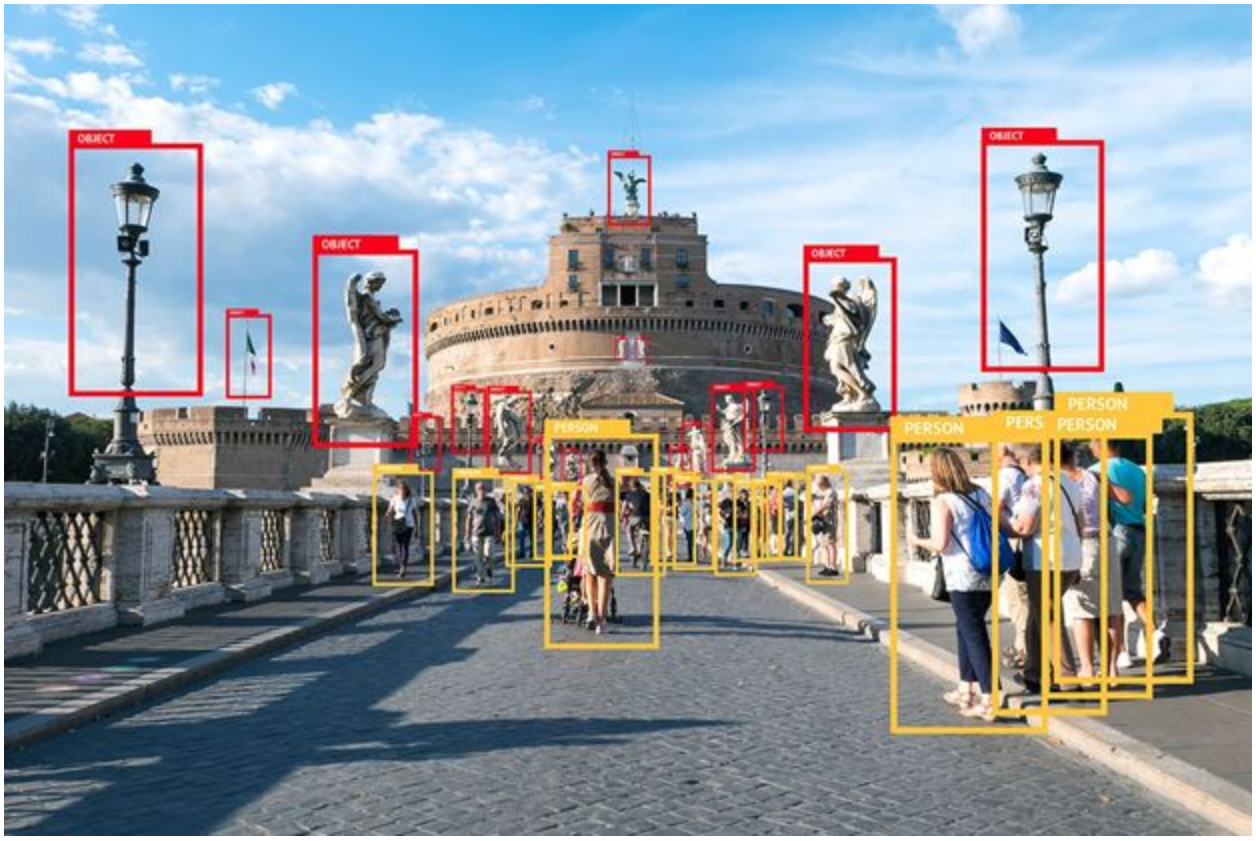
Object Detection and Segmentation

Summary	Object Detection and Segmentation
Document URL	https://github.com/rohithnagabhyrava/FinalProject_INFO7374
Application URL	http://instancesegmentation.us-east-2.elasticbeanstalk.com/
Author	Team 10: Rama Krishna Kommineni, Rohith Nagabhyrava, Umair Akthar

Introduction

What is object detection

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class in digital images.



