Smart Internz

Pothole Detection using YOLOv5

1 INTRODUCTION

1.1 Overview

Potholes are one of the greatest problems on the roads. Stuck water on the roads and overloaded vehicles are mainly responsible for surface decay and erosion of rocks under the road surface which creates potholes that cause a lot of accidents and risks for general people. A system is needed that will detect potholes not only to alert the drivers but also to alert the authorities. In this paper, we emphasized on detecting the potholes using pothole image data and normal road image data. At first, we collected the data and then preprocessed them by resizing and rescaling.

We utilized the YOLOv5 algorithm for the detection of potholes in the collected image data. YOLOv5 (You Only Look Once) is a popular real-time object detection algorithm known for its accuracy and speed. It operates by dividing the input image into a grid and predicting bounding boxes and class probabilities for each grid cell.

To train the YOLOv5 model, we annotated the collected image data by manually labeling the potholes and normal road areas. This involved marking the boundaries of potholes in the images to create bounding box annotations. We ensured a diverse range of pothole types, sizes, and road conditions were included in the dataset to enhance the model's robustness.

Following the annotation process, we split the dataset into training and validation sets to evaluate the performance of the trained model. To improve the learning capability of the model, we employed data augmentation techniques such as rotation, translation, and flipping. These techniques artificially increased the size of the training set and helped prevent overfitting.

After training the model, we evaluated its performance on the validation set. We measured metrics such as precision, recall, and F1 score to assess the model's ability to correctly detect potholes while minimizing false positives and false negatives. We also conducted additional experiments to analyze the model's performance under various lighting conditions, weather scenarios, and road types.

In conclusion, our approach focused on using YOLOv5, a state-of-the-art object detection algorithm, to detect potholes in real-time using image data. This system has the potential to significantly reduce accidents and risks associated with potholes, providing a safer and more efficient road network for all users.

1.2 Purpose

The purpose of this study is to address the pressing issue of potholes on roads by developing a reliable and efficient system for their detection using image data. Our primary objective is to enhance road safety by alerting both drivers and authorities about the presence of potholes, thereby reducing accidents and potential risks to the general public. By leveraging the YOLOv5 algorithm, we aim to achieve accurate and real-time detection of potholes, allowing for immediate action and timely repairs to ensure smooth and hazard-free road surfaces. Another important purpose of this research is to establish a comprehensive dataset of pothole images, encompassing various road conditions, pothole types, and sizes, to train and validate the detection model effectively. Through the development of this system, we aim to minimize the damage caused by potholes to vehicles, thereby reducing maintenance costs for individuals and transportation authorities.

2 LITERATURE SURVEY

2.1 Existing problem

Several studies have been conducted on deep learning-based pothole detection approaches using various machine learning-based object detection algorithms. A comparative study by Roopak Rastogi [1] et al. Al. An accuracy of 87% is achieved using the YOLO version 2 object detection algorithm. Muhammad Harun Assad et al. [31] developed an algorithm that uses YOLO v4 as the base object detection algorithm, trained on a custom dataset, and achieved 90% accuracy at 31.76 FPS. Zhang et al. [32] proposed an embedded system for road obstruction detection integrated with CNN using the Montreal Pavement dataset. The model shows that the true positive rates for potholes, patches, marks, linear cracks, and crack networks are 75.7%, 84.1%, 76.3%, 79.4%, and 83.1%, respectively. Oche et al. [33] used five binary classification models (Naive Bayes, Logistic Regression, SVM, K-Nearest Neighbors (KNN), and Random Forest Tree) to apply various classifications to data collected via smartphone and car routes. They presented a comparison of machine learning approaches. The Random Forest Tree and KNN achieved the highest accuracy of 0.8889 on the test set. To improve the accuracy of the Random Forest Tree, they tuned hyperparameters and increased accuracy up to 0.9444. The model has shown promising results on different routes and out of sample data. Arbawa et al. [34] proposed a method for detecting road potholes using the gray-level co-occurrence matrix (GLCM) feature extractor and support vector machine (SVM) as a classifier. They analyzed three features such as contrast, correlation, and dissimilarity. The results have shown that a combination of contrast and dissimilarity features exhibits better results with an accuracy of 92.033% and computing time of 0.0704 seconds per frame. Chen et al. [35] proposed a novel location-aware convolutional neural network and trained on a public pothole dataset that consists of 4,026 images as training samples and 1,650 images as test samples. The proposed model is based on 2D-vision techniques and location-aware convolutional networks. CNN networks consist of two main subnetworks; the first localization subnetwork (LCNN) finds as many candidate regions as possible by employing a high recall network model, and the second part-based subnetwork (PCNN) performs classification on the candidates on which the network is expected to focus.

