Road Damage Detection using YOLOv8

Rutuj Dhawale

1 Introduction

Road surface damage such as cracks and potholes poses serious safety risks and increases maintenance costs. Manual inspection is time-consuming, subjective, and difficult to scale. This project focuses on building an automated object detection system that can accurately identify and localize different types of road damage from images.

The task is formulated as an object detection problem, where the model predicts bounding boxes and class labels for multiple damage instances present in a road image. Model performance is evaluated using Mean Average Precision (mAP) on a hidden test set.

2 Model Architecture

We use YOLOv8 (You Only Look Once v8), a state-of-the-art, single-stage object detection model known for its balance between speed and accuracy.

1. The input image is resized and passed through the network.
2. The image is divided into a grid, where each grid cell predicts bounding boxes.
3. Each bounding box includes location, size, class label, and confidence score.
4. Non-Maximum Suppression (NMS) removes overlapping duplicate detections.

This unified approach makes YOLO fast and effective for detecting road damage of varying sizes in real-world conditions

2.1 Architecture Overview

YOLOv8 follows a modular architecture consisting of:

- **Backbone**: CSP-based convolutional layers for extracting hierarchical image features.
- **Neck**: Feature Pyramid Network (FPN) to fuse multi-scale features, enabling detection of both small and large damage regions.
- **Head**: Detection head that predicts bounding box coordinates, object confidence scores, and class probabilities.

2.2 Chosen Model Variant

The YOLOv8-Medium (`yolov8m`) model was selected as it provides better accuracy than smaller variants while remaining feasible to train on limited GPU memory.

3 Dataset and Classes

The provided dataset contains approximately 47,000 road images annotated in YOLO format. The following five damage classes are used:

Class ID	Damage Type
0	Longitudinal Crack
1	Transverse Crack
2	Alligator Crack
3	Other corruption
4	Pothole

The dataset is split into training, validation, and test sets. The test set does not contain labels and is used only for final evaluation.

4 Data Augmentation and Training Strategy

4.1 Pre-training

The model was initialized using COCO-pretrained weights, which helps accelerate convergence and improves generalization, especially when training on a limited dataset.

4.2 Data Augmentation

YOLOv8 applies built-in data augmentation during training, including:

- Mosaic augmentation
- Random horizontal flipping
- Scaling and translation
- HSV color augmentation

These augmentations help the model generalize better to real-world variations such as lighting changes and different road textures.

4.3 Training Configuration

Parameter	Value
Epochs	50
Image Size	640×640
Batch Size	2–16 (GPU dependent)
Optimizer	AdamW (YOLOv8 default)
Hardware	NVIDIA RTX 3050 (4GB VRAM)

5 Hyperparameter Tuning Experiments

Several inference and training hyperparameters were tuned to improve performance.

5.1 Tested Hyperparameters

- Confidence Threshold: 0.10 – 0.35
- IoU Threshold: 0.5 – 0.7
- Image Resolution: 416, 640
- Batch Size: Adjusted to avoid GPU out-of-memory errors

5.2 Final Selected Values

The following values provided the best balance between recall and precision:

- Confidence Threshold: 0.10
- IoU Threshold: 0.7
- Image Size: 640

6 Evaluation Metrics and Results

Model performance was evaluated using Mean Average Precision (mAP), the standard metric for object detection tasks.

- **mAP@0.5**: Measures detection quality at IoU = 0.5
- **mAP@0.5:0.95**: Averaged across multiple IoU thresholds

6.1 Validation Results

Metric	Value
Precision	0.52
Recall	0.47
mAP@0.5	0.44
mAP@0.5:0.95	0.22

7 Techniques Used to Improve the Baseline

The following strategies were used to improve performance over a basic baseline model:

- Use of a larger YOLOv8 model variant (YOLOv8-Medium)
- Fine-tuning confidence and IoU thresholds for better recall
- Strong data augmentation during training
- Transfer learning using pretrained weights
- GPU-accelerated training and optimized inference

8 Test Set Prediction and Submission

The trained model was run on the test dataset to generate predictions. For each test image, a corresponding text file was created containing:

```
<class_id> <x_center> <y_center> <width> <height> <confidence>
```

All prediction files were placed in a `predictions/` folder and compressed into `submission.zip`https://github.com/username/road-damage-detection-yolov8 for the submission.

9 Reproducibility and Source Code

All code used for training, evaluation, and prediction generation is provided in the GitHub repository below:

`https://github.com/your-username/road-damage-detection-yolov8`

The repository includes training scripts, configuration files, and instructions to reproduce the results.

10 Conclusion

This project demonstrates the effectiveness of YOLOv8 for automated road damage detection. The model successfully identifies and localizes multiple damage types in challenging road images. Future improvements could include longer training schedules, model ensembling, and training on higher-resolution images with more powerful GPUs.