Smart Drone Surveillance

Intelligent Monitoring: Utilizing Drones for Effective and Adaptive Surveillance Solutions

Project Roadmap

1. Introduction
2. Methodology: Data Collection
3. Exploratory Data Analysis (EDA)
4. Feature Engineering
5. Model Selection
6. Model Training
7. Model Evaluation

1 Introduction

1.1 Overall aim of project:

The demand for real-time and adaptive surveillance has led to the integration of advanced AI-driven models into Computer Vision systems. This project presents an innovative approach for real-time drone surveillance, evaluating both untrained and dataset-trained models for general security and smart farming applications. The models are trained and deployed to analyze live drone footage, mapping detected objects onto the surveillance environment.

This project explores the design, implementation, and performance evaluation of surveillance approach that leverages drones to enhance situational awareness and operational effectiveness across various sectors, including public safety, border control, and environmental monitoring., highlighting its potential to revolutionize traditional surveillance methods.

The proposed system features drones equipped with advanced autonomous imaging technology, Data processing capabilities, and machine learning algorithms, enabling real-time monitoring, threat detection, and in-depth data analysis.

1.2 Research Question to be answered:

How can drones integrated with AI-powered Computer Vision enhance real-time surveillance, smart farming, improve threat detection, and optimize environmental security?

1.3 Keywords

Keywords: Drones, Computer Vision, AI, Smart Surveillance, Smart Farming

1.4 Usecase

Environmental monitoring and object detection for surveillance and smart farming.

