JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Real-Time Detection of Road Damages Using YOLOv8: An Innovative Deep Learning Approach

[1]Dasu Dasari, [2]D.N.V.S.Vijaya Lakshmi, [3]Tatapudi Ashok,

[1]Assistant Professor, [2]Associate Professor, [3]Assistant Professor
[1]Dept. of CSE, [2]Dept. of ECE, [3]Dept. of CSE,
[1]Adikavi Nannaya University, Rajamahendravaram, Andhra Pradesh, India
[2]ISTS Women's Engineering College, Rajamahendravaram, Andhra Pradesh, India
[3]Adikavi Nannaya University, Rajamahendravaram, Andhra Pradesh, India

Abstract: This research presents an innovative approach to the automated detection of road damage using deep learning, emphasizing the efficiency and accuracy of YOLOv8, a cutting-edge object detection algorithm. Maintaining road infrastructure is crucial for safe and efficient transportation, yet manual inspections are time-consuming and hazardous. By leveraging high-resolution image data and the YOLOv8 model, the proposed system enables real-time detection and classification of various road damages, including cracks and potholes. The methodology involves training the YOLOv8 model on a diverse dataset, ensuring its ability to recognize distinct damage patterns with high precision. The results demonstrate the potential of this automated system to enhance road maintenance processes, offering a cost-effective and scalable solution for safer transportation networks.

IndexTerms **- Road damage detection, YOLOv8, deep learning, automated inspection, object detection, transportation safety.**

I. INTRODUCTION

The maintenance of road infrastructure is essential for ensuring safe and efficient transportation systems. However, traditional manual inspection methods are time-intensive, laborious, and sometimes hazardous. In recent years, advancements in deep learning have paved the way for automated road damage detection systems that offer significant improvements in efficiency and accuracy. Among these, the YOLO (You Only Look Once) family of algorithms has emerged as a popular choice due to its real-time processing capabilities and high precision.

YOLO-based systems have shown great promise in identifying and classifying various types of road damage, such as cracks, potholes, and surface deformations. The lightweight YOLO-LRDD model, for instance, offers a balance between accuracy and computational efficiency, making it suitable for real-time applications [1]. Similarly, YOLOv8-PD has been optimized to address specific challenges in pavement distress detection, demonstrating enhanced detection performance [2].

Further improvements in YOLO architectures, such as RDD-YOLO, integrate attention mechanisms and optimized convolutional modules, ensuring better recognition of smaller and more complex damage patterns [3]. Additionally, innovations like LAG-YOLO introduce lightweight attention modules to improve accuracy while maintaining computational efficiency [6]. Beyond these, the integration of YOLO with smartphone-based systems has enabled mobile solutions for road damage monitoring, underscoring its practical utility [7].

Despite these advancements, challenges remain in the form of diverse environmental conditions, varying damage patterns, and the need for extensive annotated datasets [4]. Future research continues to explore ways to enhance model robustness and scalability, ensuring these systems are adaptable to real-world applications [5].

II. Proposed System

This research proposes a robust system for real-time road damage detection using YOLOv8, a state-of-the-art object detection algorithm known for its speed and accuracy. The system leverages high-resolution image data and advanced deep learning techniques to efficiently identify and categorize road damages such as cracks, potholes, and surface deformations.

The architecture of the proposed system is designed to optimize performance at every stage. It begins with a user-friendly interface that allows users to upload images or videos or use a camera for real-time detection. The uploaded data is preprocessed to ensure compatibility with the YOLOv8 model. The core of the system involves YOLOv8's unique architecture, which includes a custom CSPDarknet backbone for feature extraction, a C2f module for multi-scale feature fusion, and detection heads for precise bounding box and class predictions.

The system is trained on a diverse dataset that includes annotated images of road damages from multiple sources, ensuring the model's ability to generalize across varying conditions and damage types. During training, hyperparameters are fine-tuned to enhance detection accuracy, especially for small or subtle damages.

The detection process is carried out in real-time, making it suitable for on-the-fly analysis during road inspections or integration into vehicle-mounted systems. The system outputs annotated images or video frames with bounding boxes and labels, providing users with clear and actionable information about the detected damages.

