Spring 2022
Department of Computer Science
Bahria University, Karachi Campus

INTRODUCTION TO DATA SCIENCE LAB [CSL-487]

Project Name: Image Segmentation

SEMESTER PROJECT

Maximum Marks: 30

Submission Due Date: 7th July, 2022

Sr.no	Name	Enrolment	Semester
01	Abdullah Abdul Wahid	02-134192-015	6-B
02	Fazeel Zafar	02-134192-010	6-B
03	Zoha Zehra	02-134192-058	6-B

Name	Designation
Dr. Kashif Hussain	Course Instructor
Ms. Salas Akbar	Lab Engineer

Acknowledgement

I would like to express my special thanks of gratitude to my professor, Dr. Kashif Hussain as well as our lab instructor, Ms. Salas Akbar, who gave me the golden opportunity to do this wonderful project on the topic **Image Masking**, which also helped me in doing a lot of Research and I came to know about so many new things, I am really thankful to them.

Contents

1. Chapter 1

1.1. Problem Statement

Image segmentation is a technique that divides a digital image into various subgroups known as Image segments, which serves to simplify future processing or analysis of the image by decreasing the complexity of the original image. Image segmentation can be done through two approaches, Similarity Approach and Discontinuity Approach.

The technique for image segmentation is Faster R-CNN. Faster R-CNN is a deep convolutional network used for object detection that appears to the user as a single, end-to-end, unified network. The network can accurately and quickly predict the locations of different objects.

2. Chapter 2

2.1. Literature Review

Object detection based on RGC mask R-CNN

It was published on 13th May 2020, by a group of researchers under the program *"National Natural Science Foundation of China"*. Object detection is frequently used in intelligent surveillance, autonomous driving, and surgical tool positioning, among other applications. Item detection seeks to extract categorization and position information about a specific object from complex scenarios, which may subsequently be utilised for more advanced tasks like object tracking. Furthermore, not only must object categorization and location be established concurrently in object detection, but the amount and size of objects must also be determined. Object identification, as a result, remains a difficult topic in computer vision research.

https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/iet-ipr.2019.0057

ABSTRA

Instance segmentation for real-time video detection using FPN and Mask R-CNN

It was published on 4th April 2022, by *Anu Yadav and 2Dr. Ela Kumar*. Object instance segmentation is an important step in real-time video detection. Detecting the information of all sorts of items in an image by marking the discovered object's position in the picture with a rectangular box is known as object detection. Deep learning achieves a breakthrough in object detection research by utilising its powerful feature learning capabilities. Many researchers have employed various machine learning techniques to accomplish object recognition and have concentrated on improving feature extraction accuracy.

https://assets.researchsquare.com/files/rs-1477072/v1_covered.pdf?c=1649083287

Wu, Xin & Wen, Shiguang & Xie, Yuan-Ai. (2019). Improvement of Mask-RCNN Object Segmentation Algorithm. 10.1007/978-3-030-27526-6_51.