2. Methodology

2.1 Dataset Preparation

- **KITTI-3D-Object-Detection-Dataset**

  KITTI 3D Object Detection Dataset For PointPillars Algorithm Access dataset

Exploratory Data Analysis

Step 1: Import Libraries

```
!pip install ultralytics
!pip install kaggle

import os
import datetime
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import tensorflow as tf
from tensorflow.keras import Input, Model
from tensorflow.keras.callbacks import EarlyStopping, TensorBoard
from tensorflow.keras.preprocessing.image import load_img,
img_to_array, ImageDataGenerator
from tensorflow.keras.layers import Conv2D, DepthwiseConv2D, Dense,
Concatenate, Dropout, MaxPooling2D, GlobalAveragePooling2D
from tensorflow.keras.utils import plot_model
from ultralytics import YOLO

print("All imports have been successfully added.")
```

```
Collecting ultralytics
  Downloading ultralytics-8.3.78-py3-none-any.whl.metadata (35 kB)
Requirement already satisfied: numpy<=2.1.1,>=1.23.0 in
/usr/local/lib/python3.11/dist-packages (from ultralytics) (1.26.4)
Requirement already satisfied: matplotlib>=3.3.0 in
/usr/local/lib/python3.11/dist-packages (from ultralytics) (3.10.0)
Requirement already satisfied: opencv-python>=4.6.0 in
/usr/local/lib/python3.11/dist-packages (from ultralytics) (4.11.0.86)
Requirement already satisfied: pillow>=7.1.2 in
/usr/local/lib/python3.11/dist-packages (from ultralytics) (11.1.0)
Requirement already satisfied: pyyaml>=5.3.1 in
/usr/local/lib/python3.11/dist-packages (from ultralytics) (6.0.2)
Requirement already satisfied: requests>=2.23.0 in
/usr/local/lib/python3.11/dist-packages (from ultralytics) (2.32.3)
Requirement already satisfied: scipy>=1.4.1 in
/usr/local/lib/python3.11/dist-packages (from ultralytics) (1.13.1)
Requirement already satisfied: torch>=1.8.0 in
```

```
/usr/local/lib/python3.11/dist-packages (from ultralytics)
(2.5.1+cu124)
Requirement already satisfied: torchvision>=0.9.0 in
/usr/local/lib/python3.11/dist-packages (from ultralytics)
(0.20.1+cu124)
Requirement already satisfied: tqdm>=4.64.0 in
/usr/local/lib/python3.11/dist-packages (from ultralytics) (4.67.1)
Requirement already satisfied: psutil in
/usr/local/lib/python3.11/dist-packages (from ultralytics) (5.9.5)
Requirement already satisfied: py-cpuinfo in
/usr/local/lib/python3.11/dist-packages (from ultralytics) (9.0.0)
Requirement already satisfied: pandas>=1.1.4 in
/usr/local/lib/python3.11/dist-packages (from ultralytics) (2.2.2)
Requirement already satisfied: seaborn>=0.11.0 in
/usr/local/lib/python3.11/dist-packages (from ultralytics) (0.13.2)
Collecting ultralytics-thop>=2.0.0 (from ultralytics)
  Downloading ultralytics_thop-2.0.14-py3-none-any.whl.metadata (9.4
kB)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib>=3.3.0-
>ultralytics) (1.3.1)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.11/dist-packages (from matplotlib>=3.3.0-
>ultralytics) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib>=3.3.0-
>ultralytics) (4.56.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib>=3.3.0-
>ultralytics) (1.4.8)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib>=3.3.0-
>ultralytics) (24.2)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib>=3.3.0-
>ultralytics) (3.2.1)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.11/dist-packages (from matplotlib>=3.3.0-
>ultralytics) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.11/dist-packages (from pandas>=1.1.4-
>ultralytics) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.11/dist-packages (from pandas>=1.1.4-
>ultralytics) (2025.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.23.0-
>ultralytics) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in
```

```
/usr/local/lib/python3.11/dist-packages (from requests>=2.23.0-
>ultralytics) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.23.0-
>ultralytics) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.23.0-
>ultralytics) (2025.1.31)
Requirement already satisfied: filelock in
/usr/local/lib/python3.11/dist-packages (from torch>=1.8.0-
>ultralytics) (3.17.0)
Requirement already satisfied: typing-extensions>=4.8.0 in
/usr/local/lib/python3.11/dist-packages (from torch>=1.8.0-
>ultralytics) (4.12.2)
Requirement already satisfied: networkx in
/usr/local/lib/python3.11/dist-packages (from torch>=1.8.0-
>ultralytics) (3.4.2)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.11/dist-packages (from torch>=1.8.0-
>ultralytics) (3.1.5)
Requirement already satisfied: fsspec in
/usr/local/lib/python3.11/dist-packages (from torch>=1.8.0-
>ultralytics) (2024.10.0)
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch>=1.8.0-
>ultralytics)
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from torch>=1.8.0-
>ultralytics)
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from torch>=1.8.0-
>ultralytics)
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch>=1.8.0-
>ultralytics)
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.4.5.8 (from torch>=1.8.0-
>ultralytics)
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.2.1.3 (from torch>=1.8.0-
>ultralytics)
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.5.147 (from torch>=1.8.0-
>ultralytics)
```

```
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch>=1.8.0-
>ultralytics)
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch>=1.8.0-
>ultralytics)
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in
/usr/local/lib/python3.11/dist-packages (from torch>=1.8.0-
>ultralytics) (2.21.5)
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in
/usr/local/lib/python3.11/dist-packages (from torch>=1.8.0-
>ultralytics) (12.4.127)
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch>=1.8.0-