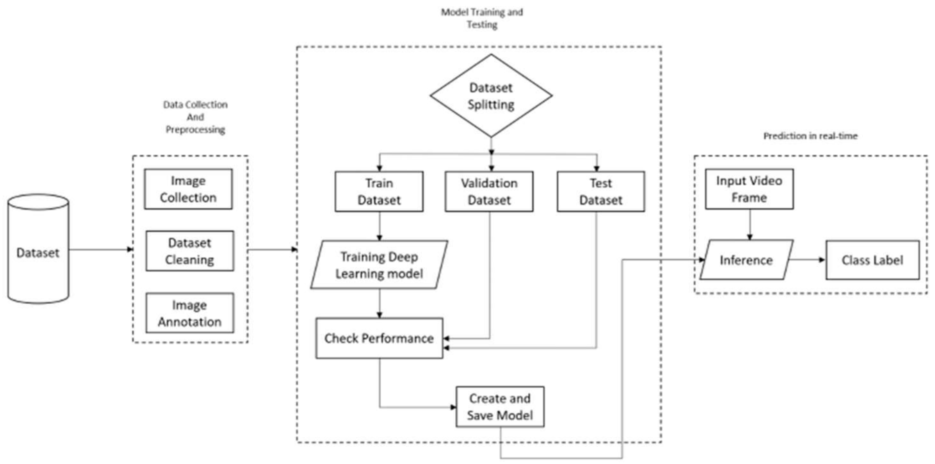
2.2 Proposed solution

Improving the safety of traffic is an important issue in Indian telecom services (ITC). This proposed model is a pothole detection system based on machine learning and image processing using the YOLO Algorithm. This system will help to detect the potholes present on the road and give a warning to the driver. The motivation here is to serve humanity better with the help of technology.

3 THEORITICAL ANALYSIS

3.1 Block diagram

Diagrammatic overview of the project.



3.2 Hardware / Software designing

In terms of hardware, a computer or server with a powerful GPU is necessary to handle the computational demands of training and running the YOLOv5 algorithm efficiently. GPUs such as NVIDIA GeForce RTX series or Tesla series are recommended for faster training and inference.

As for software requirements, the following components are necessary:

1. Operating System: Any major operating system such as Windows, Linux, or macOS can be used.

2. Python: The system requires a Python programming environment (preferably version 3.6 or higher) to execute the necessary code.

3. Deep Learning Framework: Install PyTorch, which is a popular deep learning framework used for implementing YOLOv5 and other neural network models.

4. YOLOv5: Clone or download the YOLOv5 repository from the official GitHub page (https://github.com/ultralytics/yolov5) to access the necessary code and scripts for the detection model.

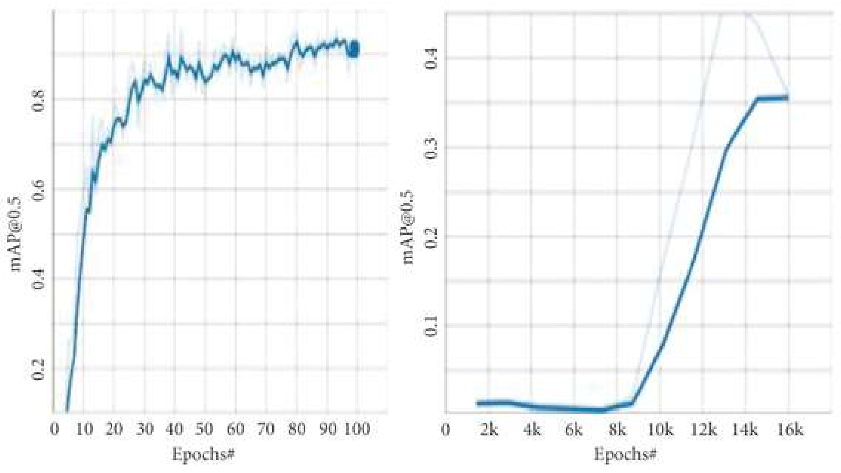
5. Dependencies: Install the required Python packages and dependencies, including NumPy, OpenCV, and other necessary packages.

Additionally, to annotate and preprocess the image data, image editing software or tools can be used, such as Adobe Photoshop, GIMP, or specialized annotation tools like LabelImg or RectLabel.

Overall, a powerful GPU, along with the appropriate software stack, will enable efficient training and inference of the YOLOv5-based pothole detection system.

