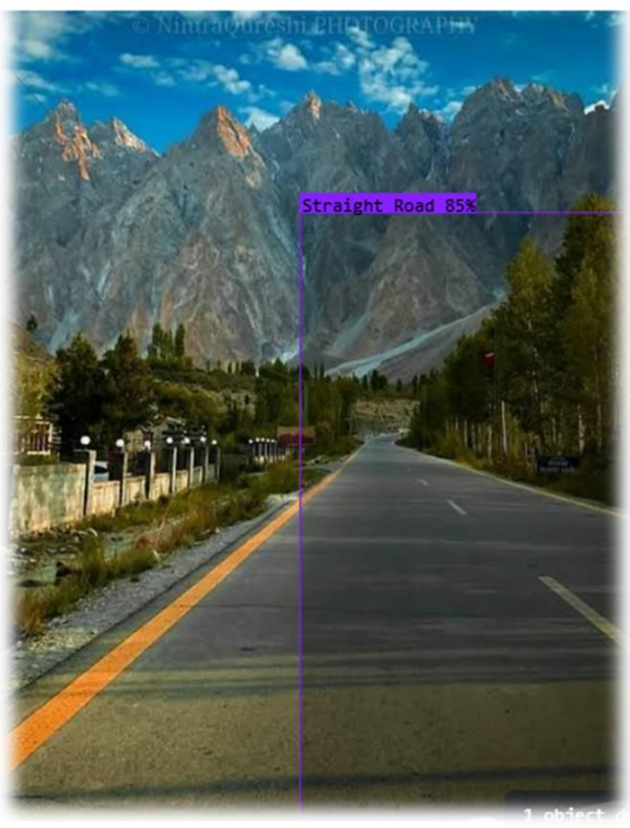
High Impact Skills Development Program

Road Detection Project Report



Authorized by: Ms Chand Safi

Reg-No: GIL-DSAI-011

Email Address: chandsafi87@gmail.com

GitHub: https://github.com/chandsafi/chandsafi

1. Introduction

1.1 Project Overview

This project involves detecting various types of road segments and characteristics across the Gilgit-Baltistan region. The aim is to classify different types of roads based on their unique attributes to assist with traffic and safety assessments in challenging terrains. Using YOLOv5 and YOLOv8, this project employs deep learning models for accurate object detection and classification.

1.2 Objectives

- To create a labeled dataset for different road types.
- To train an object detection model for real-time road type classification.
- To evaluate the model's performance on a diverse dataset representative of the Gilgit-Baltistan region's road conditions.

2. Data Collection

2.1 Data Sources

Images were sourced from multiple roads in Gilgit-Baltistan, emphasizing segments that are frequently encountered, such as straight roads, turns, and damaged roads.

2.2 Data Collection Method

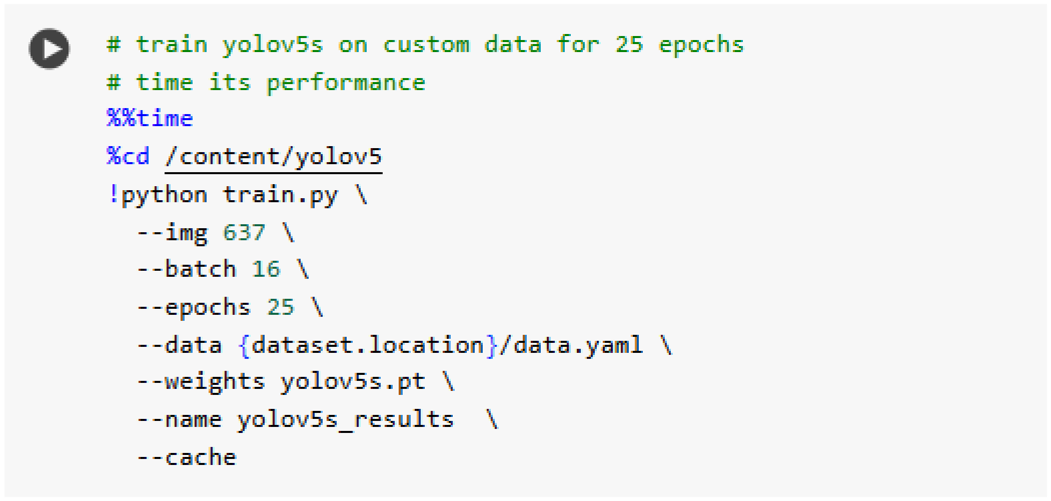
The data was collected using a combination of field photography and existing geographic datasets where available. Each road image was captured at various angles and lighting conditions to ensure diversity and robustness of the dataset. The images were stored in high resolution to retain details that are essential for road type classification.

