Definition

PROJECT OVERVIEW

Due to the presence of potholes in the road many accidents happen every year. .So, if we make a model which can detect the potholes of any road and can give a pre warning to the riders it can minimize the chances of accidents..But only detecting them is not enough ,so we need to track the location of the potholes so that we can send the exact location to our municipality to take appropriate action to fix them..This was the inspiration for our project..It can also be used as a feature for self-driving vehicles..

So, in this project we have created a pothole detection model using yolov4(tiny- for real time detection) architecture with a mix of our self annotated data and a little pre-annotated Roboflow data..

PROBLEM STATEMENT

Firstly, the goal is to create a better model which can detect the potholes in the road in real time without compromising accuracy..
1. Make a proper dataset for better accuracy…
2. Train the model with any detection architecture for real time detection..
3. Test the model on a local server..
4. Deploy the model using opencv on Flask Api…
5. Finding out the location of the potholes (here we have used the latitude & longitude of any location based on the camera location)…
6. Add the pothole location into a Database to keep the record for municipality or other sources…

The final model is expected to be useful for finding the potholes with their proper location, alongside generating the warning for the riders also and keeping the generated data in a proper database for fair use..

Analysis

DATA EXPLORATION

At first, we had taken only the Roboflow dataset and trained it with large YOLOv4 model (width=512,Height=512)..It has around 665 images originally shared by Atikur Rahman Chitholian as part of his undergraduate thesis and was originally shared on Kaggle.. Originally it wasn't divided into train, valid & test but Roboflow have re-shuffled the images into a train-valid-test split for various usage.

Train Dataset :- https://drive.google.com/file/d/11emYCzLS9Zb9G3GFeUhAlwCFyBEW1QGV/view?usp=sharing
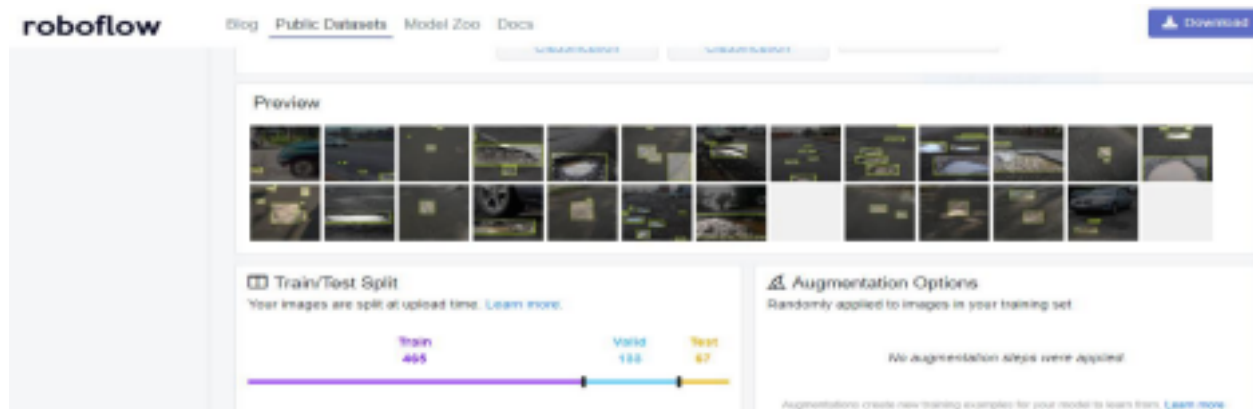Test Dataset :- https://drive.google.com/file/d/1f50OMVjHSYERf6yhKLqIB1AGEi_9EXUj/view?usp=sharing

