INM705 Deep Learning for Image Analysis

MEHMET BARIS CEKIC / 200007494
SELIM BAYRAK / 200056225

@cekicbaris
@scbayrak

Table of Contents

1. Introduction

1.1. Problem Definition

The objective of this work is to perform instance segmentation on the Microsoft COCO dataset. A few classes will be selected from the dataset and the pre-trained Mask R-CNN model will be fine-tuned to detect objects of this class with instance segmentation. This segmentation will produce pixel-wise masks for each object detected in the image where each object is annotated with a different color. The detection performance will be evaluated with the suitable metrics as will be explained in the next sections.

1.2. Background

The development of the image recognition domain can be viewed as a progression from coarse to fine image inference. The bottom step in progression starts from image classification where one category class is assigned to an image. The next step is image tagging where more than one object in the image is recognized. One step further is object detection where objects are recognized and located in the image with a bounding box. The next step in the progression is semantic segmentation which assigns labels for each image pixel. The next step in the progression is instance segmentation where different labels are assigned to separate instances of objects belonging to the same object class. Instance segmentation can be viewed as solving object detection simultaneously with semantic segmentation. The next step in the continuing progression is part-based segmentation where segmented objects are decomposed into their respective sub-components.

Before being popularized by COCO, instance segmentation was introduced by Hariharan et. Al. (Hariharan B, 2014). The technique included mask proposal generation followed by generated by classification of the proposals. With deep learning becoming more popular, new techniques with a more efficient structure emerged such as RCNN (Girshick, et al., 2014). Despite its accuracy, RCNN suffered from low speed and optimization problems. Fast RCNN (Girshick, 2015) and Faster RCNN (Ren, et al., 2016) came along to resolve these problems.

The popular approach for instance segmentation involves object detection using a box followed by object-box segmentation (Abdul Mueed Hafiz, 2020). Mask RCNN is one of the most successful techniques that use this approach by employing a mask detector (He, et al., 2018). Other techniques use sliding-window techniques (Lin T, 2017) and region-based techniques (Ren, et al., 2016).

A different approach for instance segmentation employs methods for semantic segmentation. This approach labels every image pixel categorically after which pixels are grouped into object instances with a clustering algorithm. This approach has a lower accuracy on popular benchmarks and requires more computation.

Another approach uses dense sliding window methods. This category includes methods such as DeepMask (Pinheiro PO, 2015) and InstanceFCN (Dai J, 2016). They use CNNs for mask proposal generation. Tensormask (Chen X, 2019) belongs to the same category but uses a different architecture to perform classification in parallel to predicting masks. DeepMask and InstanceFCN cannot perform classification for multiple classes. TensorMask

possesses this capability and performs decently on datasets such as COCO. However, it has a complicated algorithm.

1.3. Dataset

We used the Microsoft COCO dataset in our CW. MS COCO is a large-scale object detection, segmentation, and captioning dataset. It has more than 330K images, 1,5M objects instances, 80 object categories, 91 stuff categories, 5 caption per image, 250,000 people with keypoints. (Lin, et al., 2017)

1.4. Data Pre-processing

We use data loaders to call input images and labels in batches for training and prediction. Pycocotools is the main tool used to load the annotations for ground truths.

COCO.getCatIds: The chosen category names are entered as arguments; it generates the category ids.
COCO.getImgIds: Category ids generated from the previous step are provided as arguments, it generates the ids of the images belonging to the selected categories.
COCO.loadImgs: Loads the images with the image ids from the previous step.
COCO.getAnnIds: Generates the annotation ids from image and category ids.
COCO.loadAnns: Annotation ids entered as arguments, loads the GT annotations.

To create the ground truth, from the annotations, we extract the category ids, bounding boxes, and masks. Bounding boxes come in format [x, y, width, height]. We convert these to [x1, y1, x2, y2] format. COCO.annToMask creates the mask from the annotation. We convert these to tensors and save them in a dictionary called image_labels. This dictionary is returned as the ground truth.

While we load the image as the input to the network, we use torchvision.tranforms.Normalize() to standardize it with mean 0 and standard deviation of 1 to help our network learn faster.

