AI PROJECT REPORT

IMPLEMENTATION OF RESEARCH PAPER BASED ON POTHOLE DETECTION USING ARTIFICIAL INTELLIGENCE

TEAM MEMBERS

ARIHANT JAIN	-	2019213
ARJUN KHARE	-	2019214

INSTRUCTOR - Dr.Kusum Kumari Bharti

INTRODUCTION

India is the second-largest road network in the world. Therefore, the road network plays an important role in Indian economic development and social functioning. According to the report in the last ten years, the Road Transport sector GDP grew at an annual average rate close to 10% compared to the overall annual GDP growth of 6%. Nowadays road construction is done very rapidly by the government of India. But road maintenance is a challenging task because of the poor drainage system and overloaded vehicles. Due to poor maintenance of roads, potholes get appeared on the road which causes road accidents. According to statistics submitted by the government of India from 2013 to 2016 potholes claimed 11,836 lives and 36,421 people got injured. Pothole problems cannot be resolved easily because every year almost all the places suffer from floods, disasters, heavy rainfall, etc. Though we cannot maintain the road, we can reduce the number of accidents which are getting increased every year.

Detection of potholes using artificial intelligence can be game changer as the results can be used to fill them up as early as possible and also warn people about upcoming potholes so that they can reduce their speed accordingly to be safe.

We will be using F-RCNN model to train implement our model and see the results.

APPROACH

The proposed modern based pothole detection method using the transfer learning technique detects potholes in videos/images in real-time. The method uses common techniques like “use of F-RCNN”, “inception v2 model” which are described below. The main advantage of this method is small training time with an easy training process and higher accuracy.

A. Tensorflow Object-Detection API

Tensorflow's object detection API is a very powerful tool that can quickly enable anyone to build and deploy a powerful image recognition system. It provides many pre-trained models (trained on different datasets) which can be used to build customized classifiers/detector/recognizer after fine-tuning. We've selected the model named “F-RCNN inception v2”.

B. Transfer Learning

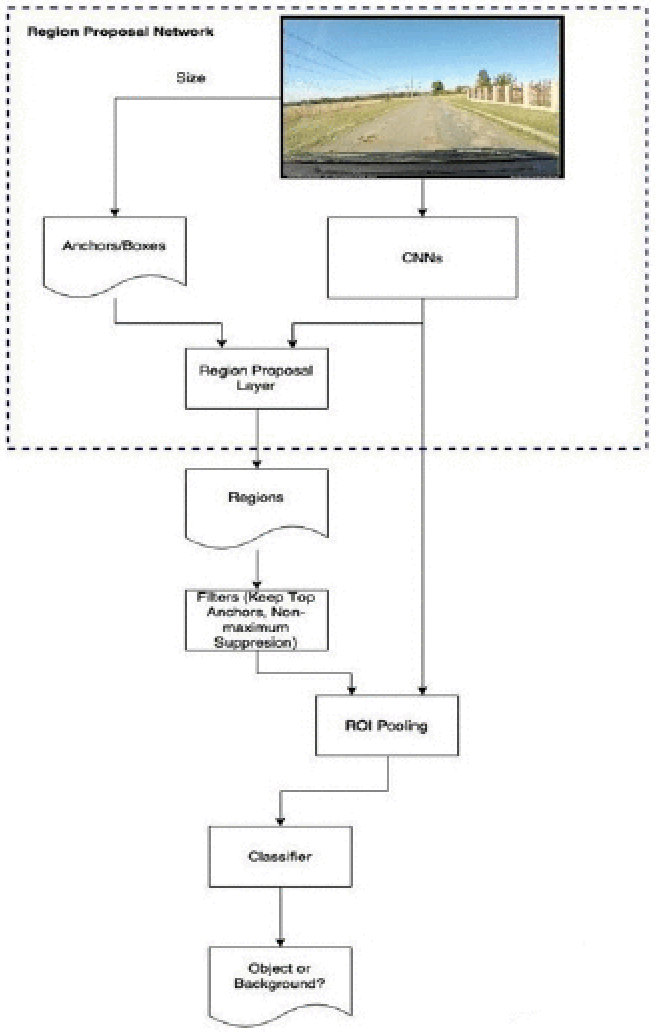
Transfer Learning makes use of knowledge gained while solving one problem and apply it to a different but related problem. With this technique, we can save a lot of time. In this technique below first, we select any pre-trained model (In which all the parameters are trained), then we perform fine-tuning. We fetch the new dataset to fine-tune the pre-trained CNN. If the new dataset is similar to the original dataset (with which the model has been trained), the same weights can be used for extracting the features from the new dataset. In our case, the dataset is very different from the original dataset. The Earlier layers of CNN contain more generic features (edge detector, colour blob detectors), but the later layers of CNN become progressively more specific to the details of the classes contained in the original dataset. So, earlier layers can help to extract the features of the new data. We fixed the earlier layers and Re-train the rest of the layers (because of the small amount of data).