2.3 Dataset Overview

The dataset was organized into the following classes:

- **Straight Road:** 120 images (84 training, 25 validation, 11 test)
- **Damaged Road:** 62 images (43 training, 14 validation, 5 test)
- **Right Turn:** 27 images (20 training, 4 validation, 3 test)

- **Left Turn:** 12 images (6 training, 4 validation, 2 test)
- **Inclined Road:** 9 images (7 training, 2 validation, 0 test)
- **Steep Road:** 5 images (3 training, 2 validation, 0 test)
- **U Turn:** 5 images (3 training, 2 validation, 0 test)
- **Junction:** 4 images (3 training, 1 validation, 0 test)
- **Two Way:** 4 images (3 training, 0 validation, 1 test)



```
# train yolov5s on custom data for 25 epochs
# time its performance
%%time
%cd /content/yolov5
!python train.py \
  --img 637 \
  --batch 16 \
  --epochs 25 \
  --data {dataset.location}/data.yaml \
  --weights yolov5s.pt \
  --name yolov5s_results  \
  --cache
```

Each class was split into training, validation, and test sets to support the model's learning and evaluation phases.

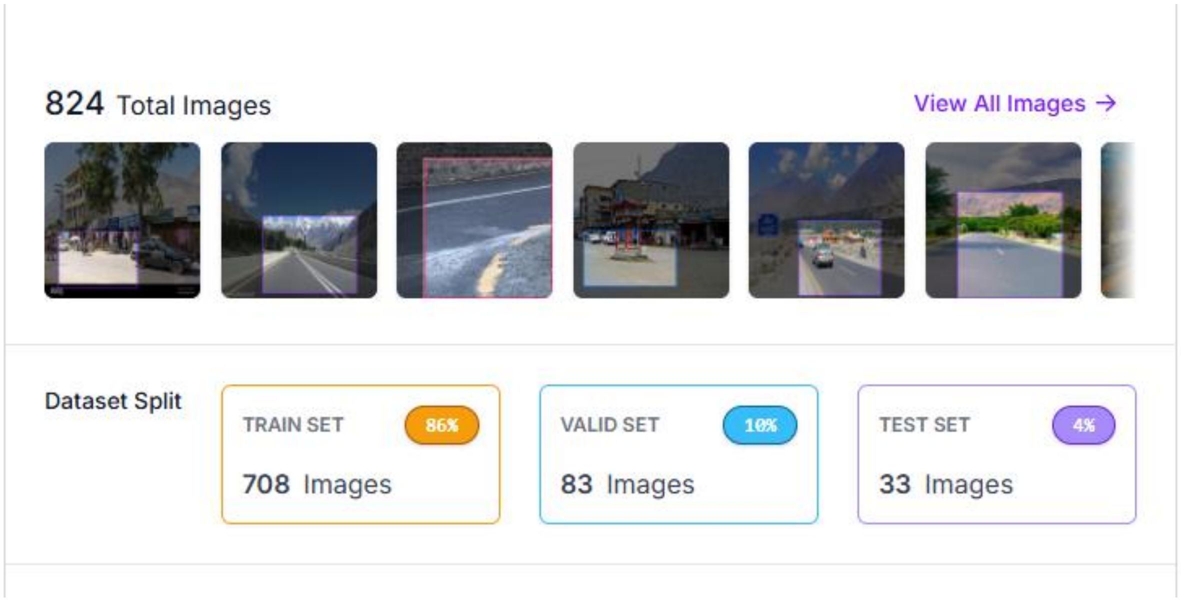
3. Data Annotation

3.1 Annotation Tool: Roboflow

The annotation process was conducted using Roboflow, an intuitive and powerful annotation tool that facilitated efficient labeling of each road segment. Roboflow's capabilities allowed for quick dataset handling, including resizing, augmenting, and exporting images in a format compatible with YOLO.

