Report

Project Title:

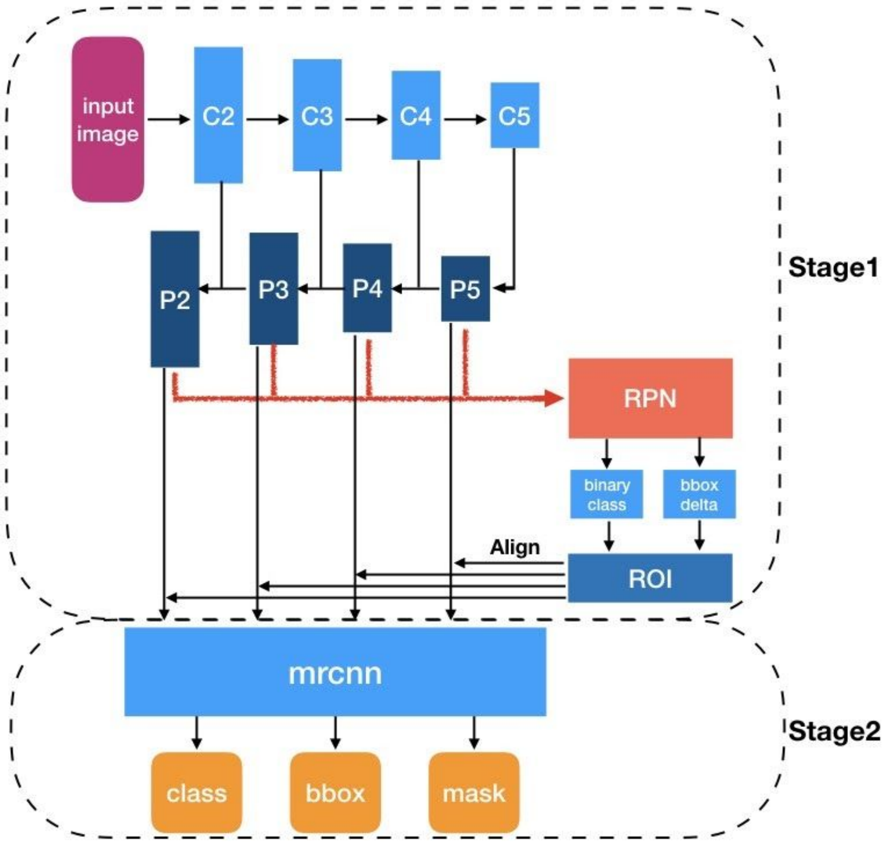
Pothole Detection on Roads using Mask R-CNN

Problem Statement

Due to the suboptimal nature of Indian roads, autonomous vehicles also need to factor in the tracks of roads that are safe for driving i.e. that do not have potholes. Our model aims to detect potholes in real-time to provide feedback to the driving mechanism using Mask R-CNN.

What is Mask R-CNN?

Mask RCNN is a deep neural network aimed to solve the instance segmentation problem in machine learning or computer vision. In other words, it can separate different objects in an image or a video. You give it an image, it gives you the object bounding boxes, classes, and masks.



There are two stages of Mask RCNN.

- First, it generates proposals about the regions where there might be an object based on the input image.
- Second, it predicts the class of the object, refines the bounding box and generates a mask in the pixel level of the object based on the first stage proposal. Both stages are connected to the backbone structure.

Goals of Mask R-CNN

- Meta- Algorithm
- Good Speed
- Good Accuracy
- Intuitive
- Easy to Use

The repository includes:

- Source code of Mask R-CNN and ResNet101.
- Training code for MS COCO
- Pre-trained weights for MS COCO

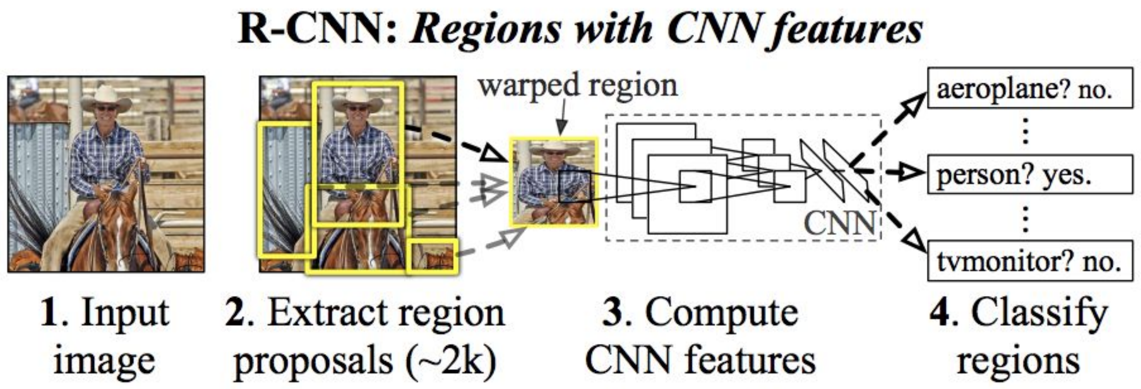
Why Mask R-CNN?

The major reason why you cannot proceed with this problem by building a standard convolutional network followed by a fully connected layer is that, the length of the output layer is variable — not constant, this is because the number of occurrences of the objects of interest is not fixed. Therefore, algorithms like R-CNN, YOLO etc have been developed.

Comparing all the models:

R-CNN

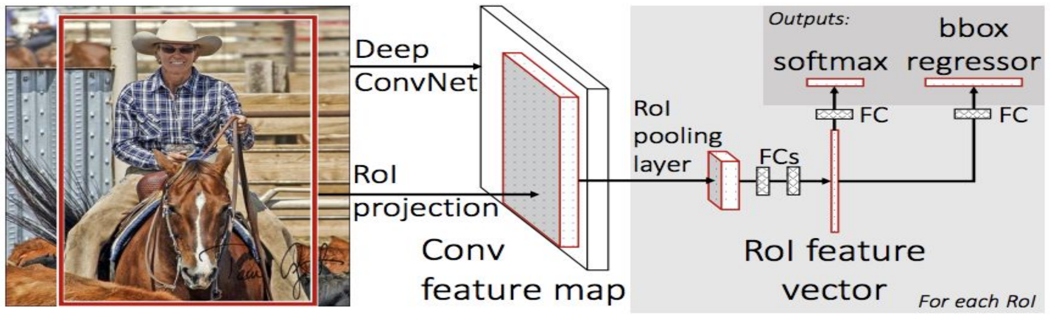
It still takes a huge amount of time to train the network as you would have to classify 2000 region proposals per image. It cannot be implemented in real time as it takes around 47 seconds for each test image.The selective search algorithm is a fixed algorithm. Therefore, no learning is happening at that stage. This could lead to the generation of bad candidate region proposals.