C. F-RCNN (Faster Region-Based Convolutional Neural network)

It stands for Faster Region-based Convolutional Neural Network. Before talking about Faster R-CNN we should know about Fast R-CNN. Fast R-CNN is a detector that uses an external proposal or external selective search. It consists of External selective search, CNN with max pooling, ROI (Region of Interest) pooling layer, fully connected layers, and output layers. Fast R-CNN takes an input image and then with the help of CNN & max-pooling layers, a convolutional feature map is extracted from the image. The ROI pooling layer performs a very important task here. We know that fully connected layers can accept only certain sizes, So ROI pooling layers converts the output of the CNN into certain fixed sizes. When we put Fast R-CNN with the RPN (Region proposal network), it becomes Faster R-CNN. So, basically, the difference between Fast R-CNN and Faster R-CNN is the Region proposal. In Fast R-CNN, there is an external selective search whereas in Faster R-CNN

RPN is combined with the Architecture.

RPN is the Architecture that makes Fast R-CNN a Faster R-CNN. In order to reduce the computational complexity and requirement, RPN decides where to look in the image. It scans the image and gives k output boxes each with 2 scores indicating the probability of availability of an object. Different size and different aspect ratio of boxes are selected in order to accommodate different types of objects.



In order to get a trained working Faster R-CNN architecture, we need to perform the following steps -

- Training of RPN - first of all, we need to Train the RPN architecture with the dataset so that it can propose the expected region.
- Training of Fast R-CNN - As we know, Faster R-CNN is a combination of RPN and Fast R-CNN. So, we've to train a Fast R-CNN with the proposals obtained by RPN (after training) in order to make a Faster R-CNN.
- Fixing Convolutional layers, fine-tuning unique layers to RPN.
- Fixing Convolutional layers, fine-tuning fully connected layers of Fast R-CNN

DATASET INFORMATION

Dataset Link - https://www.kaggle.com/sachinpatel21/starter-code-to-view-dataset-images/data

Image labelling using - https://github.com/tzutalin/labelImg

I had got the Pothole images dataset and I had used labelImg which is an open tool for labelling your images.

FUNCTIONS AND METHOD USED

We used multiple funtions and methods which are as follows -

1. TensorFlow provides several object detection models (pre-trained classifiers with specific neural network architectures) in its model zoo.

   link to the models - https://github.com/tensorflow/models/tree/master/research/object_detection/models

   Installed the object detection api using command

   ```
   pip install tensorflow-object-detection-api
   ```

2. **Imports made for various uses**

   ```
   import numpy as np

   import os
   ```

```
import six.moves.urllib as urllib

import sys

import tarfile

import tensorflow as tf

import zipfile

import matplotlib.pyplot as plt

from collections import defaultdict

from io import StringIO

from PIL import Image

from object_detection.utils import label_map_util

from object_detection.utils import visualization_utils as vis_util
```

3. **Defining various path variables having info of prediction model and testing images**

```
MODEL_NAME = 'inference_graph'

PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'

PATH_TO_LABELS = 'labelmap.pbtxt'

NUM_CLASSES = 4
```

4. **Bringing the tensorflow model into memory to training it detection_graph - Modifying Graph() to provide the model we declared in** PATH_TO_CKPT **method provided in the tensorflow library and afterwards using its methods**

```
detection_graph = tf.Graph()
```

```
with detection_graph.as_default():
    od_graph_def = tf.compat.v1.GraphDef()
    with tf.io.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')
```

5. **Loading label map - we used internal utility functions to map integers to appropriate string label**

```
label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)
```

6. **function to convert image into numpy array - load_image_into_numpy_array(image)**

```
def load_image_into_numpy_array(image):
  (im_width, im_height) = image.size
  return np.array(image.getdata()).reshape(
      (im_height, im_width, 3)).astype(np.uint8)
```

7. **function to create a session for prediction of potholes -**
   tf.Session(graph=detection_graph) as session