The proposed system aims to transform traditional road inspection methods by providing a faster, safer, and more cost-effective alternative, ultimately contributing to improved road infrastructure management and transportation safety.

III. ARCHITECTURE OF YOLOV8

YOLOv8, the latest in the YOLO series, features an optimized architecture for real-time object detection, comprising three main components: Backbone, Neck, and Head. The **Backbone** uses a customized CSPDarknet to extract features efficiently with cross-stage partial connections, enhancing learning capabilities while minimizing computational costs. The **Neck**, incorporating a novel C2f module, fuses multi-scale features for improved detection accuracy, especially for small or subtle objects. The **Head** predicts bounding boxes, confidence scores, and class probabilities using a grid-based approach for precise localization. This single-pass architecture delivers high-speed, scalable, and accurate object detection, making YOLOv8 ideal for applications like road damage detection.

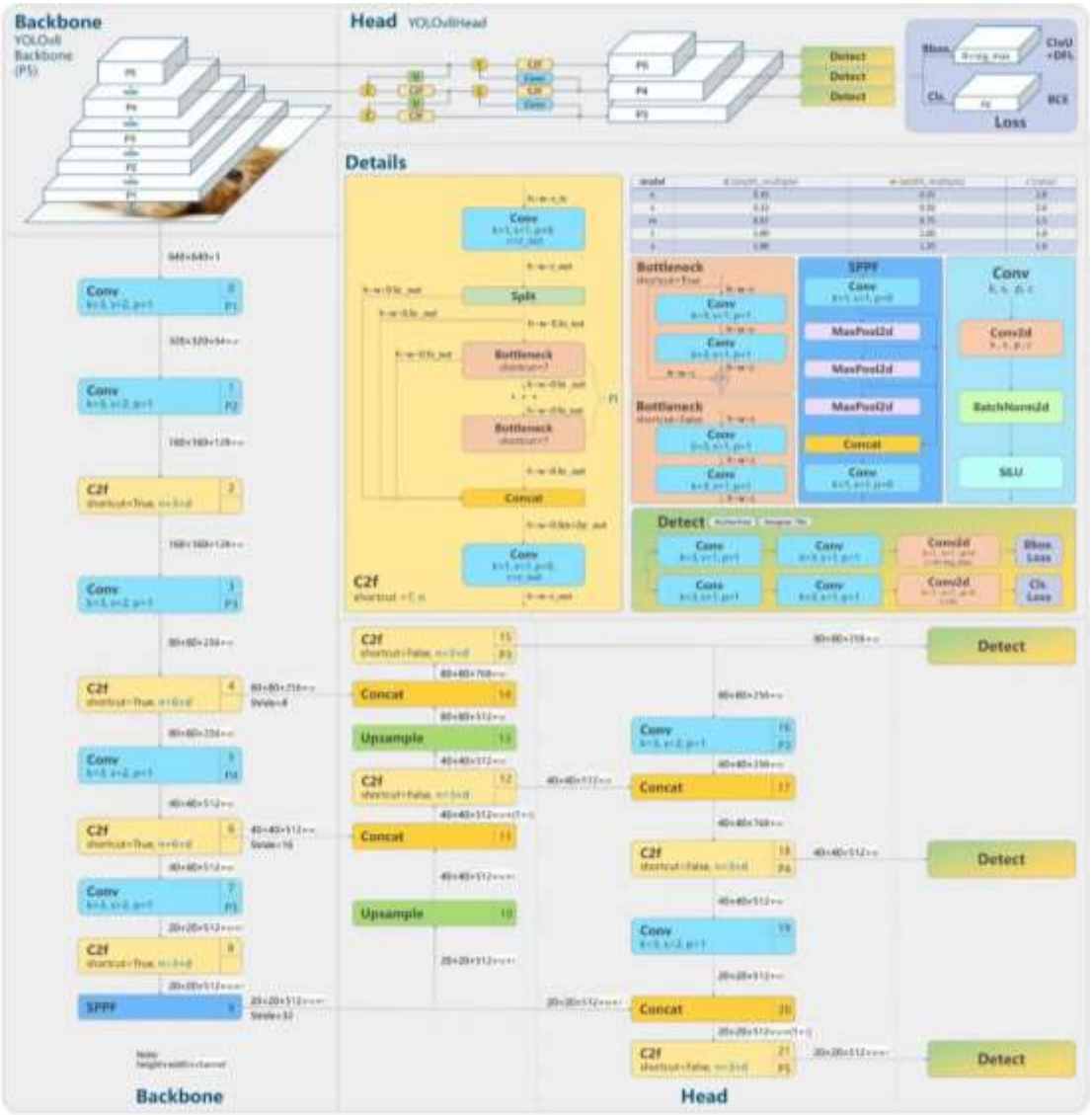

Fig 3.1: Architecture of YOLOv8

PROCEDURE

A step-by-step procedure for understanding YOLOv8 architecture:

Step 1: Downloading Images from Google

Step 2: Annotating Images in YOLO Format

Step 3: Setting up YOLO V8 on Local Machine

Step 4: Training the YOLO V8 Object Detection Model

Step 5: Running Custom Object Detection

Step 1: Downloading Images from Google

The first step is to download images of our interest from Google. We will use a simple script to automate this process and save the images in a designated folder. By specifying keywords, we can obtain Relevant images that we can later use for training our custom object detection model.

Step 2: Annotating Images in YOLO Format

Next, we will annotate the downloaded images in YOLO format. For each image, we will manually create bounding boxes around the objects we want to detect. We will also assign labels to these objects, defining the classes the model will be trained to detect. This annotation process is crucial as it provides ground truth data for our model's training.

Step 3: Setting up YOLO V8 on Local Machine

To train and run custom object detection, we need to set up YOLO V8 on our local machine. This involves creating a virtual environment and installing the necessary libraries and dependencies. By following the provided instructions, we will ensure that our environment is ready for training and prediction.

Step 4: Training the YOLO V8 Object Detection Model

