|  |  |
| --- | --- |
|  |  |
|  |
|  |

**High Impact Skills Development Program for Gilgit Baltistan**

# YOLOv5 Object Detection Model Documentation

**Student Name: Kumail Abbas  
GitHub Link: <https://github.com/KumailAbbas1/Road_Detection_Project>**

**Registration : GIL-DASI-048**

**Email:kumail8407@gmaill.com**

## Overview

This project uses YOLOv5, a state-of-the-art object detection model, to detect specific objects in images. The model was trained on a custom dataset using Roboflow for dataset management and TensorFlow for model evaluation. Key steps include dataset preparation, environment setup, model training, and evaluation.

## Prerequisites

* **Python** (version 3.7+ recommended)
* **Jupyter Notebook** for code execution
* **Git** for repository cloning
* **YOLOv5** dependencies and **Roboflow** library for dataset management

## Setup

### 1. Clone YOLOv5 Repository

First, clone the YOLOv5 repository to obtain its scripts and files:

Python coad

!git clone https://github.com/ultralytics/yolov5

%cd yolov5

### 2. Install Required Packages

Install the necessary packages using pip, including Roboflow for dataset handling:

python

Copy code

%pip install -qr requirements.txt # Installs YOLOv5 dependencies

%pip install -q roboflow

### 3. Import Libraries

Import essential libraries for managing images, running inference, and clearing output in Jupyter:

python

Copy code

import torch

import os

from IPython.display import Image, clear\_output

## Dataset Preparation

### 1. Load and Download Dataset from Roboflow

Roboflow allows easy access and integration of custom datasets:

python

Copy code

from roboflow import Roboflow

# Replace with your own Roboflow API key

rf = Roboflow(api\_key="your\_api\_key\_here")

project = rf.workspace("workspace\_name").project("project\_name")

version = project.version(5)

dataset = version.download("yolov5")

### 2. Dataset Configuration

Ensure the dataset includes:

* **Data structure**: Separate folders for train, val, and test images.
* **Annotation format**: Labels in YOLO format, defined in a data.yaml file.

### 3. Data YAML File

The data.yaml file should specify:

* Paths for training and validation images
* Class labels

Example format:

yaml

Copy code

train: ../train/images

val: ../valid/images

nc: 7 # number of classes

names: ['angry', 'disgust', 'fear', 'happy', 'sad', 'surprise', 'neutral']

## Model Training

Use YOLOv5's train.py script to train the model:

python

Copy code

!python train.py --img 640 --epochs 120 --data {dataset.location}/data.yaml --weights yolov5s.pt

Parameters:

* --img 640: Image size for training.
* --epochs 120: Number of epochs.
* --data: Path to the data.yaml file.
* --weights yolov5s.pt: Pretrained weights file.

## Evaluation and Inference

After training, evaluate the model's performance on the validation dataset:

python

Copy code

!python val.py --data {dataset.location}/data.yaml --weights runs/train/exp/weights/best.pt

For inference, test the model on sample images:

python

Copy code

!python detect.py --source /path/to/test/images --weights runs/train/exp/weights/best.pt --img 640

## Results and Metrics

Results can include:

* **Confusion Matrix**: Visualizing true positives, false positives, and false negatives.
* **Precision, Recall, and F1 Score**: Evaluating detection accuracy.
* **mAP (mean Average Precision)**: Standard metric for object detection accuracy.

## Conclusion

This YOLOv5 model is now trained and ready for real-world object detection applications. For deployment, consider converting the model to a format compatible with mobile or web applications if necessary.