and Mask R-ConvNet + RbRG

We also compared the performance of mask R-ConvNet and mask R-ConvNet + RbRG per image by computing the precision and recall for different thresholds, as shown in Tables 4.6 and 4.5. To establish the balance between precision and recall, we calculated the F_1 score per image and made the comparison based on this score. It is evident that RbRG indeed improves the outcome of segmentation for almost all the images.

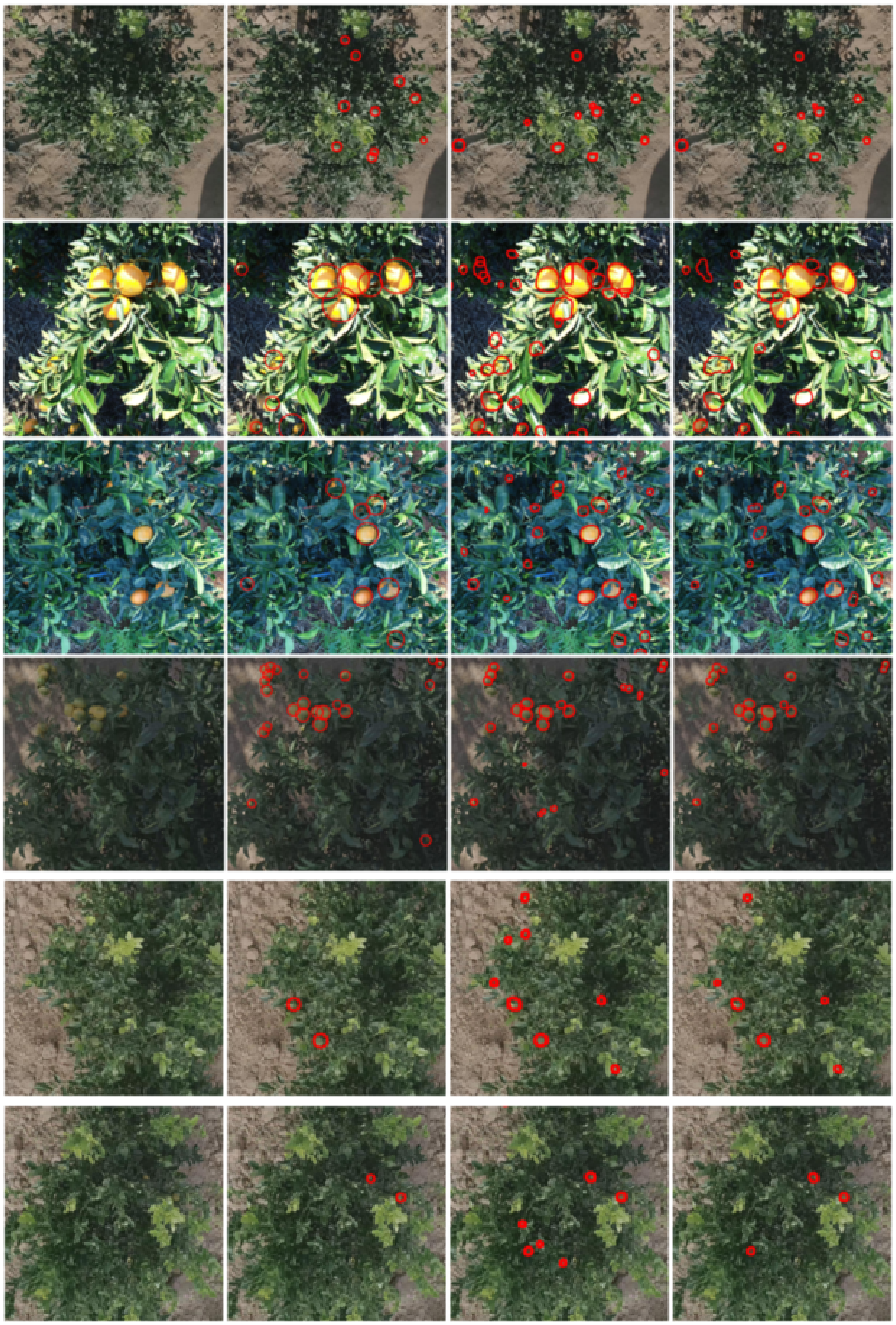


Figure 4.3: **Best viewed when coloured.** Results on cropped images. From left to right: (a) RGB images, (b) ground-truth masks, (c) mask R-ConvNet output and (d) mask R-ConvNet+RbRG output.