2. Methodology

We studied object detection and instance segmentation algorithms when working on this CW. The difference between image classification and object detection algorithms is that, in detection algorithm, we try to draw a bounding box around the object in the image and try to locate its region (Girshick, et al., 2014). In the case of more than one object in the image, then we try to draw more than one bounding box, one for each object. It is different than the Convolutional Neural Networks because there is no fixed output in the network. So, to overcome this problem, one can try to take different regions of the image and try to classify those regions or use a sliding window over the image. This naïve approach may lead to a huge number of regions and it becomes computationally inefficient. The second challenge is to detect the shape of the object inside the bounding box. This is what instance segmentation does. It creates a pixel-wise mask for each object.

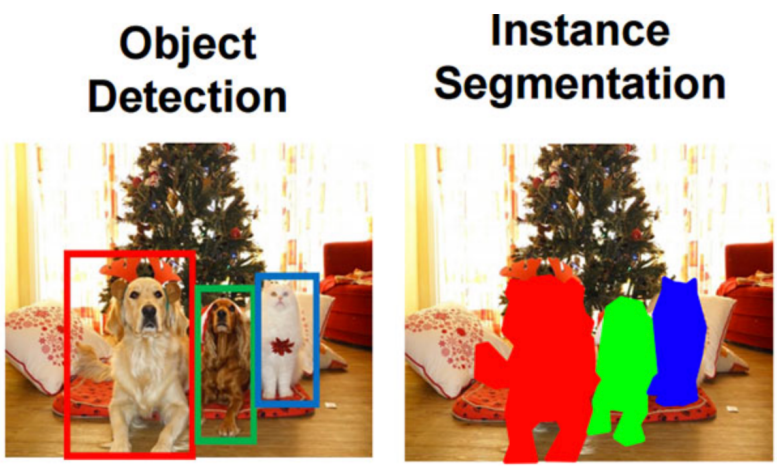


Figure 1 – Object Detection vs Instance Segmentation

We will first discuss object detection algorithms such as R-CNN , Fast R-CNN, Faster R-CNN, and lastly Mask R-CNN for instance segmentation. In our implementation, we used Mask R-CNN to predict bounding boxes and masks for the objects.

2.1. R-CNN

This algorithm is developed by Girshick, et al. in 2014. They suggested a paradigm called "recognition using regions" to solve the localization problem in CNN which solves both object detection and semantic segmentation. (Girshick, et al., 2014) Only 2000 regions were selected, so-called region proposals, by the selective search algorithm. This proposal reduces the total number of regions in the image. This approach extracts a fixed length, 4096-dimensional feature vector from each proposal by using CNN(Convolutional Neural Network) (Girshick, 2015) and classifies each region with a category-specific SVM (Support Vector Machine). It also predicts 4 values for the bounding box.

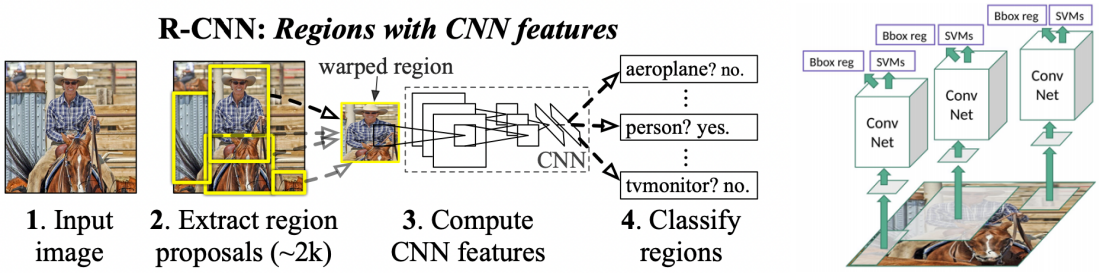


Figure 2 – R-CNN architecture

It has some disadvantages such as R-CNN is slow, the selective search algorithm is fixed, there is no learning which could lead to bad region candidate proposal generation. (Gandhi, 2018). It forward passes 2000 proposals for every image which is slow. It also needs 3 different models trained for feature extraction, classification, and bounding box regression, so it is costly. (Girshick, 2015)