Semantic maps play a key role in tasks such as navigation of mobile robots. However, the visual SLAM algorithm based on multi-objective geometry does not make full use of the rich semantic information in space. The map point information retained in the map is just a spatial geometric point without semantics. Since the algorithm based on convolutional neural network has achieved breakthroughs in the field of target detection, the target segmentation algorithm MASK-RCNN is combined with the SLAM algorithm to construct the semantic map. However, the MASK-RCNN algorithm easily treats part of the background in the image as foreground, which results in inaccuracy of target segmentation. Moreover, Grubcut segmentation algorithm is time-consuming, but it’s easy to take foreground as background, which leads to the excessive edge segmentation. Based on these, our paper proposes a novel algorithm which combines MASK-RCNN and Grubcut segmentation. By comparing the experimental results of MASK-Rcnn, Grubcut and the improved algorithm on the data set, it is obvious that the improved algorithm has the best segmentation effect and the accuracy of image target segmentation is significantly improved. These phenomenons demonstrate the effectiveness our proposed algorithm.. Wu, Xin & Wen, Shiguang & Xie, Yuan-Ai. (2019). Improvement of Mask-RCNN Object Segmentation Algorithm. 10.1007/978-3-030-27526-6_51. Semantic maps play a key role in tasks such as navigation of mobile robots. However, the visual SLAM algorithm based on multi-objective geometry does not make full use of the rich semantic information in space. The map point information retained in the map is just a spatial geometric point without semantics. Since the algorithm based on convolutional neural network has achieved breakthroughs in the field of target detection, the target segmentation algorithm MASK-RCNN is combined with the SLAM algorithm to construct the semantic map. However, the MASK-RCNN algorithm easily treats part of the background in the image as foreground, which results in inaccuracy of target segmentation. Moreover, Grubcut segmentation algorithm is time-consuming, but it's easy to take foreground as background, which leads to the excessive edge segmentation. Based on these, our paper proposes a novel algorithm which combines MASK-RCNN and Grubcut segmentation. By comparing the experimental results of MASK-Rcnn, Grubcut and the improved algorithm on the data set, it is obvious that the improved algorithm has the best segmentation effect and the accuracy of image target segmentation is significantly improved. These phenomenons demonstrate the effectiveness our proposed algorithm.

https://www.researchgate.net/publication/334850808_Improvement_of_Mask-RCNN_Object_Segmentation_Algorithm

3. Chapter 3

3.1. Methodology

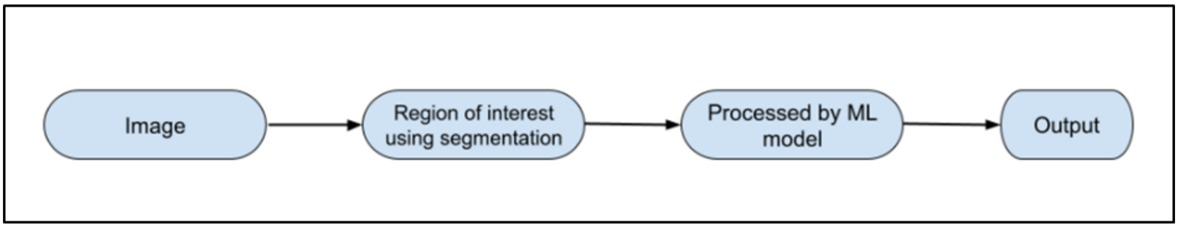
There are two approaches in image segmentation:

- **Similarity Approach:** This approach is based on detecting similarity between image pixels to form a segment, based on a threshold. ML algorithms like clustering are based on this type of approach to segment an image.

- **Discontinuity Approach:** This approach relies on the discontinuity of pixel intensity values of the image. Line, Point, and Edge Detection techniques use this type of approach for obtaining intermediate segmentation results which can be later processed to obtain the final segmented image.

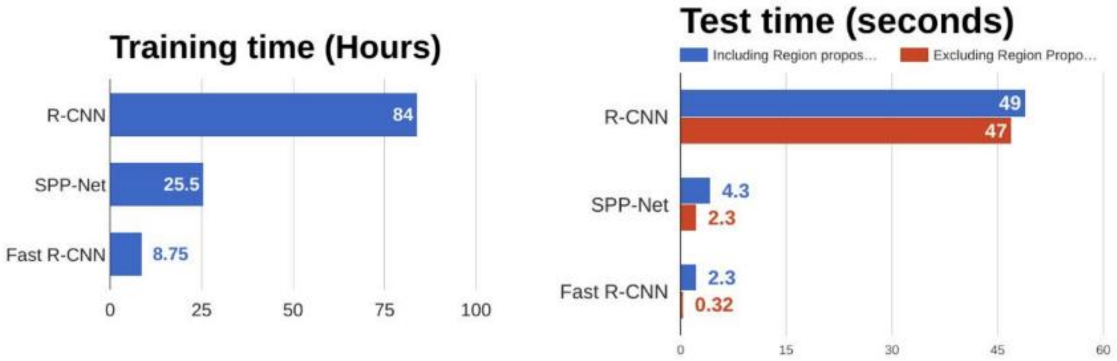
Faster R-CNN algorithm to detect images can be summarized into following steps:

1. Take an input image and pass it to the ConvNet which returns feature maps for the image
2. Apply Region Proposal Network (RPN) on these feature maps and get object proposals
3. Apply ROI pooling layer to bring down all the proposals to the same size
4. Finally, pass these proposals to a fully connected layer in order to classify any predict the bounding boxes for the imag



Both of the above algorithms(R-CNN & Fast R-CNN) uses selective search to find out the region proposals. Selective search is a slow and time-consuming process affecting the performance of the network.

“Fast R-CNN” is faster than R-CNN is because you don’t have to feed 2000 region proposals to the convolutional neural network every time. Instead, the convolution operation is done only once per image and a feature map is generated from it.



Comparison of object detection algorithms

From the above graphs, you can infer that Fast R-CNN is significantly faster in training and testing sessions over R-CNN. When you look at the performance of Fast R-CNN during testing time, including region proposals slows down the algorithm significantly when compared to not using region proposals. Therefore, region proposals become bottlenecks in Fast R-CNN algorithm affecting its performance.

4. Chapter 4

4.4. Code Snippet

Training Code:

```
import random
from torchvision.models.detection.faster_rcnn import FastRCNNPredictor
import numpy as np
import torch.utils.data
import cv2
import torchvision.models.segmentation
import torch
import os
batchSize=2
imageSize=[600,600]
device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
trainDir="D:\\UNIVERSITY FILES\\BSCS-6B\\DS\\DS LAB PROJECT\\Project New\\LabPics Chemistry\\Train"

imgs=[]
for pth in os.listdir(trainDir):
    imgs .append(trainDir+"/"+pth +"//")

def loadData():
    batch_Imgs=[]
    batch_Data=[]
    for i in range(batchSize):
        idx=random.randint(0,len(imgs)-1)
        img = cv2.imread(os.path.join(imgs[idx], "Image.jpg"))
        img = cv2.resize(img, imageSize, cv2.INTER_LINEAR)
        maskDir=os.path.join(imgs[idx], "Vessels")
        masks=[]
        for mskName in os.listdir(maskDir):
            vesMask = (cv2.imread(maskDir+'/'+mskName, 0) > 0).astype(np.uint8)
            vesMask=cv2.resize(vesMask,imageSize,cv2.INTER_NEAREST)
            masks.append(vesMask)
        num_objs = len(masks)
        if num_objs==0: return loadData()
        boxes = torch.zeros([num_objs,4], dtype=torch.float32)
        for i in range(num_objs):
            x,y,w,h = cv2.boundingRect(masks[i])
            boxes[i] = torch.tensor([x, y, x+w, y+h])
        masks = torch.as_tensor(masks, dtype=torch.uint8)
        img = torch.as_tensor(img, dtype=torch.float32)
        data = {}
        data["boxes"] =  boxes
        data["labels"] =  torch.ones((num_objs,), dtype=torch.int64)
        data["masks"] = masks
        batch_Imgs.append(img)
        batch_Data.append(data)  # load images and masks
    batch_Imgs = torch.stack([torch.as_tensor(d) for d in batch_Imgs], 0)
    batch_Imgs = batch_Imgs.swapaxes(1, 3).swapaxes(2, 3)
    return batch_Imgs, batch_Data
```

```python
model = torchvision.models.detection.maskrcnn_resnet50_fpn(pretrained=True)
in_features = model.roi_heads.box_predictor.cls_score.in_features
model.roi_heads.box_predictor = FastRCNNPredictor(in_features,num_classes=2)
model.to(device)

optimizer = torch.optim.AdamW(params=model.parameters(), lr=1e-5)
model.train()

for i in range(10001):
            images, targets = loadData()
            images = list(image.to(device) for image in images)
            targets = [{k: v.to(device) for k, v in t.items()} for t in targets]

            optimizer.zero_grad()
            loss_dict = model(images, targets)

            losses = sum(loss for loss in loss_dict.values())
            losses.backward()
            optimizer.step()
            print(i,'loss:', losses.item())
            if i%500==0:
                torch.save(model.state_dict(), str(i)+".torch")
                print("Save model to:",str(i)+".torch")

