IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

IMAGE SEGMENTATION USING THE R-CNN ARCHITECTURE

[1]Abhinav Aggarwal, [2]Shikhar Yadav, [3]Shobhit Srivastava, [4]Chahat Agarwal, [5]Ankita Singh

[1]Student, [2]Student, [3]Student, [4]Student, [5]Associate Professor

Department of Computer Science and Engineering

SRM Institute of Science and Technology, Modinagar, Ghaziabad, Delhi-NCR

Abstract: In this project, with applications including scene understanding, medical image analysis, robotic perception, video surveillance, augmented reality, and image compression, among many others, image segmentation is the main key topic in this project's use of computer vision and image processing. In the literature, a number of picture segmentation methods have been created. A common label is given to all the visual components or pixels that fall under the same category. In addition to the class label and object bounding box, the Mask R-CNN returns the binary object mask as well. In terms of segmenting pixels, Mask R-CNN is effective.

Index Terms - Convolutional Neural Networks (CNNs), Convolutional Neural Networks (CNNs), Labelled Data, Binary Masks, Computer Vision.

I. INTRODUCTION

Object detection and image localization are concepts we have probably heard of. When there is the single object present in a photo or an image, we use image localization technique to draw a bounding box around that object. It provides labels in addition to bounding boxes for object detection., allowing us to predict both the location and the class that each object belongs to. Image segmentation extends the idea of object detection by providing more detailed information about an image's shape. We segment, or divide, the photos into areas of various colours, which helps in more precise item differentiation. An image is a visible representation of something and carries a lot of beneficial information. Analyzing the image and obtaining information from it in a way that does not influence the image's other features in order to get some tasks done is one of the important applications of digital image technology. In various sectors and real-world applications such as military, medical, astronomy, etc. pattern recognition, image analysis, and image disciplines are the most important subjects in computer science and computer engineering. Innovative technologies are now being developed in the fields of image processing, particularly in the area of the image segmentation. Image segmentation is an important as well as difficult process. The most important stage of picture analysis is the segmentation procedure. The technique of splitting an image into homogenous parts based on particular criteria and, ideally, corresponding to actual things in the scene is known as image segmentation. The next steps of a typical recognition system actually use image segmentation to offer simpler and easier data, such colours or texture. From a multi-media perspective, image segmentation can be applied to a single image or a series of images that form a video. In general, the basic goal of segmentation is for reducing data for an easier analysis process, where image segmentation is described as the division of a digital image into its continuous, unconnected, and nonempty subsets to facilitate attribute extraction. In the field of image processing, picture segmentation is still an active and fascinating study area. The development of a universal technique for image segmentation remains a difficult task for academics and developers. Basic prerequisites for good image segmentation include:

- The area's pixels are all part of an image.
- If any two pixels in a specified region may be connected by a line that does not exit the region, the region is linked.
- Each region is homogeneous in term of a selected characteristic. The characteristic could be syntactic (intensity, color and texture) or semantically based.
- It is impossible to combine neighboring regions into a single homogeneous region.

II. Literature Survey

In recent years, there have been many research papers that have used the Mask R-CNN architecture for image segmentation tasks. Mask R-CNN is a state-of-the-art deep learning-based object detection and segmentation algorithm that extends the popular Faster R-CNN architecture. It allows for pixel-level segmentation of objects in images and has been widely used in various computer vision applications, including autonomous driving, robotics, and medical image analysis. In a literature survey, we have found that the following are some of the key findings and trends:

- Mask R-CNN has achieved state-of-the-art performance on various benchmark datasets for image segmentation, including COCO, Cityscapes, and PASCAL VOC.
- Many researchers have proposed modifications to the original Mask R-CNN architecture to improve its performance. For example, some researchers have explored different backbone architectures, such as ResNet and ResNeXt, to improve the feature representation of the network. Others have introduced novel loss functions or regularization techniques to improve the network's segmentation accuracy.
- Some researchers have extended the Mask R-CNN architecture to handle more challenging segmentation tasks, such as instance-level segmentation of overlapping objects or semantic segmentation of 3D point clouds.
- Transfer learning is a popular technique used by many researchers to adapt pre-trained Mask R-CNN models to new segmentation tasks. This involves fine-tuning the pretrained network on a smaller dataset or modifying its architecture to fit the new task.

Table 1: Recent Literature

S.No.	Author's Name (Year Of Publication)	Title of Paper	Methodology	Scope for Improvement
1	Hyungtae Lee, Sungmin Eum, Heesung Kwon & 09 September 2019	ME R-CNN: Multi-Expert R-CNN for Object Detection	Multiple experts are used in the Multi-Expert Region-based Convolutional Neural Network (ME R-CNN), where each expert has been trained to analyse a particular class of regions of interest (RoIs).	By including extra processings (referred to as "adding bells and whistles") such multi-scale training/testing (MS), online hard example mining (OHEM), and iterative bounding box regression, the emphasis is on creating the state-of-the-art performance in benchmark datasets (PASCAL VOC and MS COCO)..
2	Puja Bharati, Ankita Pramanik & 18 August 2019	Deep Learning Techniques—R-CNN to Mask R-CNN: A Survey	It offers a thorough explanation of region-based convolutional neural network (R-CNN) as well as its most recent advancements, including fast R-CNN, faster R-CNN, region-based fully convolutional networks, single shot detector, deconvolutional single shot detector, R-CNN minus R, you only look once (YOLO), and mask R-CNN.	A strong base network enables YOLO to operate more quickly and achieve 21~155 frames per second. With the use of ResNet architecture, mask R-CNN have an average precision of 47.3 in terms of accuracy.
3	Kaiming He, Georgia, Gkioxari, Piotr Dollar, Ross Girshick & 20 March 2017	Mask R-CNN	By adding a branch for object mask prediction in parallel with the current branch for bounding box recognition, Mask R-CNN expands Faster R-CNN.	A powerful object identification and segmentation technique, Mask R-CNN, has shown outstanding results in a variety of applications. It is a useful tool for many different industries due to its capacity to precisely recognise and segment objects, and it is anticipated to continue to be a significant area of research in the field of computer vision in the years to come.
4	Lili Zhang, Jisen Wu, Yu Fan, Hongmin Gao, Yehong Shao & 6 March 2020	An Efficient Building Extraction Method from High Spatial Resolution Remote Sensing Images Based on Improved Mask R-CNN	We discuss building extraction from high spatial resolution remote sensing pictures in this work. The majority of building extraction techniques now in use rely on artificial features. Building extraction methods still face significant difficulties due to the diversity and complexity of buildings, hence deep learning-based approaches have recently been proposed.	In order to extract buildings from high-resolution remote sensing photos, deep convolutional neural networks are used in this paper. Building recognition and high-precision extraction in high-resolution remote sensing photos were realised using the deep convolutional network's properties. An optimisation strategy incorporating edge characteristics was presented to increase the effectiveness of the network model on building extraction in response to the issues of poor edge recognition and incomplete extraction of