2.2. Fast R-CNN

Fast R-CNN model is proposed by Ross Girshick to address the drawbacks of R-CNN models. Girshick suggested that using a single network instead of three different networks for feature extraction, classification, and bbox regression, speeds up the learning and

inference. Secondly, instead of the forward passing of 2000 proposals, one pass approximation is suggested. (Girshick, 2015)

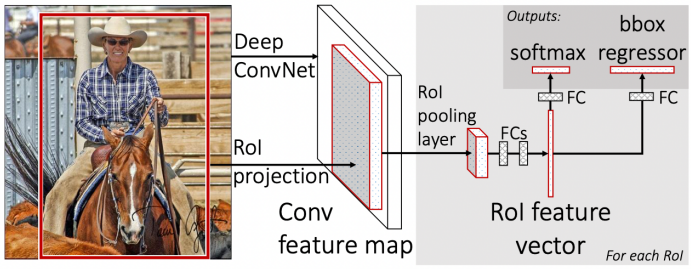


Figure 3 – Fast R-CNN Architecture

2.3. Faster R-CNN

Faster R-CNN introduces a concept called RPN (Region Proposal Network) that shares full-image convolutional features with the detection network. By doing so Ren, et al. claim that this enables nearly cost-free region proposals. (Ren, et al., 2016). As defined by Ren, et.al RPNs are fully convolutional networks that simultaneously predict object bounds and object scores at each position. Regional proposals generated by RPNs are used by Fast-RCNN for detection. And later RPNs and Fast RCNN are merged into a single network with an attention mechanism.

Faster R-CNN consists of two stages. The first stage, called a Region Proposal Network, proposes candidate object bounding boxes. The second stage extract features from each candidate box using RoI Pooling from each candidate box and performs classification and bounding-box regression as described in the Fast RCNN section.

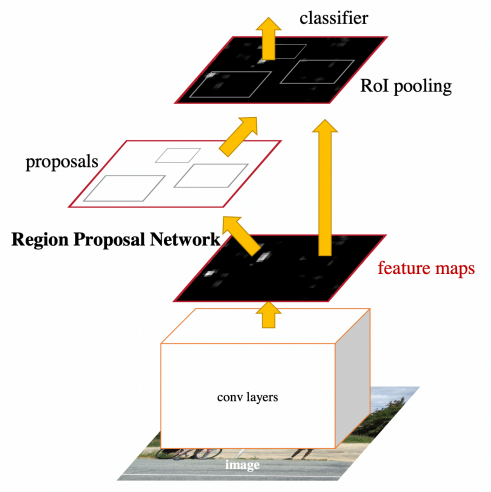


Figure 4 Faster RCNN Architecture

2.4. Mask R-CNN

Mask R-CNN is a deep neural network architecture designed to solve object/instance segmentation problems. It extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. (He, et al., 2018)

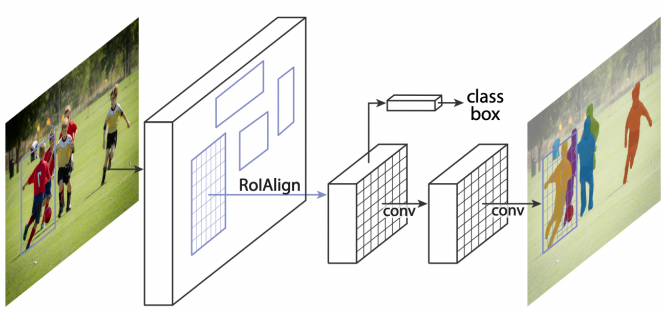


Figure 5 Masked R-CNN Architecture

Similar to Faster R-CNN, Masked R-CNN has also 2 stages, first, it generates proposals where there might be an object, the second stage predicts the object class, refines the bounding box, and generates a mask in pixel level. Both stages use the same backbone structure. The backbone is an FPN style deep neural network.