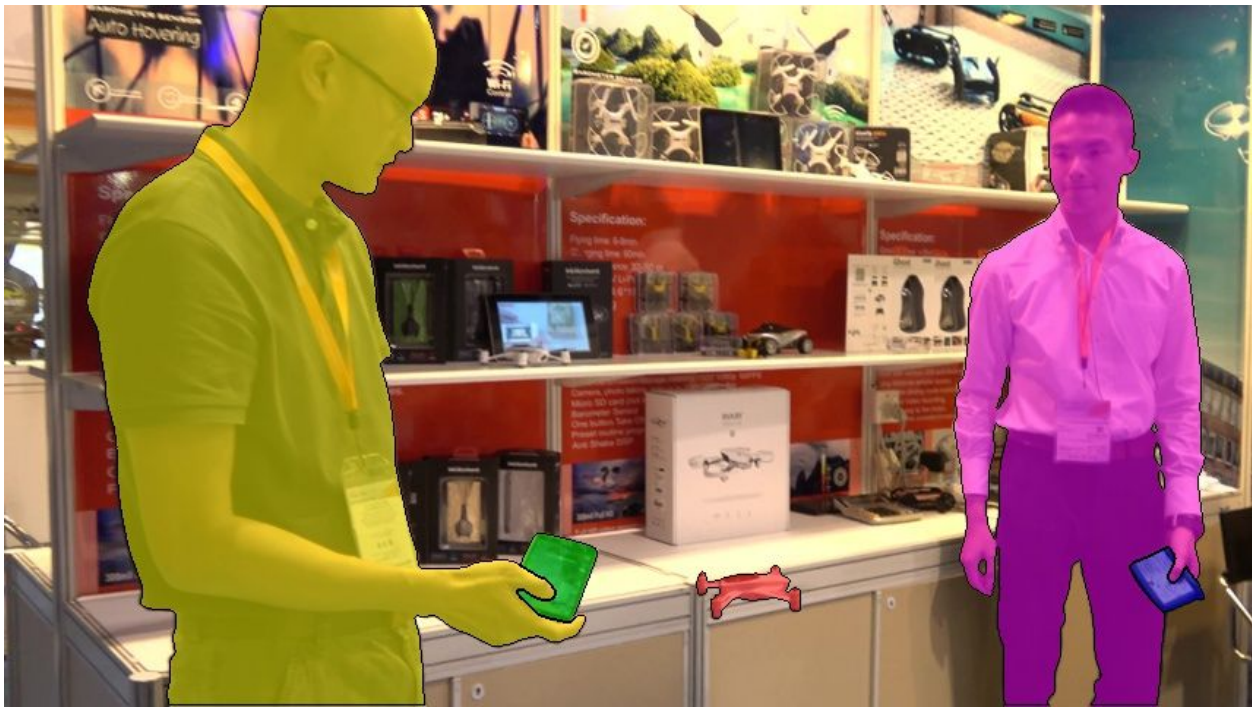
Source: LINK

Why object detection

The goal of object detection is to detect all instances of objects from a known class, such as people, cars or faces in an image. Typically, only a small number of instances of the object are present in the image, but there is a very large number of possible locations and scales at which they can occur and that need to somehow be explored.

What is Object Segmentation

In computer vision, **image segmentation** is the process of partitioning a digital image into multiple segments (sets of pixels, also known as super-pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics.



Source: LINK

What is Instance Segmentation

Instance segmentation is the combination of both Object detection and Object Segmentation. Given an image, it draws the bounding boxes around the objects and also performs the segmentation of the image by partitioning the objects in the image separately.

