Vision Based Machine Learning Model (YOLO) for Pothole Detection with Drones

Artin Yahyapour[1], Hadiseh Babazadeh[2]
[1]B.Sc Student, Faculty of Electrical Engineering, Urmia University of Technology, Urmia, Iran
[2]Assistant Professor, Faculty of Electrical Engineering, Urmia University of Technology, Urmia, Iran

Abstract

With advancing technology and using of automatic cars, the need for detecting road problems increases. Real time image processing and machine learning based pothole detection of roads is presented suitable for road monitoring drones. The camera taken videos are converted to frames, and YOLO V8 model, with manually labeled pictures, is used for training. Simulation results shows that the model can successfully identify potholes after the successful training of the system. Coordinates of the detected potholes are sent via SMS, or to a predefined server according to the user choice.

Keywords : YOLO, Deep Learning, Image Processing, Pothole Detection.

1. Introduction

A pothole is a type of road surface defect that forms when water penetrates into cracks in the pavement, freezes, and expands. This expansion causes the pavement to break apart, creating a hole. As vehicles drive over these weakened areas, the pressure further breaks down the surface, enlarging the hole and forming a pothole. Potholes can vary in size and depth and often create hazards for drivers and pedestrians, potentially causing damage to tires, wheels, suspension systems, and other parts of a vehicle and increasing the risk of accidents caused by swerving to avoid potholes or losing control when hitting one. Also, hitting a pothole can be an uncomfortable experience for passengers.

Repairing potholes typically involves cleaning the damaged area and filling it with new asphalt or other suitable materials, but first of all there must be a way to detect them, especially to be used by automatic cars[1,2]. Several ways are used for detection, such as human reports, using accelerometer sensors of cars, lidar systems, machine learning based algorithms, and etc. While traditional methods for pothole detection are costly and time consuming [3], a new method, YOLO, solves this issue using machine learning to provide real-time warnings to drivers[4]. This system has shown promising results, with an accuracy of 94.5% [5]. YOLO, or You Only Look Once, is a real-time object detection system with a range of applications [6], from medical object detection, face recognition,behaviour and object detection, motion tracking, emotion tracking, handwriting detection [7-0] to object detection in challenging weather conditions, particularly for self-driving cars[10]. These studies collectively demonstrate the versatility and potential of YOLO in various fields.

In this work, vision-based detection of potholes is of interest supposing a camera equipped drone. The drone has to take video, analyze the road surface, detect potholes using machine learning, and report the position of road damage to a center. To increase the speed frames with no object are eliminated from process step. The proposed code can also be used by drivers, as well as roads controlling and monitoring centers. To implement the idea YOLO V8 is used.

The training process is done using Roboflow platform, which is described in the first step in the following. Then Custom training is brought. Next part includes code flow chart along with its step by step description. After that, simulation results of test on real camera film, is discussed to prove the code.

2. Integration with Roboflow

To start pothole detection, first of all, there must be a collection of pictures to start training. In this research, the training images were extracted from a

video. The video was processed to convert it into multiple frames, and then used for training.

Training is done using Roboflow which is a comprehensive platform that simplifies the process of creating, training, and deploying computer vision models. It provides tools for data annotation, data augmentation, and dataset management to make it easier to create high-quality datasets for machine learning projects [11]. After uploading the images, the user can use bounding boxes and polygons in the tagging section to lable them. Roboflow automatically divides the dataset into training, validation, and testing sets. Then YOLOv8 model was trained on the labeled images using Roboflow's platform providing performance metrics and visualizations.

3. Custome Training

Google Colab is used for custom training of the YOLOv8 model. In this environment, using training dataset images, the YOLOv8 model was trained for the segmentation task.

```
!yolo task=segment mode=train model=yolov8s-seg.pt
data={dataset.location}/data.yaml epochs=100
imgsz=640
```

The steps are as follows:

1. First, the yolov8s-seg.pt model file was downloaded. This file contains the initial weights for the YOLOv8 segmentation model.
2. The data file in data.yaml format, which includes the paths to the training images and their labels, was loaded.
3. The model was trained using the YOLO command with parameters specifying the number of training epochs and the image size (imgsz).
4. After completing the training, the trained model was saved as a .pt file, which can be used for evaluation and prediction.