Fig 1:- Roboflow Image Train/Valid/Test split

The dataset was originally taken for the worst scenarios in city areas so that those could be fixed faster.
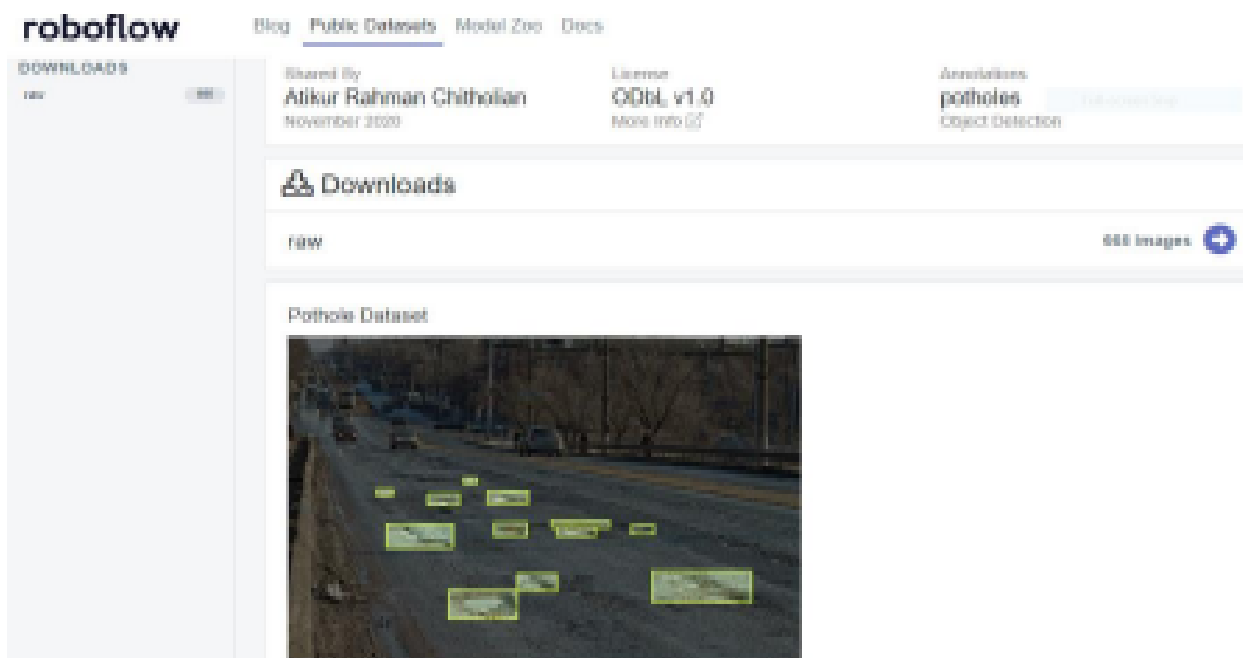

Fig 2:- Roboflow Pothole Image Example

This model was having lower mAP & precision with higher latency..So, we decided to train our model once again and this time we use a local video of approximate 1 hour from which we have taken the timestamps where the potholes are present and scrapped the frames from those clips and annotated them using labelimg tool manually..Now we got around "850" annotated images along with Roboflow Data..Total around 1500 labeled images..

Fig 3:- Annotated on Labelimg

The text annotations of the dataset have the following fields:
- ❖ utf8_string: the text itself (string)
- ❖ bbox: a bounding box around the text, in the form of [x, y, width, height] (integers)
- ❖ legibility: the readability of the text; either "legible" or "illegible" (string)
- ❖ area: the area of the bounding box (float)
- ❖ class: pothole(0)
- ❖ image_id (integer)
- ❖ id (integer)

ALGORITHMS AND TECHNIQUES

Here we are using the YOLOv4 model for training.. It is an one stage detection model which use basically "VGG 16" Convolutional Neural Network architecture for training and Softmax function for classification on darknet-53..But in this algorithm before starting the training on VGG16 it takes around 80 frames from the whole image to understand where the real object is present and this way it is able to detect any object more accurately with lower latency…
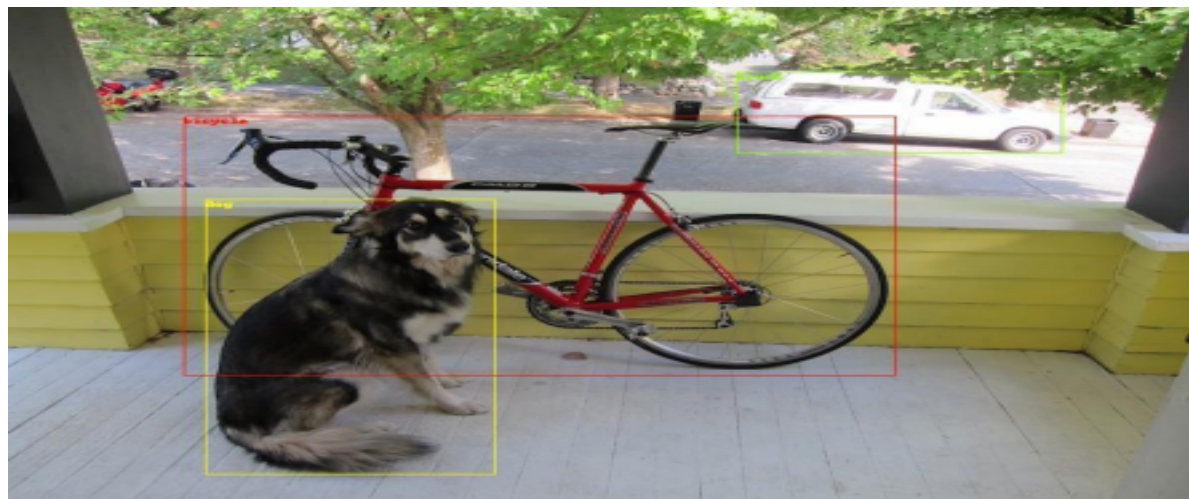

Fig 4:-Here YOLO model is detecting different objects

TRAINING ON COLAB

After making the data we have trained it on yolov4 tiny using our google colab with GPU..