3.2 Annotation Process

Each image was manually annotated by drawing bounding boxes around specific road features, tagging them with the appropriate class (e.g., straight road, right turn, damaged road). These annotations serve as ground truth labels for model training and evaluation.


824 Total Images

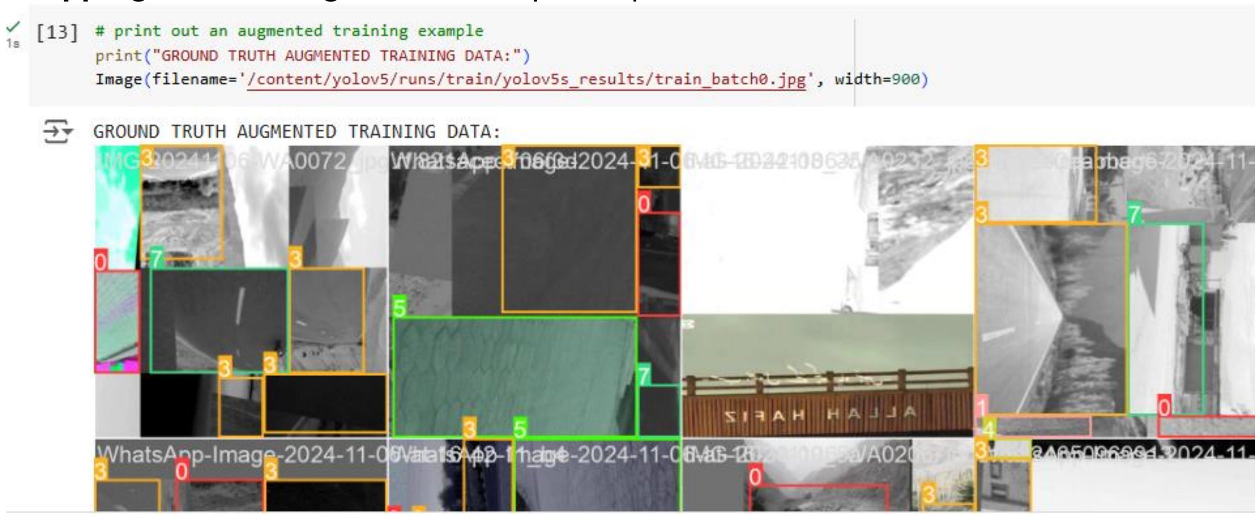
View All Images →

Dataset Split

TRAIN SET 86%	VALID SET 10%	TEST SET 4%
708 Images	83 Images	33 Images

3.3 Data Augmentation

To enhance model generalization, data augmentation techniques were applied. These included:

- **Rotation** and **flipping** to simulate different road orientations.
- **Color adjustment** to represent various lighting conditions.
- **Cropping** and **resizing** to focus on specific parts of the roads.
- 

```
[13] # print out an augmented training example
     print("GROUND TRUTH AUGMENTED TRAINING DATA:")
     Image(filename='/content/yolov5/runs/train/yolov5s_results/train_batch0.jpg', width=900)
```

```
GROUND TRUTH AUGMENTED TRAINING DATA:
```