4. Code Flow chart

The procedure of the code is shown step by step in Fig.1.

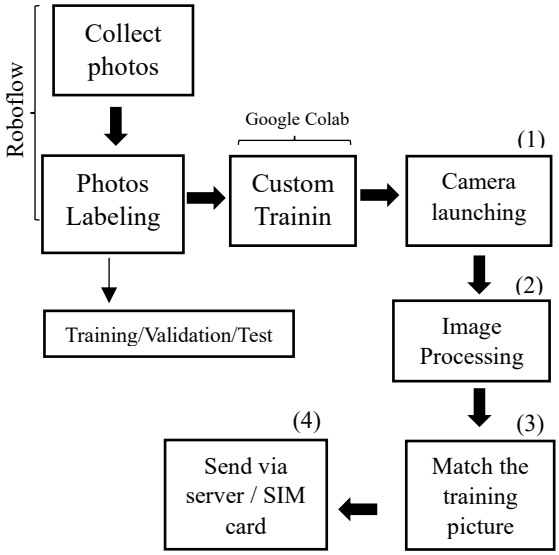

Fig 1 Code flow chart

After image labeling and custom training, it's time to analyze the real camera film taken from the road In block (1) the Raspberry Pi camera is initialized and set to capture images at a resolution of 1020x500 pixels.

```
camera = PiCamera()
camera.resolution = (1020, 500)
rawCapture = PiRGBArray(camera, size=(1020, 500))
```

In the image processing block, (2), the video is read frame by frame and every 10 frames, a frame is selected for processing. That frame is then resized to 1020x500 dimensions for faster processing:

```
    img = cv2.resize(img, (1020, 500))
    h, w, _ = img.shape
```

Next, a deep learning model, YOLO, is used to identify, predict, extract bounding boxes and masks, and categorize in the image. The detected objects are processed to extract bounding boxes and masks.

```
    for r in results:
        boxes = r.boxes
        masks = r.masks
```

In block (3), matching of frames with trained images is analyzed. the model results are processed to extract masks and bounding boxes:

```
for seg, box in zip(masks.data.cpu().numpy(), boxes):
```

The masks are resized to match the frame dimensions and contours of each mask are identified.

```
 seg = cv2.resize(seg, (w, h))
 contours, _ =
cv2.findContours((seg).astype(np.uint8),
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

Contours and class information are drawn on the image, and the processed frames are saved in a specified directory. For mask processing, OpenCV techniques are utilized.

```
cv2.polylines(img, [contour], True, color=(0, 0, 255),
thickness=2)
cv2.putText(img, c, (x, y - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)
```

In this research, the optimization of image processing for object detection is investigated using the Raspberry Pi camera and the YOLO model. One of the main goals of this research is to increase the processing speed by rejecting the frames in which no objects are detected. This process causes only frames with objects to be processed, thus improving system speed making it suitable for real time applications.

Finally, in block (4), the coordinates of the detected object should be reported. There are two possibilities: sending via SMS, or sending to the server.

To send via SMS, first the communication settings are made with the SIM card device. Then, a function is defined to send an SMS that sends the coordinates and frame number of the identified objects to the specified mobile number. Using AT commands, this function sends SMS containing x and y coordinates and frame number to the specified mobile number. First, the SMS mode is activated and then the SMS is sent.

```
message = f"Frame {frame_number}: Object detected at x={x},
y={y}"
send_sms(phone_number, count, x, y)
```

To send the processed images to the server, first the IP address and server port settings are made.