❖ Training parameters are:
➢ Width = 416, Height = 416.
➢ Max-batches = 6000.
➢ Steps = 4800, 5400.
➢ Batch-Size = 4.
➢ Class = 1.
➢ Filter = 18.
➢ Subdivision = 64.

It took around 4-5 hours to train completely and we got the following result:-

precision = 0.69, recall = 0.67, F1-score = 0.68 TP = 222, FP = 98, FN = 108, average IoU = 52.50%, mean average precision (mAP@0.50) = 0.705819, or 70.58 %

```
for conf_thresh = 0.25, precision = 0.69, recall = 0.67, F1-score = 0.68
for conf_thresh = 0.25, TP = 222, FP = 98, FN = 108, average IoU = 52.50 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
mean average precision (mAP@0.50) = 0.705819, or 70.58 %
Total Detection Time: 2 Seconds

Set -points flag:
 "-points 101" for MS COCO
 "-points 11" for PascalVOC 2007 (uncomment "difficult" in voc.data)
 "-points 0" (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset

mean average precision (mAP@0.5) = 0.705819
```

Fig 5 :- The mAP, precision & recall

TESTING & DEPLOYMENT

After Training we first test on colab with some images & then deploy on Flask using open-cv in my pycharm interpreter..

Fig 6:- Tested on colab

After doing the testing we downloaded our weights and cfg files and deployed on our local-server on a random video using OpenCV, Flask & HTML in our pycharm interpreter.. Then, we added a function (called geocoder()) which is made to calculate the latitude & the longitude of every pothole based on our camera location.. After calculating that we stored all the information in the database MongoDB in an intent called "Pothole" with the current timestamp so that we can get the detailed information for further usage…
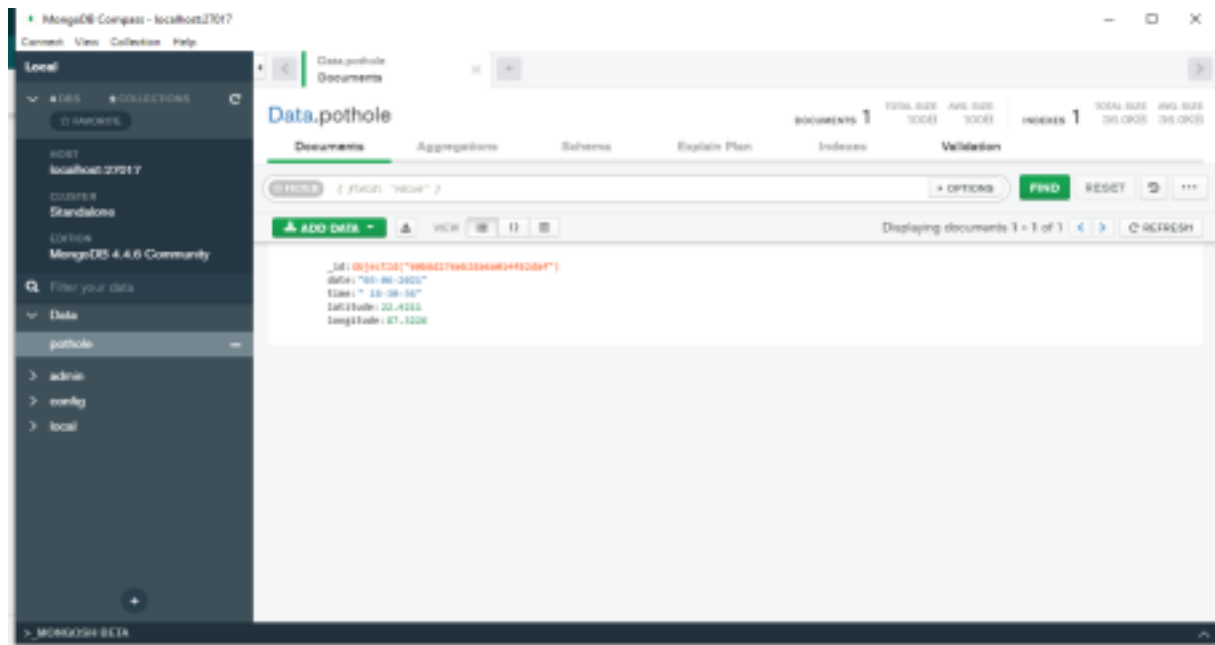

Fig7:- Pothole Detection Example

Fig 8:- Storing the data in the Database Example.

Future Note:- we can host this project in different cloud-platforms and also can make a good GUI for more handy usage of the project in future.