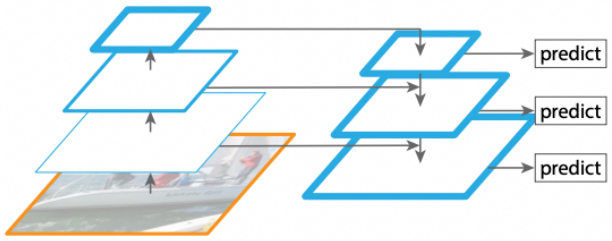


Figure 6 – Feature Pyramid Network

FPNs are deep neural network architectures that consist of the bottom-up pathway which is mostly ResNet or VGG to extract features from images, a top-down pathway that hallucinates higher resolution features by upsampling spatially coarser, but semantically stronger feature maps from higher pyramid levels, and lateral connections which are convolutions between two pathways.

Faster R-CNN uses RoI(Region of Interest) Pooling as mentioned in the earlier section, whereas Masked RCNN uses RoI Alignment which preserves spatial pixel to pixel alignment and prevents being lost as it was in RoI Pooling.

Finally, in the additional branch, an FCN (Fully Convolutional Network) generates the pixel-by-pixel masks for the image.

2.5. Accuracy Metrics

As the accuracy metrics, we have used Mean Average Precision (MAP) and Mean Average Recall (MAR).

We can start by defining precision and recall.

$$precision = \frac{tp}{tp + fp} \quad (1)$$

$$recall = \frac{tp}{tp + fn} \quad (2)$$

tp: True-positive fp: False-positive fn: False-negative

False-positive is the percentage of correct positive predictions among all predictions made, and recall is the percentage of correct positive predictions among all positive cases in reality. There is a compromise between 2 metrics where improving one is often at the expense of the other metric.

2.5.1. Mean Average Precision (MAP)

We can move one step further and define average precision (AP). This metric is the area under the precision-recall curve seen in Figure 7. The curve is created by plotting precision against recall at different confidence score thresholds. The confidence score is the probability that an anchor box contains an object. As the confidence threshold decreases, more predictions are accepted as positives. With this recall monotonically increases and precision follows a general trend of decrease with fluctuations. For calculating AP, if the curve is not monotonically decreasing, we smooth out the curve by setting the precision to maximum precision obtained for any recall value larger than the recall value at that point. In Figure 7, the green line shows the actual curve whereas the red lines show the smoothed curve.

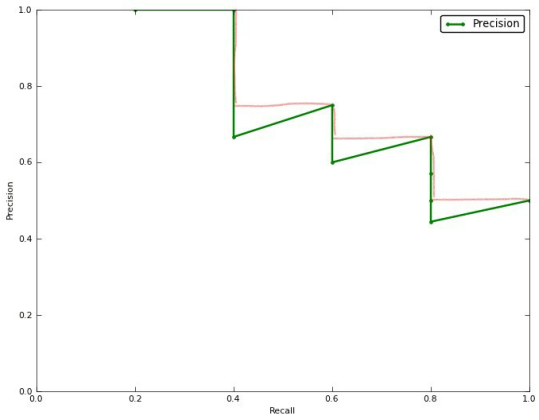


Figure 7 - Precision-Recall Curve (Liu, 2018)

The next concept that needs to be reviewed for understanding MAP is intersection over union (IOU). It is the ratio of the area of overlap to the area of union. For instance, segmentation, it is calculated between the masks of prediction and ground truth.

$$IOU = \frac{Area\ of\ overlap}{Area\ of\ union} \quad (3)$$

COCO dataset defines average precision at different IOU values as below.

AP: AP at IoU=.50:.05:.95 (primary challenge metric)
$AP^{IOU=.50}$: AP at IoU=.50 (PASCAL VOC metric)
$AP^{IOU=.75}$ AP at IoU=.75 (strict metric)

IOU=0.5 accepts a mask overlap above this value as a true positive whereas the strict metric, IOU=0.75 accepts an overlap above this value as a true positive.

Similar to how different score threshold values affect the precision-recall curve, the IOU threshold affects the precision-recall curve. Decreasing the IOU threshold for accepting predictions as positive increases the recall and decreases precision per the example below.