Source: LINK

Project goals

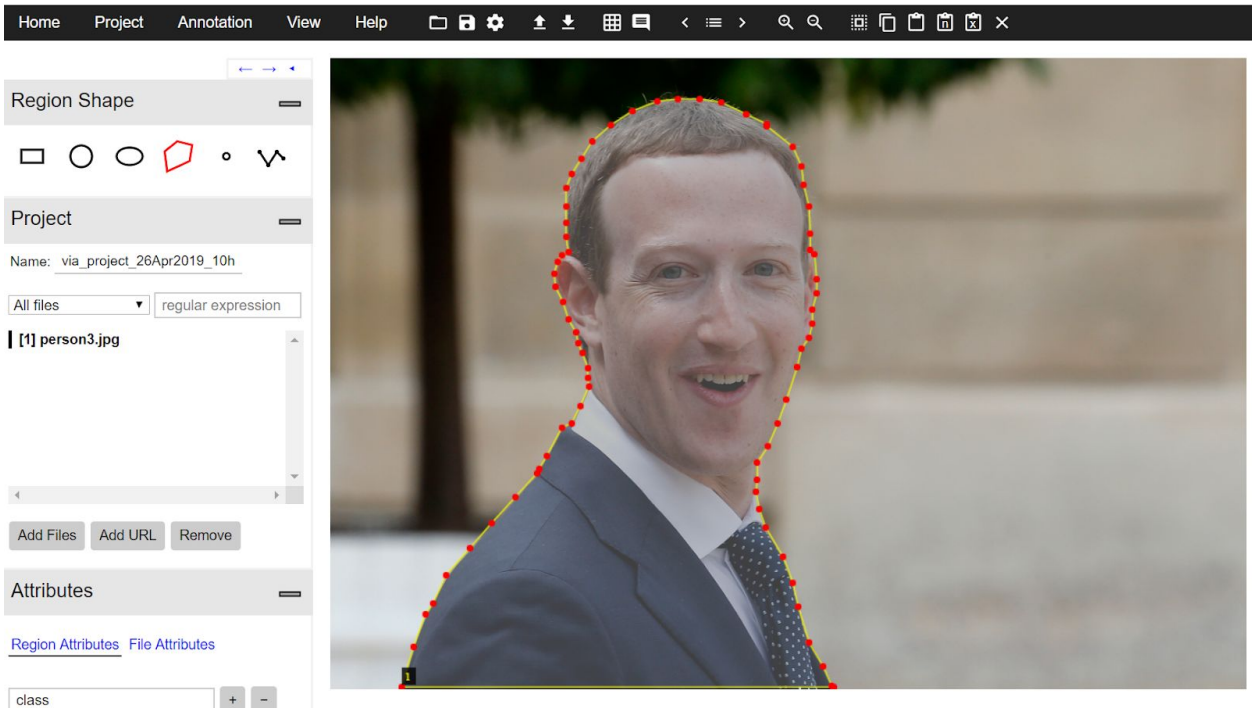
To create an Instance segmentation model which detects only the objects which the user wants in a given image. In this case, we trained our model to detect persons, cats and dogs in the given image.

In this project, we explained and showed how to create a custom dataset of objects which we want to detect in the image and the training process involved.

Data and Data Preprocessing

Experiments were conducted by using coco unlabeled image dataset released in 2017. As we want to detect only specific objects like person, cat and dog, We extracted 50 images of each class for training purpose and 20 images of each class for testing.
Link: http://cocodataset.org/#download

After the images are collected we used VGG IMAGE ANNOTATOR tool to generate the annotations of the images and saved it to the JSON file