Once the setup is complete, we can proceed to train the YOLO V8 object detection model. By providing our annotated dataset and configuration files, we will initiate the training process. We will define the number of epochs, set up the data paths, and start training our model to recognize the desired objects accurately.

Step 5: Running Custom Object Detection

After the training process is finished, we can evaluate our model's performance on unseen data. Whether it's images, videos, or webcams, we can use our trained model to detect the custom objects we specified. By adjusting the detection threshold and other parameters, we can fine-tune the model's performance and obtain accurate object detections.

IV. SYSTEM DESIGN

YOLOv8 (You Only Look Once version 8) is an advanced real-time object detection algorithm optimized for various computer vision tasks, including identifying road damage.

When a user opens a user interface, first login page will open after login he/she can be allowed to upload a file then the system will detect the damages and generate the results to the user.

Following is the detailed explanation of the process:

i. *User Interface:*
   Opening the user interface initiates the process.
   The login page is displayed.
   If not logged in, users need to register; otherwise, they provide their username and password.

ii. *Upload Page:*
   After successful login, users are directed to the upload page.
   Here, they can choose the "upload file" option to select a file from their device.

iii. *Data Processing:*
   The system takes the uploaded file as a dataset and send for processing.
   Data preprocessing steps are applied to prepare the data for the YOLOv8 algorithm.

iv. *YOLOv8 Algorithm:*
   The YOLOv8 algorithm is employed for damage prediction.
   The processed data is input into the YOLOv8 model, which is capable of detecting and predicting damages.

v. *Output Generation:*
   Once the YOLOv8 algorithm completes its predictions, the system generates output results.
   Users can view the results, which may include identified damages and relevant information.

vi. *User Interaction:*
   Users interact with the system through the user interface, navigating through login, file upload, and result viewing stages.
   The system provides a seamless experience, guiding users through the entire process.

The YOLOv8-based system follows a user-friendly flow, starting with login, proceeding to file upload, utilizing the YOLOv8 algorithm for damage prediction, and concluding with the presentation of results to the user.