Fast R-CNN

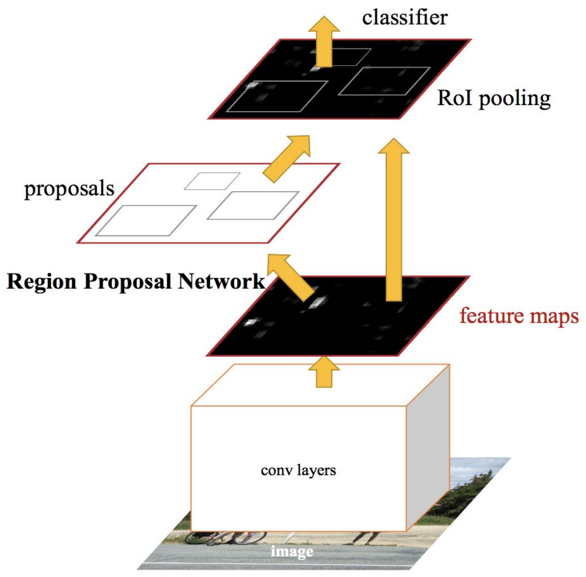
The reason “Fast R-CNN” is faster than R-CNN is because you don’t have to feed 2000 region proposals to the convolutional neural network every time. Instead, the convolution operation is done only once per image and a feature map is generated from it.

When you look at the performance of Fast R-CNN during testing time, including region proposals slows down the algorithm significantly when compared to not using region proposals. Therefore, region proposals become bottlenecks in Fast R-CNN algorithm affecting its performance.



Faster R-CNN

Similar to Fast R-CNN, the image is provided as an input to a convolutional network which provides a convolutional feature map. Instead of using selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals.The predicted region proposals are then reshaped using a RoI pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes.



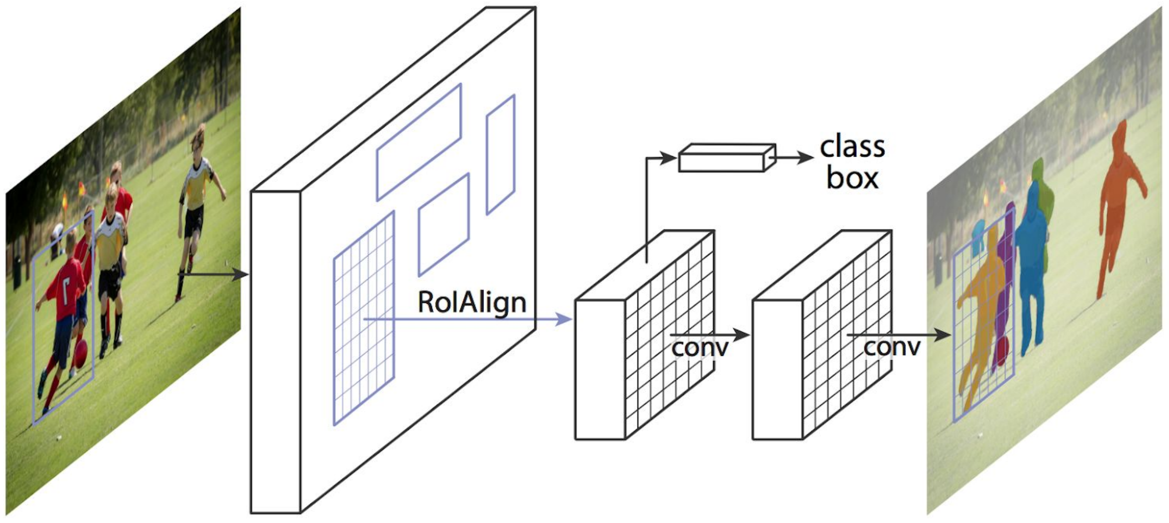
Mask R-CNN — Extending Faster R-CNN for Pixel Level Segmentation

Mask R-CNN does this 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.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).

But the Mask R-CNN authors had to make one small adjustment to make this pipeline work as expected.



When run without modifications on the original Faster R-CNN architecture, the Mask R-CNN authors realized that the regions of the feature map selected by RoIPool were slightly misaligned from the regions of the original image. Since image segmentation requires pixel level specificity, unlike bounding boxes, this naturally led to inaccuracies. This problem by cleverly adjusting RoIPool to be more precisely aligned using RoIAlign.

Importance and Relevance in future

India, being the second most populated country, has a detailed network of streets. Streets are the common mode of transportation in India bearing 90% of the country's traveler activity and about 65% of its cargo. Over the span of the recent two decades, there has been a colossal increase in the vehicle populace. Potholes formed because of substantial downpours, ill-conceived drainage framework in urban areas and transportation of heavy vehicles, turn into a major purpose behind high-chance of mishaps. Not only these, but autonomous vehicles are in regular use, but only in discrete locations. Soon these automated vehicles would be on Indian roads. Indian road conditions are absolutely non-conducive for autonomous driving. To address the referenced issues, a cost-effective solution is required that encourages drivers to drive safely. Our model provides a solution to this problem.