```

```
d:\UNIVERSITY FILES\BSCS-6B\DS\DS LAB PROJECT\Project New
(Triggered internally at  C:\actions-runner\_work\pytorch
  masks = torch.as_tensor(masks, dtype=torch.uint8)
0 loss: 148.19711303710938
Save model to: 0.torch
1 loss: 201.9141387939453
2 loss: 56.23267364501953
3 loss: 58.21120071411133
4 loss: 26.661148071289062
5 loss: 37.886043548583984
6 loss: 29.087425231933594
7 loss: 15.810129165649414
8 loss: 14.253555297851562
9 loss: 36.90418243408203
10 loss: 21.309555053710938
11 loss: 23.339401245117188
12 loss: 67.54883575439453
13 loss: 12.201528549194336
14 loss: 22.421964645385742
15 loss: 9.706744194030762
16 loss: 7.446390628814697
17 loss: 11.574146270751953
```

```
499 loss: 1.4417924880981445
500 loss: 0.7946979999542236
Save model to: 500.torch
501 loss: 0.9892111420631409
502 loss: 1.5675773620605469
503 loss: 1.0496221780776978
504 loss: 1.5338184833526611
505 loss: 0.8509167432785034
506 loss: 0.8278596997261047
507 loss: 1.4927784204483032
508 loss: 0.698151171207428
509 loss: 1.20204496383667
510 loss: 0.7276288270950317
511 loss: 1.172558307647705
512 loss: 1.6938422918319702
513 loss: 1.1485438346862793
514 loss: 0.9479849338531494
515 loss: 1.4398906230926514
516 loss: 1.4234057664871216
517 loss: 1.100693702697754
518 loss: 0.902740478515625
519 loss: 0.6490820050239563
520 loss: 0.8304839730262756
521 loss: 0.7874974608421326
522 loss: 0.900202751159668
```

This screenshot represents iterations and loss of data at each iteration (we performed 10,000 iterations)

Testing Code:

```
import random
from torchvision.models.detection.faster_rcnn import FastRCNNPredictor
import numpy as np
import cv2
import torchvision.models.segmentation
import torch
imageSize=[600,600]
imgPath="D:\\UNIVERSITY FILES\\BSCS-6B\\DS\\DS LAB PROJECT\\Project New\\LabPics Chemistry\\Test\\59Eval\\Image.jpg"

device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
model = torchvision.models.detection.maskrcnn_resnet50_fpn(pretrained=True)
in_features = model.roi_heads.box_predictor.cls_score.in_features
model.roi_heads.box_predictor = FastRCNNPredictor(in_features,num_classes=2)
model.load_state_dict(torch.load("10000.torch"))
model.to(device)
model.eval()

images=cv2.imread(imgPath)
images = cv2.resize(images, imageSize, cv2.INTER_LINEAR)
images = torch.as_tensor(images, dtype=torch.float32).unsqueeze(0)
images=images.swapaxes(1, 3).swapaxes(2, 3)
images = list(image.to(device) for image in images)

with torch.no_grad():
    pred = model(images)

im= images[0].swapaxes(0, 2).swapaxes(0, 1).detach().cpu().numpy().astype(np.uint8)
im2 = im.copy()
for i in range(len(pred[0]['masks'])):
    msk=pred[0]['masks'][i,0].detach().cpu().numpy()
    scr=pred[0]['scores'][i].detach().cpu().numpy()
    if scr>0.8 :
        im2[:,:,0][msk>0.5] = random.randint(0,255)
        im2[:, :, 1][msk > 0.5] = random.randint(0,255)
        im2[:, :, 2][msk > 0.5] = random.randint(0, 255)
cv2.imshow(str(scr), np.hstack([im,im2]))
cv2.waitKey()
```