		mask R-ConvNet						**mask R-ConvNet + RbRG**					
Image	GT	TP	FP	Missed	Prec	Rec	F1	TP	FP	Missed	Prec	Rec	F1
1	49	41	83	0	0.33	0.84	0.47	42	44	1	0.49	0.86	0.62
2	114	62	32	36	0.66	0.54	0.60	75	14	40	0.84	0.66	0.74
3	262	103	17	129	0.86	0.39	0.54	106	12	157	0.90	0.41	0.56
4	238	131	15	79	0.90	0.55	0.68	144	8	95	0.95	0.61	0.74
5	296	149	27	115	0.85	0.50	0.63	159	15	135	0.91	0.54	0.68
6	65	37	40	26	0.48	0.57	0.52	38	31	25	0.55	0.59	0.57
7	140	125	122	13	0.51	0.89	0.65	119	88	21	0.58	0.85	0.69
8	15	8	16	7	0.33	0.53	0.41	9	13	6	0.41	0.60	0.49
9	59	25	25	30	0.50	0.42	0.46	25	13	34	0.66	0.42	0.52
10	11	6	7	5	0.46	0.55	0.50	5	4	6	0.56	0.46	0.50
11	281	211	274	55	0.44	0.75	0.55	184	125	93	0.60	0.66	0.62
12	75	45	23	28	0.66	0.60	0.63	43	28	32	0.61	0.57	0.59
13	203	131	227	43	0.37	0.65	0.47	117	131	78	0.47	0.58	0.52
14	18	9	20	9	0.31	0.50	0.38	7	10	11	0.41	0.40	0.40
15	35	8	11	27	0.42	0.23	0.30	8	6	27	0.57	0.23	0.33
16	26	16	27	9	0.37	0.62	0.46	14	17	11	0.45	0.54	0.49
17	52	35	15	11	0.70	0.67	0.69	42	2	13	0.96	0.81	0.88
18	112	69	10	29	0.87	0.62	0.72	66	4	45	0.94	0.60	0.73
19	7	7	2	0	0.78	1.00	0.88	7	0	0	1.00	1.00	1.00
20	47	30	35	14	0.46	0.64	0.54	24	23	20	0.51	0.51	0.51
21	173	78	17	79	0.82	0.45	0.58	79	8	93	0.91	0.46	0.61
22	276	137	11	111	0.93	0.50	0.65	130	7	146	0.95	0.47	0.63
23	12	12	11	0	0.52	1.00	0.69	12	4	0	0.75	1.00	0.86
24	176	107	99	57	0.52	0.61	0.56	112	44	64	0.72	0.64	0.68
25	24	19	14	2	0.58	0.79	0.67	20	6	4	0.77	0.83	0.80
26	14	7	3	6	0.70	0.50	0.58	10	1	4	0.91	0.71	0.80
27	5	5	19	0	0.21	1.00	0.34	5	11	0	0.31	1.00	0.48
28	10	8	3	1	0.73	0.80	0.76	7	0	3	1.00	0.70	0.82
29	8	7	17	1	0.29	0.88	0.44	7	10	1	0.41	0.88	0.56
30	10	7	13	3	0.35	0.70	0.47	6	8	4	0.43	0.60	0.50
31	183	160	48	9	0.77	0.87	0.82	176	17	10	0.91	0.96	0.94
32	10	6	6	3	0.50	0.60	0.55	7	3	3	0.70	0.70	0.70
33	112	106	39	1	0.73	0.95	0.83	114	11	1	0.91	0.99	0.95
34	32	28	59	0	0.32	0.88	0.47	29	41	0	0.41	0.91	0.57
35	247	202	54	11	0.79	0.82	0.80	242	19	16	0.93	0.98	0.95
36	6	4	7	2	0.36	0.67	0.47	4	4	2	0.50	0.67	0.57
37	173	159	59	6	0.73	0.92	0.81	169	28	7	0.86	0.98	0.91
38	163	144	113	2	0.56	0.88	0.69	167	51	4	0.77	0.98	0.86
39	107	97	65	0	0.60	0.91	0.72	112	36	0	0.76	1.00	0.86
40	210	195	92	3	0.68	0.93	0.79	214	30	6	0.88	0.95	0.91
41	80	67	126	1	0.35	0.84	0.49	69	56	0	0.55	0.86	0.67
42	177	151	56	7	0.73	0.85	0.79	179	17	9	0.91	0.98	0.95