```
 S=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
 s.sendto(x_as_byte, (server_ip, server_port))
```

After image processing and object detection, the image is converted to JPEG format using the OpenCV library. The cv2.imencode function converts the image to an array of bytes. The compressed image is then converted to binary codes, so that it can be sent over the network. This is done using the pickle library. At the final step, the image is sent to the server using the UDP socket. The most important features are:

1- Compressing the image reduces data size, convenient for faster transmission,
2- Using UDP protocol, reducing the delay of data transmission,
3- Flexibility in adjusting the image compression quality according to application.

The YOLOv8 model is one of the most advanced and recent object recognition models, significantly improving recognition accuracy and speed using a lighter neural network architecture and optimization techniques like pruning and quantization. This model excels due to the integration of cutting-edge deep learning techniques and advanced optimizations, allowing it to perform exceptionally well under various conditions, including the presence of noise in the images.

5. Simulation Results

A video is used as input of the code to see its performance. The video is chopped into frames and for example in one of its frame there are some potholes. Fig. 2 shows the code result for such a frame.

Fig 2. Simulation result for a frame of video with detected potholes

As shown in Fig. 2, five objects (road potholes) were detected in the image. Detected Objects Coordinates are as follows:

Processing Speed: Preprocess time: 0.0ms / Inference time: 100.8ms /Postprocess time: 883.1ms

For frame 10, multiple objects (road potholes) are detected with the following coordinates:

Object 1: (x=256, y=270) Object 2: (x=619, y=138) Object 3: (x=288, y=138) Object 4: (x=198, y=163) Object 5: (x=632, y=188)

The output shows the model's efficiency and the exact positions of detected objects, indicating successful implementation of the object detection process.

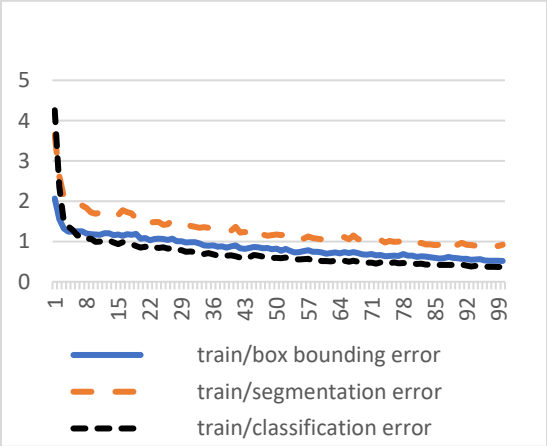

Fig 3 errors of training phase

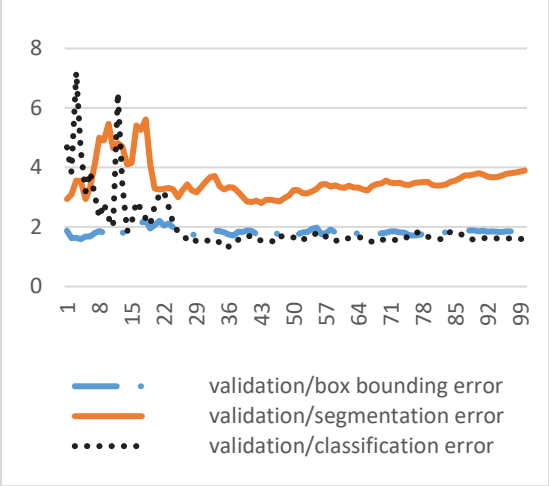

Fig 4 errors of validation phase

Figures 3, and 4 depict box bounding, segmentation and classification error during training and validation phases respectively.

Fig. 5, show the Precision-Confidence curve for the YOLOv8 model in detecting road potholes. The light line represents the model's performance for road potholes, and the dark one represents the performance for all classes. As the confidence level increases, the precision also increases and shows good prediction.

In Fig. 6, the Recall-Confidence curve is presented. The light line represents the model's performance for road potholes, and the dark one, for all classes. As the confidence level increases, recall generally decreases but has acceptable area under the curve.

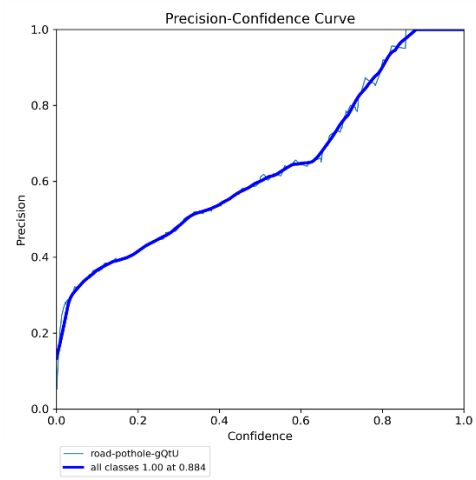

