Using Computer Vision methods for Pothole Detection in Roads

DSE312: Project Report

Submitted by: Devashish Tripathi
Roll Number: 21093
Email: devashish21@iiserb.ac.in

Abstract

Potholes are a common cause of accidents and traffic congestion and must be dealt with urgently. This project involved studying past works and taking own approach towards the problem. Three approaches were taken: simple edge detection, a YOLOv5-based model and an AutoML model made using Roboflow 3.0. The analysis of past research shows excellent works and that the field has the potential for more research. The results indicated that the YOLOv5 approach works satisfactorily.

Introduction

With the number of road accidents increasing daily, it has become imperative that a solution to this severe problem is found. Among various reasons, such as rash driving, absence of proper barricading, etc., a prominent reason is the presence of Potholes in the roads. Potholes are road depressions caused by bad construction materials and heavy traffic, which lead to road accidents and traffic congestion. Potholes are prevalent on Indian roads, meaning the issue is very relevant.

In this project, past works were analysed. A basic algorithm utilising edge detection was implemented, along with a YOLOv5-based model and an AutoML model trained using the online resource Roboflow 3.0. The results indicated that YOLOv5 can be used to give satisfactory performance.

Past Literatures

For the project, several past works were consulted. Some of them include:

- **(Jo and Ryu 2015)** used a black-box camera to detect potholes. Images extracted from the live video feed were cropped, followed by Gray-scaling and thresholding using Otsu's method (Otsu 1979) (as cited in (Jo and Ryu 2015)). Candidate extraction was performed by line segmentation, lane detection, etc., to detect lanes to distinguish between potholes and similar dark regions such as tires. A static lane marking was used when lanes weren't present. Finally, a cascade detector was applied, based on suitable thresholds on width, variance, etc., to remove minor cracks, shades, etc. The method had a sensitivity of 71%, with 88% Precision.
- **(Shaghouri, Alkhatib, and Berjaoui 2021)** utilized the use of SSD, YOLOv3 and YOLOv4 for the real-time pothole detection. The images used were primarily of Lebanese roads. It used the TensorFlow framework to implement an SSD(Single Shot Multibox Detector) architecture and a Darknet framework to implement YOLOv3 and YOLOv4. The training data for all three was prepared using LabelImg software to create annotations, and Google Colab was used for implementation. YOLOv3 was trained six times, and the 3rd run gave the best results. YOLOv4 was trained twice, and the first run gave the best results. The statistics of the best runs of SSD, YOLOv3 and YOLOv4 are mentioned in Table 1.

Model	Prec.	Recall	mAP
SSD	73%	37.5%	32.5%
YOLOv3	84%	78%	83.62%
YOLOv4	85%	81%	85.39%

Table 1: Comparison of models used in (Shaghouri, Alkhatib, and Berjaoui 2021)

- **(Hu and Zhao 2010)** used ULBP for simplicity, scale and rotation invariance, dividing local patterns into five sub-classes. Uniform patterns were merged, and nonuniform patterns were converted to familiar uniform patterns. A roughness measurement estimated local variance in a window, with windows with a smaller variance seen as a plain area. A lookup table improved implementation and noise robustness. Assuming a crack to be composed of a strong edge and corners with a plain area inside, local patterns were redefined. An edge-finding step was performed after label assignment to each pixel, followed by a fake-crack-eliminating process based on area, pixels, and orientation. Experimental results detected cracks less than 1 mm wide.
- **(Maeda et al. 2018)** utilised the SSD algorithm to create a mobile app to detect road damage. A dataset consisting of 9053 images taken in various weather and illuminance conditions using a mobile camera affixed to

the dashboard of a car was produced. Based on a study done on the COCO dataset (Lin et al. 2015) (as cited in (Maeda et al. 2018)), the SSD model was trained on the MobileNet and Inception V2 frameworks for their relatively small CPU loads and low memory consumption. Both the models were trained on images reduced to a 300x300 size, and in each case, the dataset was divided randomly into 8:2(train: evaluation). Recall and Precision greater than 75% were achieved with an inference time of 1.5 s on a smartphone. MobileNet performed better than Inception V2. The results, the dataset produced and the mobile app are available on this Github Link: https://github.com/sekilab/RoadDamageDetector