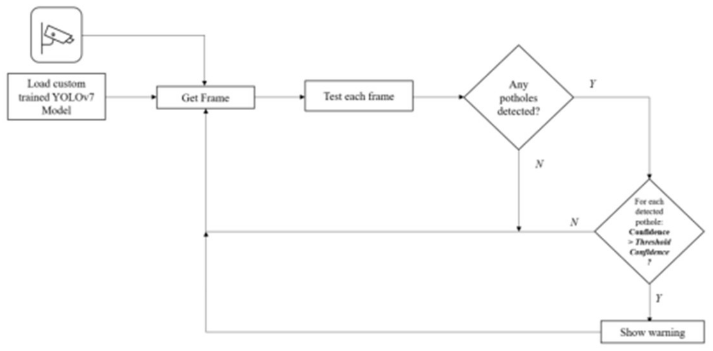
4 EXPERIMENTAL INVESTIGATIONS

The training of the YOLO and its variants are carried out on a system having Intel (R) Xeon (R) CPU at 3.0 GHz, RAM of 64 GB, and NVIDIA Titan Xp GPU. The dataset is split into 80% (1,066 images) training of the model and 20% (264 images) for the testing with labels of each image. The files needed for training are obj.names (names of the classes), obj.data (the number of classes), a path to train, test, and a backup folder. The backup folder saves the weights after every 100th iteration. The major file required for training is the configuration file which changes according to the model requirements. In our case, each model is trained for 20,000 max iterations with the batch size of 64 having subdivisions of 32 and a learning rate of 0.001 enclosed in the .cfg configuration file. The filters are set to 18 according to the formula filter size = (class+5) $*$ 3 where class = 1 in a .cfg file. For the training of YOLOv5, same parameters are used. According to the model and performance, we used pretrained weights from Google. YOLOv5 has less training time as compared to the rest of the YOLO models as 1 hour max.



5 FLOWCHART

Diagram showing the control flow of the solution

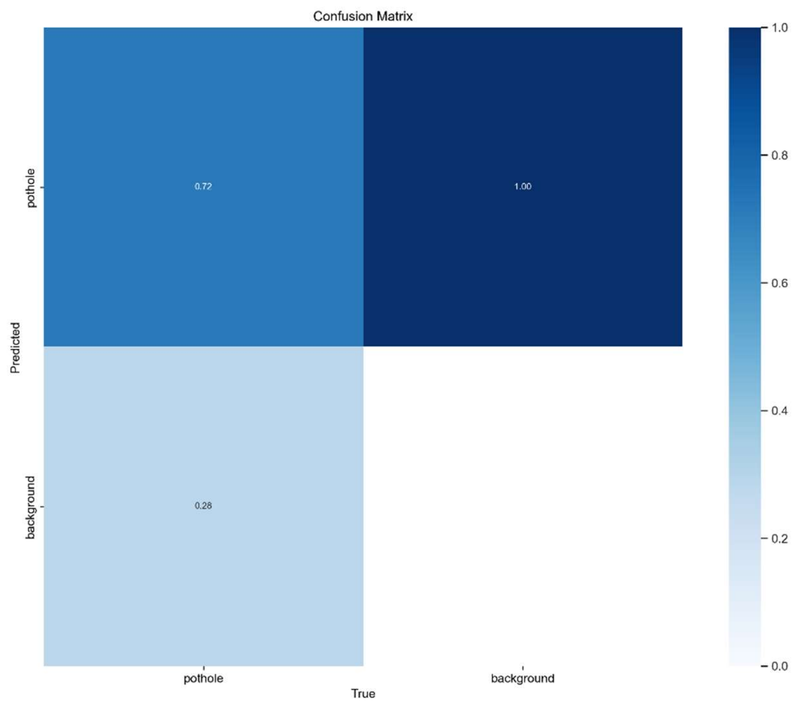

Load custom trained YOLOv7 Model
Get Frame
Test each frame
Any potholes detected?
Y
N
For each detected pothole: Confidence > Threshold Confidence ?
N
Y
Show warning

6 RESULTS

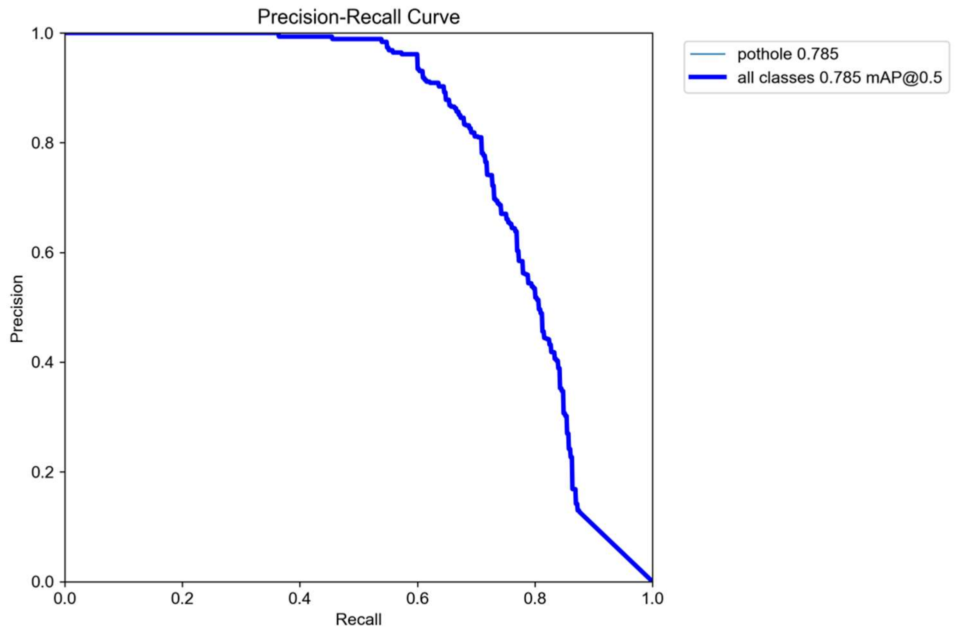
We measure key metrics such as precision, recall, and F1 score to assess the accuracy of the model.