				the convolutional networks on the building extraction from remote sensing photos.
5	Qinzhe Han, Qian Yin, Xin Zheng, Ziyi Chen & 27 March 2021	Remote sensing image building detection method based on Mask R-CNN	An essential component of catastrophe assessment is the quick and convenient identification of buildings in disaster zones. In order to address the technical requirements of flood disaster relief initiatives, this research proposes a building extraction approach for use with remote sensing pictures that blends traditional digital image processing methods and convolution neural networks.	The benefits of the conventional unsupervised object segmentation method are applied in this study to create a data collection. Using the residual network's enhanced Mask R-CNN technique, a building detection method is created for remote sensing photos of flood disaster zones.
6	Kaihan Lin, Huimin Zhao, Jujian Lv, Canyao Li, Xiaoyong Liu, Rongjun Chen, Ruoyan Zhao & 01 May 2020	Face Detection and Segmentation Based on Improved Mask R-CNN	In this we introduce the G-Mask face detection and segmentation approach, which unifies face recognition and segmentation into a single framework with the goal of obtaining more precise face information. This method is based on an enhanced Mask R-CNN.	In this study, a G-Mask approach for segmenting and detecting faces was proposed. The method can construct RoIs using RPN, extract features using ResNet-101, maintain accurate spatial location using RoIAlign, and produce binary masks using the full convolutional network (FCN).

III. Methodology

In PyTorch, the Mask R-CNN framework is a well-liked one for image segmentation. The following steps make up the general process for Mask R-CNN:

1. Data Preparation: Any machine learning project must start with gathering and preparing the data. Labelling the areas of interest in the photos is often involved in segmenting them. The model is then trained using these labels.
2. Model Architecture: A variation of the Faster R-CNN object detection model is Mask R-CNN. It is divided into two stages: a segmentation network and a region proposal network (RPN). The segmentation network creates masks for each area proposal that the RPN generates.
3. Training: The model is trained using a training dataset when the data and model architecture have been established. The model gains the ability to categorise and segment the objects in the photos during training.
4. Evaluation: After training, a validation dataset is used to assess the model. Precision, recall, and F1 score are a few examples of measures that are used to evaluate the model's performance.
5. Inference: The model can be applied to inference on fresh photos after being trained and assessed. The model creates a set of region proposals from an input image. After that, the segmentation network generates a mask for each suggestion that may be used to divide the image into segments.

The methodology involves the following steps:

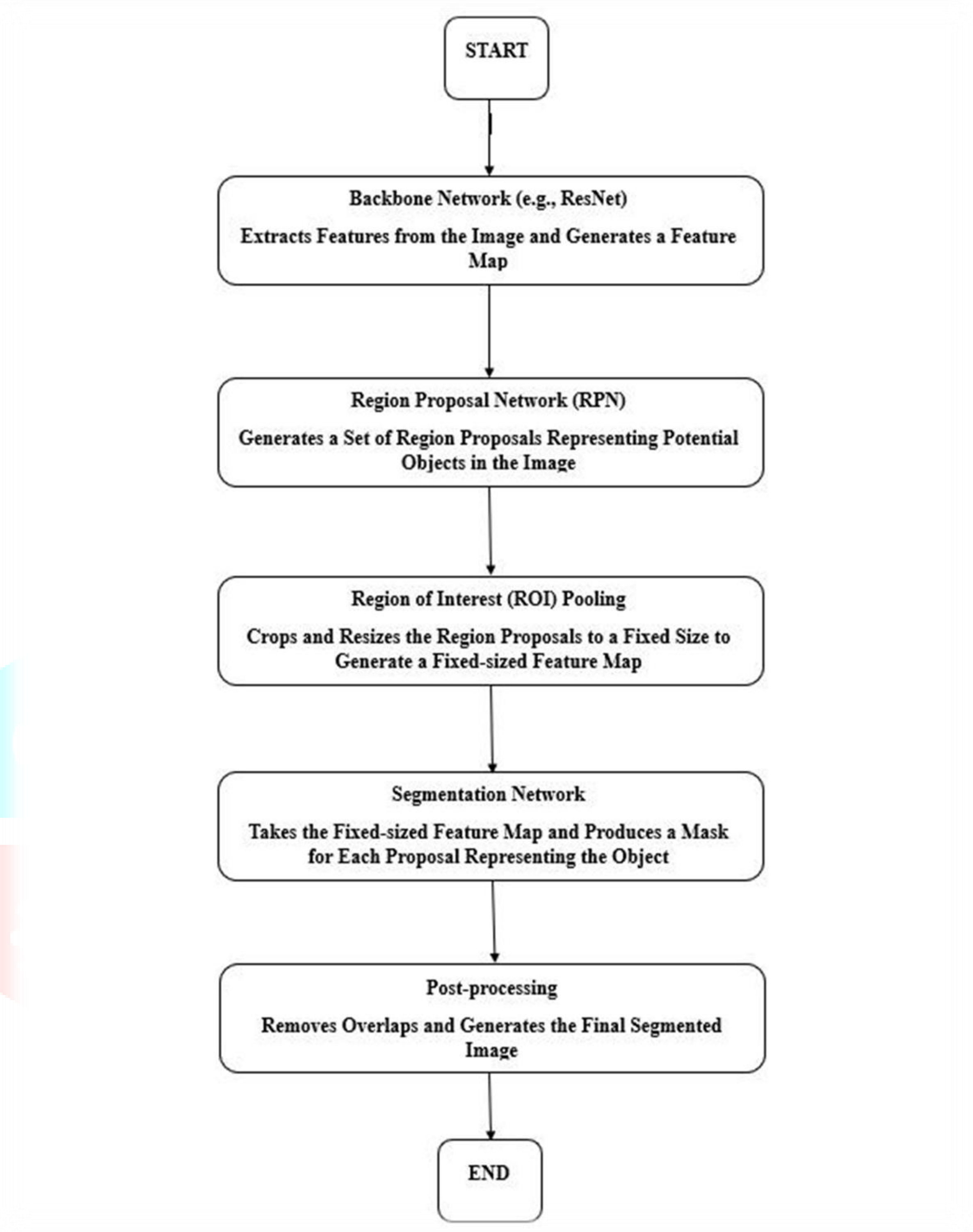

START
Backbone Network (e.g., ResNet)
Extracts Features from the Image and Generates a Feature Map
Region Proposal Network (RPN)
Generates a Set of Region Proposals Representing Potential Objects in the Image
Region of Interest (ROI) Pooling
Crops and Resizes the Region Proposals to a Fixed Size to Generate a Fixed-sized Feature Map
Segmentation Network
Takes the Fixed-sized Feature Map and Produces a Mask for Each Proposal Representing the Object
Post-processing
Removes Overlaps and Generates the Final Segmented Image
END

Fig: - Workflow of the Model

3.1 System Architecture

The open-source machine learning package PyTorch was created primarily by Facebook's AI research team. It offers a flexible framework for creating and training deep neural networks and is based on the Torch library. In order to efficiently compute gradients during backpropagation, PyTorch offers a dynamic computational network, which makes it suitable for training sophisticated models. Additionally, a variety of pre-built modules for creating neural networks are available, along with tools for loading and processing data. Because of its simplicity, adaptability, and easy integration with other Python tools, PyTorch has become quite well-liked in the deep learning community.

An open-source library for computer vision and machine learning is called OpenCV (Open Source Computer Vision). It was initially created by Intel, and the OpenCV community currently looks after it. A set of tools and algorithms are provided by OpenCV that can be used to create real-time computer vision applications. Filtering, feature detection, segmentation, and machine learning algorithms for object detection and recognition are just a few of the many image processing operations it offers. Other computer vision-related tasks supported by OpenCV include camera calibration, stereo vision, and optical flow. With interfaces for Python, Java, and MATLAB in addition to being built in C++, it is usable by a variety of developers. In computer vision research as well as in commercial applications like robotics, surveillance, and self-driving cars.

A Python data visualisation toolkit called Matplotlib offers a large selection of tools and functions for producing static, animated, and interactive visualisations. It was created as an open-source tool that enables quick and simple plot and graph development. Line plots, scatter plots, bar plots, histograms, and many other types of graphs are supported by Matplotlib. It has a customizable user interface that allows you to change the colours, markers, labels, and annotations of plot elements. Additionally, Matplotlib includes a variety of built-in features for working with datasets, such as support for Pandas and NumPy dataframes and arrays. Additionally, SciPy, Pandas, and NumPy are just a few of the scientific computing Python libraries that work nicely with Matplotlib. The scientific and data analysis groups now frequently utilise Matplotlib for data visualisation because of its simplicity, adaptability, and wide range of features.

3.2 Steps involved in project are:

3.2.1 The first step is to import and install the required libraries and getting the mask-rcnn model.

3.2.2 Checking the System Specification of GPU (NVIDIA System Management Interface) to run the model.

3.2.3 Implementing the coco classes in the coco instance category names.

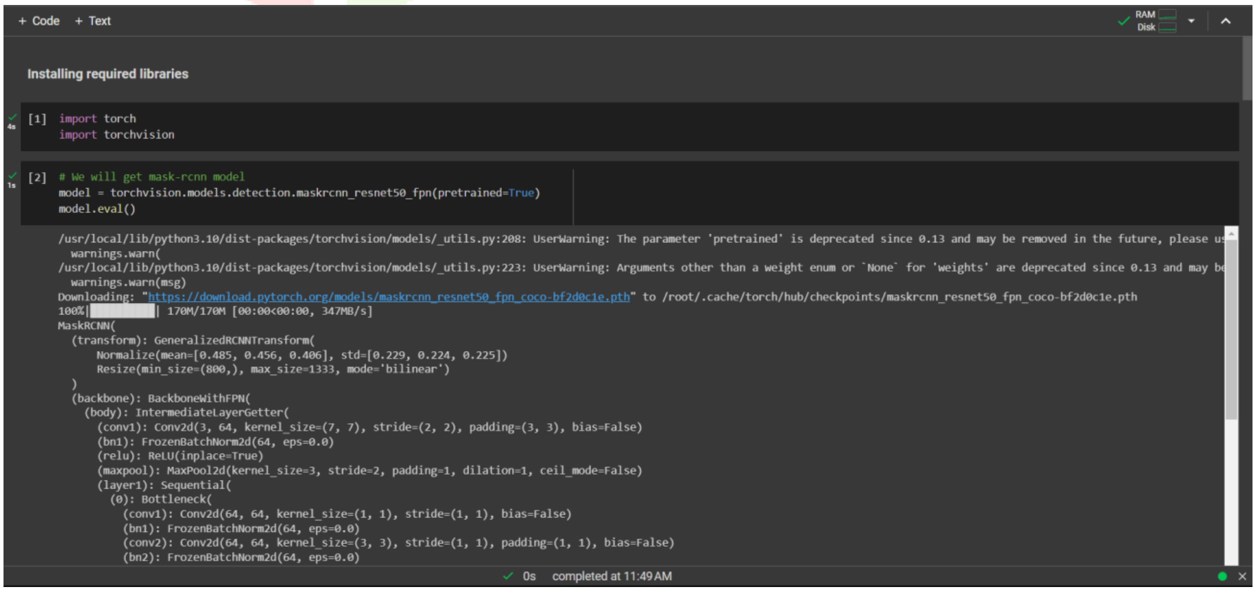
3.2.4 Then we will create a prediction model in which it will take the image and extract the bounding boxes and masks.