>ultralytics)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Requirement already satisfied: triton==3.1.0 in
/usr/local/lib/python3.11/dist-packages (from torch>=1.8.0-
>ultralytics) (3.1.0)
Requirement already satisfied: sympy==1.13.1 in
/usr/local/lib/python3.11/dist-packages (from torch>=1.8.0-
>ultralytics) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.11/dist-packages (from sympy==1.13.1-
>torch>=1.8.0->ultralytics) (1.3.0)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7-
>matplotlib>=3.3.0->ultralytics) (1.17.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.11/dist-packages (from jinja2->torch>=1.8.0-
>ultralytics) (3.0.2)
Downloading ultralytics-8.3.78-py3-none-any.whl (921 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 921.5/921.5 kB 12.8 MB/s eta
0:00:00
anylinux2014_x86_64.whl (363.4 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 4.8 MB/s eta
0:00:00
anylinux2014_x86_64.whl (13.8 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 70.6 MB/s eta
0:00:00
anylinux2014_x86_64.whl (24.6 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 59.2 MB/s eta
0:00:00
e_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (883 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 38.5 MB/s eta
```

```
0:00:00
anylinux2014_x86_64.whl (664.8 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 2.0 MB/s eta
0:00:00
anylinux2014_x86_64.whl (211.5 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 6.2 MB/s eta
0:00:00
anylinux2014_x86_64.whl (56.3 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 12.1 MB/s eta
0:00:00
anylinux2014_x86_64.whl (127.9 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 7.7 MB/s eta
0:00:00
anylinux2014_x86_64.whl (207.5 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 5.6 MB/s eta
0:00:00
anylinux2014_x86_64.whl (21.1 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 18.9 MB/s eta
0:00:00
e-cu12, nvidia-cuda-nvrtc-cu12, nvidia-cuda-cupti-cu12, nvidia-cublas-
cu12, nvidia-cusparse-cu12, nvidia-cudnn-cu12, nvidia-cusolver-cu12,
ultralytics-thop, ultralytics
  Attempting uninstall: nvidia-nvjitlink-cu12
    Found existing installation: nvidia-nvjitlink-cu12 12.5.82
    Uninstalling nvidia-nvjitlink-cu12-12.5.82:
      Successfully uninstalled nvidia-nvjitlink-cu12-12.5.82
  Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.6.82
    Uninstalling nvidia-curand-cu12-10.3.6.82:
      Successfully uninstalled nvidia-curand-cu12-10.3.6.82
  Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.2.3.61
    Uninstalling nvidia-cufft-cu12-11.2.3.61:
      Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
  Attempting uninstall: nvidia-cuda-runtime-cu12
    Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
    Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
    Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-cupti-cu12
    Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
    Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
  Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.5.3.2
    Uninstalling nvidia-cublas-cu12-12.5.3.2:
```

```
      Successfully uninstalled nvidia-cublas-cu12-12.5.3.2
  Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.5.1.3
    Uninstalling nvidia-cusparse-cu12-12.5.1.3:
      Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
  Attempting uninstall: nvidia-cudnn-cu12
    Found existing installation: nvidia-cudnn-cu12 9.3.0.75
    Uninstalling nvidia-cudnn-cu12-9.3.0.75:
      Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
  Attempting uninstall: nvidia-cusolver-cu12
    Found existing installation: nvidia-cusolver-cu12 11.6.3.83
    Uninstalling nvidia-cusolver-cu12-11.6.3.83:
      Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
Successfully installed nvidia-cublas-cu12-12.4.5.8 nvidia-cuda-cupti-
cu12-12.4.127 nvidia-cuda-nvrtc-cu12-12.4.127 nvidia-cuda-runtime-
cu12-12.4.127 nvidia-cudnn-cu12-9.1.0.70 nvidia-cufft-cu12-11.2.1.3
nvidia-curand-cu12-10.3.5.147 nvidia-cusolver-cu12-11.6.1.9 nvidia-
cusparse-cu12-12.3.1.170 nvidia-nvjitlink-cu12-12.4.127 ultralytics-
8.3.78 ultralytics-thop-2.0.14
Requirement already satisfied: kaggle in
/usr/local/lib/python3.11/dist-packages (1.6.17)
Requirement already satisfied: six>=1.10 in
/usr/local/lib/python3.11/dist-packages (from kaggle) (1.17.0)
Requirement already satisfied: certifi>=2023.7.22 in
/usr/local/lib/python3.11/dist-packages (from kaggle) (2025.1.31)
Requirement already satisfied: python-dateutil in
/usr/local/lib/python3.11/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in
/usr/local/lib/python3.11/dist-packages (from kaggle) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-
packages (from kaggle) (4.67.1)
Requirement already satisfied: python-slugify in
/usr/local/lib/python3.11/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: urllib3 in
/usr/local/lib/python3.11/dist-packages (from kaggle) (2.3.0)
Requirement already satisfied: bleach in
/usr/local/lib/python3.11/dist-packages (from kaggle) (6.2.0)
Requirement already satisfied: webencodings in
/usr/local/lib/python3.11/dist-packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in
/usr/local/lib/python3.11/dist-packages (from python-slugify->kaggle)
(1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests->kaggle)
(3.4.1)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.11/dist-packages (from requests->kaggle) (3.10)
Creating new Ultralytics Settings v0.0.6 file □
View Ultralytics Settings with 'yolo settings' or at
```

```
'/root/.config/Ultralytics/settings.json'
Update Settings with 'yolo settings key=value', i.e. 'yolo settings
runs_dir=path/to/dir'. For help see
https://docs.ultralytics.com/quickstart/#ultralytics-settings.
All imports have been successfully added.
```