Research Papers Referred

Paper 1-**Real-Time Pothole Detection Using Android Smartphones with Accelerometer**
Conference: 2011 7th IEEE International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)

Paper 2- **Pothole Detection: An Efficient Vision-Based Method Using RGB Color Space Image Segmentation(May 2017)**
Conference: The 40th International Convention on Information and

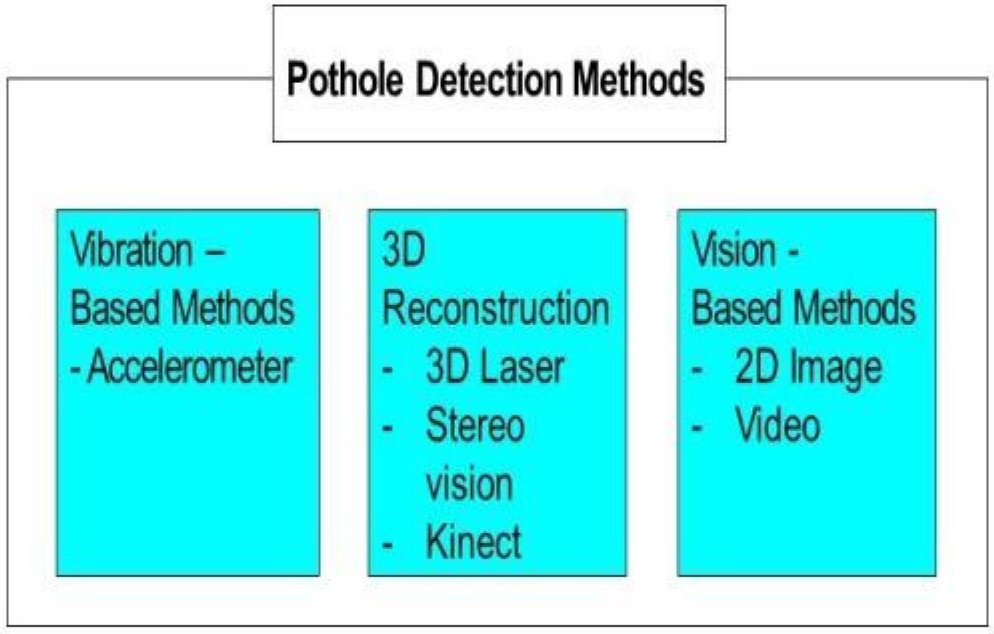
Communication Technology, Electronics and Microelectronics, Opatija, Croatia

Review Paper - **A REVIEW PAPER ON EXISTING POTHOLE DETECTION METHODS** International Research Journal of Engineering and Technology (IRJET) issued on 12th Dec 2018

Overview of the existent Pothole Detection Methods

13 research papers are briefly explained in this review paper. Many of these methods had certain limitations. Some cannot be implemented since the system is expensive and some work only under specific conditions. Some had the limitation that the system is that it provides warnings only after detecting the potholes. We picked and studied some of them that were similar to our model.

Paper 3- **Review and Analysis of Pothole Detection Methods**


Pothole Detection Methods
Vibration – Based Methods - Accelerometer
3D Reconstruction - 3D Laser - Stereo vision - Kinect
Vision - Based Methods - 2D Image - Video

Some Vibration-based methods for detecting potholes can be summarized as:

1. **Authors: Yu and Yull; Methods: Using acceleration records; Equipments:- ICP accelerometer, PC-oscilloscope, Test driving on I-90, USA**

2. **Authors: De Zoysa et al; Methods: Using acceleration and BusNet; Equipments:- A MICAz mote with acceleration sensor, Test in Sri Lanka**
3. **Authors: Erikson et al; Methods: Using accelerometer data; Equipments:-P2 (three-axis acceleration sensor and GPS devices), 7 taxis running in Boston**

3D reconstruction methods for detecting potholes:

1. **Authors: Chang et al; Methods: Using 3D laser scanning; Equipments:- 3D laser (MENSi GS 100), Simulation dataset & real data set**
2. **Authors: Li et al; Methods: Using 3D laser scanning; Equipments:- 3D laser & digital camera**
3. **Authors: Wang; Methods: Using stereo vision; Equipments:- Two cameras - Feasibility study**

Paper 4- **RIPD: Route Information and Pothole Detection**

For detecting potholes, **the integrated smartphone will use Accelerometer, GPS and various Z-axis algorithms**. Once all these sensors are activated, the accelerometer will continuously record the Z-axis readings and use the algorithms to detect potholes.

Whenever a pothole is detected the location coordinates are recorded using GPS sensor and sent to the server along with the timestamp. In our system we are using a combination of two Z-axis algorithms – Z-THRESH and Z-DIFF

Paper 5- **A Real-Time Pothole Detection Approach for Intelligent Transportation System**

Image Recognition Method: Yu and Salari proposed a pothole detection approach based on laser imaging techniques to collect road information. Then the artificial neural network algorithm (ANN) was used to analyze the road information and detect potholes [6]. However, this approach which requires a big computation power to recognize the laser images is unsuitable for mobile devices.

Mobile Sensing Method: The proposed real-time pothole detection method based on mobile sensing includes three steps: accelerometer data normalization, pothole detection approaches, and pothole location determination.

Pothole Detection Technology Research Announced By Jaguar Land Rover

Dr Mike Bell, Global Connected Car Director, Jaguar Land Rover, said:

“Our MagneRide equipped Range Rover Evoque and Discovery Sport vehicles feature sophisticated sensors that allow the vehicle to profile the road surface under the wheels and identify potholes, raised manholes and broken drain covers. By monitoring the motion of the vehicle and changes in the height of the suspension, the car is able to continuously adjust the vehicle’s suspension characteristics, giving passengers a more comfortable ride over uneven and damaged road surfaces.”

Comparing Paper 1 & 2 with pur model

	Our model	Paper 1	Paper 2
Input	image	accelerometer , gyroscope	image
Framework/Tools used	Keras(Tensorflow)	encog(android framework)	Matlab(image processing toolbox)
Approach	Mask R-CNN	ANN	Region-based segmentation
Passive/Proactive	proactive	passive	passive
Real-time or not	real-time	real-time	Not real-time
Accuracy	-	90% - 95%	82%

Major Drawbacks:

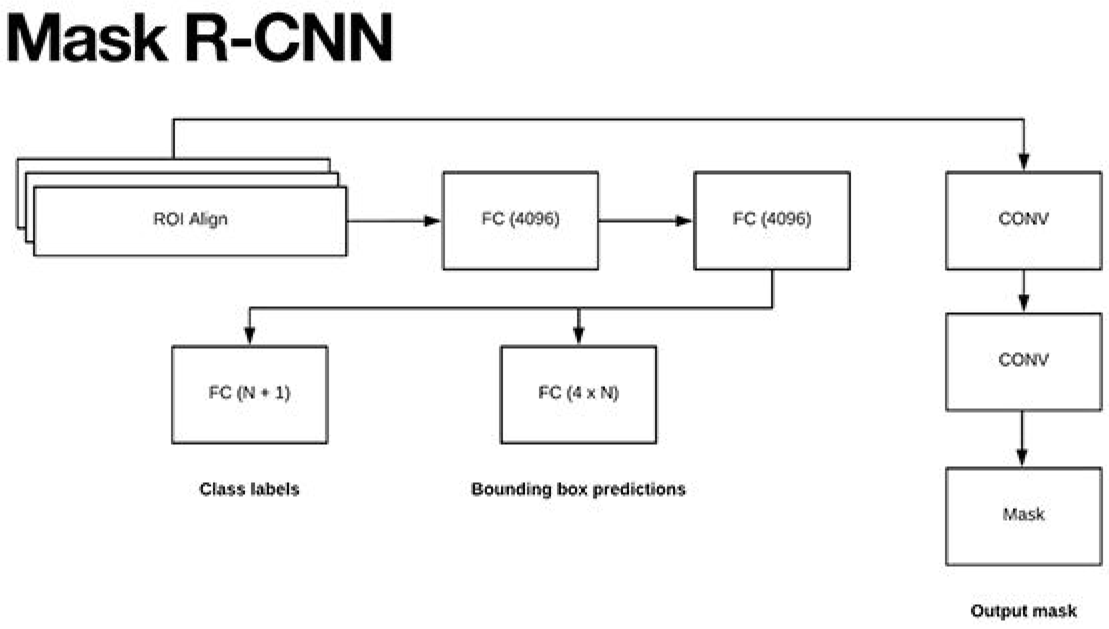
Paper 1 - **Real-Time Pothole Detection Using Android Smartphones with Accelerometer**

When using an accelerometer and gyroscope, the limitation to this system is that it provides warnings only after detecting the potholes which do not viably assist drivers with avoiding the chance of an accident.

Paper 2 - **A REVIEW PAPER ON EXISTING POTHOLE DETECTION METHODS**

It doesn't work in real-time, it processes the picture and the latter detects the potholes using Matlab.

Work Plan



Input - Using the smartphone camera, we take a video feed and splice atmost 5 frames per second(which is the processing limit of masked R-CNN)

Image segmentation creates a pixel-wise mask for each pothole in the image. This technique gives us a far more granular understanding of the pothole(s) in the image(i.e. Instance detection).

Backbone Model

Similar to the ConvNet that we use in Faster R-CNN to extract feature maps from the image, we use the ResNet 101 architecture to extract features from the images in Mask R-CNN. So, the first step is to take an image and extract features using the ResNet 101 architecture. These features act as an input for the next layer.

Region Proposal Network (RPN)

Now, we take the feature maps obtained in the previous step and apply a region proposal network (RPM). This basically predicts if an object is present in that region (or not). In this step, we get those regions or feature maps that the model predicts contain some object.

Region of Interest (RoI)

Then we apply a pooling layer and convert all the regions to the same shape. Next, these regions are passed through a fully connected network so that the class label and bounding boxes are predicted. Finally, the proposals are passed to a fully connected layer to classify and output the bounding boxes for objects. It also returns the mask for each proposal.

The segmentation mask.

For that, we first compute the region of interest so that the computation time can be reduced. For all the predicted regions, we compute the Intersection over Union (IoU) with the ground truth boxes. We can computer IoU like this:

IoU = Area of the intersection / Area of the union

Now, only if the IoU is greater than or equal to 0.5, we consider that as a region of interest. Otherwise, we neglect that particular region. We do this for all the regions and then select only a set of regions for which the IoU is greater than 0.5.

About Hardware, Data and Processing:

K80 Gpu with 24 gb memory to fit 3 images i.e. batch size=3

Dataset- 1457 images

Dataset link-
https://drive.google.com/file/d/16-xNP_Ez-3WgFF3vfsP9KJl4ka9hXDlV/view?usp=sharing

After training we have used pretrained weights. Since we didn't had that much processing power to train again.

Annotation- We have also done annotation of all the images, for Annotation of images we have used robots.ox.ac.uk:
http://www.robots.ox.ac.uk/~vgg/software/via/via-1.0.6.html

Annotation format: (The VIA tool saves image in JSON format)

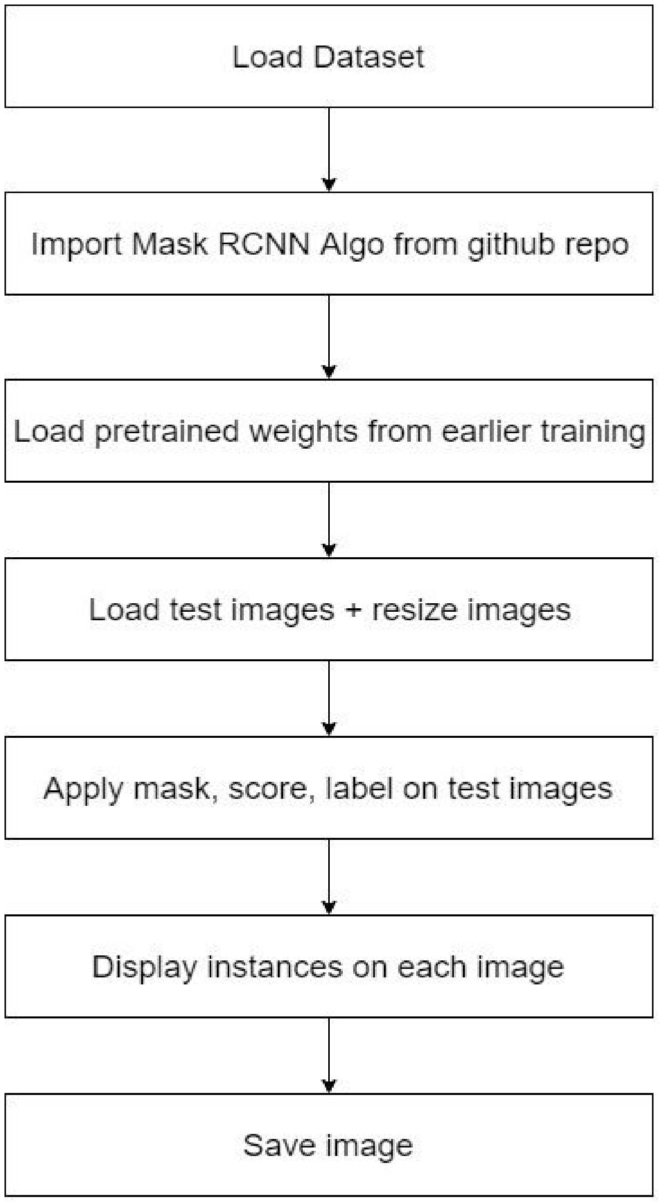
Load
annotations

```
VGG Image Annotator saves each image in the form:
{ 'filename': '28503151_5b5b7ec140_b.jpg',
  'regions': {
      '0': {
          'region_attributes': {},
          'shape_attributes': {
              'all_points_x': [...],
              'all_points_y': [...],
              'name': 'polygon'}},
      ... more regions ...
  },
```