Below are a few images of our results that show the accuracy of the pothole detection using YOLO v5.

Pothole Detection using YOLOv5

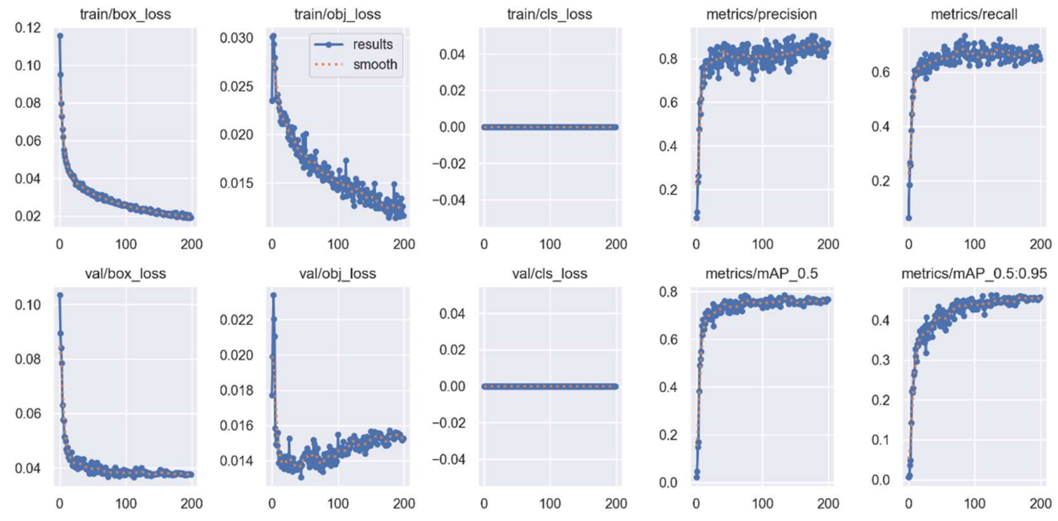


Above figure shows the Confusion Matrix, it says 72% of the potholes in images were identified correctly.

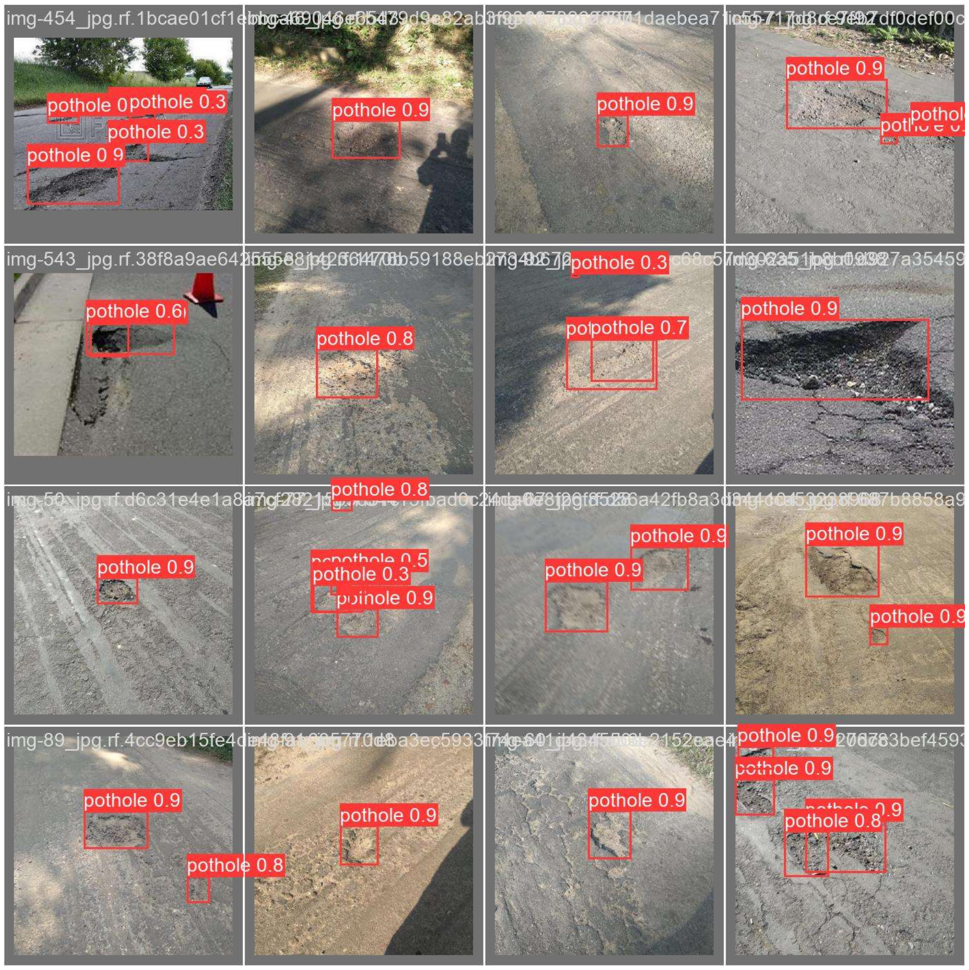


The precision-recall curve demonstrates the trade-off between the precision (accuracy of positive predictions) and recall (ability to detect all positive instances) of our pothole detection system, providing insights into its performance across different thresholds.

