

IMAGE PROCESSING FOR INDIAN ROADS

- **Problem Description**

In developing nations such as India, the vehicular growth rate is increasing exponentially which is worsening the traffic operations. The Ministry of Road Transport and Highways report on Road accidents in India stated that road accidents increased by a rate of 0.46 % in the year 2020 when compared to 2019. The major challenges faced in the road transportation sector are: **Potholes which are formed due to heavy rains and dense movement of vehicles on the poorly constructed roads. Pothole formation has given rise to accidents and loss of human lives**

Due to this there is a need to develop a model which can analyze and detect poor road conditions like potholes. This project aims in building a system which can detect the poor road conditions and can notify the driver as well as the government beforehand to improve the road conditions.

- **SYSTEM MODEL**

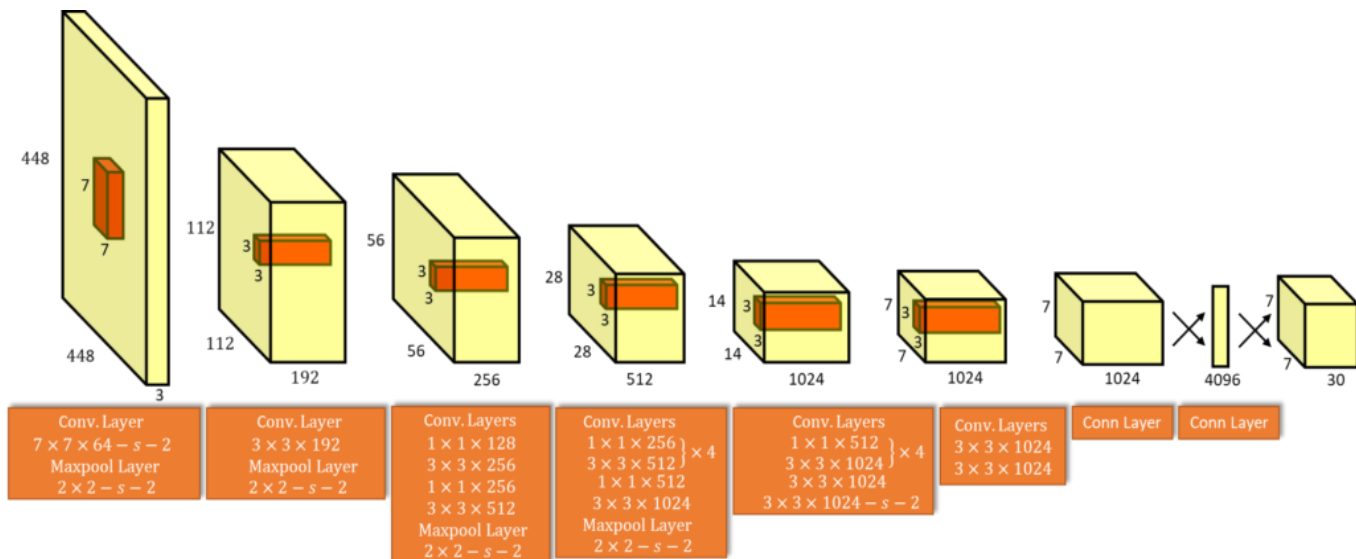
Image Processing Technique:

Following steps were performed in respective order to gain the output:

- a)Median Blur
- b)Erosion
- c)Canny-edge detection
- d)Contour-detection
- e)BoundingBox prediction

Deep Learning Approach:YOLO:

1. Yolo is a Deep Learning algorithm specially designed for image classification and Object Detection (Deep Convolutional Neural Network). It contains a very large number of hidden layers for processing the input data.
2. The architecture of YOLO contains 24 convolutional layers, 4 max-pooling layers, and 2 fully connected layers.
3. The architecture of the YOLO base model is shown below:



• **PROPOSED WORK**

Pothole detection is being carried out using two techniques namely image processing and machine learning techniques. Those two techniques are used for a study of the detection and occurrence of potholes. In this project, we propose both of them individually and then a combination of the techniques to see how image pre-processing can affect the performance of a deep learning model.