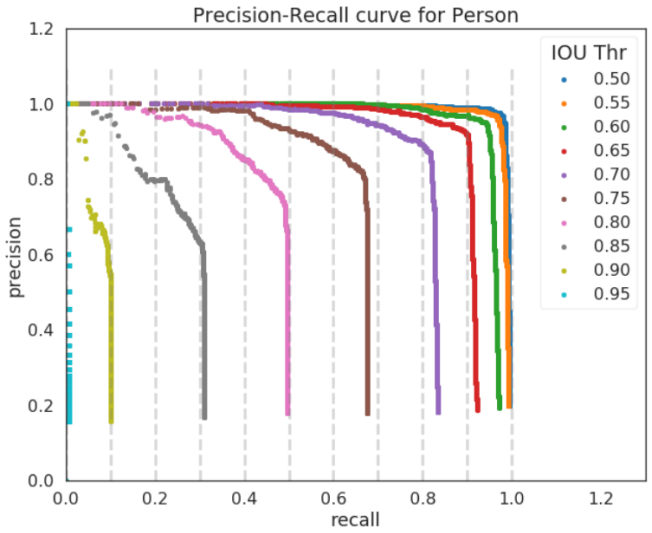


Figure 8 - Example Precision-Recall curves for different IOUs (Arlen, 2018)

Finally, to calculate MAP, AP values are averaged over all IOU values. For the MS COCO dataset, AP is averaged over 10 IOU values from 0.5 to 0.95 at 0.05 increments.

2.5.2. Mean Average Recall (MAR)

For detection proposals, it is important to have good coverage of the objects of interest in the image, since missed objects cannot be recovered in the subsequent classification stage (Hosang, et al., 2016).

In contrast to AP calculation, for computing Average Recall (AR), confidence scores of predictions aren't taken into account. All predictions are accepted as positive ones, effectively taking the confidence threshold as zero. In general, AR is defined as twice the area under the Recall-IOU curve.
For the COCO dataset, MAR is calculated as an average over 10 IOU values from 0.5 to 0.95 at 0.05 increments similar to MAP calculation. For this dataset, AR is defined as the maximum recall given a fixed number of detections per image, averaged over categories and IoUs. The number of detections is taken as 1, 10, or 100.

$AR^{max=1}$: AR given 1 detection per image
$AR^{max=10}$: AR given 10 detections per image
$AR^{max=100}$: AR given 100 detections per image

For COCO dataset the calculation can be given as below:

$$AR = \frac{1}{O}\sum_{o=1}^{O} max\ \{R_t(o)(\tau(k))\}\ where\ \{k \mid P_{t(o)}(\tau(k)) > 0\} \quad (4)$$

$P_{t(o)}$ and $R_t(o)$ are the precision x recall points for a confidence $\tau(k)$, with IOU threshold $t(o)$. This equation corresponds to the average of the largest recall values such that the precision is greater than zero for each IOU threshold (Rafael Padilla, 2021).

2.6. Implementation

2.6.1. Classes, Functions and Notebooks

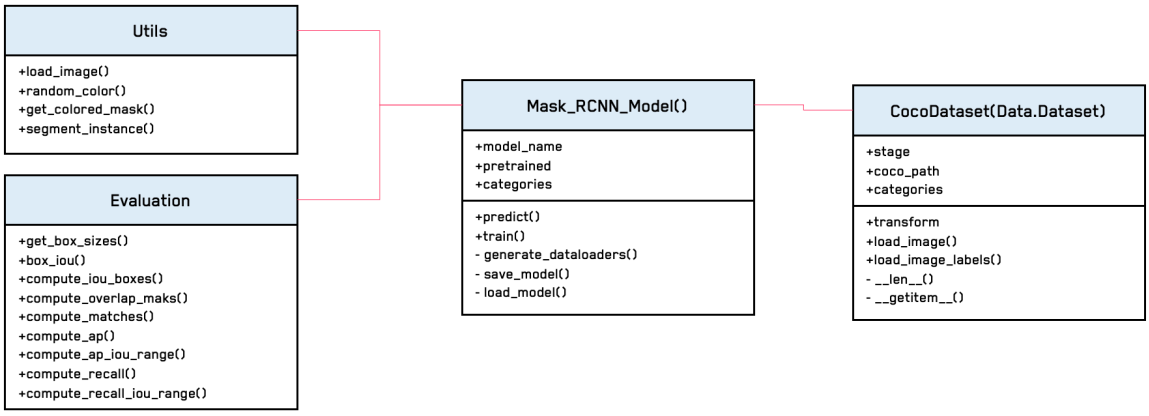


Figure 9 Implementation Classes and Functions

Mask_RCNN_Model : This class is the main class where model training and prediction are implemented. It uses the "**maskrcnn_resnet50_fpn**" model from the torch library and depending on the hyperparameters, it may run training from scratch, load model parameters from a *.pth file that's already saved in earlier trainings, or use the pretraining weights for inference.