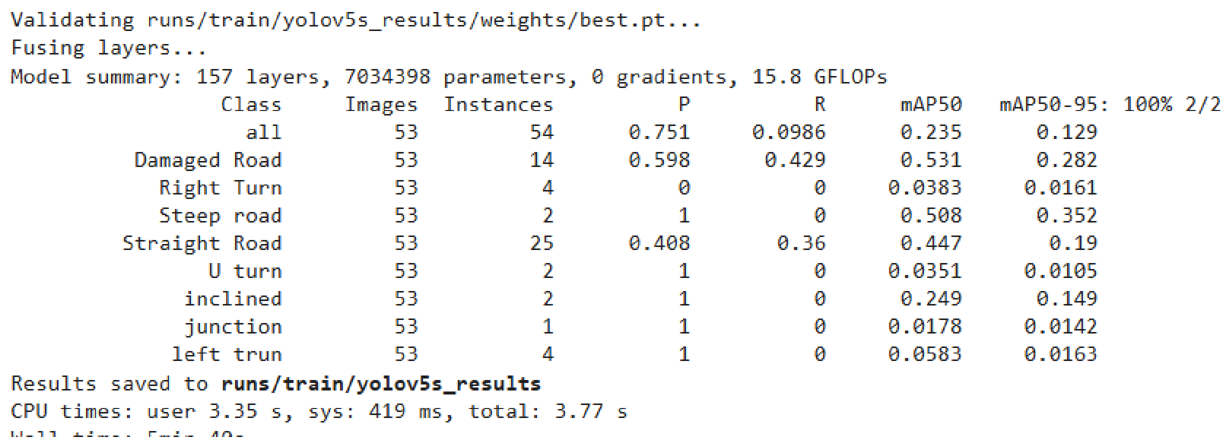
4. Model Training

4.1 Model Selection: YOLOv5

YOLOv5 were selected for their high performance in real-time object detection tasks. Both models are optimized for fast and accurate detection, ideal for deployment in regions requiring efficient processing.

4.2 Training Environment

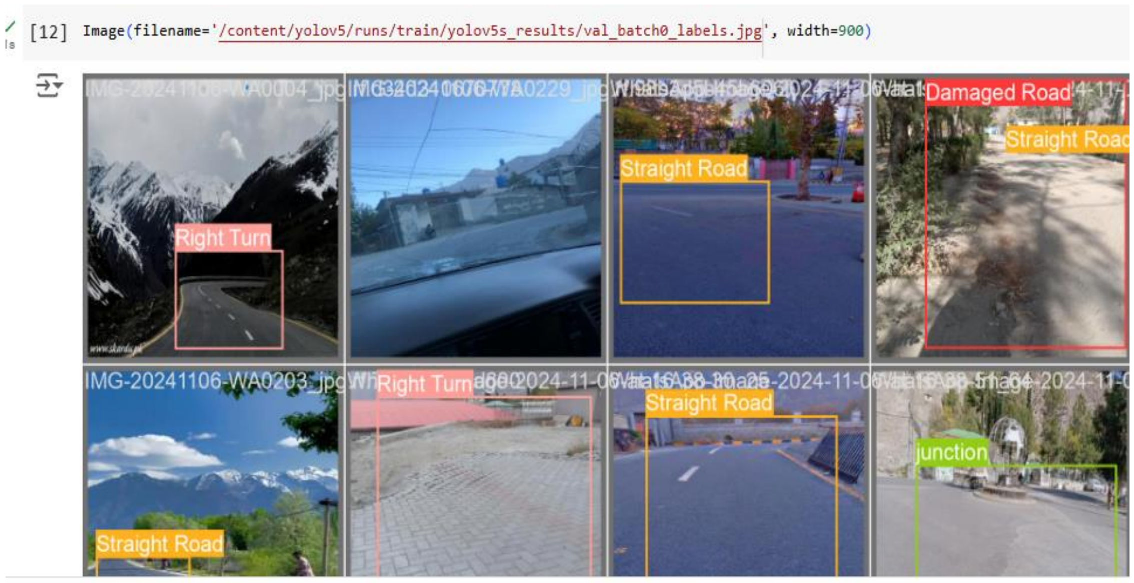
Training was conducted in Google Colab, leveraging GPU resources for accelerated model training. Due to challenges with internet connectivity and power, training sessions were strategically timed, and the models were regularly saved to prevent data loss.



```
Validating runs/train/yolov5s_results/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7034398 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 2/2
                   all         53         54      0.751     0.0986      0.235      0.129
          Damaged Road         53         14      0.598      0.429      0.531      0.282
            Right Turn         53          4          0          0     0.0383     0.0161
            Steep road         53          2          1          0      0.508      0.352
         Straight Road         53         25      0.408       0.36      0.447       0.19
                U turn         53          2          1          0     0.0351     0.0105
              inclined         53          2          1          0      0.249      0.149
              junction         53          1          1          0     0.0178     0.0142
             left trun         53          4          1          0     0.0583     0.0163
Results saved to runs/train/yolov5s_results
CPU times: user 3.35 s, sys: 419 ms, total: 3.77 s
```