First of all, we implemented the image processing techniques on a single image in the order: median blur, erosion, canny edge detection, contour detection, bounding box prediction. After that we labelled a dataset of around 800 images and passed it to the YOLOv5 model and noted the results.

Secondly, we applied median blur on the already labelled dataset and then passed it to the YOLOv5 model and noted the results. Lastly, we applied median blur and erosion both and passed it to the model and noted the results. We compared the results at last. These experiments gave us information on how image pre-processing could affect the accuracy of the model.

Experiment-1: Image Processing Technique

We first performed image processing techniques using python and openCv on images and following are the results on one of the image:

Steps:

a)Median Blur:

The Median blur operation is similar to the other averaging methods. Here, the central element of the image is replaced by the median of all the pixels in the kernel area. This operation processes the edges while removing the noise. It is very useful in removing salt and pepper noise.



b)Erosion:

Erosion erodes away the boundaries of the foreground objects. It is used to diminish the features of an image.



c)Canny Edge detection:

The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images.



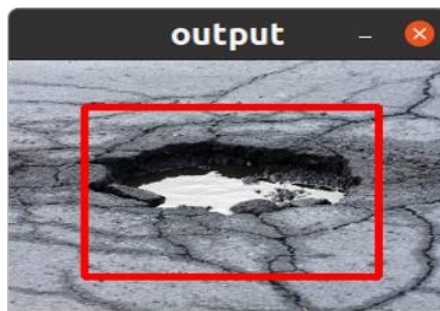
d)Contour Detection:

When we join all the points on the boundary of an object, we get a contour. Typically, a specific contour refers to boundary pixels that have the same color and intensity.



e) Bounding Box Prediction:

Finally, place a bounding box across the contour.



Experiment-2: Deep Learning Approach: USING YOLO

About YOLO:

- 1) Yolo is a Deep Learning algorithm specially designed for image classification and Object Detection (Deep Convolutional Neural Network).
- 2) The model was trained over a week by the authors of the paper “You Only Look Once: Unified, Real-Time Object Detection” and obtained a whopping accuracy of 88%.
- 3) Yolo had been trained on a very large dataset.
- 4) YOLO runs at 45 Frames per second so it has a latency of 15ms.
- 5) It has an mAP (mean Average Precision) of 63.4% which is the highest among real-time object detectors.

Our Approach: Transfer Learning

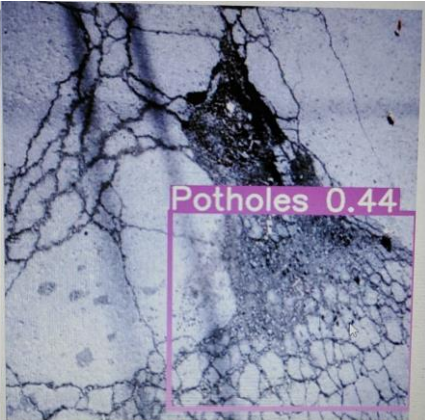
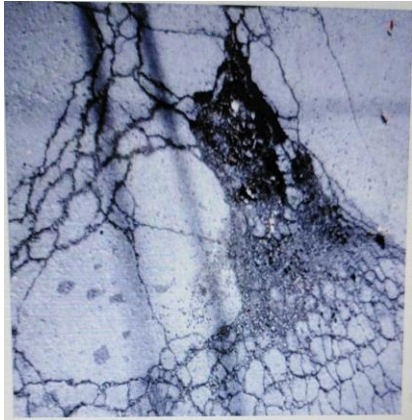
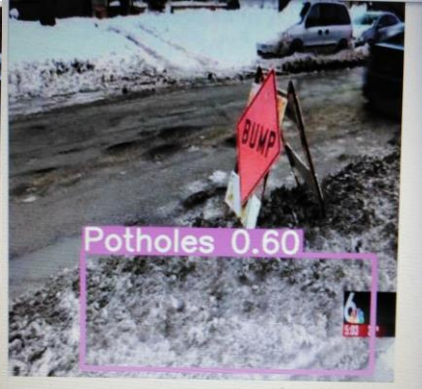
- 1) Transfer learning is about speeding up a new learning task by reusing the results of previous learning.
- 2) The already-trained model is referred to as the base model. Transfer learning involves retraining the base model or creating a new model on top of the base model.
- 3) In our project, we will be using the YOLO v5 model for our custom object detection, that is, pothole detection.
- 4) YOLO was chosen after going through a number of research papers and techniques used in them and it was found by us that it has the best real-time object detection accuracy.