- **(Thompson et al. 2022)** involves the SHREC 2022 competition in which the participants utilised Semantic Segmentation and Masks to identify road cracks and potholes. The dataset consisted of 4340 images collected from 5 public datasets. A total of 7 methods were proposed, out of which 1 was the baseline made by the sponsors. All submissions used DL techniques based on criteria such as Weighted Pixel Accuracy. PUCPUnet++ and HCMUS-CPS-DLU-Net stood out as the most valuable runs.
- **(Salman et al. 2013)** utilised the Gabor filter and generated a filter bank containing multi-orientations. The Gabor filter of a given orientation was then convolved with the input pre-processed image. After the completion of convolution, the real component of the response out of the kernel was thresholded to generate a binary output image. Finally, the binary images from the differently oriented filters were combined by logical OR operation to produce an output image containing detected crack segments. Results with Precision up to 95% were achieved.
- **(Nienaber, Booysen, and Kroon 2015)** utilised Canny Edge detection and Contour detection on greyscale images to detect potholes. It first converted the colour image to greyscale, used a convex hull algorithm to improve segmentation, applied the Canny Filter, dilated the output and used contour detection to detect contours. The sample study indicated a precision of 81.8% and a recall of 74.4%

Work Process

Dataset Preprocessing

Two datasets were used for the problem. The Roboflow model was trained on the dataset downloaded from Kaggle, containing 2077 images, while the YOLOv5 implementation was trained on the dataset downloaded from Roboflow Universe, containing 665 images. Both the image sets were pre-processed by converting all images to 640x640 size. For augmentation, all the images were flipped Horizontally and rotated between -15°to 15°. For the second dataset, additional augmentations were applied, including clockwise, anti-clockwise and flip rotation of 90°. This led to a total of 3532 images in the first and 1595 images in the second dataset. About 80% of the images were used for training purposes. The links to the augmented datasets are given in the Appendix.

Implementations

Edge Detection based: This simple approach used edge detection methods for Pothole detection. The following algorithm roughly shows the process followed:

Algorithm 1: Edge Detection-based Algorithm for Pothole Detection

Require: Img - Input Image **Output:** - $CImg$ - Image with highlighted potholes

1: $GreyImg \leftarrow$ Convert Img to Grey-scale
2: $GaussBlurImg \leftarrow$ Apply Gaussian Blur to $GreyImg$ to reduce noise
3: $CannyImg \leftarrow$ Apply Canny Edge detection with suitable thresholds on $GaussBlurImg$
4: $DilatedImg \leftarrow$ Apply dilation a suitable number of times(default is 5) on $GaussBlurImg$ to connect edges.
5: $contours \leftarrow$ The contours detected from contour identification done on $DilatedImg$ using the appropriate methods to get contours around the potholes
6: $Cimg \leftarrow$ Overlay $contours$ on Img
7: Display $Cimg$

Roboflow 3.0 AutoML based: Roboflow is a website catered to Computer Vision developers. Their Train 3.0 technique uses the AutoML model and the MS COCO v7 model as a benchmark. The training took about 1.5 hours, and an API key was generated, which could be used in Python code to run the model locally.

YOLOv5 based: YOLOv5 is the fifth iteration of the "You Only Look Once" object detection model, built on top of PyTorch. The general layout of YOLO is present in the Appendix. YOLO is available in different versions, based on the computational complexity. Due to resource constraints, YOLOv5-s, a smaller version of YOLOv5, was used.

Google Colab was used for implementing the YOLOv5s (Jocher 2020) approach. A T4 GPU was allocated, having a GPU RAM of 15GB. Images were trained in batches of 16, and cache was enabled. Due to the model used(YOLOv5s with only 100 epochs) and the comparatively smaller dataset, the training took only about 40 minutes to complete. Some of the result curves can be found in the Appendix.

Results

The image in Figure 1 was used for comparison between the three different approaches.

A brief analysis of the results of applying the different approaches and the outcome received upon applying them in the above image are as follows:

Edge-Detection approach:

The edge detection approach performed the worst. Parameters had to be tuned per image, and many times, it would detect potholes in locations where they were not present. Additionally, the contours were not law appropriately detected, and thus, the surroundings of the pothole would be covered.

Figure 1: Original image

The following shows the result of applying this approach to the original image. While the general location of the Potholes has been detected very well, the individual potholes aren't separated, and some false positives have also been picked up.

Figure 2: Edge detected Potholes

Roboflow Train 3.0 approach:

The Roboflow model gave a mAP50 of 80%, Precision of 81% and Recall of 75.1%. However, the flaw with the model is that it is a black box, i.e. the hyperparameters or even the approach utilised by the model is not fully known. As such, any modifications that may be needed later would be difficult to implement.

The following is the result of the Roboflow approach. Note that the potholes in the back are not recognised. It may be due to the dataset used.

