PROJECT REPORT

HUMAN DETECTION IN FLOODS USING MASK-RCNN

Made By: Vibhor Singh

TABLE OF CONTENTS

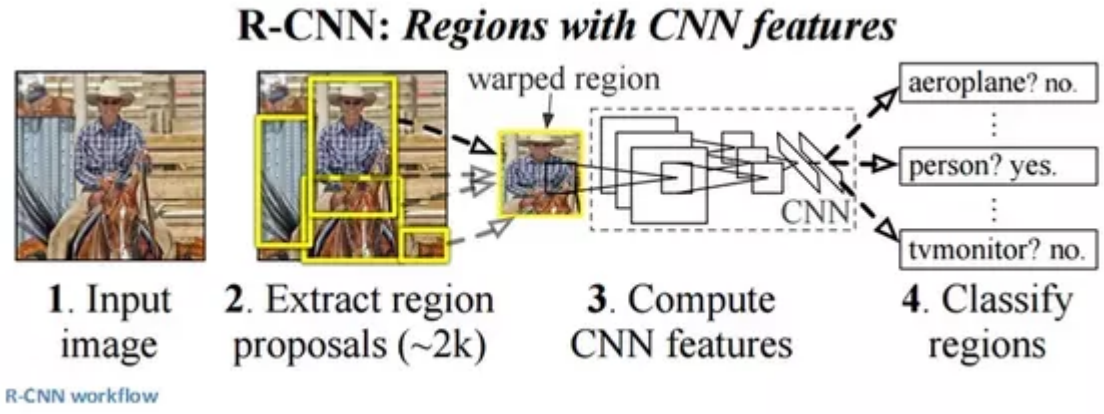
PROBLEM STATEMENT

Floods are a powerful force of nature and can cause different disaster situations like collapsed buildings, landslides etc. During these emergency situations, and specially in urban disaster areas, many different people are deployed (policeman, fire fighters, medical assistance, army etc.). They need to cooperate with each other in order to save lives, protect structural infrastructure, and evacuate victims to safety as efficiently as possible. In these situations, human rescuers must make quick decisions under stress, and try to get maximum victims to safety often at their own risk. They must gather data and determine the location and status of victims as quickly as possible so that the rescue force can enter the flooded area and save victims. All of these tasks are performed mostly by humans, and often are very dangerous situations and moreover it is important to allocate human resources judiciously. This is why since some years, mobile robots have been proposed to help them and to perform tasks that humans nor existing tools can do. In the past, there have been development of robots like the one developed by USAR, but these have become primitive now and there has been very minimal thought given to upgrading systems like these. So, with this project, the aim is to use a state-of-the-art algorithm, Mask-RCNN, to detect humans and check its viability as if it is possible to use it in real life flood situations.

INTRODUCTION

1.1 RCNN (Region Based Convolution Neural Network)

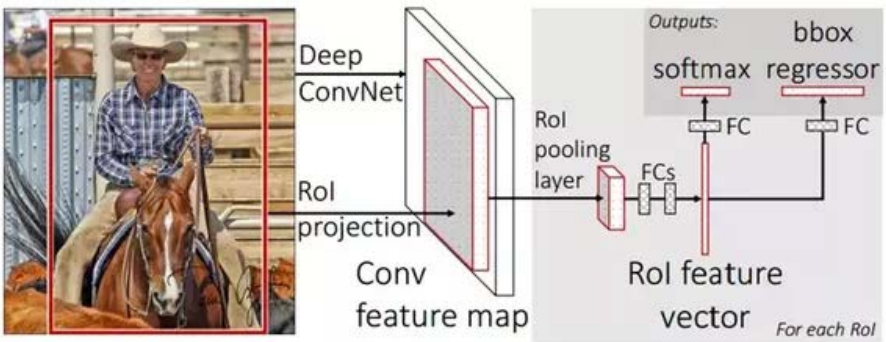
The purpose of R-CNNs (Region Based Convolution Neural Network) is to solve the problem of object detection. Given a certain image, we want to be able to draw bounding boxes over all of the objects. The process can be split into two general components, the region proposal step and the classification step.



R-CNN workflow

1.2 FAST RCNN

Fast R-CNN was able to solve the problem of speed by basically sharing computation of the convolution layers between different proposals and swapping the order of generating region proposals and running the CNN. In this model, the image is first fed through a ConvNet, features of the region proposals are obtained from the last feature map of the ConvNet.

