PART B:

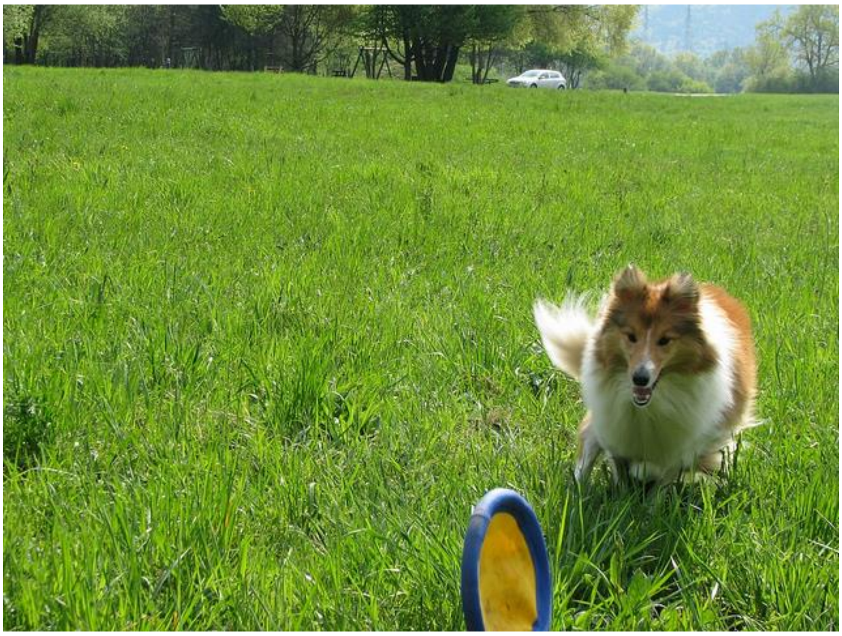
Preface:

First we need to generate the ROI/Thumbnail using the *generate _roi(input img, salmap, thresh, alpha, beta)* function.

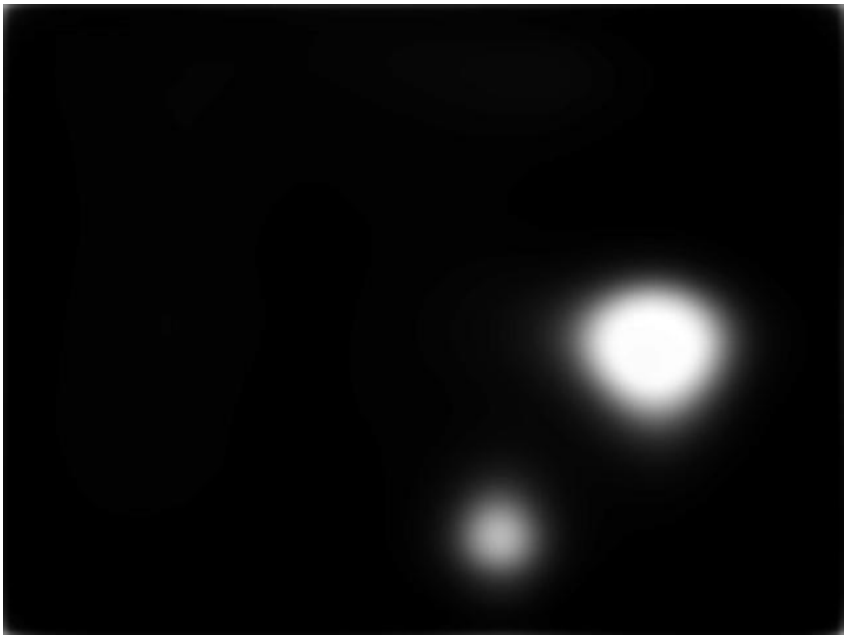
```
def generate_roi(salmap, thresh, alpha, beta):

    image=salmap
    # height, width, number of channels in image
    height = image.shape[0]
    width = image.shape[1]
    aspect=width/height
    threshMap = cv2.threshold(image, int(2.55*thresh), 255,cv2.THRESH_BINARY)[1]

    # find contours and get the external one
    contours, hier = cv2.findContours(threshMap, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    roi=[]
    for c in contours:
        # get the bounding rect
        x, y, w, h = cv2.boundingRect(c)
        if w>15 and h>15:
            roi.append([x, y, w, h])
            # cv2.rectangle(image_orig, (x, y), (x+w, y+h), (0, 0, 255), 2)

    # print(roi)
    for r in roi:
        h_opti=round((r[2]/aspect))
        w_opti=round(r[2])
        # draw a red rectangle to visualize the bounding rect
        cv2.rectangle(image_orig, (x, y), (x+w, y+h), (0, 0, 255), 2)
        x1=r[0]
        y1=int(r[1]/4)

        Rs=0
        roi_final=[]
        for j in range(r[3]-h_opti):
            dr=alpha*np.sum(threshMap[r[1]+j:r[1]+j+h_opti, r[0]:r[0]+r[3]])-beta*(h_opti*r[3])
            if dr>Rs:
                Rs=dr
                roi_final=[r[0],r[1]+j,r[3],h_opti]

    return roi_final
```

