**Image Segmentation using Mask R-CNN on COCO-2017 Subset**

**Name: Aishwarya Sukumaran Student ID : 22058088**

**GitHub:** [**Link**](https://github.com/17031910/Research-Assignment/blob/main/Ass1_Image_segmentation.ipynb)

**Introduction**

This project focuses on segmenting four specific object classes—cake, car, dog, and person—from images using a deep learning model called Mask R-CNN. I used a subset of the COCO-2017 dataset and built a simple but effective pipeline in PyTorch. The goal was to train a model that can identify and draw masks around these objects and then visualize the results on test images. This task is useful in real-world situations like Wildlife Monitoring, Medical Imaging, and Automated Checkout.

**Literature Review**

Image segmentation is a crucial problem in computer vision that aims to precisely locate and delineate objects at the pixel level. Unlike object detection, which only draws bounding boxes, segmentation identifies the exact shape of each object.

**Ramesh and Kumar (2021)** reviewed the Mask R-CNN architecture, highlighting how it extends Faster R-CNN by adding a mask prediction branch for simultaneous detection and segmentation, making it a strong standard for instance segmentation tasks. **Anantharaman et al. (2018)** demonstrated the use of Mask R-CNN for detecting and segmenting oral diseases, showing its potential in medical imaging applications. More recently, **Sapkota et al. (2024)** compared Mask R-CNN with YOLOv8 for instance segmentation in complex orchard environments and found Mask R-CNN effective in handling varied and overlapping objects in natural settings.

From these studies, I selected Mask R-CNN with a ResNet-50 backbone because it is well-tested, comes pretrained on COCO, and fits well with my dataset and project goals.

**Exploratory Data Analysis (EDA)**

I started by analysing annotations using the COCO API, focusing on the distribution and size of the objects in the four target classes.

A graph of blue rectangular bars

AI-generated content may be incorrect.

**Figure 1: Images per Class**

A graph with orange bars

AI-generated content may be incorrect.

**Figure 2: Instances per Class**

A graph with numbers and a line

AI-generated content may be incorrect.

**Figure 3: Object Size Distribution**

1. **Images per Class (EDA 1):** The fig 1 bar chart shows how many images contain each class. We can see that 'car' and 'person' appear in most images, while 'cake' and 'dog' show up much less. This means some classes have a lot more examples than others.
2. **Instance Distribution per Class (EDA 2):** The fig 2 horizontal bar chart shows the total number of object instances for each class across all images. Like before, 'person' and 'car' have many more objects than 'cake' and 'dog', which again shows the classes are unbalanced.
3. **Object Size Distribution (EDA 3) :** Th3 fig 3 histogram shows how big the objects are, based on their bounding box sizes. Most objects are small, with fewer big ones. Having many small objects can make it harder for the model to detect them well.

**Methodology**

**Dataset**

I used a subset of COCO-2017. My dataset folders had:

* 300 training images + annotations
* 300 validation images + annotations
* 30 test images (unlabelled, used for inference only)

I wrote a custom data loader that picks only images with the four classes and prepares masks, boxes, and labels for training.

**Model**

I used the pretrained maskrcnn\_resnet50\_fpn model from torchvision, Mask R-CNN model that was already trained on COCO data. I changed it to detect the 4 classes I needed plus background. Using a pretrained model helped me train faster and get better results since training from scratch needs a lot of time and power.

**Training**  
I trained the model with these settings:

|  |  |
| --- | --- |
| **Training Parameter** | **Value** |
| Optimizer | SGD with momentum |
| Learning Rate | 0.001 |
| Batch Size (Training) | 2 |
| Batch Size (Validation) | 1 |
| Maximum Epochs | 10 |
| Early Stopping | Yes (patience = 3) |

**Table 1: Model Training Parameters**

I monitored validation loss after each epoch. If it didn’t improve for 3 epochs, training stopped early. Validation loss improved in the first 3 epochs, so I saved the best model then. When it stopped improving, early stopping triggered at epoch 6. The final model was saved after that. This helped the model learn well without overfitting or wasting time.

**Evaluation**  
I used training and validation loss to check how the model learned during training.

A graph with blue and orange lines

AI-generated content may be incorrect.

**Figure 4: Training and validation loss. Early stopping triggered at epoch 6.**

The Fig 4 graph shows training and validation losses over epochs. Both losses went down at first, with the best validation loss in the first three epochs. When validation loss stopped improving, early stopping stopped training at epoch 6 to prevent overfitting.

**Results and Discussion**

After training, I tested the model on 30 new images without labels. The model predicted objects and showed bounding boxes with class names and confidence scores above 0.72. If no confident objects were found, a message was shown. The visual results helped check how well the model worked on unseen data.

|  |  |  |  |
| --- | --- | --- | --- |
| **Image** | **Description** | **Detected Classes** | **Detection Accuracy (Confidence Scores)** |
| A person in a green shirt and tie  AI-generated content may be incorrect. | Two persons standing opposite sides | Person (2 instances) | 0.90, 0.92 |
| A person surfing on a wave  AI-generated content may be incorrect. | Person water skiing in the ocean | Person | 0.92 |
| A group of men walking on a red carpet  AI-generated content may be incorrect. | Group of people walking + car | Person (multiple), Car | Person: 0.91 to 0.96, Car: 0.73 |

**Table 2: Detection Results on Test Images**

**Critical Analysis**

The model worked really well for detecting people, with confidence scores above 0.90 in all test images as mentioned in the table 2. This is probably because there were more training examples for the ‘person’ class.

However, it didn’t do as well for other classes like ‘car’, which had a lower confidence score of 0.73. Classes like ‘dog’ and ‘cake’ were even harder to detect. This could be because there weren’t enough examples of those classes in the dataset. Also, smaller objects were sometimes missed or not segmented properly. To improve this, I could try adding more data for those classes, using data augmentation, or training the model for more epochs.

**Improvements**

* Train for more epochs to help the model learn better details.
* Add data augmentations like flips or crops to improve robustness.
* Use evaluation metrics like IoU or mAP for better performance measurement.
* Adjust the model or input resolution to better detect small objects.

**Conclusion**

The model gave good results for detecting the ‘person’ class, even in different scenes like standing, action, and group settings. It was able to find people with high accuracy. But for other classes like ‘car’, the results were not as strong. This shows the model is good with frequent and clear objects but needs some improvement for less common or smaller ones. With more balanced data and a bit more training, the overall performance can be made even better.

**References**

1. Anantharaman, R., Velazquez, M. and Lee, Y. (2018) ‘Utilizing Mask R-CNN for Detection and Segmentation of Oral Diseases’, *2018 IEEE International Conference on Healthcare Informatics*, pp. 249–254. Available at: <https://ieeexplore.ieee.org/document/8621112>.
2. Ramesh, S.C. and Kumar, V. (2021) ‘A Review on Instance Segmentation Using Mask R-CNN’, *SSRN Electronic Journal*. Available at: <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3794272>.
3. Sapkota, R., Ahmed, D. and Karkee, M. (2024) ‘Comparing YOLOv8 and Mask R-CNN for instance segmentation in complex orchard environments’, *Computers and Electronics in Agriculture*, 213, p. 107833. Available at: <https://www.sciencedirect.com/science/article/pii/S258972172400028X>.