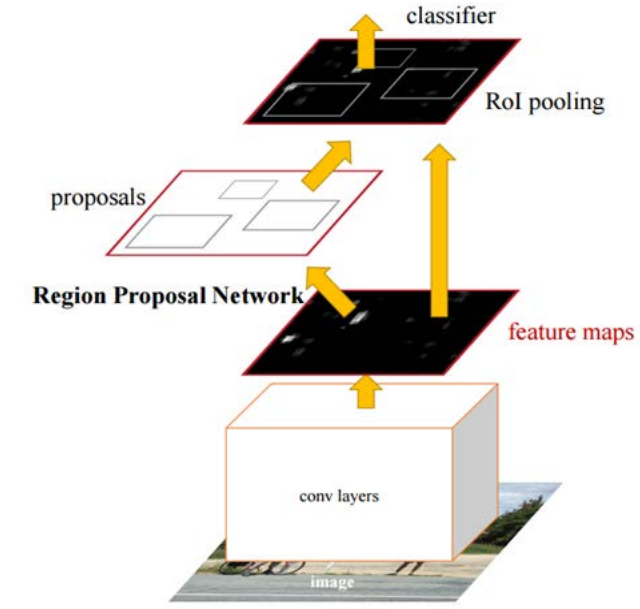


Fast R-CNN workflow

An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

1.3 FASTER RCNN

Faster R-CNN works to combat the somewhat complex training pipeline that both R-CNN and Fast R-CNN exhibited. Faster R-CNN, is composed of two modules. The first module is a deep fully convolutional network that proposes regions, and the second module is the Fast R-CNN detector that uses the proposed regions. The entire system is a single, unified network for object detection



Faster R-CNN workflow

1.4 MASK RCNN

Mask R-CNN works by adding a branch to Faster R-CNN that outputs a binary mask that says whether or not a given pixel is part of an object. The branch (in white in the above image), as before, is just a Fully Convolutional Network on top of a CNN based feature map. Here are its inputs and outputs:

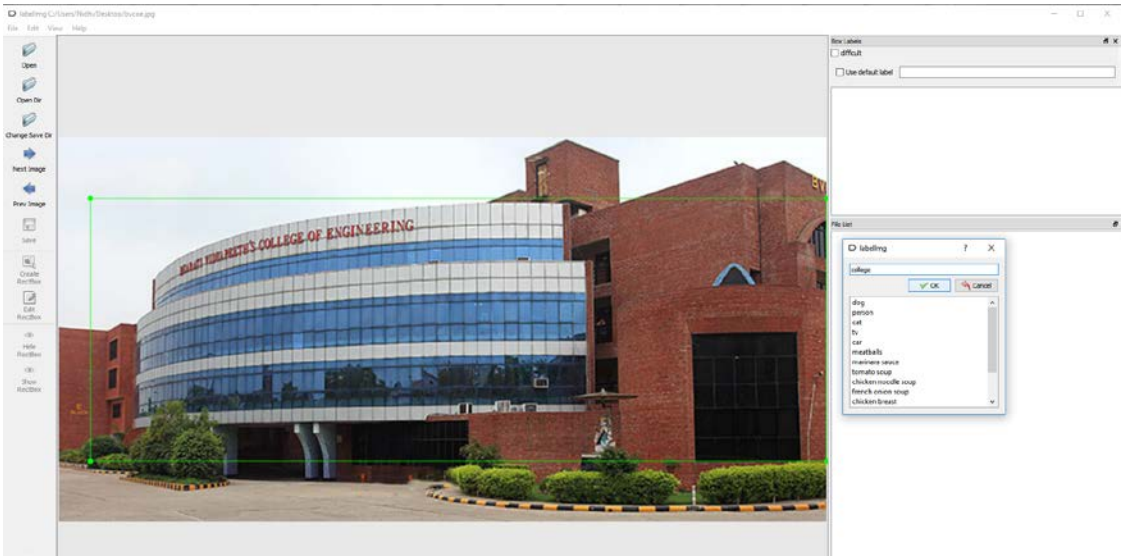
- **Inputs**: CNN Feature Map.
- **Outputs**: Matrix with 1s on all locations where the pixel belongs to the object and 0s elsewhere (this is known as a binary mask).