3.2.5 We will give the list of colours in the model and it will return the colored mask of an image.

3.2.6 After the training of the model we will download a random image and it will be tested through the data which we have implemented and gives the output as segmented image.

3.3 Result and Analysis: The objective of this study was to separate an input image into several regions or segments, each of which corresponds to a particular object or area of interest. A series of binary masks, one for each detected object, are produced as the algorithm's output and show which pixels in the input image correspond to each object. The masks are useful in a variety of applications, including self-driving cars, robotics, and medical imaging, and may be used to carry out tasks including object detection, tracking, and instance segmentation. Mask RCNN lets machines to more effectively comprehend and interpret visual data, which is an essential first step towards creating more intelligent and autonomous systems.

Here is the implementation of the code and the result we get:

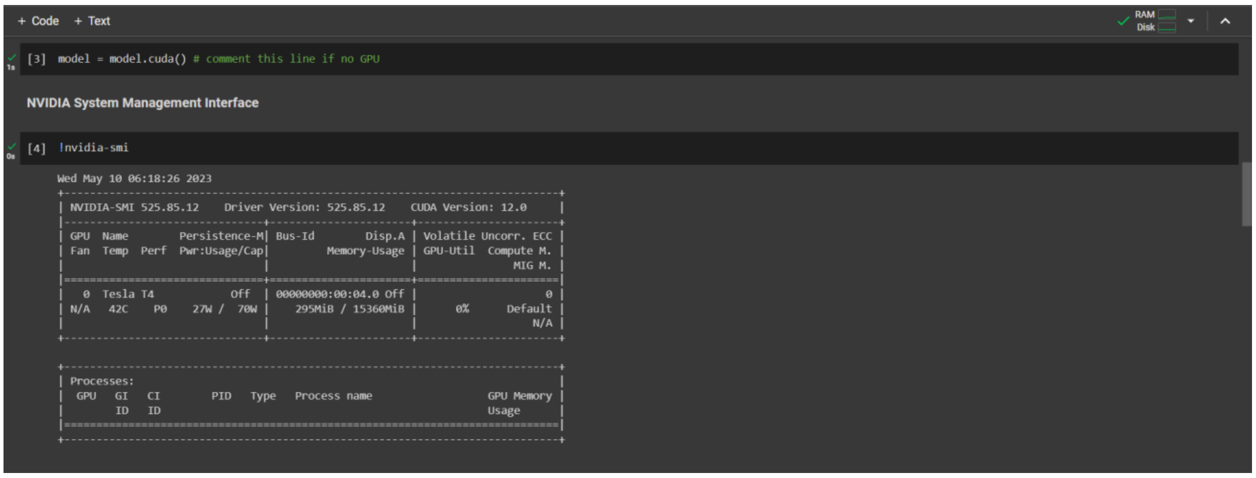


```
[1] import torch
    import torchvision
```

```
[2] # We will get mask-rcnn model
    model = torchvision.models.detection.maskrcnn_resnet50_fpn(pretrained=True)
    model.eval()
```

```
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be removed in the future, please us
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or `None` for 'weights' are deprecated since 0.13 and may be
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/maskrcnn_resnet50_fpn_coco-bf2d0c1e.pth" to /root/.cache/torch/hub/checkpoints/maskrcnn_resnet50_fpn_coco-bf2d0c1e.pth
100%|██████████| 170M/170M [00:00<00:00, 347MB/s]
MaskRCNN(
  (transform): GeneralizedRCNNTransform(
      Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
      Resize(min_size=(800,), max_size=1333, mode='bilinear')
  )
  (backbone): BackboneWithFPN(
    (body): IntermediateLayerGetter(
      (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
      (bn1): FrozenBatchNorm2d(64, eps=0.0)
      (relu): ReLU(inplace=True)
      (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
      (layer1): Sequential(
        (0): Bottleneck(
          (conv1): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (bn1): FrozenBatchNorm2d(64, eps=0.0)
          (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (bn2): FrozenBatchNorm2d(64, eps=0.0)
```

Fig: - Installing the required libraries

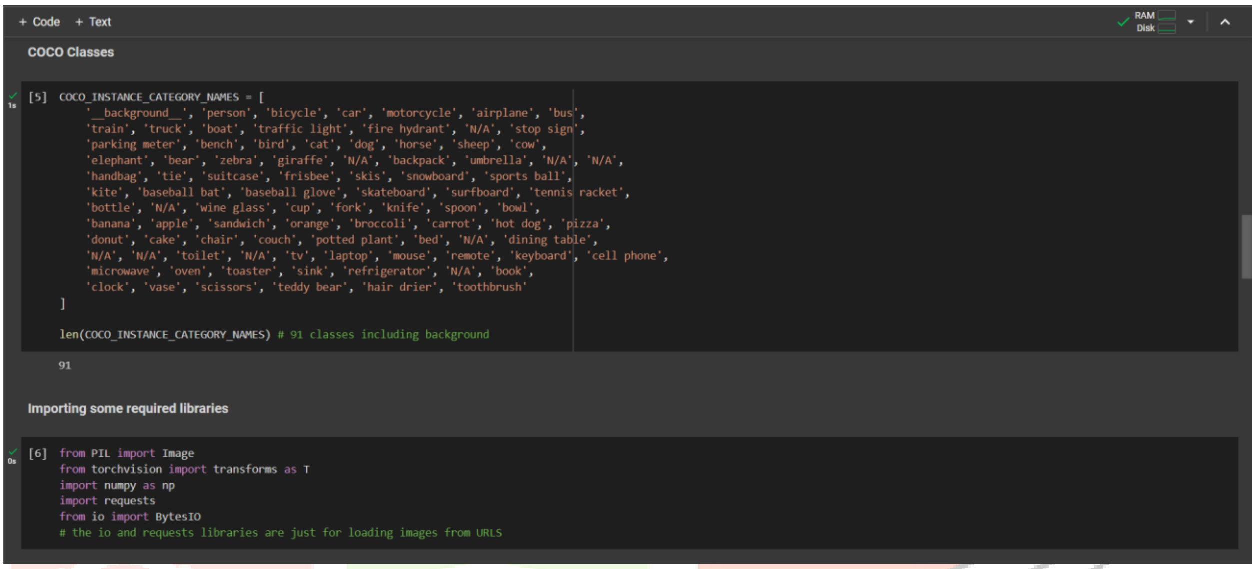


```
[3] model = model.cuda() # comment this line if no GPU
```