Step 2 : Download and connect dataset

Download Dataset

Downlod the KITTI dataset from Kaggle using kaggle library.

```
import os

# Set the environment variable for Kaggle API to find kaggle.json
os.environ["KAGGLE_CONFIG_DIR"] = "/kaggle/config"
!chmod 600 /kaggle/config/kaggle.json

print("Successfuly connected to Kaggle.")
!kaggle datasets list

import os

# Ensure Kaggle API key is set correctly
os.environ["KAGGLE_CONFIG_DIR"] = "/kaggle/config"

# Download the dataset to a writable directory
!kaggle datasets download -d garymk/kitti-3d-object-detection-dataset
-p /kaggle/working/kitti

print("Dataset downloaded successfully to /kaggle/working")
```

```
chmod: cannot access '/kaggle/config/kaggle.json': No such file or
directory
Successfuly connected to Kaggle.
Traceback (most recent call last):
  File "/usr/local/bin/kaggle", line 4, in <module>
    from kaggle.cli import main
  File "/usr/local/lib/python3.11/dist-packages/kaggle/__init__.py",
line 7, in <module>
    api.authenticate()
  File
"/usr/local/lib/python3.11/dist-packages/kaggle/api/kaggle_api_extende
d.py", line 407, in authenticate
    raise IOError('Could not find {}. Make sure it\'s located in'
OSError: Could not find kaggle.json. Make sure it's located in
/kaggle/config. Or use the environment method. See setup instructions
at https://github.com/Kaggle/kaggle-api/
Dataset URL: https://www.kaggle.com/datasets/garymk/kitti-3d-object-
detection-dataset
```

```
License(s): unknown
Downloading kitti-3d-object-detection-dataset.zip to
/kaggle/working/kitti
100% 30.0G/30.0G [05:18<00:00, 29.9MB/s]
100% 30.0G/30.0G [05:19<00:00, 101MB/s]
Dataset downloaded successfully to /kaggle/working
```

Step3 : Load and Preprocess dataset

To use the KITTI dataset for training, the images are split into training and validation sets. Shuffle and split (80% train, 20% val).

```
import os
import shutil
import random
import zipfile
import os

zip_path = "/kaggle/working/kitti/kitti-3d-object-detection-
dataset.zip"
extract_path = "/kaggle/working/kitti"  # Change if needed

# Ensure the extraction directory exists
os.makedirs(extract_path, exist_ok=True)

# Unzip the file
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)

print(f"Dataset extracted to: {extract_path}")


# Paths
image_folder = "/kaggle/working/kitti/testing/image_2"
train_folder = "/kaggle/working/kitti/images/train"
val_folder = "/kaggle/working/kitti/images/val"

# Create train/val folders if they don't exist
os.makedirs(train_folder, exist_ok=True)
os.makedirs(val_folder, exist_ok=True)

# Get all image files
image_files = [f for f in os.listdir(image_folder) if
f.endswith((".png", ".jpg", ".jpeg"))]

# Shuffle and split (80% train, 20% val)
random.shuffle(image_files)
split_idx = int(0.8 * len(image_files))
train_files, val_files = image_files[:split_idx],
image_files[split_idx:]
```

```
# Move files
for file in train_files:
    shutil.move(os.path.join(image_folder, file),
os.path.join(train_folder, file))

for file in val_files:
    shutil.move(os.path.join(image_folder, file),
os.path.join(val_folder, file))

print("Dataset split into training and validation sets.")
```

```
Dataset extracted to: /kaggle/working/kitti
Dataset split into training and validation sets.
```