TECHNOLOGY USED

2.1 FOR DATA COLLECTION AND ANNOTATION

Python- It has become the world's most widely used programming language in recent times and is a free, open source software that can be used to do anything – ranging from writing code for competitive programming to writing code for quantum computing. It is also generally regarded as the standard language to make Machine Learning models with.

LabelImg- LabelImg is a graphical image annotation tool. It is written in Python and uses Qt for its graphical interface. Below is a picture of its GUI -:



2.1.1 Libraries used:

- **Opencv**: OpenCV (Open Source Computer Vision Library) is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms. OpenCV has a modular structure, which means that the package includes several shared or static libraries.

- **Numpy**- NumPy is the fundamental package for scientific computing with Python. It is a powerful N-dimensional array object, a sophisticated (broadcasting) functions tools for integrating C/C++ code. It is also useful for linear algebra. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

- **Time-** This module provides various time-related functions.

- **Imgaug**- Imgaug is a library for image augmentation in machine learning experiments. It supports a wide range of augmentation techniques, allows to easily combine these, has a simple yet powerful stochastic interface, can augment images and keypoints/landmarks on these and offers augmentation in background processes for improved performance.

- **TensorFlow**- TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

- **Scipy**- The *SciPy ecosystem* is a collection of open source software for scientific computing in Python. The community of people who use and develop this stack. The SciPy library, one component of the SciPy stack, provides many numerical routines. The SciPy ecosystem includes general and specialized tools for data management and computation, productive experimentation and high-performance computing.

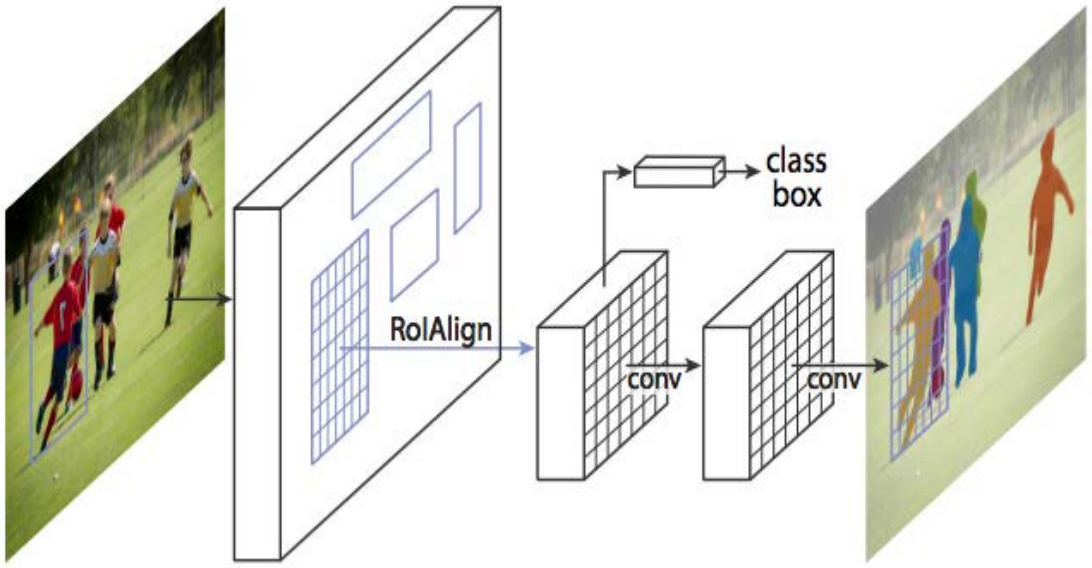
- **Skimage**- scikit-image is a collection of algorithms for image processing. It is available free of charge and free of restriction. Is an open source image processing library for the Python programming language. It includes algorithms for segmentation, geometric transformations, color space manipulation, analysis, filtering, morphology, feature detection, and more. It is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

- **Keras**- Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit or Theano. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

2.1 FOR DETECTION (Mask-RCNN)

Mask-RCNN is a conceptually simple, flexible, and general framework for object instance segmentation. Its approach efficiently detects objects in an image while simultaneously generating a high-quality segmentation mask for each instance. 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 outperforms all existing, single-model entries on every task, including the COCO 2016 challenge winners.