Table 4.5: **Best Viewed when coloured:**Table of detailed results for mask R-ConvNet and mask R-ConvNet + RbRG. IoU=0.3. Performance indicators based on F_1 score: blue text colour for improvement and red otherwise. GT represents actual fruit count and F1 is the F_1 score. Prec stands for precision and Rec stands for recall.

		mask R-ConvNet						**mask R-ConvNet + RbRG**					
Image	GT	TP	FP	Missed	Prec	Rec	F1	TP	FP	Missed	Prec	Rec	F1
1	49	24	101	0	0.19	0.49	0.28	27	56	1	0.33	0.55	0.41
2	114	57	35	36	0.62	0.50	0.55	74	15	40	0.83	0.65	0.73
3	262	88	27	129	0.77	0.34	0.47	105	13	15	0.89	0.40	0.55
4	238	112	28	79	0.80	0.47	0.59	139	13	95	0.91	0.58	0.71
5	296	137	34	115	0.80	0.46	0.59	154	18	135	0.90	0.52	0.66
6	65	30	47	26	0.39	0.46	0.42	30	39	25	0.44	0.46	0.45
7	140	115	132	13	0.47	0.82	0.60	101	103	21	0.50	0.72	0.59
8	15	7	17	7	0.29	0.47	0.36	9	13	6	0.41	0.60	0.49
9	59	23	27	30	0.46	0.39	0.42	22	15	34	0.60	0.37	0.46
10	11	2	10	5	0.17	0.18	0.17	4	5	6	0.44	0.36	0.40
11	281	160	324	55	0.33	0.57	0.42	157	147	93	0.52	0.56	0.54
12	75	44	24	28	0.65	0.59	0.62	39	32	32	0.55	0.52	0.53
13	203	84	274	43	0.24	0.41	0.30	90	153	78	0.37	0.44	0.40
14	18	8	21	9	0.28	0.44	0.34	7	10	11	0.41	0.39	0.40
15	35	6	13	27	0.32	0.17	0.22	8	6	27	0.57	0.22	0.33
16	26	14	29	9	0.33	0.54	0.41	12	19	11	0.39	0.46	0.42
17	52	30	19	11	0.61	0.58	0.59	37	7	13	0.84	0.71	0.77
18	112	54	21	29	0.72	0.48	0.58	62	8	45	0.89	0.55	0.68
19	7	7	3	0	0.70	1.00	0.82	7	0	0	1.00	1.00	1.00
20	47	20	44	14	0.31	0.43	0.36	13	34	20	0.28	0.28	0.27
21	173	71	24	79	0.75	0.41	0.53	57	27	93	0.68	0.33	0.44
22	276	123	23	111	0.84	0.45	0.58	123	12	146	0.91	0.47	0.60
23	12	7	16	0	0.30	0.58	0.40	8	8	0	0.50	0.67	0.57
24	176	102	104	57	0.50	0.58	0.53	105	51	64	0.67	0.60	0.63
25	24	18	15	2	0.55	0.75	0.63	18	8	4	0.69	0.75	0.72
26	14	7	3	6	0.70	0.50	0.58	9	2	4	0.82	0.64	0.72
27	5	3	21	0	0.13	0.60	0.21	2	14	0	0.13	0.4	0.19
28	10	5	6	1	0.46	0.50	0.48	5	2	3	0.71	0.50	0.59
29	8	7	17	1	0.29	0.88	0.44	7	10	1	0.41	0.78	0.56
30	10	5	15	3	0.25	0.50	0.33	4	10	4	0.29	0.40	0.33
31	183	141	67	9	0.68	0.77	0.72	171	22	10	0.89	0.93	0.91
32	10	3	9	3	0.25	0.30	0.27	2	7	3	0.22	0.20	0.21
33	112	74	73	1	0.50	0.66	0.57	77	45	1	0.63	0.69	0.67
34	32	11	76	0	0.13	0.34	0.18	19	49	0	0.28	0.59	0.38
35	247	168	87	11	0.66	0.68	0.67	215	42	16	0.84	0.87	0.85
36	6	4	7	2	0.36	0.67	0.47	4	4	2	0.50	0.67	0.57
37	173	123	96	6	0.56	0.71	0.63	147	46	7	0.76	0.85	0.80
38	163	97	160	2	0.38	0.60	0.46	117	89	4	0.57	0.72	0.64
39	107	78	85	0	0.48	0.73	0.58	86	56	0	0.61	0.80	0.69
40	210	176	116	3	0.60	0.84	0.70	180	59	6	0.75	0.86	0.80
41	80	36	159	1	0.19	0.45	0.26	40	82	0	0.33	0.50	0.40
42	177	130	82	7	0.61	0.73	0.67	153	37	9	0.81	0.86	0.83