NVIDIA System Management Interface

```
[4] !nvidia-smi

Wed May 10 06:18:26 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.85.12    Driver Version: 525.85.12    CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   42C    P0    27W /  70W |    295MiB / 15360MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

Fig: - Checking the System GPU Specification



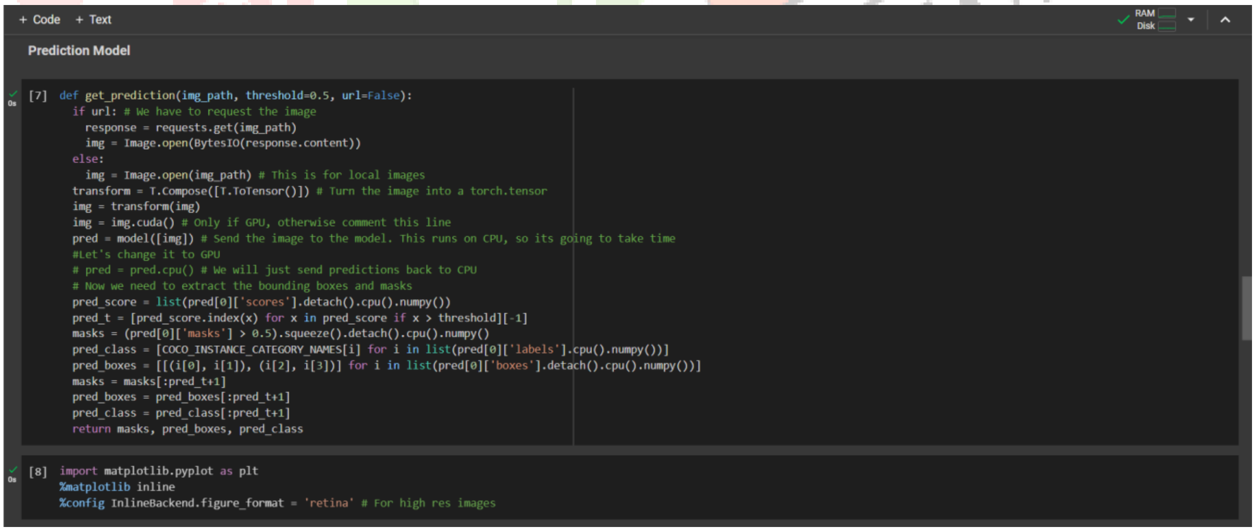
COCO Classes

```
[5] COCO_INSTANCE_CATEGORY_NAMES = [
        '__background__', 'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus',
        'train', 'truck', 'boat', 'traffic light', 'fire hydrant', 'N/A', 'stop sign',
        'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow',
        'elephant', 'bear', 'zebra', 'giraffe', 'N/A', 'backpack', 'umbrella', 'N/A', 'N/A',
        'handbag', 'tie', 'suitcase', 'frisbee', 'skis', 'snowboard', 'sports ball',
        'kite', 'baseball bat', 'baseball glove', 'skateboard', 'surfboard', 'tennis racket',
        'bottle', 'N/A', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl',
        'banana', 'apple', 'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza',
        'donut', 'cake', 'chair', 'couch', 'potted plant', 'bed', 'N/A', 'dining table',
        'N/A', 'N/A', 'toilet', 'N/A', 'tv', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone',
        'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'N/A', 'book',
        'clock', 'vase', 'scissors', 'teddy bear', 'hair drier', 'toothbrush'
    ]

    len(COCO_INSTANCE_CATEGORY_NAMES) # 91 classes including background

    91
```

Importing some required libraries

```
[6] from PIL import Image
    from torchvision import transforms as T
    import numpy as np
    import requests
    from io import BytesIO
    # the io and requests libraries are just for loading images from URLS
```

Fig: - Importing COCO Classes

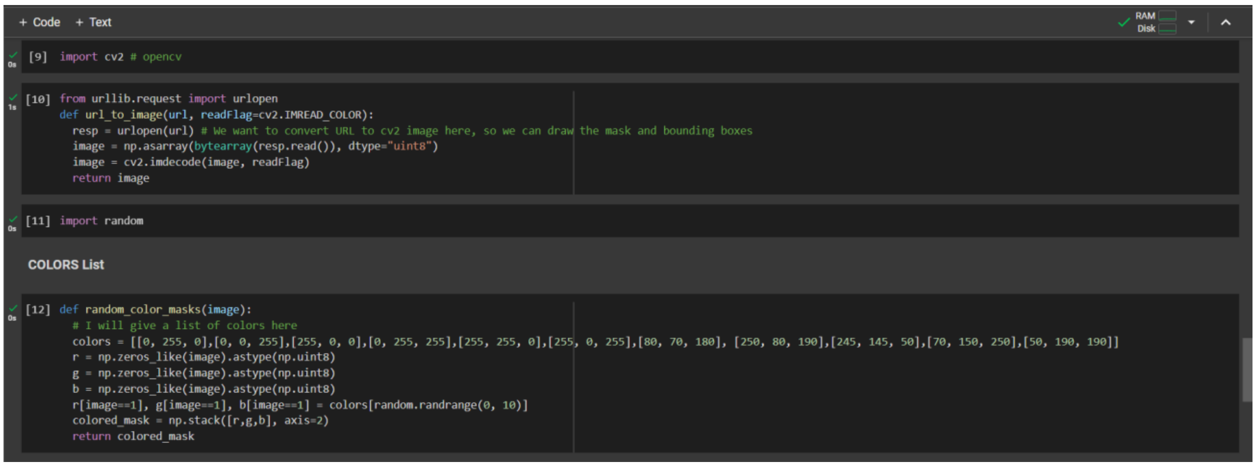


Prediction Model