Fig 5 Precision-Confidence Curve

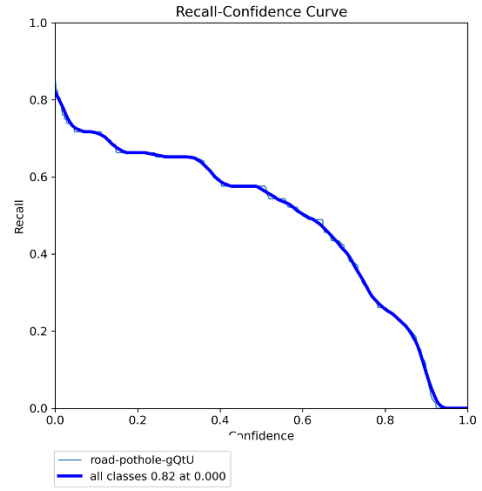

Fig 6 Recall-Confidence Curve

6. Conclusion

An image recognition and processing system based on the YOLOv8 machine learning model is introduces, which utilizes a Raspberry Pi-connected camera to receive and process images in real-time. Leveraging

the advanced YOLOv8 model, this system accurately and swiftly detects objects and extracts their coordinates. These coordinates are then efficiently transmitted via SMS or server to the intended destinations, enabling remote monitoring. Additionally, the system's capability to send images to a server facilitates the storage and subsequent analysis of data.

The use of more advanced deep learning and image processing techniques is suggested for future studies, making this system a powerful tool for addressing complex image processing challenges, even in noisy environments or bad wather conditions.

References

[1] Kavitha, R., and S. Nivetha. "Pothole and object detection for an autonomous vehicle using yolo." *2021 5th international conference on intelligent computing and control systems (ICICCS)*. IEEE, 2021.

[2] Chitale, Pranjal A., et al. "Pothole detection and dimension estimation system using deep learning (yolo) and image processing." 2020 35th International Conference on Image and Vision Computing New Zealand (IVCNZ). IEEE, 2020.

[3] Riya, P. D., K. R. Nakulraj, and A. A. Anusha. "Pothole detection methods." 2018 3rd International Conference on Inventive Computation Technologies (ICICT). IEEE, 2018.

[4] Fong, Simon, Nilanjan Dey, and Amit Joshi. "ICT Analysis and Applications Proceedings of ICT4SD 2023, Volume 2." Proceedings of ICT4SD 2 (2023):1.

[5] Lincy, A., et al. "Road Pothole Detection System." ITM Web of Conferences. Vol. 53. EDP Sciences, 2023.

[6] Park, Sung-Sik, Van-Than Tran, and Dong-Eun Lee. "Application of various yolo models for computer vision-based real-time pothole detection." Applied Sciences 11.23 (2021): 11229.

[7] Ragab, Mohammed Gamal, et al. "A Comprehensive Systematic Review of YOLO for Medical Object Detection (2018 to 2023)." IEEE Access (2024).

[8] Hegde, Shambhu, Harish V. Mekali, and Golla Varaprasad. "Pothole detection and inter vehicular communication." 2014 IEEE international conference on vehicular electronics and safety. IEEE, 2014.

[9] Kaushik, Vineet, and Birinderjit Singh Kalyan. "Pothole Detection System: A Review of Different Methods Used for Detection." 2022 Second International Conference on Computer Science, Engineering and Applications (ICCSEA). IEEE, 2022.

[10] He, Tianyu, and Yi Li. "An improved method for object detection in raining and foggy conditions for self-driving cars." Third International Conference on Artificial Intelligence and Electromechanical Automation (AIEA 2022). Vol. 12329. SPIE, 2022.

[11] Alexandrova, Sonya, Zachary Tatlock, and Maya Cakmak. "RoboFlow: A flow-based visual programming language for mobile manipulation tasks." 2015 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2015.