Examle:

Input Image:



SalMap:



Calculation of Aspect Ratio:

```
height = image.shape[0]
width = image.shape[1]
aspect=width/height
```

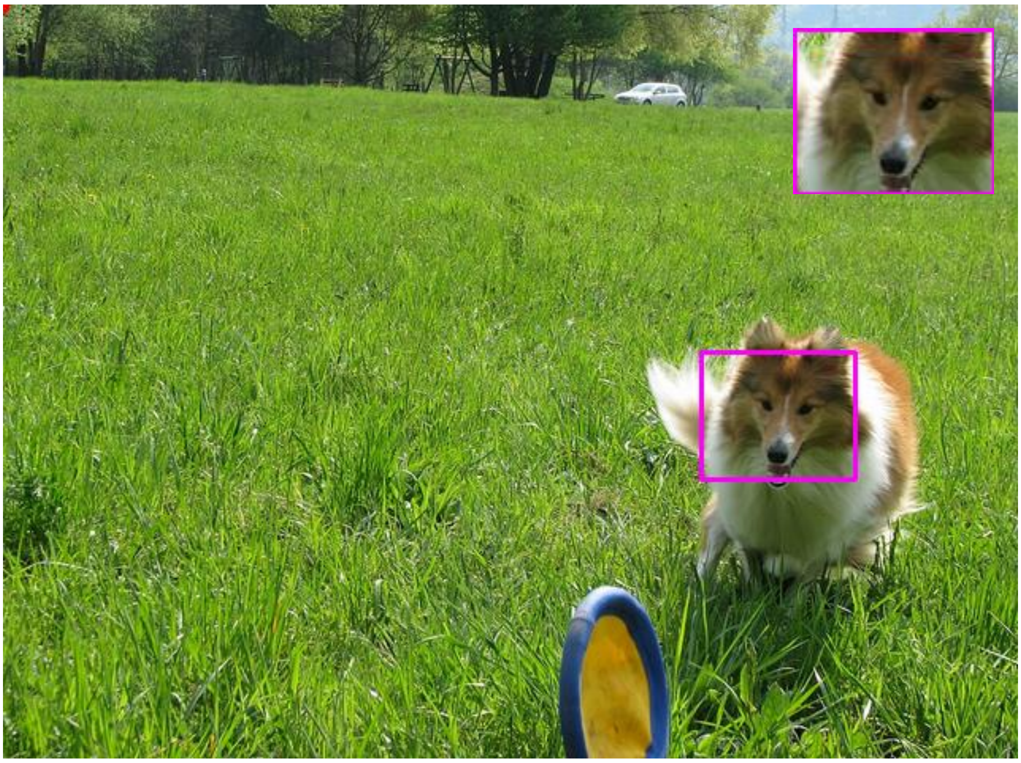
ROI Score:

$$R_s = \alpha \left(\sum_{P \in \mathbf{R}} S_P P \right) - \beta(N)$$

equivalent code:

```
Rs=0
roi_final=[]
for j in range(r[3]-h_opti):
    dr=alpha*np.sum(threshMap[r[1]+j:r[1]+j+h_opti, r[0]:r[0]+r[3]])-beta*(h_opti*r[3])
    if dr>Rs:
        Rs=dr
        roi_final=[r[0],r[1]+j,r[3],h_opti]
```