Step 3 : Preprocess the dataset

```
import pickle
import os
import yaml

# Load and Inspect the Pickle File

# Read KITTI dataset .pkl files, print its structure

def load_and_inspect_pkl(file_path):
    """Load and inspect a pickle file."""
    with open(file_path, 'rb') as f:
        data = pickle.load(f)

    print(f"\nLoaded: {file_path}")
    print("Data Type:", type(data))  # Likely a list or dict
    if isinstance(data, dict):
        print("Keys:", data.keys())  # Print dictionary keys if
applicable
    else:
        print("First 5 items:", data[:5])  # Print first 5 items if
it's a list

    return data


# Convert KITTI Annotations to YOLO Format

# convert_kitti_to_yolo(kitti_data, output_folder) extracts bounding
# box annotations from the KITTI dataset and converts them into the
YOLO
# format (class x_center y_center width height). It normalizes
bounding box
# coordinates based on the image width (1242) and height (375).
```

```python
def convert_kitti_to_yolo(kitti_data, output_folder, image_width=1242,
image_height=375):
    """Convert KITTI labels to YOLO format and save them."""

    # Debugging: Check if kitti_data is a dictionary
    if not isinstance(kitti_data, dict):
        raise TypeError(f"Expected kitti_data to be a dictionary, got
{type(kitti_data)}")

    # Ensure 'infos' key exists
    if "infos" not in kitti_data:
        raise KeyError("Invalid KITTI data format. 'infos' key
missing.")

    os.makedirs(output_folder, exist_ok=True)

    for i, entry in enumerate(kitti_data["infos"]):
        label_file = os.path.join(output_folder, f"{i:06d}.txt")

        with open(label_file, "w") as f:
            for obj, bbox in zip(entry["annos"]["name"],
entry["annos"]["bbox"]):
                if obj not in CLASS_MAPPING:
                    continue  # Skip unknown classes

                class_id = CLASS_MAPPING[obj]
                x_min, y_min, x_max, y_max = bbox
                x_center = ((x_min + x_max) / 2) / image_width
                y_center = ((y_min + y_max) / 2) / image_height
                width = (x_max - x_min) / image_width
                height = (y_max - y_min) / image_height

                f.write(f"{class_id} {x_center:.6f} {y_center:.6f}
{width:.6f} {height:.6f}\n")

# create_yaml(dataset_path, class_names) creates a dataset.yaml file
# specifying paths for YOLO training and the number of object classes.

def create_yaml(dataset_path, class_names):
    """Generate a dataset YAML file for YOLOv8 training."""
    data_yaml = {
        'train': os.path.join(dataset_path, 'images', 'train'),
        'val': os.path.join(dataset_path, 'images', 'val'),
        'nc': len(class_names),
        'names': class_names
    }

    with open(os.path.join(dataset_path, 'dataset.yaml'), 'w') as f:
```

```
        yaml.dump(data_yaml, f, default_flow_style=False)

# Define class mapping for outdoor drone footage (250 classes)
CLASS_MAPPING = {
    "Car": 0, "Pedestrian": 1, "Cyclist": 2, "Truck": 3, "Bus": 4,
"Motorcycle": 5, "Bicycle": 6,
    "Van": 7, "Taxi": 8, "SUV": 9, "Emergency Vehicle": 10, "Trailer":
11, "Boat": 12, "Train": 13,
    "Helicopter": 14, "Airplane": 15, "Ambulance": 16, "Fire Truck":
17, "Police Car": 18,
    "Horse": 19, "Cow": 20, "Dog": 21, "Cat": 22, "Sheep": 23, "Goat":
24, "Pig": 25,
    "Deer": 26, "Bear": 27, "Elephant": 28, "Giraffe": 29, "Zebra":
30, "Camel": 31, "Kangaroo": 32,
    "Rabbit": 33, "Squirrel": 34, "Monkey": 35, "Bird": 36, "Eagle":
37, "Owl": 38, "Pigeon": 39,
    "Crow": 40, "Seagull": 41, "Duck": 42, "Goose": 43, "Swan": 44,
"Peacock": 45, "Parrot": 46,
    "Stork": 47, "Hawk": 48, "Falcon": 49, "Turtle": 50, "Snake": 51,
"Lizard": 52, "Frog": 53,
    "Crocodile": 54, "Alligator": 55, "Shark": 56, "Dolphin": 57,