Pothole Detection using YOLOv5



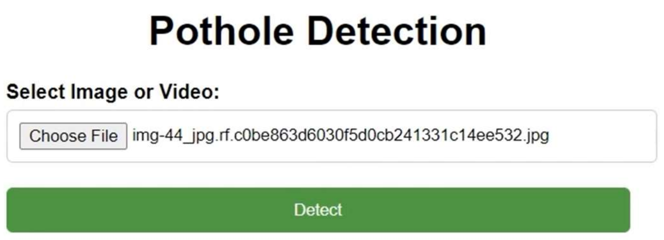
Above figure shows the trend in the loss of various parameters during training.



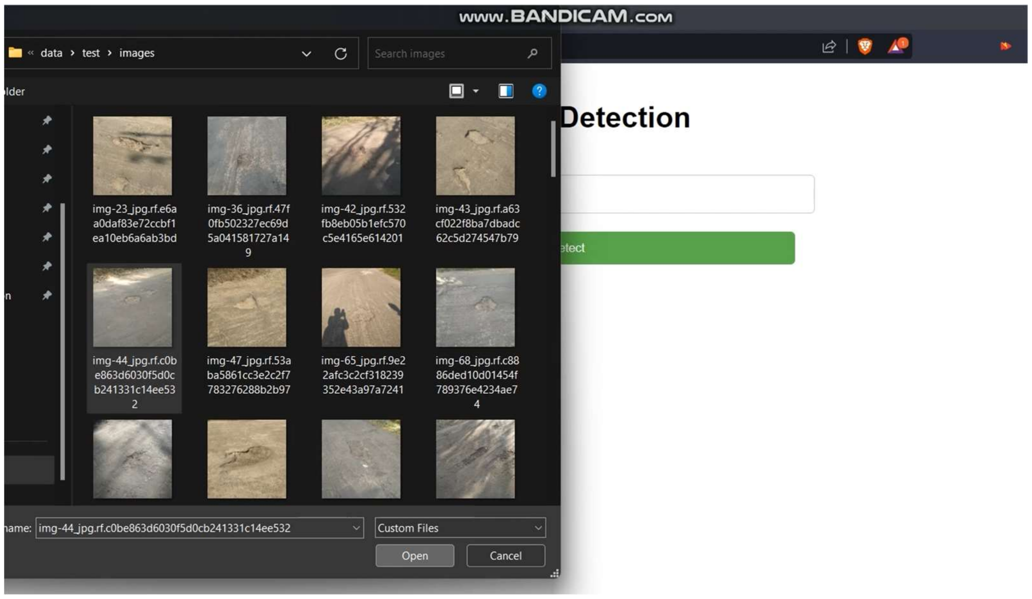
This image shows the results of the validation phase.

Visual examples of detected potholes in both images and videos are provided, demonstrating the effectiveness of our system in identifying and alerting the presence of potholes.