Output:

Figure 01

Figure 02

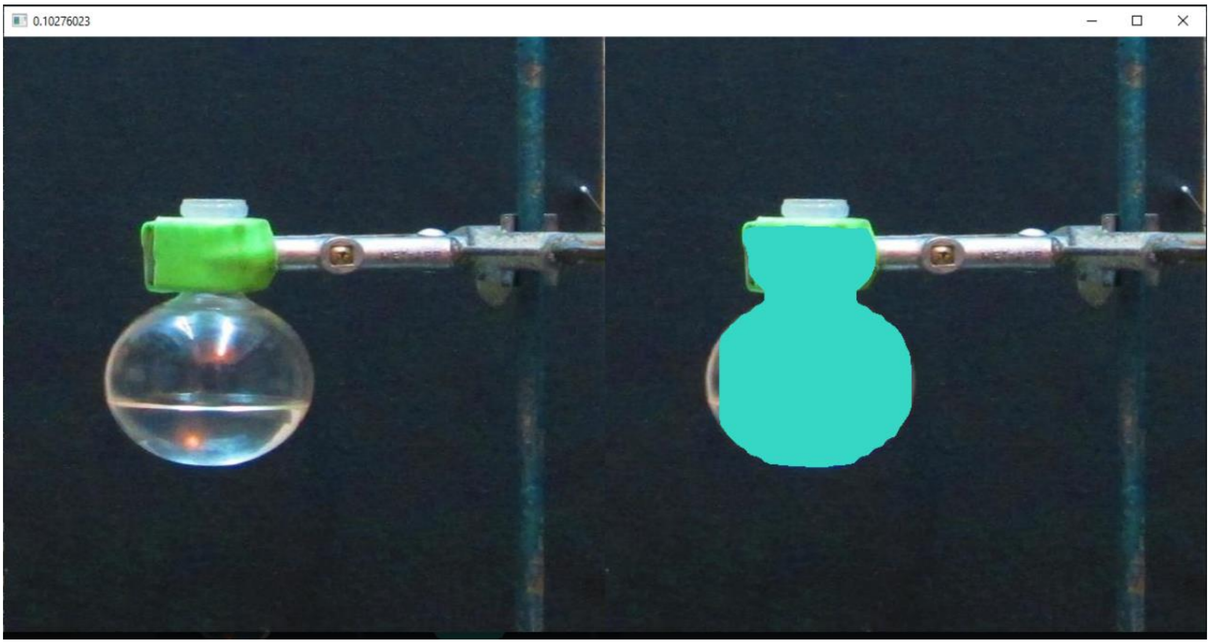


Figure 03

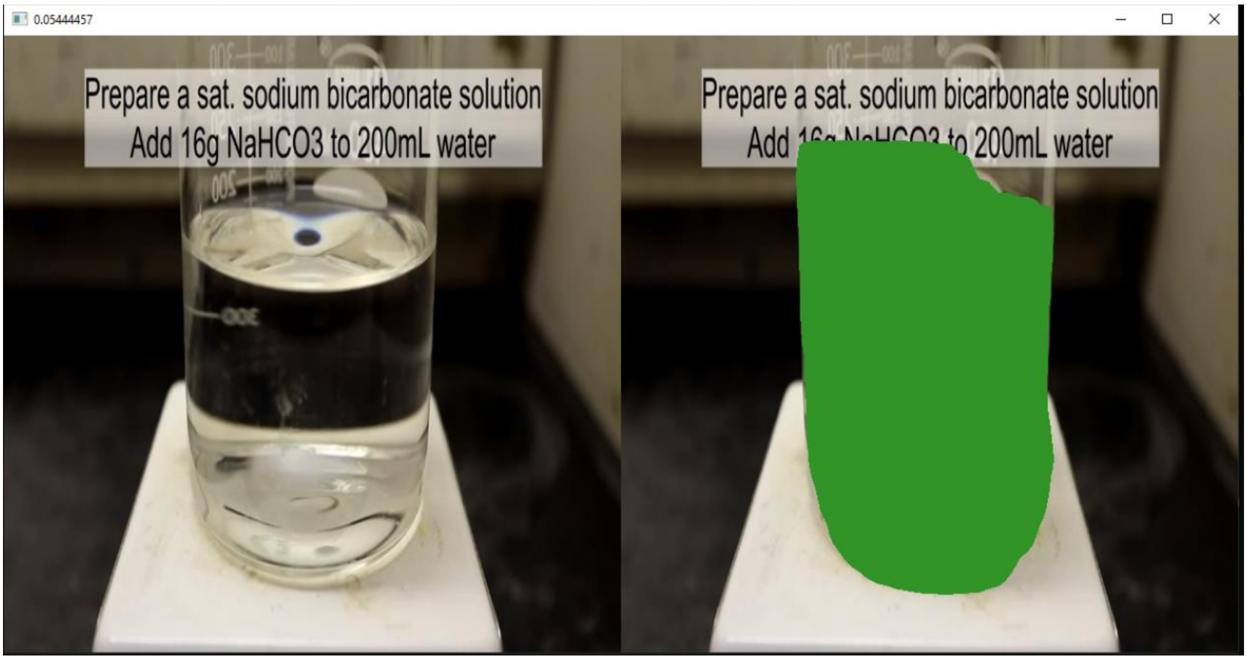


Figure 04

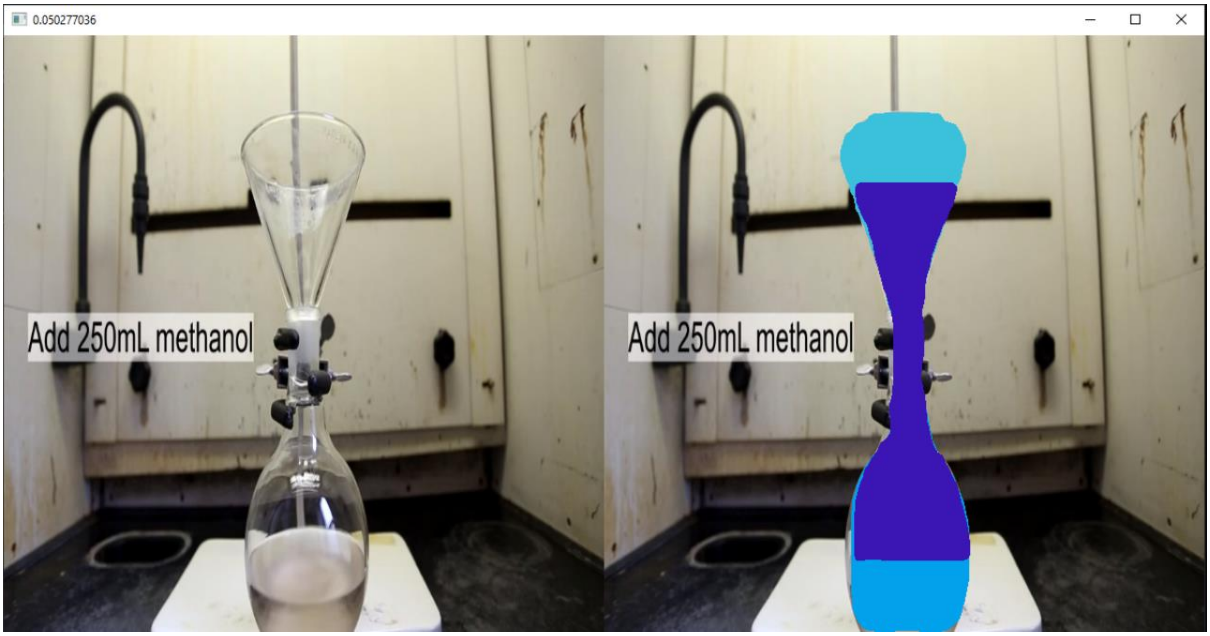


Figure 05

Figure 06

Pytorch Files

Important (D:) > UNIVERSITY FILES > BSCS-6B > DS > DS LAB PROJECT > Project New

Name	Date modified	Type	Size
vitz.jpg	06/07/2022 11:38 pm	JPG File	299 KB
test.py	07/07/2022 12:55 am	PY File	2 KB
new.py	07/07/2022 12:50 am	PY File	3 KB
index.jpg	06/07/2022 11:36 pm	JPG File	7 KB
Image.jpg	06/07/2022 11:20 pm	JPG File	240 KB
Deso2.jpg	03/07/2022 7:34 pm	JPG File	76 KB
cars.jpg	06/07/2022 11:30 pm	JPG File	72 KB
10000.torch	06/07/2022 11:18 pm	TORCH File	172,184 KB
9500.torch	06/07/2022 11:08 pm	TORCH File	172,184 KB
9000.torch	06/07/2022 10:59 pm	TORCH File	172,184 KB
8500.torch	06/07/2022 10:50 pm	TORCH File	172,184 KB
8000.torch	06/07/2022 10:41 pm	TORCH File	172,184 KB
7500.torch	06/07/2022 10:33 pm	TORCH File	172,184 KB