We have trained model till 160 epochs.

Brief Explanation

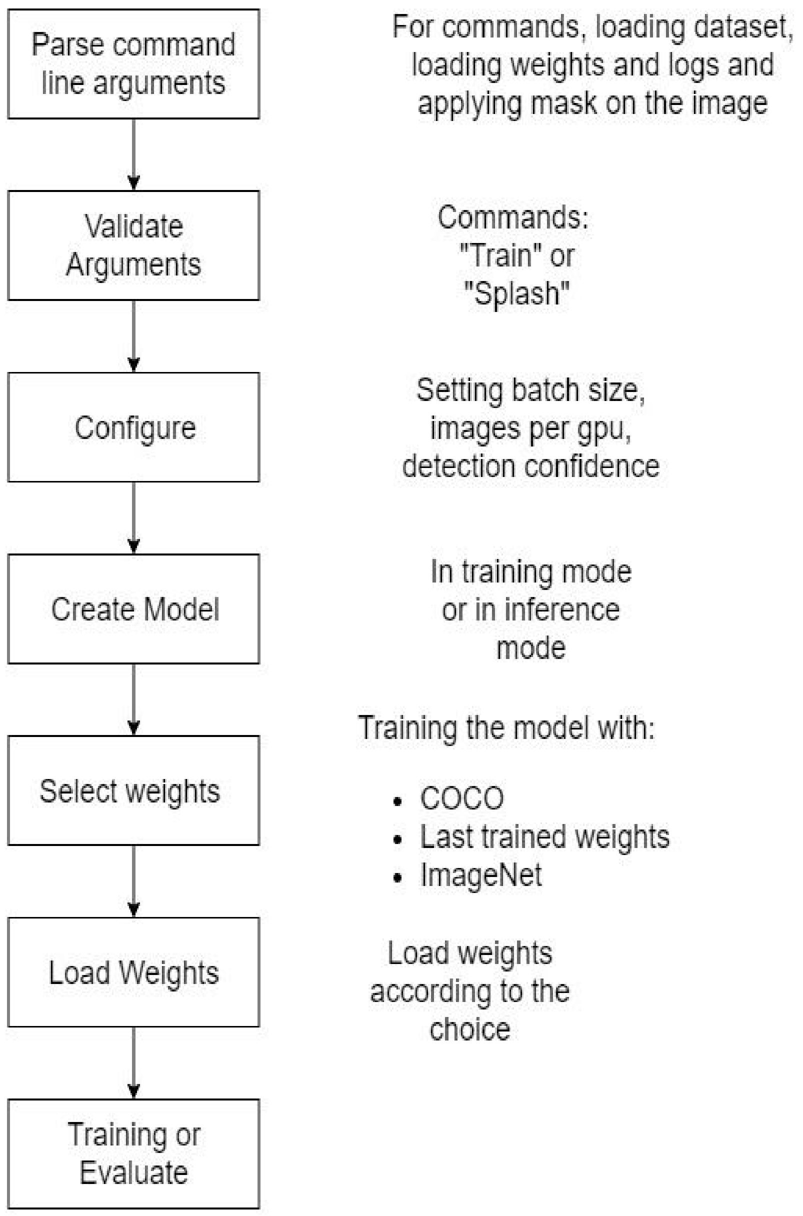
Main File Flowchart and function documentation


Load Dataset
Import Mask RCNN Algo from github repo
Load pretrained weights from earlier training
Load test images + resize images
Apply mask, score, label on test images
Display instances on each image
Save image

Documentation on functions:

- random_colors() - add random colors on the mask
- apply_mask() - apply mask to image
- display_instances - take the image and the result and apply the mask, box and the label
- save_image() - save final image