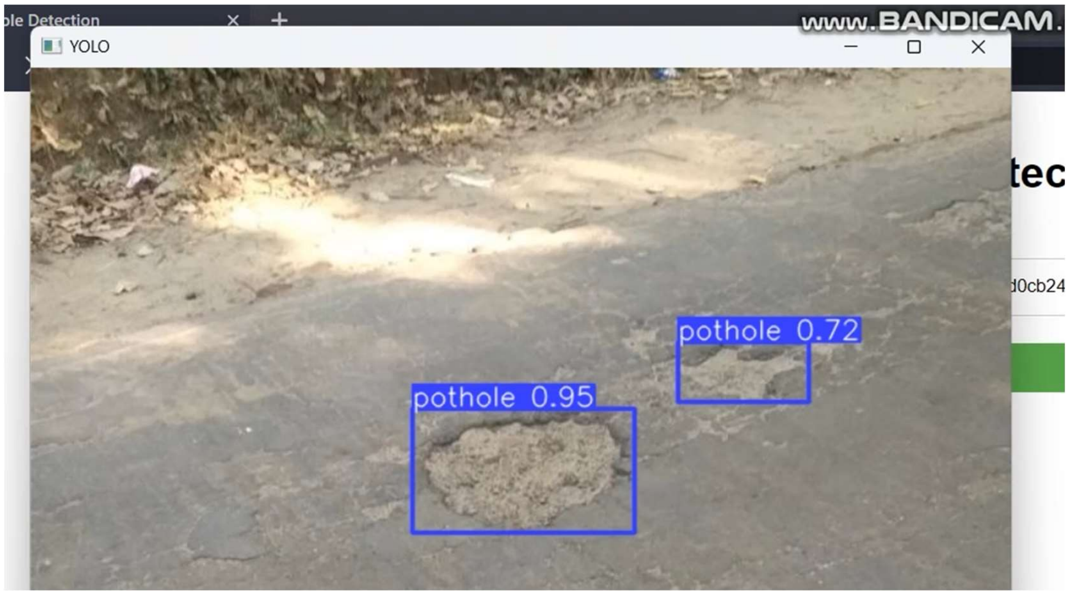
Pothole Detection using YOLOv5



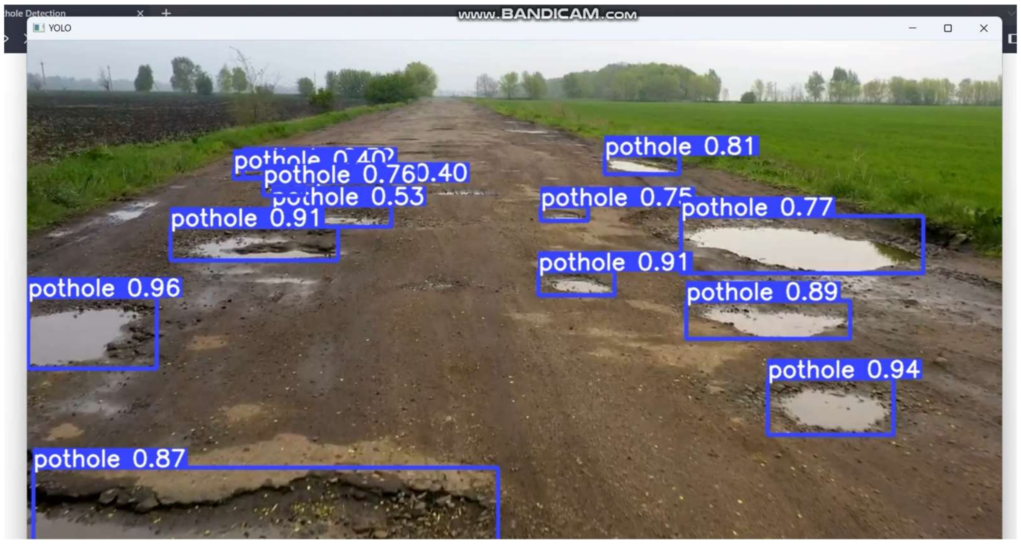
This is the front-end page made using Flask, wherein the user can insert an image or video to detect potholes.


www.BANDICAM.com
data › test › images
Search images
Detection
img-23_jpg.rf.e6aa0daf83e72ccbf1ea10eb6a6ab3bd
img-36_jpg.rf.47f0fb502327ec69d5a041581727a149
img-42_jpg.rf.532fb8eb05b1efc570c5e4165e614201
img-43_jpg.rf.a63cf022f8ba7dbadc62c5d274547b79
img-44_jpg.rf.c0be863d6030f5d0cb241331c14ee532
img-47_jpg.rf.53aba5861cc3e2c2f7783276288b2b97
img-65_jpg.rf.9e22afc3c2cf318239352e43a97a7241
img-68_jpg.rf.c8886ded10d01454f789376e4234ae74
name: img-44_jpg.rf.c0be863d6030f5d0cb241331c14ee532
Custom Files
Open
Cancel

Detection of potholes in images:



Detection of potholes in videos:



As we can infer from the screenshots above, the model is detecting potholes with a pretty good accuracy.

7 ADVANTAGES & DISADVANTAGES

List of advantages and disadvantages of the proposed solution

Advantages:

1. Improved road safety: Potholes are a common road hazard that can cause accidents, vehicle damage, and injury to drivers and pedestrians. Detecting potholes allows for timely repairs, reducing the risk of accidents and improving overall road safety.
2. Preventive maintenance: Early detection of potholes enables proactive maintenance strategies. By identifying and repairing potholes in their early stages, before they become larger and more dangerous, transportation authorities can save on costly repairs and minimize disruptions to traffic flow.
3. Cost savings: Detecting and repairing potholes promptly can help reduce long-term maintenance costs. By addressing smaller potholes early on, transportation agencies can prevent the need for more extensive repairs that become necessary when potholes worsen over time.
4. Enhanced road infrastructure planning: Pothole detection data can provide valuable insights for road infrastructure planning and management. By analyzing the frequency, distribution, and characteristics of potholes, transportation authorities can identify areas with recurring issues and prioritize road improvement projects accordingly.
5. Efficient resource allocation: Pothole detection allows transportation agencies to allocate their resources more efficiently. Instead of conducting routine inspections or relying on

manual reports, automated detection systems can identify potholes, enabling maintenance crews to focus their efforts on specific locations that require attention.