The key point is to decouple the classification and the pixel-level mask prediction tasks. Based on the framework of Faster R-CNN, it adds a third branch for predicting an object mask in parallel with the existing branches for classification and localization. The mask branch is a small fully-connected network applied to each RoI, predicting a segmentation mask in a pixel-to-pixel manner. Because pixel-level segmentation requires much more fine-grained alignment than bounding boxes, mask R-CNN improves the RoI pooling layer (named “RoIAlign layer”) so that RoI can be better and more precisely mapped to the regions of the original image.



The RoIAlign layer is designed to fix the location misalignment caused by quantization in the RoI pooling. RoIAlign removes the hash quantization, for example, by using x/16 instead of [x/16] (approximating to an integer value), so that the extracted features can be properly aligned with the input pixels.

CODE

'"Only the testing code has been published in this report for sake of compactness"'

```
import os
import sys
import random
import math
import numpy as np
import skimage.io
import matplotlib
import matplotlib.pyplot as plt

import coco
import utils
import model as modellib
import visualize

%matplotlib inline

# Root directory of the project
ROOT_DIR = os.getcwd()

# Directory to save logs and trained model
MODEL_DIR = os.path.join(ROOT_DIR, "logs")

# Local path to trained weights file
COCO_MODEL_PATH = os.path.join(ROOT_DIR, "mask_rcnn_coco.h5")

# Download COCO trained weights
if not os.path.exists(COCO_MODEL_PATH):
utils.download_trained_weights(COCO_MODEL_PATH)

# Directory of images to run detection on
IMAGE_DIR = os.path.join(ROOT_DIR, "images")

class InferenceConfig(coco.CocoConfig):
GPU_COUNT = 1
IMAGES_PER_GPU = 1

config = InferenceConfig()
config.display()

# Create model object in inference mode.
model = modellib.MaskRCNN(mode="inference", model_dir=MODEL_DIR,
config=config)
```

```
# Load weights trained on MS-COCO
model.load_weights(COCO_MODEL_PATH, by_name=True)

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

# Load a random image from the images folder
file_names = next(os.walk(IMAGE_DIR))[2]
filename = os.path.join(IMAGE_DIR, 'flood1.jpg')
image = skimage.io.imread(filename)

# Run detection
results = model.detect([image], verbose=1)

# Visualize results
r = results[0]
visualize.display_instances(image, r['rois'], r['masks'],
r['class_ids'],
class_names, r['scores']
```

RESULTS

6.1 DATASET

For this project, I constituted a novice dataset that was handpicked in a non-deterministic fashion. The images and videos were scoured from Google resources like Google Images and Youtube.

Number of pictures – **300**
Number of videos - **10**

6.2 PERFORMANCE

After training the architecture on our own dataset, promising results came to front. Below are some of the screenshots of them.

Left: *Screenshot of a video* **Right:** *Screenshot of the same after training*

Left: *A test picture* **Right:** *The test picture after training*

mAP (Mean average precision – accuracy metric for videos) -: **29.44**

FUTURE ENHANCEMENTS

1. While the results have been promising, due to the niche nature of the given problem statement, there isn't a plethora of high-quality data available on public domains for the given problem and hence it's only natural to concur that with more amount of high-quality data, the results and accuracy will only become higher.

2. In this project, we have not trained the dataset from scratch as Mask-RCNN is a heavy architecture, it isn't possible to train it from scratch on a normal laptop or desktop even with a relatively high-powered GPU and hence pre-trained weights were used for training and testing in this use case as it made predictions very fast. My aim is to train it from scratch and hence only predict PERSON and remove all other categories.

3. Extending the above point, if the real-time predictions can be made to be fast enough, the project can be embedded into drones as using a fleet of drones is a lot easier for surveying the disaster struck areas than sending a fleet of helicopters along with human personnel and hence would make the rescue operations a lot more efficient and will also lead to saving of time needed to rescue disaster victims which will be better for both the rescuers as well as the victims.

REFERENCES

1. He K, Gkioxari G, Dollár P, Girshick R. Mask r-cnn. InComputer Vision (ICCV), 2017 IEEE International Conference on 2017 Oct 22 (pp. 2980-2988). IEEE.https://data.gov.in/
2. matterpot/Mask_RCNN. (2019). Retrieved from https://github.com/matterport/Mask_RCNN
3. (2019, February 4). Mask R-CNN with OpenCV. Retrieved from https:// www.pyimagesearch.com/2018/11/19/mask-r-cnn-with-opencv/