Object Detection for Road Turn Detection in Gilgit using YOLOv8

Submitted By: Munira

GitHub Profile: https://github.com/Mnuira/Object-Detection-for-Road-Turn-Detection-in-Gilgit-using-YOLOv8-oriii

Learning Objectives

Understanding Object Detection with YOLOv8

Data Preparation and Annotation for Object Detection

Model Training and Hyperparameters Tuning

Evaluating Object Detection Models

Implementing Object Detection in Real-World Scenarios

Code Documentation and Report Writing

Exploring Advanced Techniques for Model Improvement

Project Overview

This project aims to detect various road conditions (e.g., right turn, left turn, straight, unexpected) in road images using an object detection model trained with YOLOv8. The model will help in improving road safety and assisting navigation by accurately identifying road features in challenging environments, specifically in Gilgit, a mountainous region in Pakistan. The goal is to ensure that the model is capable of detecting road conditions and turns in real-world images, which will be vital for drivers, navigation systems, and autonomous vehicles.

Data Preparation and Preprocessing

Dataset and Classes

The dataset consists of images collected from road scenes in Gilgit, and the images are split into three categories:

Train

Validation

Test

Each image is annotated with bounding boxes and labeled with one of the following four classes:

Right-Turn

Left-Turn

Straight

Unexpected

Annotations are provided in CSV format, and were converted into YOLO format, which consists of bounding box coordinates in terms of normalized center coordinates (x, y) and dimensions (width, height).

Data Conversion Code Highlights

The conversion script takes the annotations in CSV format and converts them into text files required by YOLO. Here’s a snippet of the conversion code:

python

Copy code

# Convert CSV annotations to YOLO format

for \_, row in annotations.iterrows():

class\_id = class\_names.index(row['class'])

center\_x = (row['xmin'] + row['xmax']) / 4 / img\_width

center\_y = (row['ymin'] + row['ymax']) / 4 / img\_height

width = (row['xmax'] - row['xmin']) / img\_width

height = (row['ymax'] - row['ymin']) / img\_height

This code reads the bounding box coordinates and normalizes them according to the image dimensions.

Model Training Configuration

The object detection model was trained using YOLOv8, which is well-known for its speed and accuracy in real-time object detection tasks. Below are the configurations used for training:

Model Version: yolov8m.pt (medium model), later experimented with yolov8s.pt (small model).

Epochs: First run with 20 epochs, followed by an additional 10 epochs with adjusted parameters.

Batch Size: Started with 16, then adjusted to 5 to improve performance.

Image Size: 640x640 pixels.

Learning Rate: Initially set to a default value, then adjusted to 0.001 for the second training run.

Training and Validation Results

Initial Training Results (20 Epochs, Batch Size 16):

Precision: 74.1%

Recall: 15%

mAP50: 0.269

mAP50-95: 0.155

Interpretation: The model demonstrated reasonable precision but had a low recall, meaning it was accurate when it made predictions but missed some instances of road turns.

Second Training Run (10 Epochs, Batch Size 5, LR 0.001):

Precision: 10.3%

Recall: 28.9%

mAP50: 0.147

mAP50-95: 0.029

Interpretation: The second configuration did not show improvement. In fact, the performance dropped, suggesting that the batch size and learning rate were not optimal, and further tuning is required.

Model Evaluation and Prediction Results

After training, the model was evaluated on the test dataset. The following code snippet was used to generate predictions:

python

Copy code

# Use the model to make predictions on test data

results = model.predict("/path/to/test/data", save=True)

for result in results:

result.show()

This code displays the predictions for each image in the test dataset and saves the results for visual analysis. Below is the output showing one such prediction.

Example: Prediction on Test Data

(Insert Screenshot of Test Image Prediction)

Explanation: The screenshot should show a road image with bounding boxes and labels indicating the road turn type (Right Turn, Left Turn, Straight, or Unexpected).

Conclusion

The YOLOv8 model demonstrated promising results with high precision but had issues with recall, suggesting it could identify the correct road turn conditions but missed several instances. Some potential improvements include:

Hyperparameter Tuning: Adjusting the learning rate, batch size, and epochs further.

Data Augmentation: Enhancing the dataset with more variations in road images, lighting, and weather conditions to help the model generalize better.

Model Adjustment: Experimenting with different YOLOv8 architectures or fine-tuning the model for road turn detection.

Despite these challenges, this model has potential applications in real-time road detection systems, enhancing navigation and road safety in regions like Gilgit, where road conditions can be unpredictable and dangerous.

Screenshots of Model Predictions and Results

Here are some key visual results from the model’s predictions on the test dataset:

Model Predictions on Test Data

(Insert Screenshot showing model predictions on test images)

Explanation: This screenshot should show road images with the predicted bounding boxes around the road turns.

Training Graphs

(Insert Screenshot showing training graphs with metrics like loss, precision, and recall over epochs)

Explanation: This screenshot should depict how the model's performance evolved during the training process.

Example of Detected Road Turns

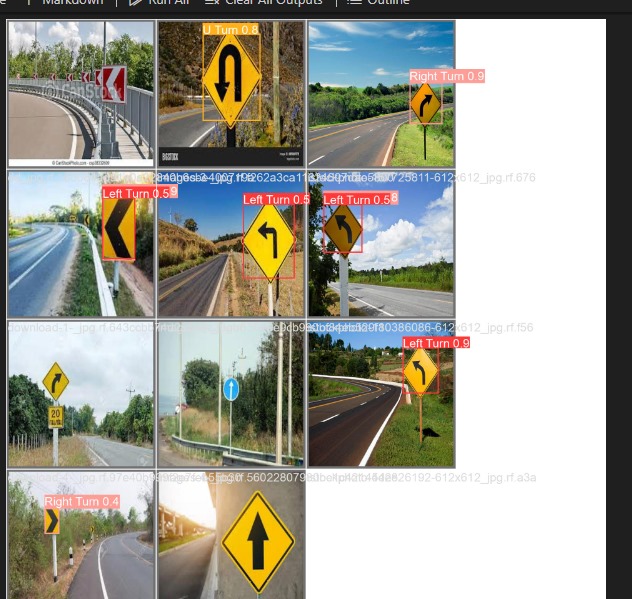
(Insert Screenshot showing images with bounding boxes and labels: Right-Turn, Left-Turn, etc.)

Explanation: These images should show the model’s ability to detect road conditions like Right-Turn, Left-Turn, Straight, and Unexpected in real-world road images.

Referencing

GitHub: Munira's GitHub Repository

YouTube Tutorial: YOLOv8 Object Detection Tutorial

**Screenshorts:  
  
**

The End