"Whale": 58, "Fish": 59,
    "Octopus": 60, "Crab": 61, "Jellyfish": 62, "Starfish": 63,
"Coral": 64, "Tree": 65,
    "Palm Tree": 66, "Bush": 67, "Flower": 68, "Grass": 69, "Rock":
70, "Boulder": 71,
    "Hill": 72, "Mountain": 73, "River": 74, "Lake": 75, "Pond": 76,
"Waterfall": 77,
    "Bridge": 78, "Dam": 79, "Cave": 80, "Cliff": 81, "Canyon": 82,
"Desert": 83,
    "Sand Dune": 84, "Oasis": 85, "Glacier": 86, "Volcano": 87,
"Island": 88,
    "Lighthouse": 89, "Windmill": 90, "Water Tower": 91, "Skyscraper":
92,
    "Building": 93, "House": 94, "Apartment": 95, "Warehouse": 96,
"Factory": 97,
    "Gas Station": 98, "Shop": 99, "Mall": 100, "Hospital": 101,
"School": 102,
    "University": 103, "Church": 104, "Mosque": 105, "Temple": 106,
"Stadium": 107,
    "Playground": 108, "Parking Lot": 109, "Road": 110, "Highway":
111, "Intersection": 112,
    "Traffic Light": 113, "Stop Sign": 114, "Road Sign": 115,
"Billboard": 116,
    "Fence": 117, "Wall": 118, "Gate": 119, "Pavement": 120,
"Sidewalk": 121,
    "Crosswalk": 122, "Train Tracks": 123, "Tunnel": 124, "Metro
Station": 125,
    "Airport": 126, "Runway": 127, "Helipad": 128, "Dock": 129,
```

```
"Pier": 130,
    "Crane": 131, "Construction Site": 132, "Scaffolding": 133,
"Bulldozer": 134,
    "Excavator": 135, "Dump Truck": 136, "Cement Mixer": 137,
"Forklift": 138,
    "Wind Turbine": 139, "Solar Panel": 140, "Power Line": 141,
"Electric Tower": 142,
    "Satellite Dish": 143, "Radio Tower": 144, "Cell Tower": 145,
"Fire Hydrant": 146,
    "Mailbox": 147, "Street Lamp": 148, "Bench": 149, "Trash Can":
150, "Recycling Bin": 151,
    "Bike Rack": 152, "Fountain": 153, "Statue": 154, "Sculpture":
155, "Public Art": 156,
    "Billboard Sign": 157, "Graffiti": 158, "Drone": 159, "Camera":
160, "Radar": 161,
    "Satellite": 162, "Rocket": 163, "Meteor": 164, "Sun": 165,
"Moon": 166, "Star": 167,
    "Cloud": 168, "Rain": 169, "Snow": 170, "Fog": 171, "Tornado":
172, "Lightning": 173,
    "Rainbow": 174, "Tsunami": 175, "Hurricane": 176, "Earthquake
Ruins": 177,
    "Wildfire": 178, "Lava Flow": 179, "Avalanche": 180, "Solar
Eclipse": 181,
    "Lunar Eclipse": 182, "Wave": 183, "Iceberg": 184, "Coral Reef":
185,
    "Submarine": 186, "Fishing Boat": 187, "Cargo Ship": 188,
"Speedboat": 189,
    "Jet Ski": 190, "Yacht": 191, "Cruise Ship": 192, "Sailboat": 193,
"Rowboat": 194,
    "Canoe": 195, "Kayak": 196, "Barge": 197, "Tank": 198, "Military
Jeep": 199,
    "Aircraft Carrier": 200, "Missile": 201, "Warplane": 202, "UFO":
203,
    "Drone Swarm": 204, "Hot Air Balloon": 205, "Paraglider": 206,
"Skydiver": 207,
    "Windsurfer": 208, "Kite": 209, "Sandcastle": 210, "Surfer": 211,
"Lifeguard": 212,
    "Diver": 213, "Snorkeler": 214, "Scuba Tank": 215, "Underwater
Cave": 216,
    "Seashell": 217, "Fishing Net": 218, "Fishing Rod": 219,
"Lifebuoy": 220,
    "Floating Dock": 221, "Underwater Wreck": 222, "Sunken Ship": 223,
    "Underwater Pipeline": 224, "Oil Rig": 225, "Coral Bleaching":
226,
    "Plastic Pollution": 227, "Bird Nest": 228, "Beehive": 229,
"Spider Web": 230,
    "Forest Fire": 231, "Logging Site": 232, "Deforested Land": 233,
"Reforestation Area": 234,
    "Solar Farm": 235, "Wind Farm": 236, "Hydroelectric Dam": 237,
```

```
"Geothermal Plant": 238,
    "Nuclear Plant": 239, "Coal Mine": 240, "Gold Mine": 241, "Diamond
Mine": 242,
    "Oil Pipeline": 243, "Gas Flare": 244, "Fossil Fuel Plant": 245,
"Carbon Emission": 246,
    "Air Pollution": 247, "Waste Dump": 248, "Landfill": 249,
"Recycling Plant": 250
}



# Load train pickle file
pkl_files = [
    "/kaggle/working/kitti/kitti_infos_train.pkl",
]

data_infos = load_and_inspect_pkl(pkl_files[0])