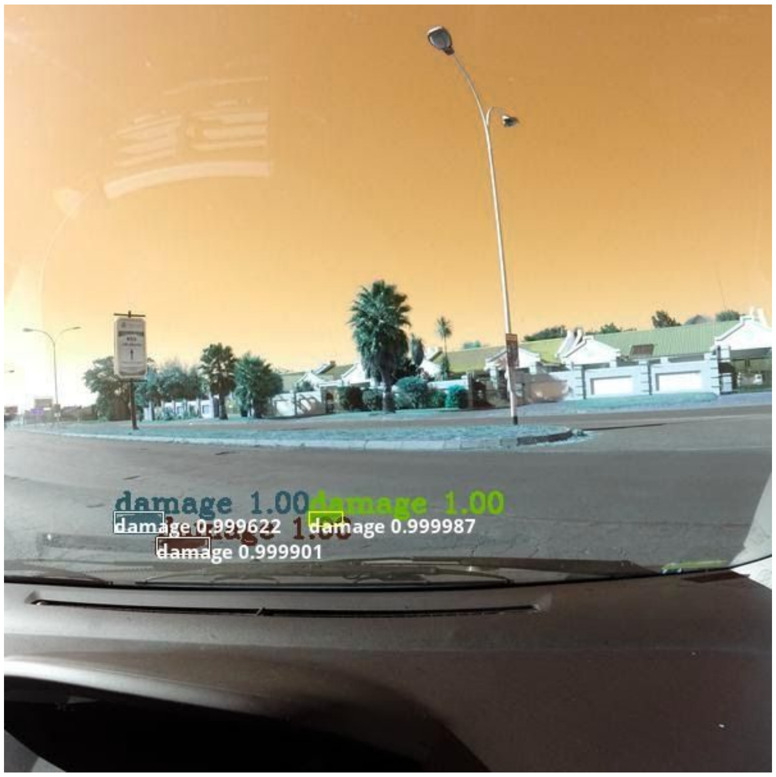
Custom file and function documentation

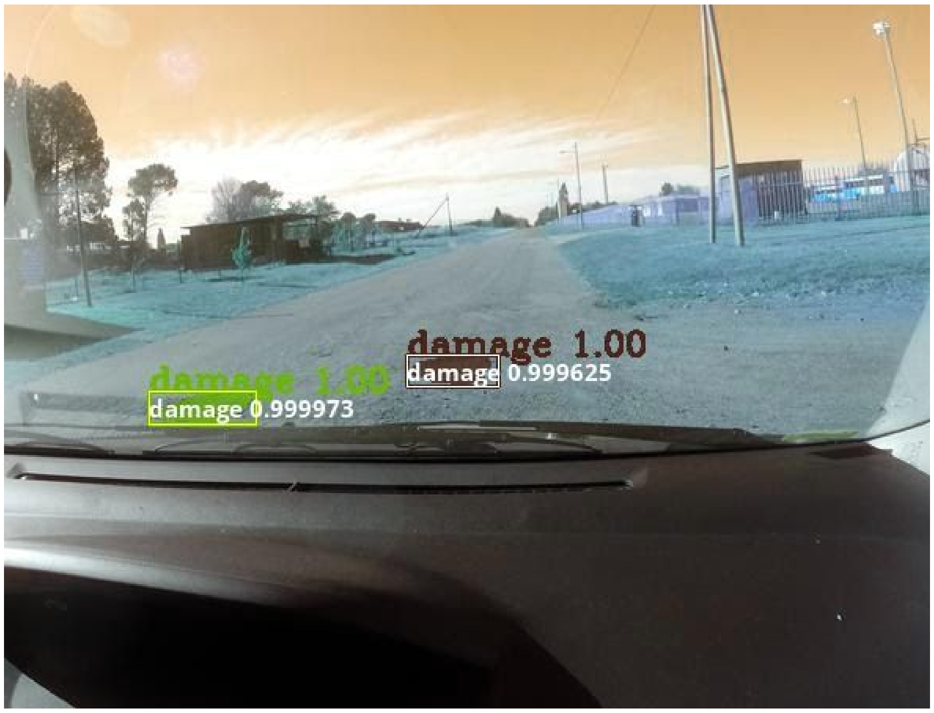

Parse command line arguments
For commands, loading dataset, loading weights and logs and applying mask on the image
Validate Arguments
Commands: "Train" or "Splash"
Configure
Setting batch size, images per gpu, detection confidence
Create Model
In training mode or in inference mode
Select weights
Training the model with:
• COCO
• Last trained weights
• ImageNet
Load Weights
Load weights according to the choice
Training or Evaluate

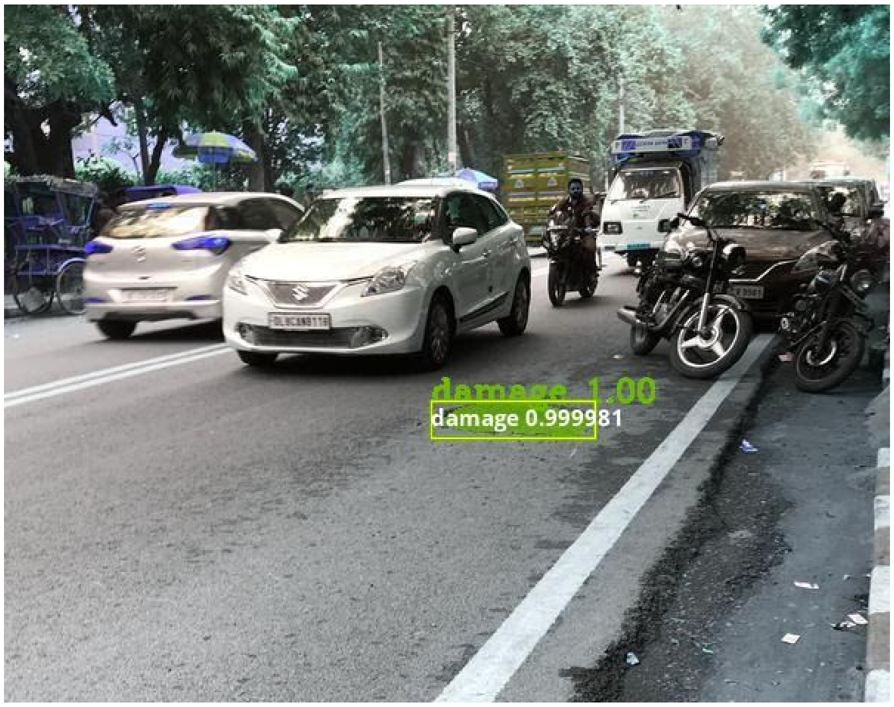
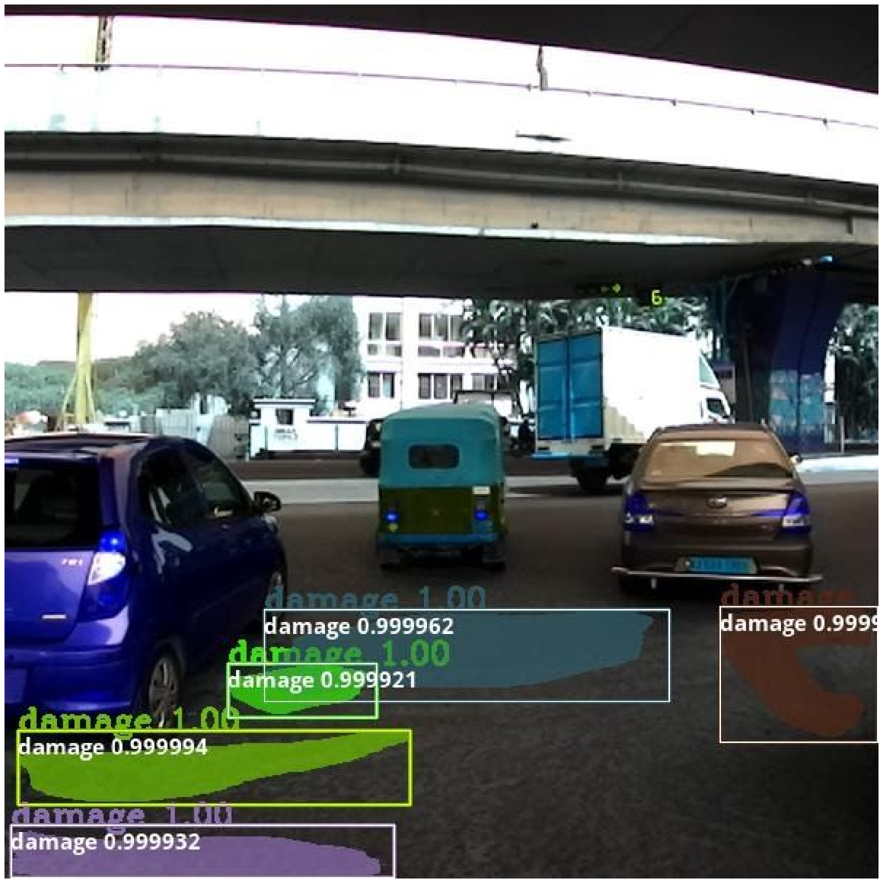
Documentation on functions:

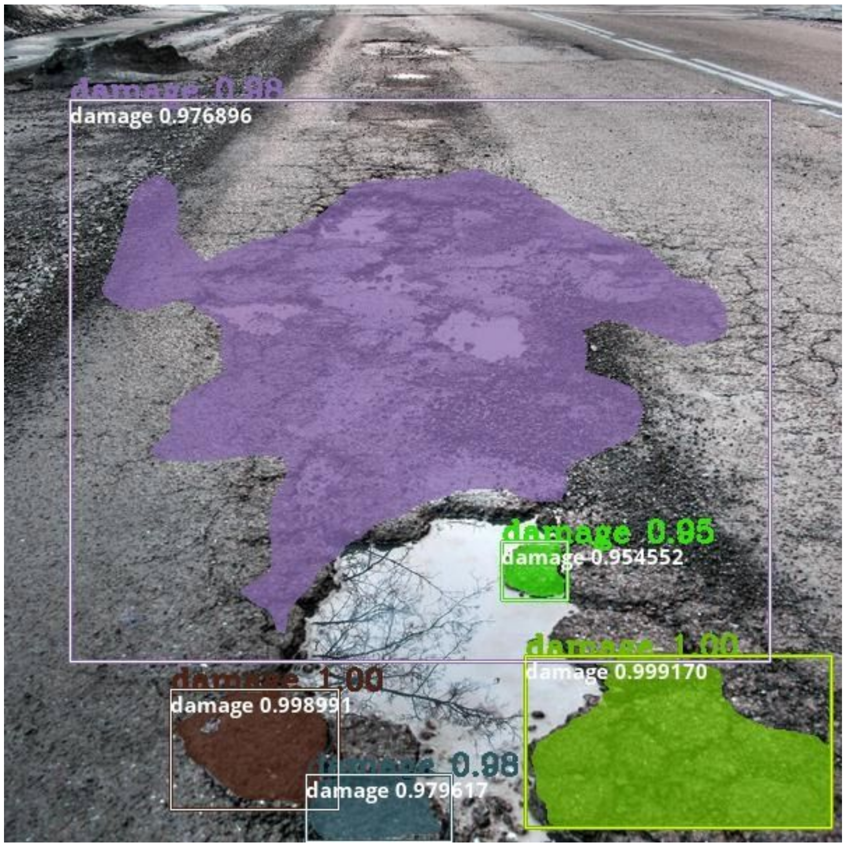
- load_custom() - Load dataset and annotaion
- load_mask() - generate instance mask of an image
- image_reference() - returning reference of an image
- train() - train dataset
- color_splash() - applying color splash effect (makes gray scale of the img)
- detect_and_color_splash() - model detect and add color splash effect

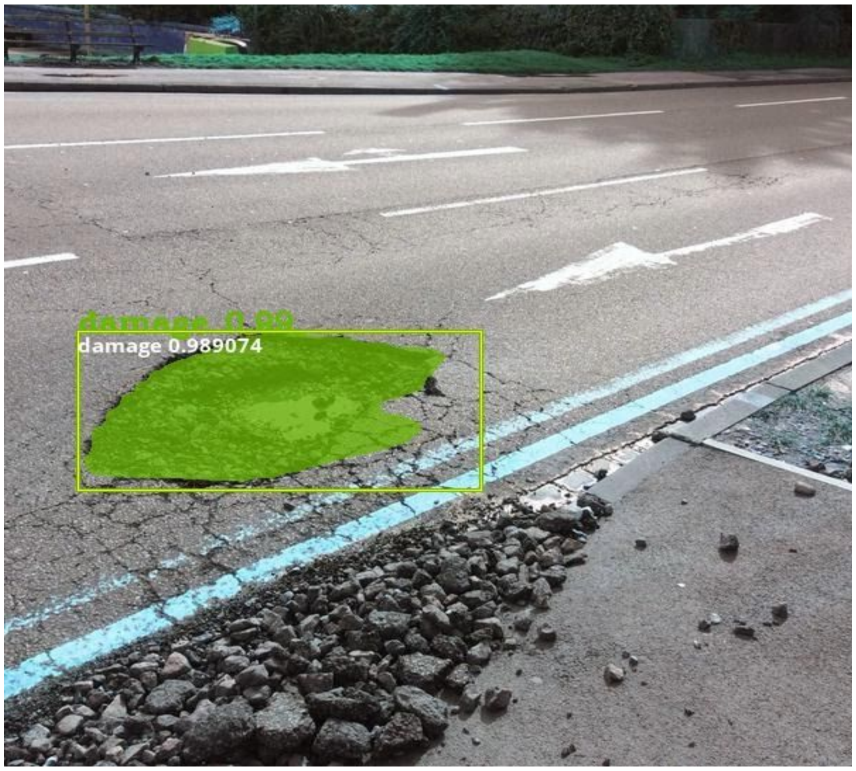
Some Results of detected potholes:

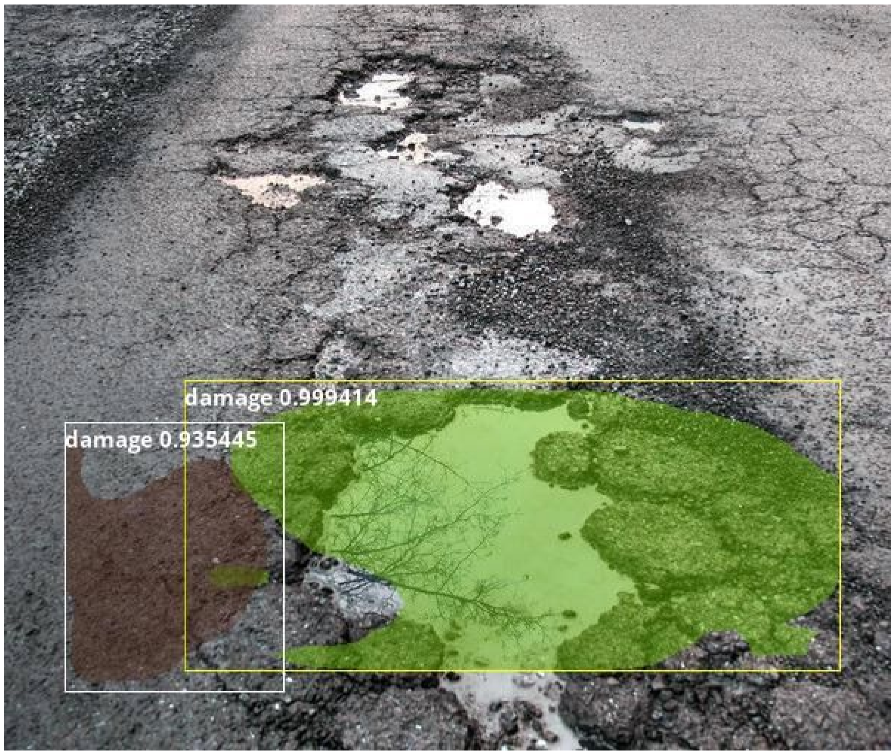


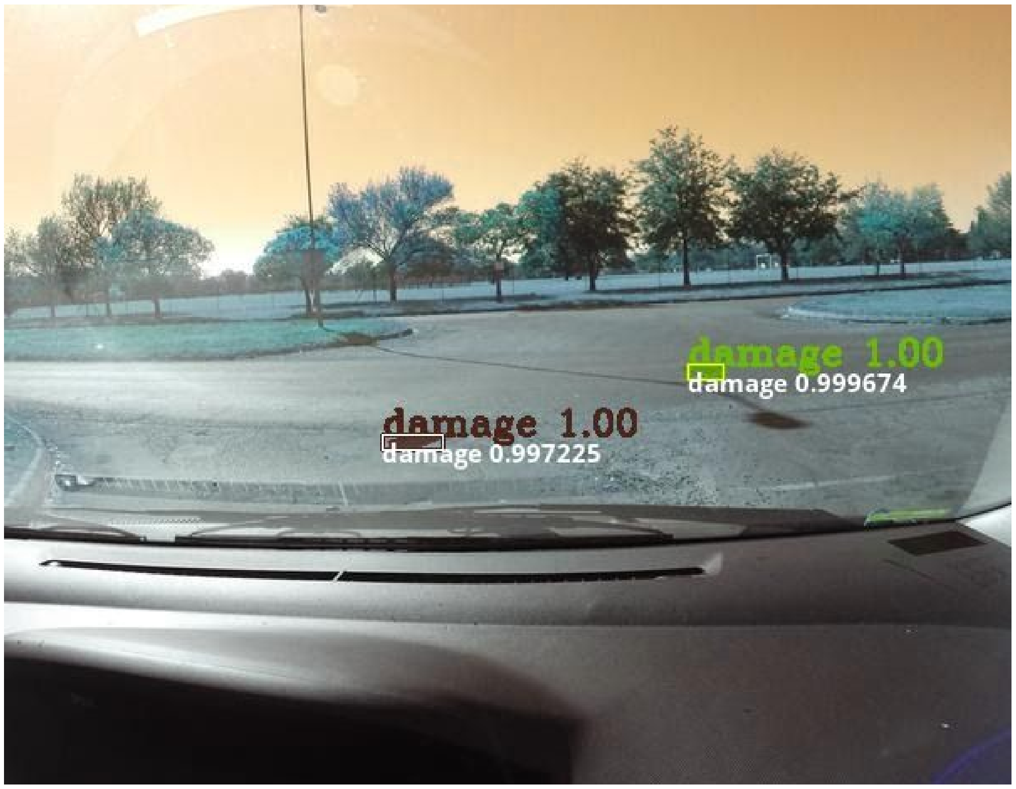



damage 1.00
damage 0.999981

damage 1.00
damage 0.999962
damage 0.9999
damage 1.00
damage 0.999921
damage 1.00
damage 0.999994
damage 1.00
damage 0.999932


damage 0.98
damage 0.976896
damage 0.95
damage 0.954552
damage 1.00
damage 0.999170
damage 1.00
damage 0.998991
damage 0.98
damage 0.979617


damage 0.99
damage 0.989074


damage 0.999414
damage 0.935445


damage 1.00
damage 0.999674
damage 1.00
damage 0.997225

Scope for Improvement

- Distance error can be removed
- We had varied and a very small dataset, better accuracy can be achieved by training with a large but not so varied dataset.
- Better hardware can be used.

Tools used

1. Tensorflow
2. Keras
3. OpenCV
4. Python 3.7
5. Spyder3
6. Matplotlib

References

1. https://medium.com/@alittlepain833/simple-understanding-of-mask-rcnn-134b5b330e95
2. https://paperswithcode.com/paper/mask-r-cnn: https://arxiv.org/pdf/1703.06870v3.pdf
3. **A REVIEW PAPER ON EXISTING POTHOLE DETECTION METHODS:** https://www.irjet.net/archives/V5/i12/IRJET-V5I12107.pdf
4. https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e
5. https://github.com/matterport/Mask_RCNN