The system architecture used in this project is given below-

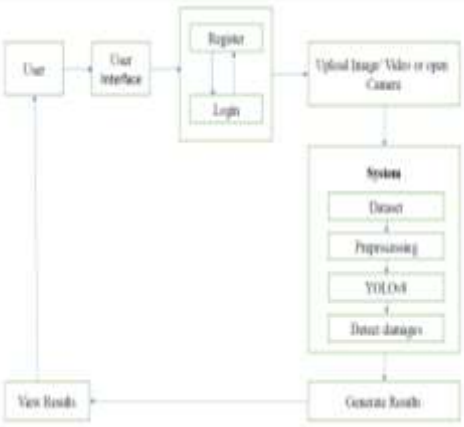

Fig 4.1: System Architecture

V. IMPLEMENTATION

5.1 Dataset

A YAML (YAML Ain't Markup Language) file is a human-readable data serialization format used for configuration files, data exchange between languages with different data structures, and sometimes for data storage. YAML is often preferred for its simplicity and readability, making it easy for both humans and machines to understand. For training a model, yolov8n.yaml and coco128.yaml files is used.

a) The COCO (Common Objects in Context) dataset is a large-scale image recognition, segmentation, and captioning dataset designed for training and evaluating computer vision models. It contains over 330,000 images with more than 2.5 million object instances labeled across 80 object categories. The dataset is widely used in research and industry for object detection, instance segmentation, and other computer vision tasks.

The COCO dataset has large number of images with labels. The COCO128.yaml was used for training the model to identify the objects. The COCO dataset was taken from the website- GitHub- dksfal/coco128.yaml

b) To train the model for detecting road damages, YOLOv8n.yaml is used for this project. It has large 7,000 plus images of road damages.

Ultralytics recently released the YOLOv8 family of object detection models. These models outperform the previous versions of YOLO models in both speed and accuracy on the COCO dataset. But about the performance on custom datasets, we will train YOLOv8 models on a custom dataset. Specifically, we will train it on a large-scale pothole detection dataset.

While fine tuning object detection models, we need to consider a large number of hyperparameters into account. Training the YOLOv8 models is no exception, as the codebase provides numerous hyperparameters for tuning. Moreover, we will train the YOLOv8 on a custom pothole dataset which mainly contains small objects which can be difficult to detect. These are considered as 3 types:

- YOLOv8n (Nano model)
- YOLOv8s (Small model)
- YOLOv8m (Medium model)

Among these YOLOv8n dataset is used for training. To detect very small objects also.

Cracks and Potholes Detection Dataset to Train YOLOv8

This dataset contains more than 7000 images collected from several sources. To give a brief overview, the dataset includes images from:

- Roboflow pothole dataset
- Dataset from a research paper publication
- Images that have been sourced from YouTube videos and are manually annotated
- Images from the RDD2022 dataset

After going through several annotation corrections, the final dataset now contains:

- 6962 training images
- 271 validation images

Here are a few images from the dataset, along with the annotations.

Fig 5.1: Annotated images from the pothole dataset to train the YOLOv8 model on custom dataset.

It is very clear from the above image that training YOLOv8 on a custom pothole dataset is a very challenging task. The potholes can be of various sizes, ranging from small to large.

5.2 Coding

A) Setting Up YOLOv8 to Train on Dataset

To train YOLOv8 on a custom dataset, we need to install the ultralytics package. This provides the yolo Command Line Interface (CLI). One big advantage is that we do not need to clone the repository separately and install the requirements.

B) Import Libraries

Import necessary libraries required for detecting the objects. Those provides tools for building and training neural networks, and it is widely used in research and industry for machine learning tasks.

C) Load the Dataset

Load a dataset from a dataset loading script that downloads and generates the dataset. Use the load_dataset() function to load the dataset.

Load a dataset from a file using a python library such as pandas or numpy. Use the appropriate function to read the file and load the dataset.

D) Pre-process the data

The code preprocesses a batch of image data by moving it to a specified device (e.g., GPU) and normalizing the pixel values to a standardized range. This type of preprocessing is common in deep learning workflows, especially when training neural networks.

E) Model Training

The python script serves as an entry point for training a YOLO object detection model. It provides flexibility for users to configure the model and dataset settings through the command line. The training process is executed by initializing a YOLO model and calling the training method with the specified or default configurations.