OUTPUT ROI (With Max Rs)



PARAMETERS:

– α = 0.8, β = 0.3
– α = 1.0, β = 0.1
– α = 0.6, β = 0.4

Precision/Recall Equations:

precision=TP/(TP+FN)
recall=TP/(TP+FP)

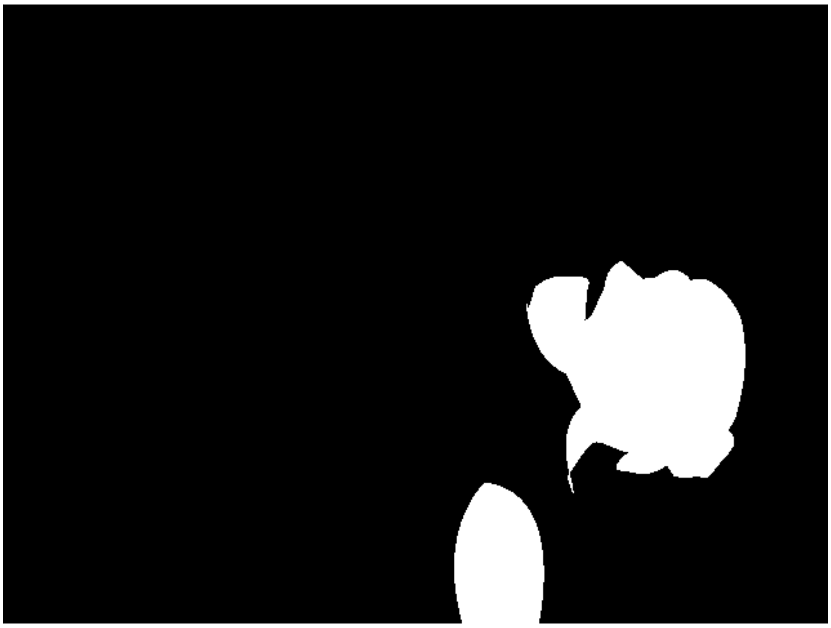
where:

TP=True Positive
FN=False Negative
FP=False Positive

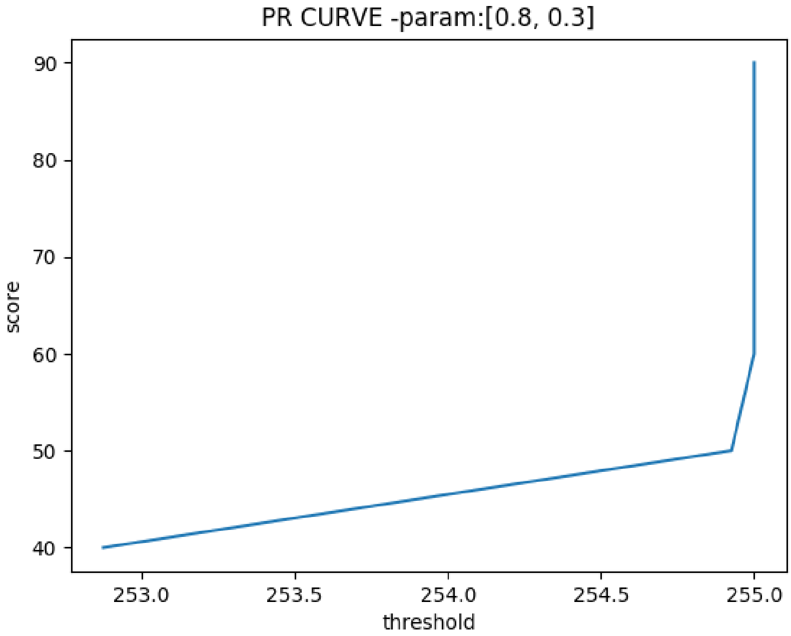
Corresponding PR Curve Code for SALGAN:

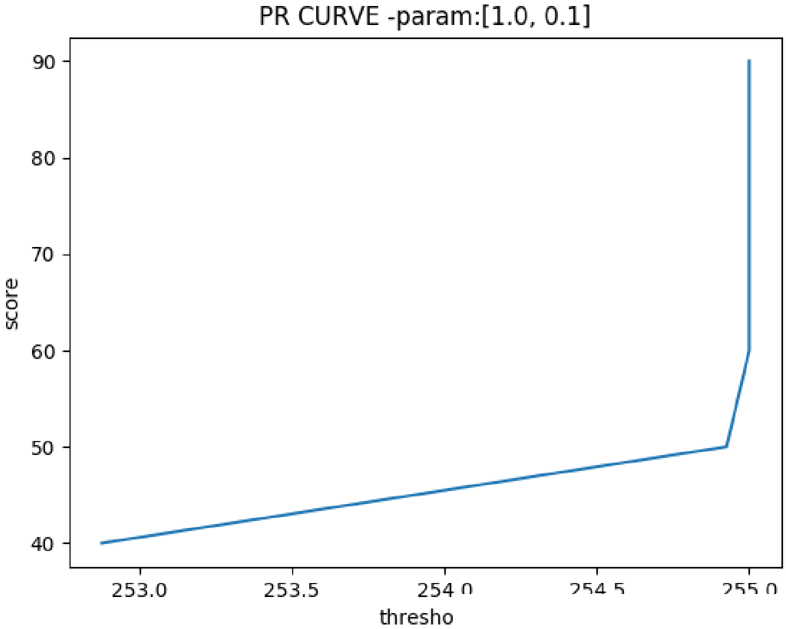
```
def mask_score_roi(mask, roi):

    temp=mask[roi[1]:roi[1]+roi[3],roi[0]:roi[0]+roi[2]]
    TP = np.sum(temp == 255)

    h = mask.shape[0]
    w = mask.shape[1]
    FN = np.sum(temp == 255) -TP

    contours, hier = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    area=0
    for c in contours:
        # get the bounding rect
        x, y, w, h = cv2.boundingRect(c)
        if cv2.contourArea(c)>area:
            r=(x, y, w, h)
        area=cv2.contourArea(c)

    temp=mask[r[1]:r[1]+r[3],r[0]:r[0]+r[2]]
    FP=abs(roi[2]-r[2]) * abs(roi[3]-r[3])

    precision=TP/(TP+FN)
    recall=TP/(TP+FP)

    return precision,recall

def salience_score_roi(map, roi):
    temp=map[roi[1]:roi[1]+roi[3],roi[0]:roi[0]+roi[2]]
    score=np.mean(temp)
    tot=np.sum(temp>0)
    missed=tot-score

    return score,missed
```