Disadvantages:

1. Complexity of detection: Pothole detection is a complex task due to the variability in pothole shapes, sizes, and appearances. Potholes can vary from small cracks to larger depressions, and their visibility may be affected by factors such as lighting conditions, shadows, and occlusions. Developing accurate and robust detection algorithms that can handle these variations can be challenging.
2. False positives and negatives: Pothole detection algorithms may produce false positives, incorrectly identifying non-pothole objects as potholes. This can lead to unnecessary maintenance and repair efforts. Conversely, false negatives occur when actual potholes are missed, potentially resulting in delayed repairs and compromised road safety. Achieving a balance between sensitivity and specificity is crucial to minimize false detections.
3. Annotation and training data: Training a pothole detection system requires a large dataset of annotated images or videos. Manually annotating such data can be time-consuming and expensive, as it involves accurately labeling potholes in various road conditions and perspectives. Acquiring diverse and representative training data can be challenging, particularly for rare or specific types of potholes.
4. Deployment and maintenance costs: Implementing a pothole detection system requires initial investment in hardware, software, and infrastructure. There may be costs associated with installing and maintaining cameras or sensors, developing and deploying the detection algorithms, and integrating the system with existing road infrastructure management systems. Ongoing maintenance and calibration of the system are also necessary for reliable performance.
5. Environmental and operational challenges: Pothole detection systems can face environmental challenges such as adverse weather conditions, poor lighting, and extreme temperatures, which can affect their performance. Operational challenges may arise from factors like heavy traffic, dynamic road conditions, and changes in infrastructure. Ensuring the robustness and reliability of the system under various operational environments can be demanding.

8 APPLICATIONS

The applications of our pothole detection system are diverse and impactful. Firstly, it can be integrated into vehicles, providing real-time alerts to drivers, thereby preventing accidents and vehicle damage. Secondly, transportation authorities can utilize this system to monitor road conditions and prioritize pothole repairs efficiently. Additionally, our system can be employed in smart city initiatives, where automated detection and reporting of potholes enable swift maintenance actions. Furthermore, insurance companies can leverage this technology to assess road conditions and offer customized policies. Ultimately, our system contributes to safer roads, cost-effective maintenance, and improved overall transportation infrastructure.

9 CONCLUSION

In conclusion, our study successfully developed a pothole detection system utilizing the YOLOv5 algorithm, which was trained on a dataset sourced from Roboflow. This dataset consisted of 665 images, divided into 465 training, 133 validation, and 67 test images. The YOLOv5s model was trained for 200 epochs, yielding promising results.

The evaluation of the trained model showed a precision of 0.87, indicating a high accuracy in identifying potholes, while the recall score was 0.65, indicating the model's ability to detect a significant portion of the actual potholes. The mean Average Precision (mAP) at a threshold of 0.5 was calculated as 0.76, further confirming the model's effectiveness.

It's important to note that the dataset used in this study contained only one class, which was Pothole. The focus on this single class allowed us to train the model specifically for pothole detection, resulting in improved accuracy and performance.

The successful implementation of the YOLOv5 model trained on this dataset demonstrates the feasibility and potential of using deep learning techniques for pothole detection. Further research and improvements in dataset size, model architecture, and training duration can enhance the system's precision, recall, and overall performance. The integration of this system into real-world applications can contribute to safer roads, improved maintenance practices, and a more efficient transportation infrastructure.

10 FUTURE SCOPE

In this paper, we proposed an efficient method to recognize a pothole on a road from the viewpoint of cost and implementation. The future scope of our pothole detection system involves expanding the dataset, exploring advanced machine learning techniques, integrating geolocation technologies, and collaborating with stakeholders to improve accuracy, efficiency, and scalability. Further research can focus on detecting other road hazards, integrating with autonomous vehicles, and implementing the system on a larger scale for proactive road maintenance. By continuously refining and enhancing the system, we can contribute to safer roads and more efficient transportation networks in the future.

11 BIBILOGRAPHY

Roboflow Dataset: Pothole Detection. Retrieved from :
https://public.roboflow.com/object-detection/pothole/1

PyTorch: Tensors and Dynamic Neural Networks in Python with Strong GPU Acceleration. Retrieved from **https://pytorch.org/**

LabelImg GitHub Repository. Retrieved from
https://github.com/tzutalin/labelImg

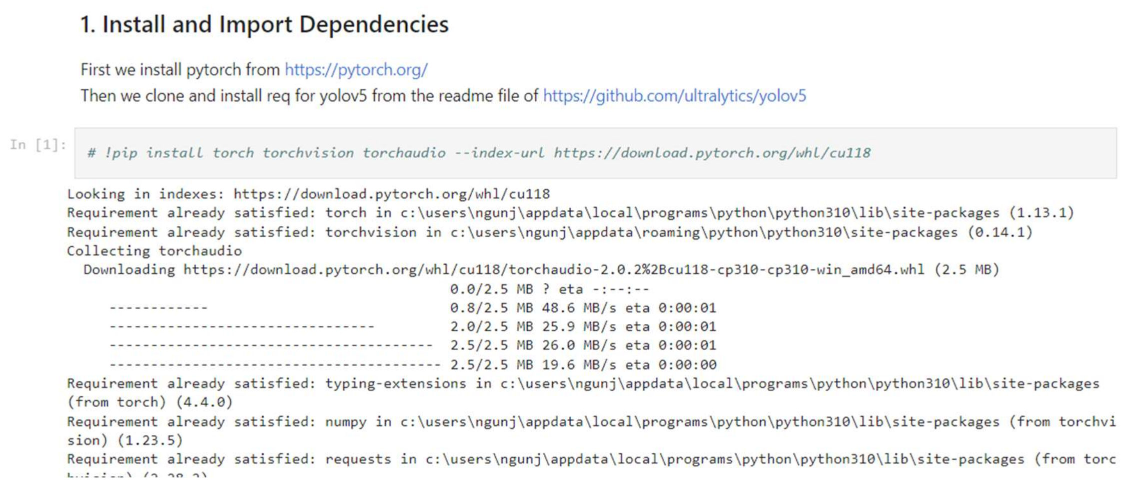
OpenCV: Open Source Computer Vision Library. Retrieved from
https://opencv.org/