```
[7] def get_prediction(img_path, threshold=0.5, url=False):
      if url: # We have to request the image
        response = requests.get(img_path)
        img = Image.open(BytesIO(response.content))
      else:
        img = Image.open(img_path) # This is for local images
      transform = T.Compose([T.ToTensor()]) # Turn the image into a torch.tensor
      img = transform(img)
      img = img.cuda() # Only if GPU, otherwise comment this line
      pred = model([img]) # Send the image to the model. This runs on CPU, so its going to take time
      #Let's change it to GPU
      # pred = pred.cpu() # We will just send predictions back to CPU
      # Now we need to extract the bounding boxes and masks
      pred_score = list(pred[0]['scores'].detach().cpu().numpy())
      pred_t = [pred_score.index(x) for x in pred_score if x > threshold][-1]
      masks = (pred[0]['masks'] > 0.5).squeeze().detach().cpu().numpy()
      pred_class = [COCO_INSTANCE_CATEGORY_NAMES[i] for i in list(pred[0]['labels'].cpu().numpy())]
      pred_boxes = [[(i[0], i[1]), (i[2], i[3])] for i in list(pred[0]['boxes'].detach().cpu().numpy())]
      masks = masks[:pred_t+1]
      pred_boxes = pred_boxes[:pred_t+1]
      pred_class = pred_class[:pred_t+1]
      return masks, pred_boxes, pred_class
```

```
[8] import matplotlib.pyplot as plt
    %matplotlib inline
    %config InlineBackend.figure_format = 'retina' # For high res images
```

Fig: - Extracting the bounding boxes and masks

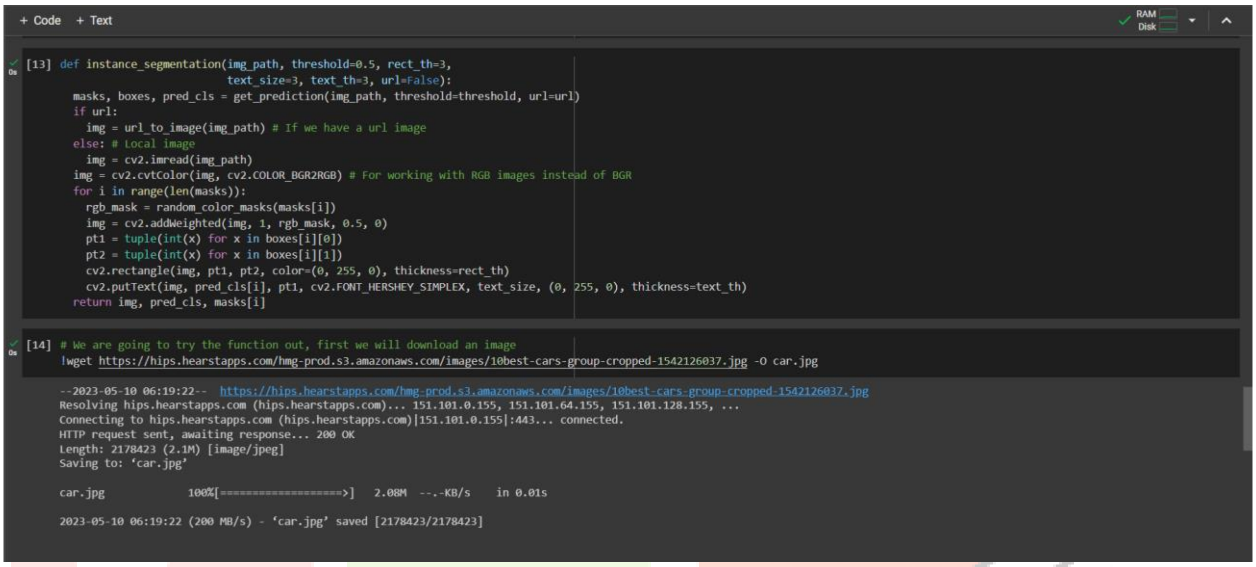


```
[9]  import cv2 # opencv

[10] from urllib.request import urlopen
     def url_to_image(url, readFlag=cv2.IMREAD_COLOR):
       resp = urlopen(url) # We want to convert URL to cv2 image here, so we can draw the mask and bounding boxes
       image = np.asarray(bytearray(resp.read()), dtype="uint8")
       image = cv2.imdecode(image, readFlag)
       return image

[11] import random
```

COLORS List

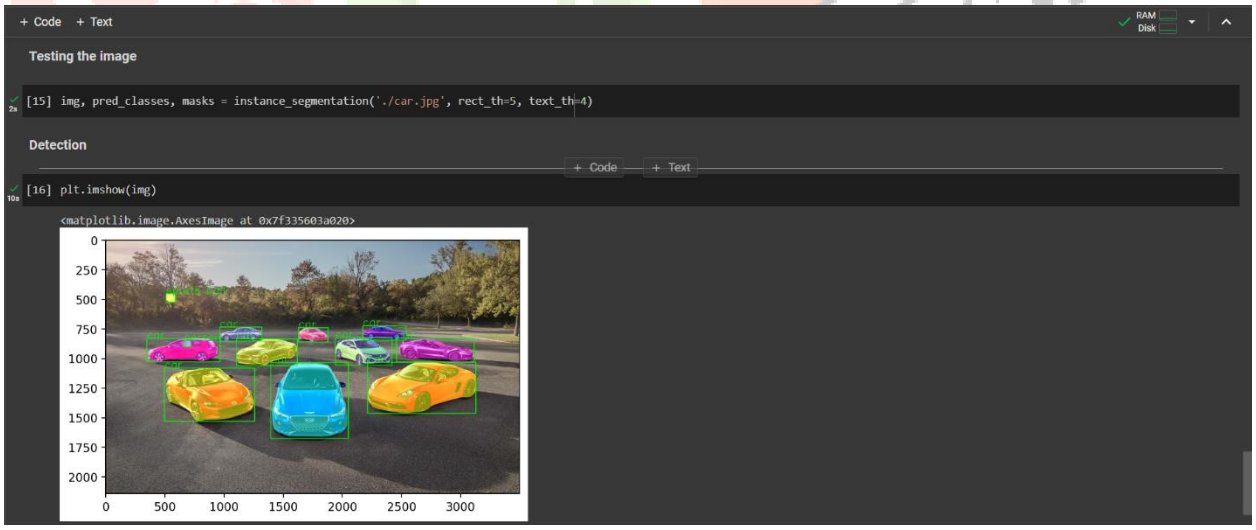
```
[12] def random_color_masks(image):
       # I will give a list of colors here
       colors = [[0, 255, 0],[0, 0, 255],[255, 0, 0],[0, 255, 255],[255, 255, 0],[255, 0, 255],[80, 70, 180], [250, 80, 190],[245, 145, 50],[70, 150, 250],[50, 190, 190]]
       r = np.zeros_like(image).astype(np.uint8)
       g = np.zeros_like(image).astype(np.uint8)
       b = np.zeros_like(image).astype(np.uint8)
       r[image==1], g[image==1], b[image==1] = colors[random.randrange(0, 10)]
       colored_mask = np.stack([r,g,b], axis=2)
       return colored_mask
```

Fig: - Giving a list of colors