CocoDataset: This class implements **data.dataset** class from torch library and it uses pycocotools to get categories, images, and image labels. It also transforms the image from the COCO library to tensors.

Evaluation: Evolution is a collection of functions that are used to calculate **mAP** and **mAR** metrics mentioned in section 2.5.

Utils: Utils is a collection of helper functions to load the image, draw the bounding box, and generate the mask.

2.6.2. Evaluation Metric Implementations

Our implementation is different from the standard COCO method of calculating MAP and MAR. We have adapted the code from Matterport's implementation on (github.com/matterport/Mask_RCNN/blob/master/mrcnn/utils.py#L715)

2.7. Customization

In our implementation, bounding box thickness changes with the confidence score where the thickness of the individual boxes are computed with the formula below:

```
box_thickness = math.ceil((scores[i]-confidence)/0.1)
```

According to this when the confidence threshold is 0.5, the thicknesses will be as below for different ranges:

1 – 0.9 : 5, **0.9 – 0.8** : 4, **0.9 – 0.8 :** 4, **0.8 – 0.7 :** 3, **0.7 – 0.6 :** 2, **0.6 – 0.5 :** 1

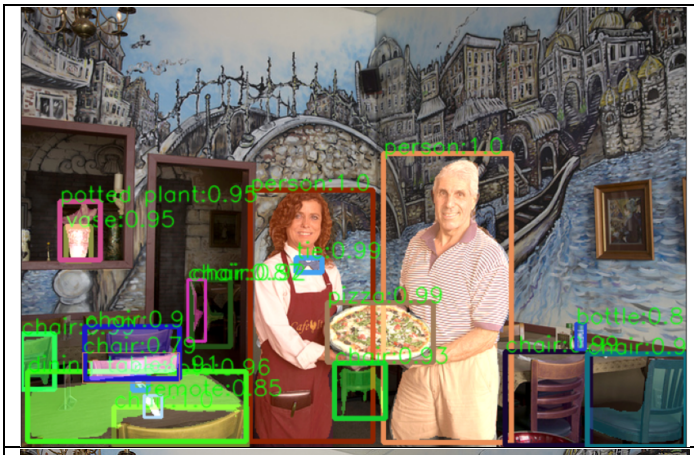
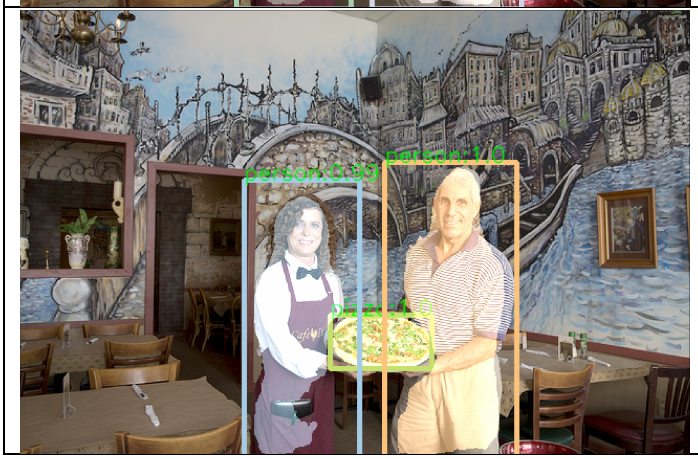
Besides, confidence scores are written with text above the boxes. Also, the bounding boxes are drawn with the same color as the mask colors.

2.8. Model Training and Results

In our model training and inference process, we use the following scenarios.