8. **using methods from detection_graph to get tensors by name for**

various uses

```
image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')

detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')

detection_scores = detection_graph.get_tensor_by_name('detection_scores:0')

detection_classes = detection_graph.get_tensor_by_name('detection_classes:0')

num_detections = detection_graph.get_tensor_by_name('num_detections:0')
```

9. **using a method from session to create a tuple using following parameters defined earlier**

```
(boxes, scores, classes, num) = session.run( [detection_boxes,
detection_scores, detection_classes, num_detections], feed_dict={image_tensor:
image_np_expanded})
```

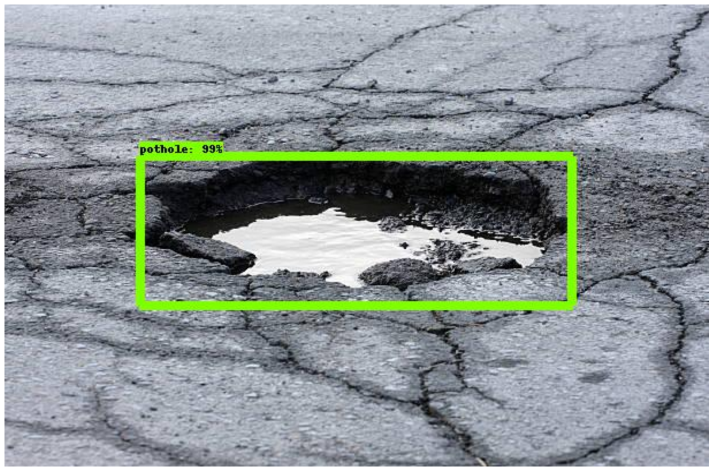
10. **Used Utility functions from object_detection_utils to visualize the classifications predicted by the model and passed the required arguments**

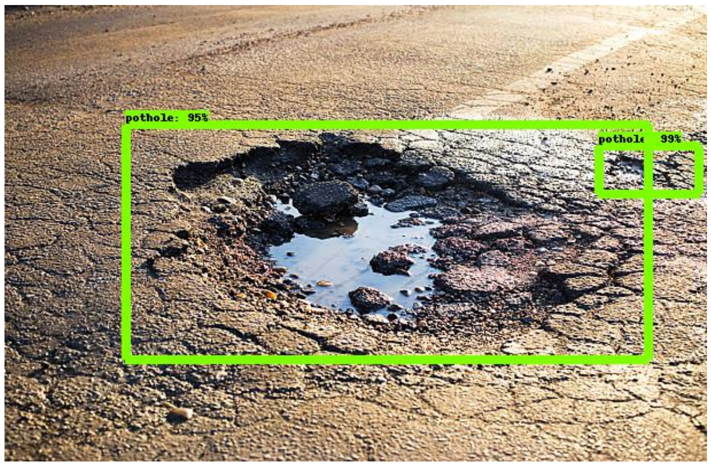
```
vis_util.visualize_boxes_and_labels_on_image_array(

        image_np,

        np.squeeze(boxes),

        np.squeeze(classes).astype(np.int32),

        np.squeeze(scores),

        category_index,

        use_normalized_coordinates=True,

        line_thickness=8)
```

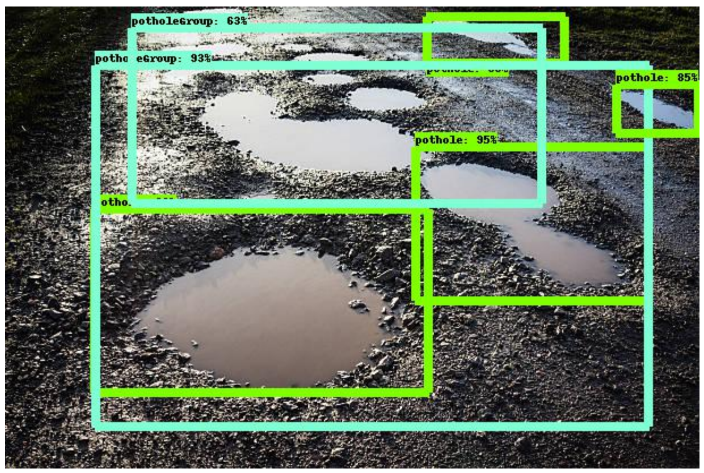
RESULTS

Input Images

Output Images







COMPARISION

The model works similar to the model suggested in the research paper, and the results are good.

CONCLUSION

The model is effectively able to predict potholes and a group of potholes from images and videos provided to it.

REFERENCES

Research Paper

Image Detection

Video Detection