Table 4.6: **Best Viewed when coloured:**Table of detailed results for mask R-ConvNet and mask R-ConvNet + RbRG. IoU=0.5. Performance indicators based on F_1 score: blue text colour for improvement and red otherwise. GT represents actual fruit count and F1 is the F_1 score. Prec stands for precision and Rec stands for recall

4.6 Conclusion

4.6.1 IoU threshold=0.3.

From the data in tables 4.1 and 4.3 we can generate the following summary table describing the performance of mask R-ConvNet and mask R-ConvNet + RbRG

	TP$_{0.3}$(%)	FP$_{0.3}$(%)	Missed$_{0.3}$(%)
mask R-ConvNet	68.49	39.87	22.49
mask R-ConvNet + RbRG	71.97	24.37	28.50
Deviation	+3.48	-15.50	+6.01

Table 4.7: Comparison between the results of mask R-ConvNet and mask R-ConvNet + RbRG with IoU threshold of 0.3.

RbRG indeed improves the performance of mask R-ConvNet model. From the data in Table 4.7, upon applying RbRG algorithm on the output of mask R-ConvNet, the percentage of correct positive detections rose by 3.48%, and the false positive detections dropped by 15.5%. The number of missed fruits, however, increased by 6.01%.

4.6.2 IoU threshold=0.5.

At a threshold of 0.5, mask R-ConvNet + RbRG is still better than mask R-ConvNet as shown in the Table below

	TP$_{0.5}$(%)	FP$_{0.5}$(%)	Missed$_{0.5}$(%)
mask R-ConvNet	55.90	50.82	22.49
mask R-ConvNet + RbRG	61.34	34.41	28.50
Deviation	+5.44	-16.41	+6.01

Table 4.8: Comparison between the results of mask R-ConvNet and mask R-ConvNet + RbRG with IoU threshold of 0.5.

The true positive detections increased by 5.44%, false positives dropped by 16.41% and the number of fruits missed by the model increased by 6.01%.

The increase in the number of missed fruits is attributed to the intrinsic limitations of the RbRG and the fact that the RbRG takes it as input the output of mask R-ConvNet which themselves contains classification errors. The inherent limitation of the model is that RbRG algorithm cannot detect a fruit that was missed by the base classifier unless we choose η_F below the threshold of the base model, η_0. The latter choice will result in more false positive detections, and therefore, we did not pursue it in this study.

5. Conclusion and Future Work

In this Chapter, we state some concluding remarks about the findings in our research project. This Chapter is divided into three sections namely Section 5.1 where we summarize our findings, Section 5.2 presents the limitations of RbRG algorithm and we finally propose the solutions for the said limitations in Section 5.3 as part of future studies.

5.1 Conclusion

In this dissertation, we presented RbRG, an approach, used to refine the output of any ConvNet. RbRG begins by thresholding the image into regions based on confidence scores and performs region growing to relabel pixels at the uncertainty region. Specifically, we applied RbRG (see Chapter 3) to refine the output of a mask R-ConvNet (see Chapter 2).

We quantified the performance of mask R-ConvNet and mask R-ConvNet + RbRG using the components of the confusion matrix, namely True Positive (TP), False Negative (FN) and False Positive (FP). These components are defined based on some threshold established by Intersection over Union (IoU) metric. After evaluating the performance of the base model (mask R-ConvNet) and the refinement module (RbRG), we conclude that indeed, the RbRG algorithm improved the performance of the model (see Section 4.6).

The summary of the results of mask R-ConvNet for different IoU thresholds were presented in Tables 4.1 and 4.2 and that of mask R-ConvNet + RbRG were shown in Tables 4.3 and 4.4. The comparison between mask R-ConvNet and mask R-ConvNet + RbRG

We also provided detailed results in tables 4.5 and 4.6 for different IoU thresholds.