# Convert and save training labels
convert_kitti_to_yolo({"infos": data_infos},
output_folder="/kaggle/working/kitti/labels/train")

# Load validate pickle file
pkl_files = [
    "/kaggle/working/kitti/kitti_infos_val.pkl",
]

data_infos = load_and_inspect_pkl(pkl_files[0])

# Convert and save validate labels
convert_kitti_to_yolo({"infos": data_infos},
output_folder="/kaggle/working/kitti/labels/val")

# Create dataset.yaml
create_yaml("/kaggle/working/kitti", list(CLASS_MAPPING.keys()))

print("YOLO labels and training_presets/dataset.yaml created
successfully!")


Loaded: /kaggle/working/kitti/kitti_infos_train.pkl
Data Type: <class 'list'>
First 5 items: [{'image_idx': 0, 'pointcloud_num_features': 4,
'velodyne_path': 'training/velodyne/000000.bin', 'img_path':
'training/image_2/000000.png', 'img_shape': array([ 370, 1224],
dtype=int32), 'calib/P0': array([[     707.05,           0,
604.08,           0],
       [          0,      707.05,      180.51,           0],
       [          0,           0,           1,           0],
       [          0,           0,           0,           1]]),
'calib/P1': array([[     707.05,           0,      604.08,     -
```

```
379.78],
       [          0,      707.05,      180.51,           0],
       [          0,           0,           1,           0],
       [          0,           0,           0,           1]]),
'calib/P2': array([[     707.05,           0,      604.08,
45.758],
       [          0,      707.05,      180.51,    -0.34542],
       [          0,           0,           1,    0.004981],
       [          0,           0,           0,           1]]),
'calib/P3': array([[     707.05,           0,      604.08,     -
334.11],
       [          0,      707.05,      180.51,      2.3307],
       [          0,           0,           1,   0.0032012],
       [          0,           0,           0,           1]]),
'calib/R0_rect': array([[    0.99991,    0.010093,  -0.0085119,
0],
       [  -0.010127,     0.99994,  -0.0040377,           0],
       [  0.0084707,   0.0041235,     0.99996,           0],
       [          0,           0,           0,           1]]),
'calib/Tr_velo_to_cam': array([[   0.006928,    -0.99997,  -0.0027578,
-0.024577],
       [  -0.001163,   0.0027498,          -1,   -0.061272],
       [    0.99998,   0.0069311,  -0.0011439,     -0.3321],
       [          0,           0,           0,           1]]),
'calib/Tr_imu_to_velo': array([[          1,  0.00075531,  -0.0020358,
-0.80868],
       [ -0.0007854,     0.99989,   -0.014823,     0.31956],
       [  0.0020244,    0.014825,     0.99989,    -0.79972],
       [          0,           0,           0,           1]]),
'annos': {'name': array(['Pedestrian'], dtype='<U10'), 'truncated':
array([          0]), 'occluded': array([0]), 'alpha': array([       -
0.2]), 'bbox': array([[      712.4,         143,      810.73,
307.92]]), 'dimensions': array([[        1.2,        1.89,
0.48]]), 'location': array([[       1.84,        1.47,        8.41]]),
'rotation_y': array([       0.01]), 'score': array([          0]),
'index': array([0], dtype=int32), 'group_ids': array([0],
dtype=int32), 'difficulty': array([0], dtype=int32),
'num_points_in_gt': array([377], dtype=int32)}}, {'image_idx': 3,
'pointcloud_num_features': 4, 'velodyne_path':
'training/velodyne/000003.bin', 'img_path':
'training/image_2/000003.png', 'img_shape': array([ 375, 1242],
dtype=int32), 'calib/P0': array([[     721.54,           0,
609.56,           0],
       [          0,      721.54,      172.85,           0],
       [          0,           0,           1,           0],
       [          0,           0,           0,           1]]),
'calib/P1': array([[     721.54,           0,      609.56,     -
387.57],