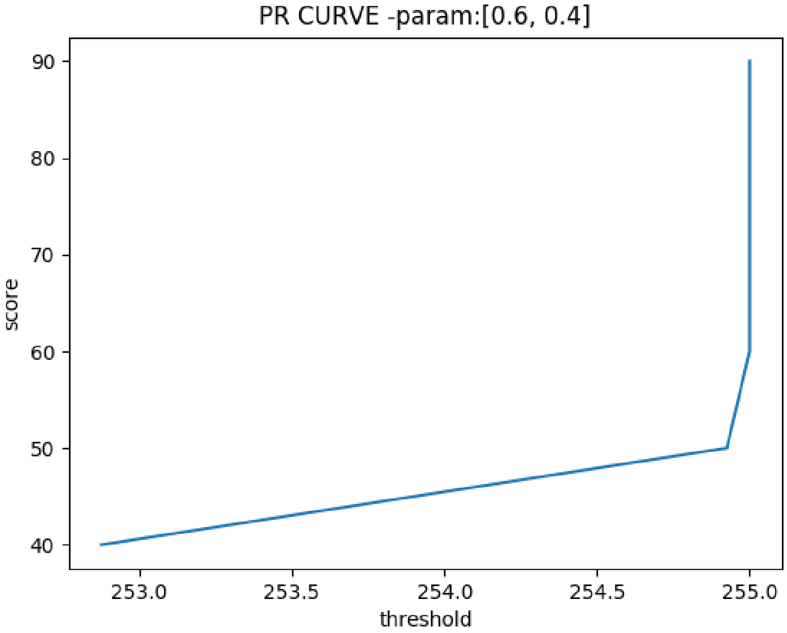
Ground Truth:



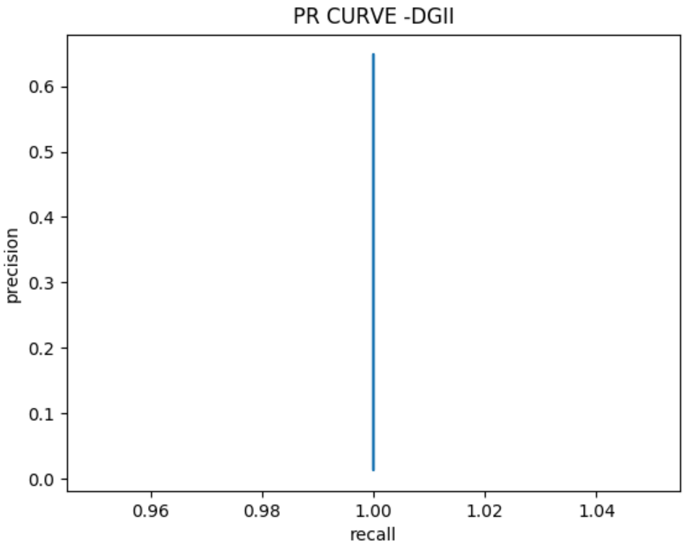
PR Curves (AIM/BMS):

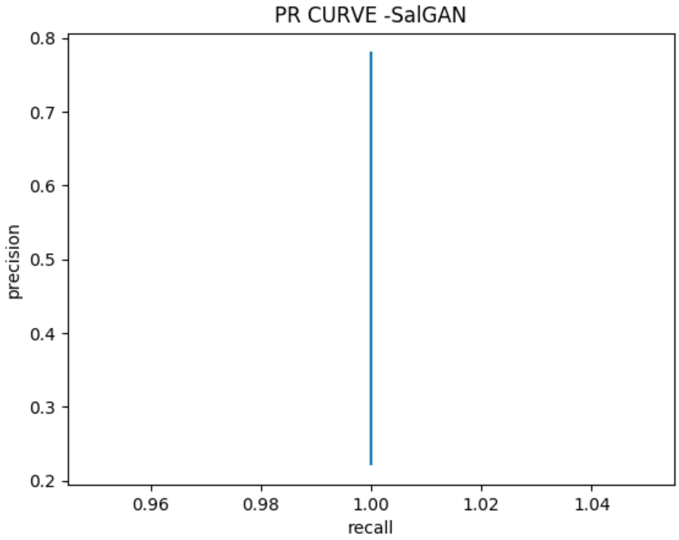



PR CURVE -param:[1.0, 0.1]
score
90
80
70
60
50
40
253.0
253.5
254.0
254.5
255.0
thresho


PR CURVE -param:[0.6, 0.4]
score
90
80
70
60
50
40
253.0
253.5
254.0
254.5
255.0
threshold

PR Curves (DGII/SALGAN ---- for all parameters):





It is clearly seen, the AUC for both DGII and SalGan is the maximum.