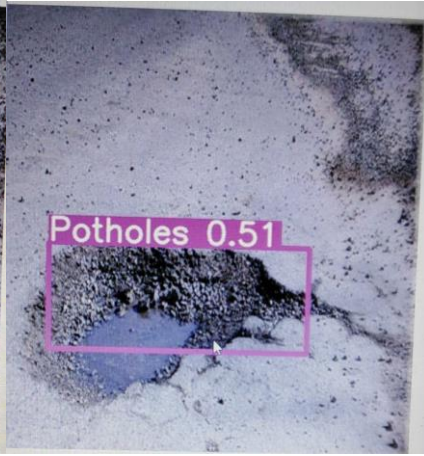
Experiment -3: Combination of 2 techniques (Image processing & Machine learning)

- 1) After performing the above techniques successfully we thought of combining the 2 techniques and verifying the result.
- 2) We first labeled the dataset and trained the YOLOv5 Model with that dataset.
- 3) After that, we applied the Median Blur technique of image processing on the same dataset and then trained the YOLOv5 model.
- 4) After that, we applied Erosion on the dataset obtained from step 3 and then again trained the model.
- 5) Finally, we compared the result in all the 3 cases and this gave us information on how image pre-processing can affect the accuracy of the deep learning model.

These are some of the output images of YOLOv5 model in the following respective order:

a) Normal Image Dataset b) Image processing techniques applied





- **IMPLEMENTATION DETAILS**

Tools and Technologies Used:

1. Python
2. OpenCv
3. Roboflow
4. Jupyter Notebook
5. Vscode
6. Darknet framework
7. CNN

Dataset:

DATASETS

LINKS

1. <https://www.kaggle.com/chitholian/annotated-potholes-dataset>
2. <https://www.kaggle.com/sachinpatel21/pothole-image-dataset>

Description

A zipped folder containing 800+ .jpg images of the Road with Potholes.

Note: These images are web scrapped from google, it might have some noisy or duplicate images.