Hyperparameter Choices to Train YOLOv8 on Custom Dataset

Here are a few pointers explaining the hyperparameter choices that we make while training:

- We will train each model for 100 epochs. As a concept project, to get started, we will try to get the best possible results with limited training. As we have almost 7000 images, even 100 epochs will take quite some time to train and should give decent results.
- As the potholes can be quite small in some images, we will set the image size to 1280 resolution while training. Although this will increase the training time, we can expect better results compared to the default 640 image resolution training.

F) Make Predictions

YOLOv8 is applied for road damage detection by training it on annotated datasets, enabling it to efficiently predict bounding boxes and confidence scores for instances of road damage. Post-processing steps enhance prediction accuracy, and the final results are visualized on input images, offering a comprehensive solution for real-time detection and localization of road damage.

G) Model Evaluation

This method checks if it's required to save JSON results, if the dataset is in COCO format and if there are predictions. If these conditions are met, it loads COCO annotations and predictions using pycocotools, performs evaluation, and updates the mAP values in the stats dictionary.

H) Visualizing the detection results

If we want to place a file name in the parameter “source”, which should be in the format of mp4, or jpg, or png then it will detect the damages in that file or else if we want to detect the damages in real time, put source= “0”, then the local camera will ON and detect damages in real time.

VI. WEB PAGE DESIGN

The road damage detection project is coded a web application using the HTML code, by a hyperlink tag. The application allows users to upload images or videos from their devices, which are then processed to identify road damages or they can use web cam of their device to detect in real-time. The efficient implementation facilitates widespread accessibility, enhancing the convenience of users from detecting the road damages.

The project is coded in python, that file is called in the HTML to detect the objects in the input file. Additionally, an HTML file, named “success.html” is used to display the output. The “success.html” is called in the “login page”. The HTML file ensures visually appealing and interactive front-end experience for users these files create a cohesive web application that efficiently processes road damage images or videos, detect damages, and display the results in a user-friendly manner. The combination of python and HTML enhances the project’s versatility and user accessibility.

VII. WEB PAGE DEPLOYMENT

The deployment of the webpage is uses the HTML file with hyperlinks, which is used to redirect the page into the background python code and will process the user uploaded image and detect the damages.

First user has to login to the webpage, if the user exists already then, the page redirects to the upload page. Otherwise, he has to register in the register page. After uploading the file, the system will give the desired output.

1) Login page

The code given below is used to create a login page in the user interface before going to the upload page. If the user already registered, he/she can go to the login page and fill the basic details, that is Username and Password.

2) *Register page*

If he is not registered, then he can go to the register page. After registering by giving the username and password, the upload page will open and the user can upload any image or video.

3) Upload Page

After successfully login then the upload page will open. First, we have to enter the purpose, but it is optional. Then we can choose upload file or open camera option. When we upload any file or open camera, the backend python code will run and process the uploaded image or video. Then it will give the desired output to the user.

The open camera option is used for real-time experience by allowing the web cam in his device and the system will give the output in real-time.

The output of the uploaded image is shown in Fig.7.1, which is having bounding boxes around the damages and also tells the name of the damage. If the user tuns on the camera, then the output in real- time detection is shown in Fig.7.2.

Fig 7.1 Output Image

Fig 7.2 Output in real-time

In real-time also it will detect the damages by using the bounding box and shows the type of damage.

VIII. Results and Observations

In the comparative analysis of road damages, the left-side image is the user uploaded image and the right-side image is the system given output. The system will generate the bounding boxes around the damages and send it to the user. The system successfully detected the damages within less amount of time and gives the best accuracy. The system will process the image by dividing the image by number of pixels. So, that will give the best accuracy. In the system generated image, the road damages are very clear and it will give which type of road damages it is. The system will generate output by drawing a bounding box for the damages and gives the name of the damages along with its label ID. If the user gives the video, then the system will divide the video into number of frames per second (FPS). Then it will detect the damages for each image separately. The system will accept the file format as .mp4 file, .png, and .jpg.

Fig.8.1 Road damages before and after detection

IX. Conclusion

Hence, this project introduced an innovative method for detecting the damages of road in real-time in an effective way within less time. Utilizing YOLOv8 for road damage detection presents a compelling solution with its strengths in accuracy, speed, and flexibility. The model's real-time inference capabilities make it suitable for applications where swift detection of road damages is crucial. The open- source nature of YOLOv8 and its active community support contribute to its accessibility and ongoing improvements.