YOLOv5 : **https://github.com/ultralytics/yolov5**

Custom data documentation :
https://docs.ultralytics.com/yolov5/tutorials/train_custom_data/

APPENDIX

A. Source Code



1. Install and Import Dependencies

First we install pytorch from https://pytorch.org/
Then we clone and install req for yolov5 from the readme file of https://github.com/ultralytics/yolov5

```
In [1]: # !pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118
```

```
Looking in indexes: https://download.pytorch.org/whl/cu118
Requirement already satisfied: torch in c:\users\ngunj\appdata\local\programs\python\python310\lib\site-packages (1.13.1)
Requirement already satisfied: torchvision in c:\users\ngunj\appdata\roaming\python\python310\site-packages (0.14.1)
Collecting torchaudio
  Downloading https://download.pytorch.org/whl/cu118/torchaudio-2.0.2%2Bcu118-cp310-cp310-win_amd64.whl (2.5 MB)
                                              0.0/2.5 MB ? eta -:--:--
     ------------                             0.8/2.5 MB 48.6 MB/s eta 0:00:01
     --------------------------------         2.0/2.5 MB 25.9 MB/s eta 0:00:01
     ---------------------------------------  2.5/2.5 MB 26.0 MB/s eta 0:00:01
     ---------------------------------------- 2.5/2.5 MB 19.6 MB/s eta 0:00:00
Requirement already satisfied: typing-extensions in c:\users\ngunj\appdata\local\programs\python\python310\lib\site-packages
(from torch) (4.4.0)
Requirement already satisfied: numpy in c:\users\ngunj\appdata\local\programs\python\python310\lib\site-packages (from torchvi
sion) (1.23.5)
Requirement already satisfied: requests in c:\users\ngunj\appdata\local\programs\python\python310\lib\site-packages (from torc
```

Pothole Detection using YOLOv5

```
In [1]:
import torch
import matplotlib.pyplot as plt
import numpy as np
import cv2
```

2. Load Model

Load model from pytorch hub ie. https://pytorch.org/hub/ in that https://pytorch.org/hub/ultralytics_yolov5/
The code written below is taken from the above website

```
In [2]:
model = torch.hub.load('ultralytics/yolov5', 'yolov5s')
```

```
Using cache found in C:\Users\ngunj/.cache\torch\hub\ultralytics_yolov5_master
requirements: Ultralytics requirement "gitpython>=3.1.30" not found, attempting AutoUpdate...
Requirement already satisfied: gitpython>=3.1.30 in c:\users\ngunj\appdata\roaming\python\python310\site-packages (3.1.30)
Requirement already satisfied: gitdb<5,>=4.0.1 in c:\users\ngunj\appdata\local\programs\python\python310\lib\site-packages (fr
om gitpython>=3.1.30) (4.0.9)
Requirement already satisfied: smmap<6,>=3.0.1 in c:\users\ngunj\appdata\local\programs\python\python310\lib\site-packages (fr
om gitdb<5,>=4.0.1->gitpython>=3.1.30) (5.0.0)

requirements: 1 package updated per C:\Users\ngunj\.cache\torch\hub\ultralytics_yolov5_master\requirements.txt
requirements:  Restart runtime or rerun command for updates to take effect

YOLOv5  2023-6-28 Python-3.10.0 torch-2.0.1+cpu CPU

Fusing layers...
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients
Adding AutoShape...
```

```
In [3]:
model
```

```
Out[3]: AutoShape(
```

6. Load Model

```
In [3]:
yolo_model = torch.hub.load('ultralytics/yolov5', 'custom', path='yolov5/runs/train/exp5/weights/last.pt', force_reload=True
```

```
Downloading: "https://github.com/ultralytics/yolov5/zipball/master" to C:\Users\ngunj/.cache\torch\hub\master.zip
YOLOv5  2023-7-1 Python-3.10.0 torch-2.0.1+cpu CPU

Fusing layers...
Model summary: 157 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
Adding AutoShape...
```

```
In [4]:
img = os.path.join('data','test','images','img-43_jpg.rf.a63cf022f8ba7dbadc62c5d274547b79.jpg')

results= yolo_model(img)
results
```

```
Out[4]: YOLOv5 <class 'models.common.Detections'> instance
image 1/1: 720x720 3 potholes
Speed: 11.0ms pre-process, 160.1ms inference, 1.0ms NMS per image at shape (1, 3, 640, 640)
```

```
In [9]:
results.show()
```