1. Use pretrained Mask R-CNN parameters from PyTorch library.
2. Train the model from scratch for the ['person', 'chair', 'bottle', 'pizza'] categories.
    a. Loss, mAP, mAR, Inference and Confidence Score after 1 epoch
    b. Loss, mAP, mAR, Inference and Confidence Score after 5 epochs
    c. Loss, mAP, mAR, Inference and Confidence Score after 10 epochs
    d. Loss, mAP, mAR, Inference and Confidence Score after 15 epochs
    e. Loss, mAP, mAR, Inference and Confidence Score after 20 epochs

The test results are as follows

	• Pretrained Weights from Pytorch • Confidence high (>0.9) for selected categories. • Masks are properly selected.
	• 1 epoch training • Confidence is not very high and wrong detections exist. There is no person with 0.77 in the image. • Masks are not properly selected. • mAP: 0.209 • mAR: 0.270 • Loss: 1.228369
	• 5 epochs training • Confidence is high (>0.8) and no wrong detection exists. • Masks are not properly selected. There are distortions in man's neck, arm, and over pants, and gaps on women cook's apron. • mAP: 0.274 • mAR: 0.334 • Loss: 0.798
	• 10 epochs training • Confidence is high (>0.9) and no wrong detection exists. • Masks are getting better but not as good as the pre-trained model yet. The distortions are lighter compared to 5 epochs • mAP: 0.308 • mAR: 0.360 • Loss : 0.558520

	• 15 epochs • Confidence is high (>0.98) and no wrong detection exists. • The masks are getting better. • mAP: 0.378 • mAR: 0.427 • Loss: 0.4374
	• 20 epochs • Confidence is high (>0.98) and no wrong detection exists. • Masks are not getting better comparing to epoch 15. • mAP: 0.374 • mAR: 0.423 • Loss: 0.3652

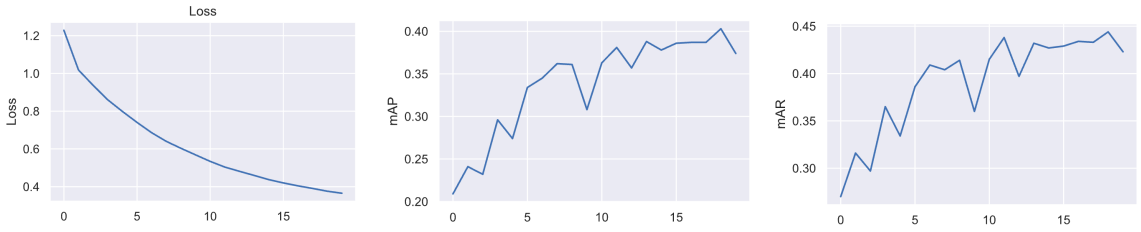


Figure 10 – Loss, mAP and mAR over training epochs

We both run inference on CPU and GPU. GPU is by far faster than CPU naturally as seen in figure 11.