However, challenges exist, particularly in the quality of the training large dataset. Accurate annotation of road damage images is vital for optimal model performance. YOLOv8 proves to be an effective and efficient solution for road damage detection tasks. Its real-time object detection capabilities, coupled with a high level of accuracy, make it a valuable tool for identifying and categorizing various types of road damages, such as potholes, cracks, or surface degradation.

X. Future Scope

The future scope for road damage detection using YOLOv8 encompasses advancements in multiple areas. These include the exploration of advanced detection architectures, integration with semantic segmentation techniques, and the incorporation of multi-modal sensing for a more comprehensive understanding of road scenes.

Optimizing models for real-time deployment on edge devices, adapting to diverse environmental conditions, and collaborating with GIS and mapping systems are key direction.

Future scope is adding an alert message, which will alert when road damages are near. And the system should detect the damages which is far from 100 meters. Also, in future we can update the damaged road in the Google maps. So, that everyone will use and select the best way to travel.

References

[1] Liu, H. Ma, and X. Zhang, "YOLO-LRDD: A Lightweight Method for Road Damage Detection," EURASIP Journal on Advances in Signal Processing, vol. 2022, no. 1, pp. 1–12, 2022. Available: https://asp-eurasipjournals.springeropen.com.

[2] Chen, L. Wang, and Q. Yang, "YOLOv8-PD: An Improved Road Damage Detection Algorithm Based on YOLOv8n," Scientific Reports, vol. 14, no. 3, 2024. Available: https://www.nature.com.

[3] Kim and S. Park, "RDD-YOLO: Road Damage Detection Algorithm Based on Improved You Only Look Once Version 8," Applied Sciences, vol. 14, no. 8, pp. 3360–3370, 2023. Available: https://www.mdpi.com.

[4] J. Wu, X. Li, and Y. Lin, "Road Damage Detection Algorithm for Improved YOLOv5," Scientific Reports, vol. 12, 2022. Available: https://www.nature.com.

[5] Sharma, P. Singh, and M. Patel, "Optimizing YOLO Architectures for Optimal Road Damage Detection," arXiv preprint arXiv:2410.08409, 2024. Available: https://arxiv.org.

[6] T. Zhou, X. Huang, and Y. Fang, "LAG-YOLO: Efficient Road Damage Detector via Lightweight Attention Ghost-YOLO," Journal of Intelligent Computing, vol. 2023, no. 4, pp. 101–120, 2023. Available: https://www.sciopen.com.

[7] Jeong, "Road Damage Detection Using YOLO with Smartphone Images," in 2020 IEEE International Conference on Smart Computing (SMARTCOMP), Atlanta, GA, USA, Dec. 2020, pp. 10–13. Available: https://ieeexplore.ieee.org.

[8] YOLOv8 Architecture online reference: https://viso.ai/deep-learning/yolov8-guide

[9] Arya, D.; Maeda, H.; Ghosh, S.K.; Toshniwal, D.; Mraz, A.; Kashiyama, T.; Sekimoto, Y.J.A.i.C. Deep learning-based road damage detection and classification for multiple countries. Autom. Constr. 2021, 132, 103935. [Google Scholar] [CrossRef]

[10] Dongjun Jeong, "Road Damage Detection Using YOLO with Smartphone Images",10-13 December 2020, Atlanta, GA, USA Publisher: IEEE

[11] Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448. [Google Scholar]

[12] Zhuohui Chen; Dahao Wang; Yezhe Wang; Shunying Lin; Haoran Jia; Peixin Lin; Yixian Liu; Ling Chen. Road Damage Detection Algorithm Based on Object Detection Network. Nanjing, China, 16-18 June 2023. Research and Implementation of Road Damage Detection Algorithm Based on Object Detection Network | IEEE Conference Publication | IEEE Xplore

[13] Brief summary of YOLOv8 model structure Issue #189 ultralytics/ultralytics GitHub

[14] Train YOLOv8 on Custom Dataset - A Complete Tutorial (learnopencv.com)