7000.torch	06/07/2022 10:25 pm	TORCH File	172,184 KB
6500.torch	06/07/2022 10:16 pm	TORCH File	172,184 KB
6000.torch	06/07/2022 10:07 pm	TORCH File	172,184 KB
5500.torch	06/07/2022 9:58 pm	TORCH File	172,184 KB
5000.torch	06/07/2022 9:50 pm	TORCH File	172,184 KB
4500.torch	06/07/2022 9:41 pm	TORCH File	172,184 KB
4000.torch	06/07/2022 9:33 pm	TORCH File	172,184 KB
3500.torch	06/07/2022 9:25 pm	TORCH File	172,184 KB
3000.torch	06/07/2022 9:17 pm	TORCH File	172,184 KB
2500.torch	06/07/2022 9:09 pm	TORCH File	172,184 KB
2000.torch	06/07/2022 9:00 pm	TORCH File	172,184 KB
1500.torch	06/07/2022 8:52 pm	TORCH File	172,184 KB
1000.torch	06/07/2022 8:44 pm	TORCH File	172,184 KB
500.torch	06/07/2022 11:52 pm	TORCH File	172,184 KB
0.torch	06/07/2022 11:43 pm	TORCH File	172,184 KB
LabPics Chemistry	06/07/2022 8:20 pm	File folder	

5. Chapter 6

5.1. Conclusion

This project is a basic solution for a image segmentation problem on chemistry lab equipments. This can be extended over other medical or industrial problems. After the segmentation, we can use these images by passing them to further more advances ML models, to solve various classification and detection problems. We have used Fast R-CNN for this purpose, but there are even better approaches available for the task at hand.

5.2. Future Work

Following are the implementations of image segmentation which can be useful:

- Blood cell detection
- Autonomous vehicles
- Retail applications

6. Chapter 6

6.1. References

http://www.cs.utoronto.ca/~strider/publications/Chapter9.pdf

https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

https://www.researchgate.net/publication/334850808_Improvement_of_Mask-RCNN_Object_Segmentation_Algorithm

https://towardsdatascience.com/image-segmentation-part-1-9f3db1ac1c50

https://www.analyticsvidhya.com/blog/2019/07/computer-vision-implementing-mask-r-cnn-image-segmentation/