4.3 Training Procedure

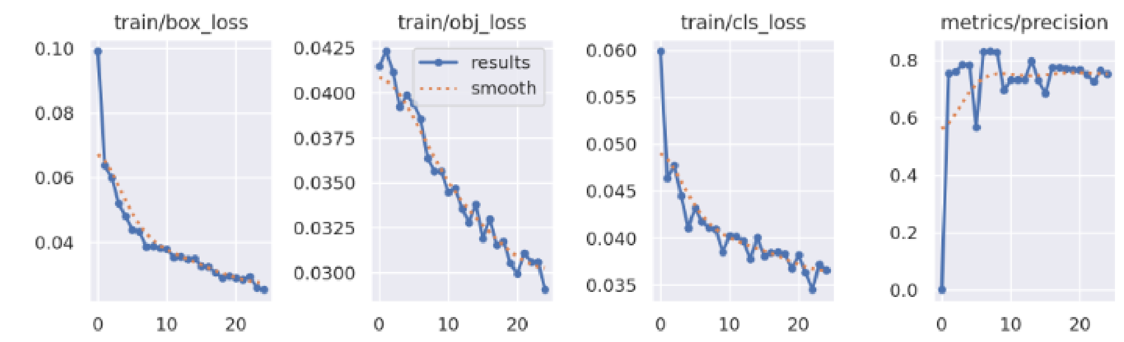
- **Configuration:** A custom YAML file was created to define the classes and data splits. Each model was configured with hyperparameters suited for small to moderate datasets, including batch size, learning rate, and image input size.
- **Model Architecture:** YOLO's backbone network was customized for the specific demands of road detection by focusing on small object detection and unique road shapes.
- **Training Process:** The model was trained over multiple epochs until convergence, with early stopping mechanisms to avoid overfitting. Each epoch's performance was monitored on validation data to assess the model's progress.



```
[12] Image(filename='/content/yolov5/runs/train/yolov5s_results/val_batch0_labels.jpg', width=900)
```

-

```
[11] from utils.plots import plot_results  # plot results.txt as results.png
     Image(filename='/content/yolov5/runs/train/yolov5s_results/results.png', width=1000)
```



4.4 Challenges and Solutions

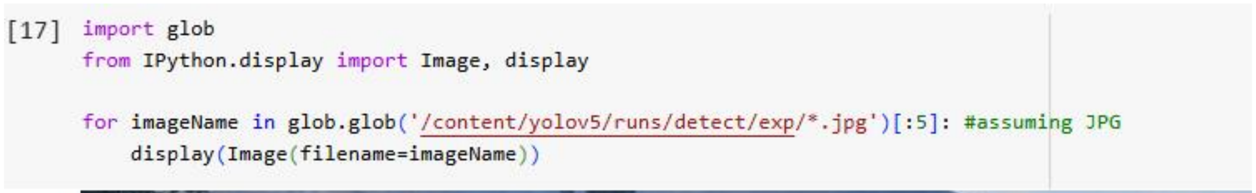
- **Power and Connectivity Issues:** Training was often interrupted. Checkpoints were created to save intermediate weights, enabling resumption without data loss.
- **Imbalanced Classes:** Techniques such as data augmentation and class weighting were used to handle classes with fewer instances, ensuring balanced learning.