```
[13] def instance_segmentation(img_path, threshold=0.5, rect_th=3,
                               text_size=3, text_th=3, url=False):
       masks, boxes, pred_cls = get_prediction(img_path, threshold=threshold, url=url)
       if url:
         img = url_to_image(img_path) # If we have a url image
       else: # Local image
         img = cv2.imread(img_path)
       img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) # For working with RGB images instead of BGR
       for i in range(len(masks)):
         rgb_mask = random_color_masks(masks[i])
         img = cv2.addWeighted(img, 1, rgb_mask, 0.5, 0)
         pt1 = tuple(int(x) for x in boxes[i][0])
         pt2 = tuple(int(x) for x in boxes[i][1])
         cv2.rectangle(img, pt1, pt2, color=(0, 255, 0), thickness=rect_th)
         cv2.putText(img, pred_cls[i], pt1, cv2.FONT_HERSHEY_SIMPLEX, text_size, (0, 255, 0), thickness=text_th)
       return img, pred_cls, masks[i]

[14] # We are going to try the function out, first we will download an image
     !wget https://hips.hearstapps.com/hmg-prod.s3.amazonaws.com/images/10best-cars-group-cropped-1542126037.jpg -O car.jpg

     --2023-05-10 06:19:22--  https://hips.hearstapps.com/hmg-prod.s3.amazonaws.com/images/10best-cars-group-cropped-1542126037.jpg
     Resolving hips.hearstapps.com (hips.hearstapps.com)... 151.101.0.155, 151.101.64.155, 151.101.128.155, ...
     Connecting to hips.hearstapps.com (hips.hearstapps.com)|151.101.0.155|:443... connected.
     HTTP request sent, awaiting response... 200 OK
     Length: 2178423 (2.1M) [image/jpeg]
     Saving to: 'car.jpg'

     car.jpg             100%[===================>]   2.08M  --.-KB/s    in 0.01s

     2023-05-10 06:19:22 (200 MB/s) - 'car.jpg' saved [2178423/2178423]
```

Fig: - Importing the random image to the Model



Testing the image

```
[15] img, pred_classes, masks = instance_segmentation('./car.jpg', rect_th=5, text_th=4)
```

Detection

```
[16] plt.imshow(img)