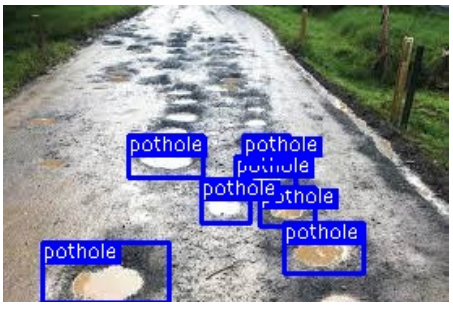

Figure 3: Roboflow detected Potholes

YOLOv5 approach:

The YOLOv5-s model gave a mAP50 of 79.8%, mAP50-95 of 49.5%, Precision of 77.8% and Recall of 70%. This indicates that the model would give less accurate results for a higher IOU. At the same time, it performs reasonably well for an IOU of .50. This means by using a larger dataset, more epochs and higher versions of YOLOv5, such as YOLOv5x6, at the cost of higher resources, we would get better results.

The following is the result of the YOLOv5-s approach. Note that more potholes in the back are recognised compared to the Roboflow approach. It may be due to the dataset used having more augmentations.

Figure 4: YOLO detected Potholes

Conclusion

The YOLOv5 approach shows potential. Past research was analysed, showing great approaches and a potential for more research using new methodologies. The work can act as an introduction to this field and can be implemented in systems such as navigation to alert drivers about potholes in advance.

References

Hu, Y.; and Zhao, C.-x. 2010.

Jo, Y.; and Ryu, S. K. 2015. Pothole Detection System Using a Black-box Camera. *Sensors*, 15: 29316–29331.

Jocher, G. 2020. Ultralytics YOLOv5.

Lin, T.-Y.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R.; Hays, J.; Perona, P.; Ramanan, D.; Zitnick, C. L.; and Dollár, P. 2015. Microsoft COCO: Common Objects in Context. arXiv:1405.0312.

Maeda, H.; Sekimoto, Y.; Seto, T.; Kashiyama, T.; and Omata, H. 2018. Road Damage Detection and Classification Using Deep Neural Networks with Smartphone Images. *Computer-Aided Civil and Infrastructure Engineering*, 33(12): 1127–1141.

Nienaber, S.; Booysen, M. T.; and Kroon, R. 2015. Detecting Potholes Using Simple Image Processing Techniques and Real-World Footage.

Otsu, N. 1979. A threshold selection method from gray level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9: 62–66.

Salman, M.; Mathavan, S.; Kamal, K.; and Rahman, M. 2013. Pavement crack detection using the Gabor filter. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, 2039–2044.

Shaghouri, A. A.; Alkhatib, R.; and Berjaoui, S. 2021. Real-Time Pothole Detection Using Deep Learning. arXiv:2107.06356.

Thompson, E. M.; Ranieri, A.; Biasotti, S.; Chicchon, M.; Sipiran, I.; Pham, M.-K.; Nguyen-Ho, T.-L.; Nguyen, H.-D.; and Tran, M.-T. 2022. SHREC 2022: pothole and crack detection in the road pavement using images and RGB-D data. arXiv:2205.13326.

Appendix

A) Datasets:

The augmented datasets can be downloaded from https://universe.roboflow.com/devashish-workspace/pothole-detection-vy8up and https://universe.roboflow.com/devashish-workspace/pothole-detection-2-28w0d.

B) YOLOv5 architecture overview:

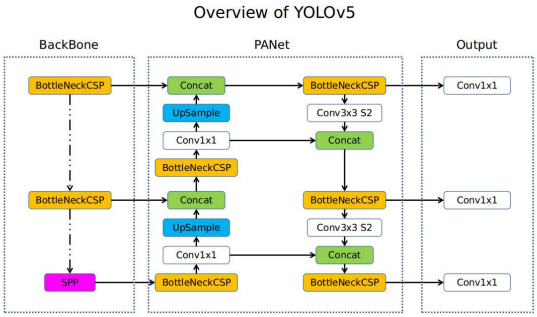

Figure 5: YOLOv5 Overview. Created by user seekFire on GitHub.

C)Relevant Curves for YOLOv5

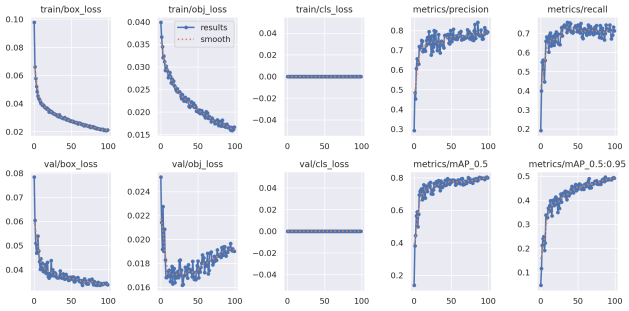

Figure 6: Loss and mAP curves