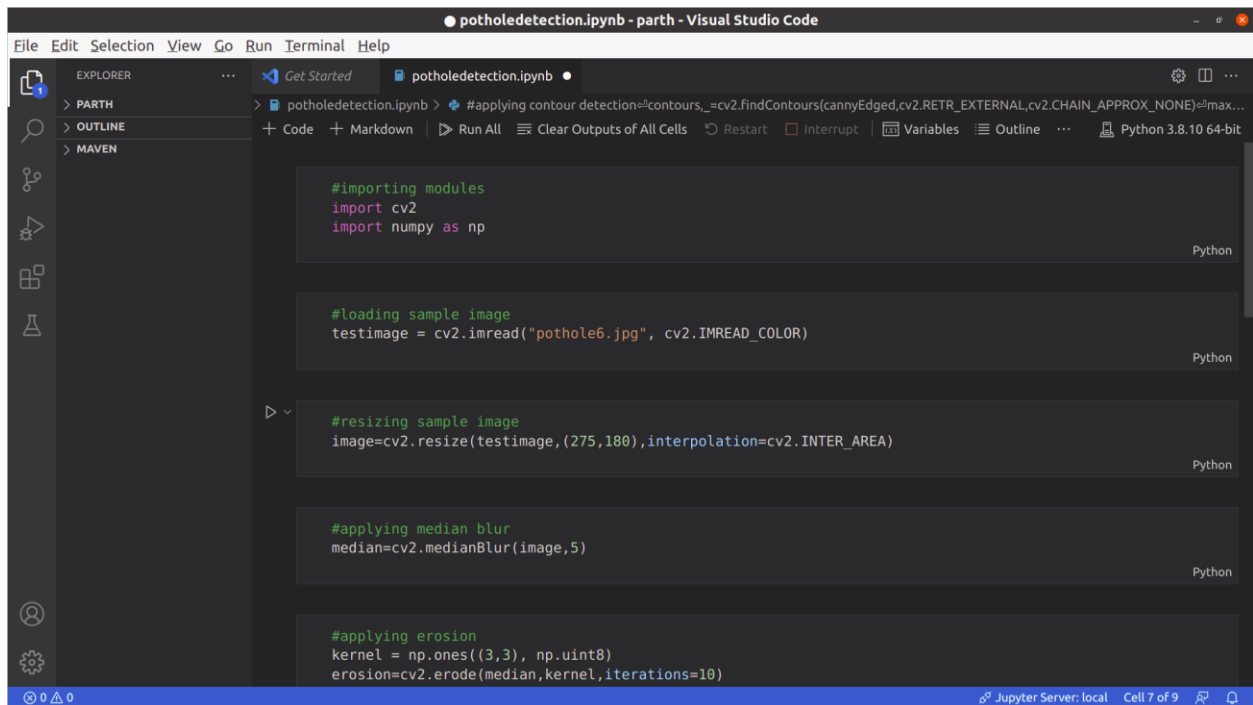
SAMPLE IMAGES:

10



5.3 Code Snippets:

Image Processing Technique:



```
File Edit Selection View Go Run Terminal Help

potholedetection.ipynb - parth - Visual Studio Code

EXPLORER
> PARTH
> OUTLINE
> MAVEN

Get Started potholedetection.ipynb
+ Code + Markdown | Run All | Clear Outputs of All Cells | Restart | Interrupt | Variables | Outline | Python 3.8.10 64-bit

# importing modules
import cv2
import numpy as np

Python

# loading sample image
testimage = cv2.imread("pothole6.jpg", cv2.IMREAD_COLOR)

Python

# resizing sample image
image = cv2.resize(testimage, (275, 180), interpolation=cv2.INTER_AREA)

Python

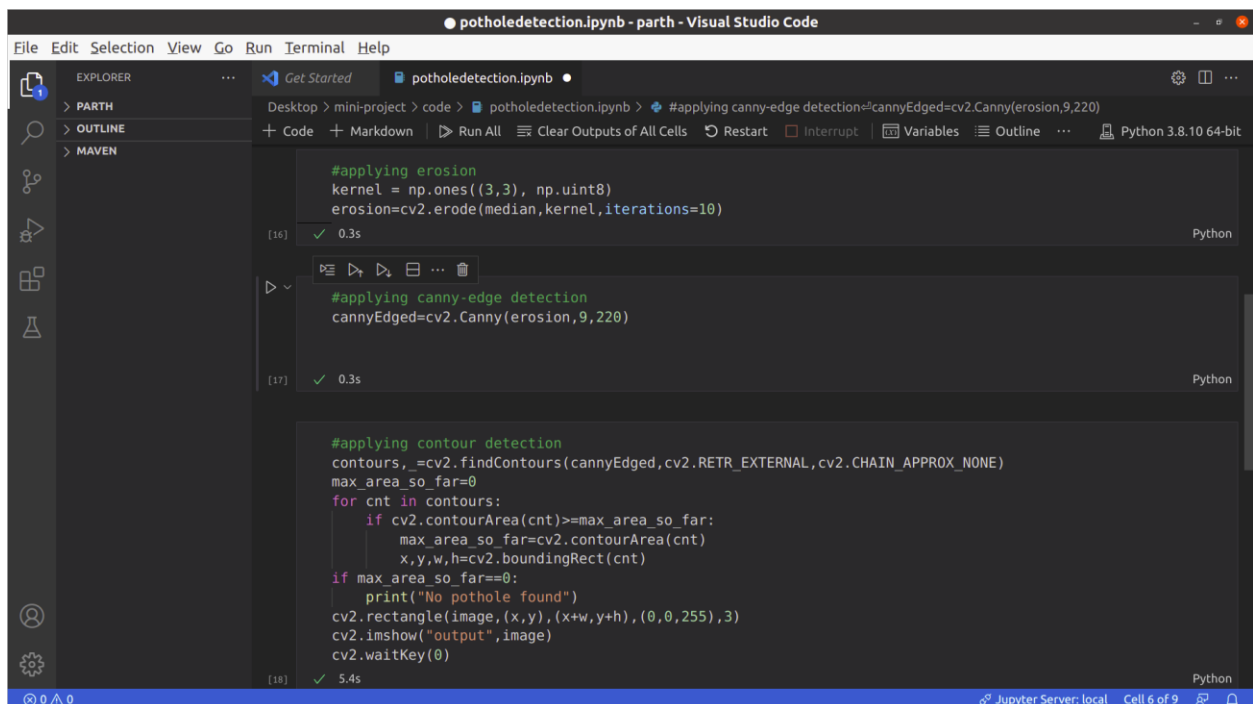
# applying median blur
median = cv2.medianBlur(image, 5)

Python

# applying erosion
kernel = np.ones((3, 3), np.uint8)
erosion = cv2.erode(median, kernel, iterations=10)

Python

0 0 Jupyter Server: local Cell 7 of 9
```



```
potholedetection.ipynb - parth - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER
> PARTH
> OUTLINE
> MAVEN

Get Started potholedetection.ipynb
+ Code + Markdown | Run All | Clear Outputs of All Cells | Restart | Interrupt | Variables | Outline | Python 3.8.10 64-bit

Desktop > mini-project > code > potholedetection.ipynb > # applying canny-edge detection=cannyEdged=cv2.Canny(erosion,9,220)

# applying erosion
kernel = np.ones((3, 3), np.uint8)
erosion = cv2.erode(median, kernel, iterations=10)

[16] ✓ 0.3s Python

# applying canny-edge detection
cannyEdged = cv2.Canny(erosion, 9, 220)

[17] ✓ 0.3s Python

# applying contour detection
contours, _ = cv2.findContours(cannyEdged, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
max_area_so_far = 0
for cnt in contours:
    if cv2.contourArea(cnt) >= max_area_so_far:
        max_area_so_far = cv2.contourArea(cnt)
        x, y, w, h = cv2.boundingRect(cnt)
if max_area_so_far == 0:
    print("No pothole found")
cv2.rectangle(image, (x, y), (x+w, y+h), (0, 0, 255), 3)
cv2.imshow("output", image)
cv2.waitKey(0)

[18] ✓ 5.4s Python

0 0 Jupyter Server: local Cell 6 of 9
```

• RESULTS AND COMPARISON WITH THE EXISTING WORK

Experiment-1: Image Processing Technique

Testing set: 34 images

Steps Performed	Images	% accuracy
Image processing techniques 1)Median blur 2)Erosion 3) Canny Edge detection 4)Contour Detection 5)Bounding box	16/34 images showed correct potholes	47.05

Experiment-2: Deep Learning Approach(YOLO)

The dataset used for this approach was divided into following set:

Training set: 734 images

validation set: 72 images

Testing set: 34 images

Steps Performed	Images	% accuracy
Without pre-processing on images	23/34 images showed correct potholes	67.647
Applied median-blur on images	25/34 images showed correct potholes	73.529
Applied median-blur+erosion on images	26/34 images showed correct potholes	76.470