JSON


```
{"person3.jpg1773955":{
        "filename":"person3.jpg",
        "size":1773955,
        "regions":[{
                "shape_attributes":
{"name":"polygon","all_points_x":
[226,264,302,326,367,442,511,586,655,662,686,734,764,792,819,816,816,799,785,778,754,
7,713,710,720,734,754,747,747,747,747,764,795,830,888,956,1035,1100,1169,1237,1316,13
,1381,1464,1491,1498,1515,1519,1519,1532,1539,1539,1525,1525,1515,1491,1477,1443,1416
385,1351,1351,1347,1357,1378,1433,1464,1481,1515,1560,1587,1594],"all_points_y":
[1988,1861,1759,1724,1635,1549,1471,1388,1313,1299,1258,1169,1155,1100,1039,1011,980,
2,902,867,826,771,723,682,644,624,614,566,521,470,411,367,319,264,213,168,137,130,130
41,178,209,216,281,343,401,470,555,607,620,699,744,795,840,884,919,977,1080,1155,1227
279,1333,1371,1426,1484,1577,1659,1735,1848,1923,1985,1988]},"region_attributes":
{"class":"1"}}],"file_attributes":{}}}
```


The JSON file consists of the coordinate details of all the points of the segmented image and the associated class of the segmented image as shown in the above figure.

Modelling and Implementation

Training Setup

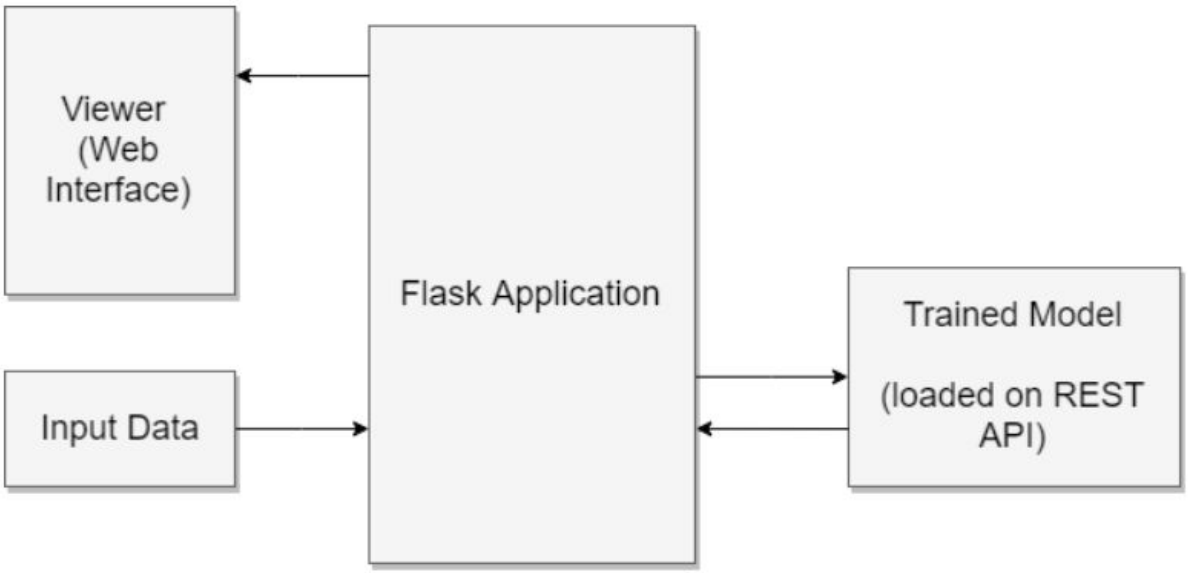
- Training Platform: AWS t3.xlarge, m5.4xlarge for training and Google colab for visualizing and testing the trained model.
- Language and framework: Python, keras with tensorflow backend.
- Evaluation: Loss = sum(box regression losses) / count(sampled anchors)

We first trained different models and ran to 5 epochs. Then took the best model out of these and trained that to 25 epochs.