     <matplotlib.image.AxesImage at 0x7f335603a020>
```

Fig: - Image segmentation

Fig: - Random Image of Cars

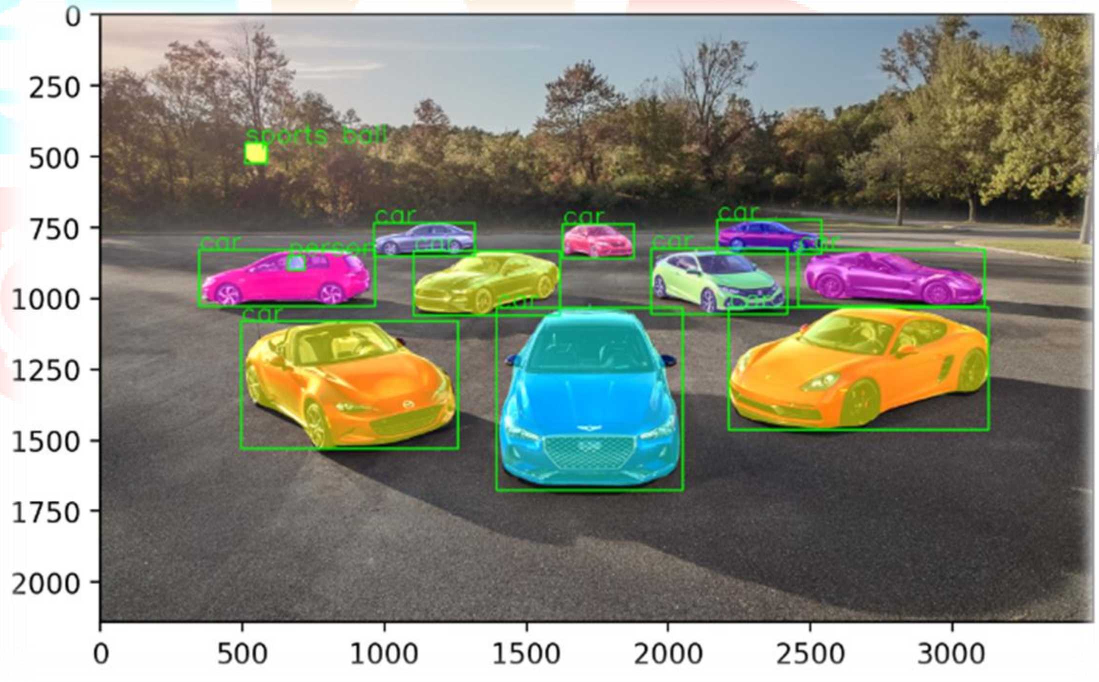
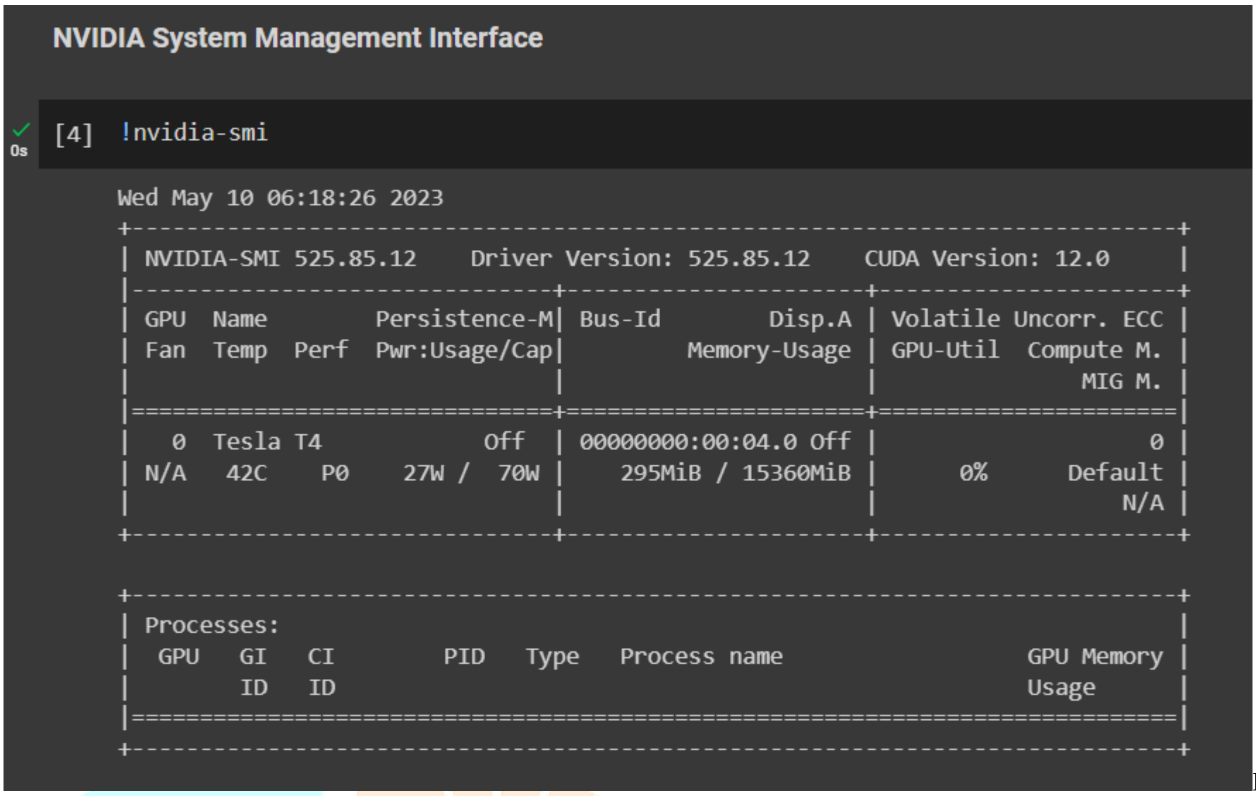


Fig: - Segmented Image

IV. System Specification



NVIDIA System Management Interface

```
[4]  !nvidia-smi

     Wed May 10 06:18:26 2023
     +-----------------------------------------------------------------------------+
     | NVIDIA-SMI 525.85.12    Driver Version: 525.85.12    CUDA Version: 12.0     |
     |-------------------------------+----------------------+----------------------+
     | GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
     | Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
     |                               |                      |               MIG M. |
     |===============================+======================+======================|
     |   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
     | N/A   42C    P0    27W /  70W |    295MiB / 15360MiB |      0%      Default |
     |                               |                      |                  N/A |
     +-------------------------------+----------------------+----------------------+

     +-----------------------------------------------------------------------------+
     | Processes:                                                                  |
     |  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
     |        ID   ID                                                   Usage      |
     |=============================================================================|
     +-----------------------------------------------------------------------------+
```

Fig: - System Specification

V. CONCLUSION

In conclusion, it is an effective computer vision technique that fuses object recognition with segmentation to precisely segment items in an input image. Mask RCNN can accurately recognise and separate objects in real-time by utilising pre-trained convolutional neural networks (CNNs) and region proposal networks (RPNs). The approach is applicable to many different fields, such as robotics, self-driving cars, and medical imaging. PyTorch is a good choice for implementing Mask RCNN and other deep learning algorithms due to its dynamic computational graph and user-friendly interface. In conclusion, Image Segmentation in PyTorch | Mask RCNN is a critical technique for empowering machines to more effectively comprehend and interpret visual input, which is a crucial step towards developing more intelligent and autonomous systems.

References

[1] Lin, K., Zhao, H., Lv, J., Li, C., Liu, X., Chen, R., & Zhao, R. (2020), "Face Detection and Segmentation Based on Improved Mask R-CNN" Discrete Dynamics in Nature and Society, pp 1–11, 2020.

[2] Han, Q., Yin, Q., Zheng, X. et al. Remote sensing image building detection method based on Mask R-CNN. Complex Intell. Syst. 8, 1847–1855 (2022)

[3] Zhang, L., Wu, J., Fan, Y., Gao, H., & Shao, Y. (2020). An Efficient Building Extraction Method from High Spatial Resolution Remote Sensing Images Based on Improved Mask R-CNN.

[4] Kaiming He, Georgia Gkioxari, Piotr Dollar, Ross Girshick; Mask R-CNN; Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017..

[5] Das, A. K., Nayak, J., Naik, B., Pati, S. K., & Pelusi, D. (Eds.). (2020). Computational Intelligence in Pattern Recognition. Advances in Intelligent Systems and Computing.

[6] Lee, H., Eum, S., & Kwon, H. (2019). ME R-CNN: Multi-Expert R-CNN for Object Detection. IEEE Transactions on Image Processing.