5.2 Limitations of RbRG

While RbRG algorithm indeed improved the performance of the mask R-ConvNet model, the algorithm has some limitations in its performance. The notable drawbacks include:

- With the current set-up, RbRG takes as input RGB images obtained under natural outdoor conditions without any preprocessing. The illumination problem caused by the sunlight affects the performance of RbRG because the core step of the algorithm implementation depends on the colour of the image.
- For fruits close to each other or the fruits occluding each other, the algorithm identified these fruits as one fruit because RbRG operates in a semantic sense.
- The fruits that were missed by the base model are highly unlikely to be detected by RbRG. This is based on the choices of parameters in RbRG, specifically, high confidence object probability (R_F) and high confidence background probability (R_B) (see Section 3.2).
- The performance of RbRG is contingent to the accuracy of the base classifier. Errors (specifically false positive detections) from the base detector may be propagated into the RbRG algorithm if the base model falsely classified background pixels as object with high confidence, that is, given a high confidence of a false positive detection generated by the initial classifier, RbRG is unable to correct such mistakes and therefore these false positives are retained on the refined output.

5.3 Possible Areas for Future Work

We believe that finding the solutions for the said limitations will make RbRG even a better refinement module. To solve the illumination problem, we suggest preprocessing of the images before passing it into the algorithm. Subjecting the images into a logarithm transform and histogram equalization might be possible ways to solve the illumination problem. RbRG should also be built in such a way that it can perform instance segmentation instead of semantic segmentation. This goes a long way in improving its performance especially in cases where the number of objects in an image is of prime concern. Finally, we hypothesize that the base model should be trained with RbRG as an additional step to training before computing the training losses. By doing so, RbRG gets a chance to correct errors related to colour confusion which might have lowered the accuracy of the base model trough false negative detections. It will be a good idea also to compare results of RbRG and other post-processing methods, for example, Bayesian approach (Davis et al., 2018), coarse-to-fine semantic segmentation (Jing et al., 2019) and LabelBank (Hu et al., 2017).

Acknowledgements

I want to acknowledge AIMS and it's funders for supporting me through this master's program, as well as my supervisor, Dr. Bubacarr Bah from the University of Stellenbosch and AIMS South Africa. A special thank you also to my co-supervisors, Munsanje Mweene and Stuart Reid, from Aerobotics, Cape Town, for the support they provided during the research. I also express my gratitude to my tutor, Alice Nanyanzi for her support during this research.

A big thank you to my family also for giving me support all along. A special mention of my mother, Jane C. Koskey; my cousin , Simon K. Korir; Saumu A. Athman; my brother, Dancun K. Maritim and my nieces Angela and Faith.

References

Achanta, R. and Susstrunk, S. Superpixels and polygons using simple non-iterative clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4651–4660, 2017.

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

Adams, R. and Bischof, L. Seeded region growing. *IEEE Transactions on pattern analysis and machine intelligence*, 16(6):641–647, 1994.

Boskovitz, V. and Guterman, H. An adaptive neuro-fuzzy system for automatic image segmentation and edge detection. *IEEE Transactions on fuzzy systems*, 10(2):247–262, 2002.

Davis, J. W., Menart, C., Akbar, M., and Ilin, R. A classification refinement strategy for semantic segmentation. *ArXiv*, abs/1801.07674, 2018.

Dias, P. A. and Medeiros, H. Semantic segmentation refinement by Monte Carlo region growing of high confidence detections. In *Asian Conference on Computer Vision*, pages 131–146. Springer, 2018.

Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1): 98–136, 2015.

Fawcett, T. An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

Girshick, R. Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

Hu, H., Deng, Z., Zhou, G.-T., Sha, F., and Mori, G. LabelBank: Revisiting global perspectives for semantic segmentation. *arXiv preprint arXiv:1703.09891*, 2017.

Hurtik, P. and Madrid, N. Bilinear interpolation over fuzzified images: enlargement. In *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8. IEEE, 2015.

Jing, L., Chen, Y., and Tian, Y. Coarse-to-fine semantic segmentation from image-level labels. *IEEE Transactions on Image Processing*, 29:225–236, 2019.

Kamdi, S. and Krishna, R. Image segmentation and region growing algorithm. *International Journal of Computer Technology and Electronics Engineering (IJCTEE)*, 2(1), 2012.

Khan, S., Rahmani, H., Shah, S., and Bennamoun, M. *A Guide to Convolutional Neural Networks for Computer Vision*, volume 8. Morgan & Claypool, 02 2018. doi: 10.2200/S00822ED1V01Y201712COV015.

Li, Y., Wang, X., and Xu, P. Chinese text classification model based on deep learning. *Future Internet*, 10(11):113, 2018.

Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

Ren, S., He, K., Girshick, R., and Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

Sande, K., Uijlings, J., Gevers, T., and Smeulders, A. Segmentation as selective search for object recognition. *Pediatric Critical Care Medicine - PEDIATR CRIT CARE MED*, pages 1879–1886, 11 2011. doi: 10.1109/ICCV.2011.6126456.