Application

- Web Application: Flask
- Deployed on: AWS Beanstalk
- Programming languages: Python and HTML

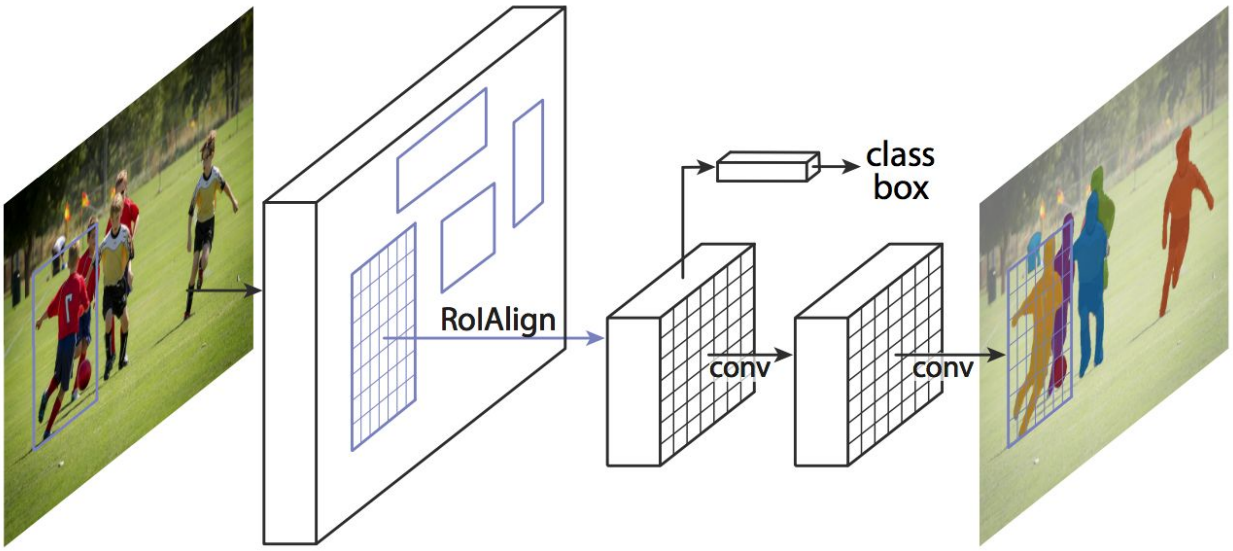
Pipeline Design



- The web application inputs and image.
- Preprocess and sends its to the model.
- Model detects the instances and returns a segmented image as output.

Models and Analysis

The method, called Mask R-CNN, extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5 fps.



Source:
https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761e238b46

ResNet50 or ResNet101, two standard convolutional neural network serves as feature extractor. The early layers detect low level features (edges and corners), and later layers successively detect higher level features (car, person, sky).

Backbone	Training	Epoch	Weights	Train Loss	Val Loss
Resnet101	Heads	1	coco	1.2885	1.2503
		5		0.4734	0.802
	4+ Layers	1		0.9323	1.2124
		5		0.3678	0.8032
	Heads	1	imagenet	3.0743	3.5848
		5		2.0718	2.6951
	Heads	1	coco	1.8895	2.2148
		5		0.9649	1.4239

Resnet50		1	imagenet	2.2133	3.3923
		5		2.203	2.4935
	4+ Layers	1		1.9194	2.3239
		5		1.8991	2.2711

We choose the best model from this and trained that to 25 epochs we observed best train and val loss at 20th epoch. We choose this as our best model.

Details on how to run the model

1. Download the flask application repository to your local device.
2. In Terminal, cd to folder where the repository has been downloaded.
3. Create an environment for downloading dependencies, conda create -n yourenvname
4. Switch to env conda activate yourenvname
5. To install dependencies on your local machine, run pip install -r requirements.txt
6. Run python application.py

Or you can view the running app here :
http://instancesegmentation.us-east-2.elasticbeanstalk.com

References

- Data: http://cocodataset.org/#download
- Annotations tool: http://www.robots.ox.ac.uk/~vgg/software/via/
- MaskRCNN: https://github.com/matterport/Mask_RCNN
- Paper: https://arxiv.org/pdf/1703.06870.pdf
- AWS : https://console.aws.amazon.com/elasticbeanstalk
- Flask Application: https://github.com/anagar20/Resnet-Image-Classification-Flask-App/blob/master/app.py