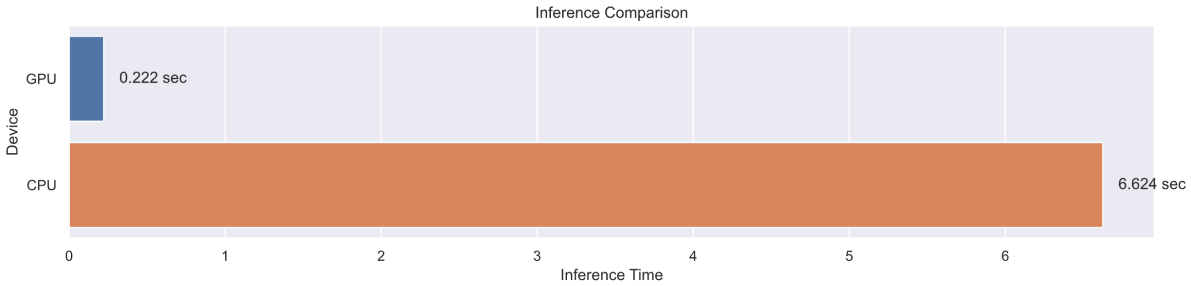


Figure 11 – Inference Comparison between CPU and GPU

3. Conclusion

The pre-trained model is successful at recognizing the selected category of objects with an accuracy above 0.9 in addition to the other categories. For training from scratch, after 15 epochs, the model can recognize the selected categories with 100% confidence and reasonably accurate masks in the selected image. The mAP and mAR results are found satisfactory after this number of epochs when the highest mAP and mAR scores on the Detection Leaderboard of the COCO dataset are 0.588 and 0.700 respectively at the time of writing this report. When we consider the models in the leaderboard are state-of-the-art models with various fine-tuning, the performance of Mask RCNN is good as an out-of-the-box model with no tweaks or fine-tuning. Only 15 epochs were sufficient for the model to perform well on the example image. Further epochs and fine-tuning can further increase the mask accuracy.

Instance segmentation is an advanced method of image recognition domain that solves object detection simultaneously with semantic segmentation. Our study shows that Mask R-CNN is successful at instance segmentation producing good accuracy. However, it is not sufficiently fast to use for real-time image recognition. As future work, one may consider working on YOLO algorithms for real-time image recognition.

4. Reflections

The evaluation metrics are not as straightforward as calculating simple precision or recall. Instance segmentation doesn't have a fully defined standard of metrics. Different datasets such as COCO and Pascal have slightly different approaches for Mean Average Precision and Mean Average recall. These metrics for COCO are not described in a big amount of detail on the website. Due to this, it was required to do research and consult to different papers to understand the evaluation methodology in full. It was found that different interpretations of the metrics exist. While conducting a detailed study of the correct COCO metrics and sharing the information in the report, we have chosen to implement an interpretation by Matterport Inc. due to its simplicity and the time restrictions of this project. As a further study, a pure implementation of the metrics as advised by the COCO dataset could be proposed. Another aspect is the model that has been evaluated which was taken as out-of-the-box. A further interesting project could be to perform tweaks to the model structure to see test much the performance could be increased.

5. Table of Figures

6. Works Cited

Abdul Mueed Hafiz, G. M. B., 2020. A Survey on Instance Segmentation: State of the art. *International Journal of Multimedia Information Retrieval,* 9(2192-662X), p. 171–189.

Arlen, T. C., 2018. *Understanding the mAP Evaluation Metric for Object Detection,* s.l.: www.medium.com.

Chen X, G. R. H. K. D. P., 2019. A Foundation for Dense Object Segmentation. *arXiv preprint arXiv:190312174.*

Dai J, H. K. L. Y. R. S. S. J., 2016. Instance-sensitive fully convolutional networks. *European Conference on Computer Vision,* pp. 534-549.

Gandhi, R., 2018. *R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms.* [Online]
Available at: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e
[Accessed 2021].

Girshick, R., 2015. Fast R-CNN.

Girshick, R., Donahue, J., Darrell, T. & Malik, J., 2014. *Rich feature hierarchies for accurate object detection and semantic segmentation.* [Online]
Available at: https://arxiv.org/pdf/1311.2524.pdf
[Accessed 2021].

Hariharan B, A. P. G. R. M. J., 2014. Simultaneous detection and segmentation.. *European Conference on Computer Vision*.

He, K., Gkioxari, G., Dollar, P. & Girshick, R., 2018. Mask R-CNN.

Hosang, J., Benenson, R., Dollár, P. & Schiele, B., 2016. What Makes for Effective Detection Proposals?. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 38(4), pp. 814 - 830.

Lin T, G. P. G. R. H. K. D. P., 2017. Focal Loss for Dense Object Detection. *2017 IEEE International Conference on Computer Vision (ICCV),* pp. 2999-3007.

Lin, T.-Y.et al., 2017. Feature Pyramid Networks for Object Detection.

Lin, T.-Y.et al., 2015. Microsoft COCO: Common Objects in Context.

Liu, Y., 2018. *The Confusing Metrics of AP and mAP for Object Detection / Instance Segmentation,* s.l.: www.medium.com.

Pinheiro PO, C. R. D. P., 2015. Learning to Segment Object Candidates. pp. 1990--1998.

Rafael Padilla, W. L. P. T. L. B. D. S. L. N. E. A. B. d. S., 2021. A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit. *Electronics,* 10(3).

Ren, S., He, K., Girshick, R. & Sun, J., 2016. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.