5. Model Evaluation

5.1 Evaluation Metrics

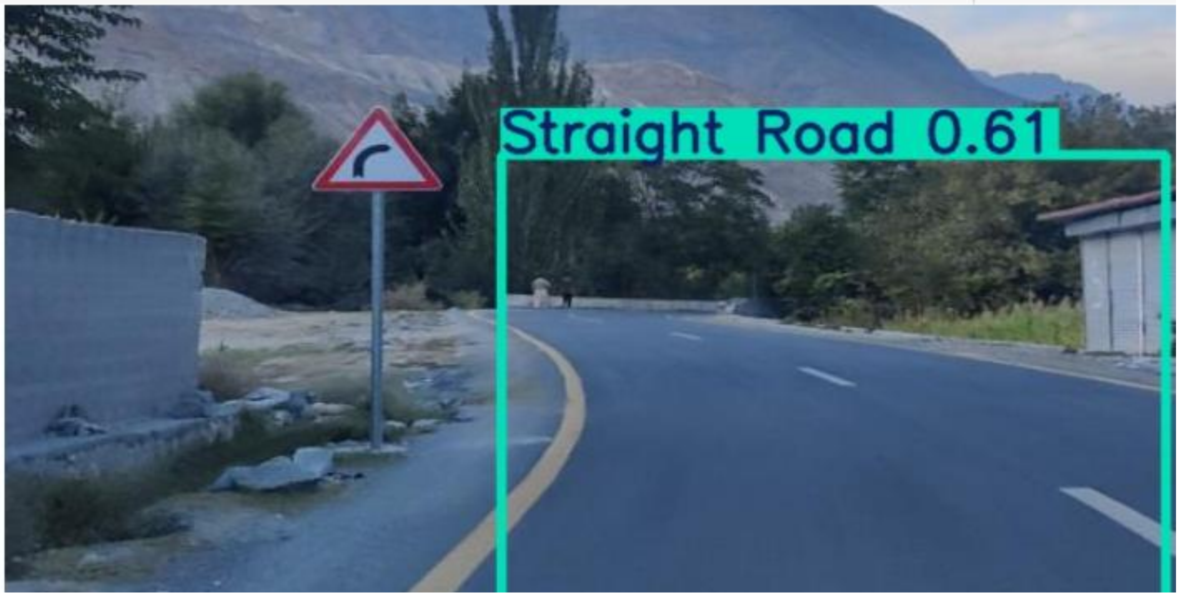
The model was evaluated using common object detection metrics:

- **Mean Average Precision (mAP):** Measures the model's overall detection accuracy.
- **Precision and Recall:** To assess the model’s ability to correctly identify road types without excessive false positives or negatives.
- **F1 Score:** A balanced measure combining precision and recall.



```
[17] import glob
     from IPython.display import Image, display

     for imageName in glob.glob('/content/yolov5/runs/detect/exp/*.jpg')[:5]: #assuming JPG
         display(Image(filename=imageName))
```



5.3 Analysis of Results

The model demonstrated high accuracy in identifying common road types (straight roads and damaged roads). Less frequent classes, such as “Junction” and “Two Way,” had lower performance due to limited samples. Data augmentation and potential synthetic data generation are considered for improving these classes' accuracy.

6. Conclusion

6.1 Summary

This project successfully developed a road detection model tailored to the unique terrains of Gilgit-Baltistan. The model effectively identifies various road types, providing a valuable tool for navigation and infrastructure management in mountainous regions.

6.2 Future Work

To further enhance model performance, the following steps are recommended:

- **Expand Dataset:** Collect more data on less-represented classes, possibly using synthetic data generation.
- **Model Tuning:** Experiment with alternative model architectures or ensemble methods.

- **Deployment:** Optimize the model for real-time inference on edge devices, making it viable for on-site applications.

6.3 Learning Outcomes

This project provided significant insights into deep learning, from dataset preparation and annotation to training and evaluation. Overcoming challenges with power and connectivity underscored the importance of resilient training practices.