       [          0,      721.54,      172.85,           0],
```

```
       [          0,           0,           1,           0],
       [          0,           0,           0,           1]]),
'calib/P2': array([[     721.54,           0,      609.56,
44.857],
       [          0,      721.54,      172.85,     0.21638],
       [          0,           0,           1,   0.0027459],
       [          0,           0,           0,           1]]),
'calib/P3': array([[     721.54,           0,      609.56,     -
339.52],
       [          0,      721.54,      172.85,      2.1999],
       [          0,           0,           1,   0.0027299],
       [          0,           0,           0,           1]]),
'calib/R0_rect': array([[    0.99992,   0.0098378,   -0.007445,
0],
       [ -0.0098698,     0.99994,  -0.0042785,           0],
       [  0.0074025,   0.0043516,     0.99996,           0],
       [          0,           0,           0,           1]]),
'calib/Tr_velo_to_cam': array([[  0.0075337,    -0.99997,  -0.0006166,
-0.0040698],
       [   0.014802,  0.00072807,    -0.99989,   -0.076316],
       [    0.99986,   0.0075238,    0.014808,    -0.27178],
       [          0,           0,           0,           1]]),
'calib/Tr_imu_to_velo': array([[          1,  0.00075531,  -0.0020358,
-0.80868],
       [ -0.0007854,     0.99989,   -0.014823,     0.31956],
       [  0.0020244,    0.014825,     0.99989,    -0.79972],
       [          0,           0,           0,           1]]),
'annos': {'name': array(['Car', 'DontCare', 'DontCare'], dtype='<U8'),
'truncated': array([          0,          -1,          -1]),
'occluded': array([ 0, -1, -1]), 'alpha': array([       1.55,
-10,         -10]), 'bbox': array([[     614.24,      181.78,
727.31,      284.77],
       [          5,      229.89,      214.12,      367.61],
       [     522.25,      202.35,      547.77,      219.71]]),
'dimensions': array([[       4.15,        1.57,        1.73],
       [         -1,          -1,          -1],
       [         -1,          -1,          -1]]), 'location': array([[
1,        1.75,       13.22],
       [      -1000,       -1000,       -1000],
       [      -1000,       -1000,       -1000]]), 'rotation_y':
array([       1.62,         -10,         -10]), 'score': array([
0,           0,           0]), 'index': array([ 0, -1, -1],
dtype=int32), 'group_ids': array([0, 1, 2], dtype=int32),
'difficulty': array([ 0,  0, -1], dtype=int32), 'num_points_in_gt':
array([674,  -1,  -1], dtype=int32)}}, {'image_idx': 7,
'pointcloud_num_features': 4, 'velodyne_path':
'training/velodyne/000007.bin', 'img_path':
'training/image_2/000007.png', 'img_shape': array([ 375, 1242],
dtype=int32), 'calib/P0': array([[     721.54,           0,
```

```
609.56,           0],
       [          0,      721.54,      172.85,           0],
       [          0,           0,           1,           0],
       [          0,           0,           0,           1]]),
'calib/P1': array([[     721.54,           0,      609.56,     -
387.57],
       [          0,      721.54,      172.85,           0],
       [          0,           0,           1,           0],
       [          0,           0,           0,           1]]),
'calib/P2': array([[     721.54,           0,      609.56,
44.857],
       [          0,      721.54,      172.85,     0.21638],
       [          0,           0,           1,   0.0027459],
       [          0,           0,           0,           1]]),
'calib/P3': array([[     721.54,           0,      609.56,     -
339.52],
       [          0,      721.54,      172.85,      2.1999],
       [          0,           0,           1,   0.0027299],
       [          0,           0,           0,           1]]),
'calib/R0_rect': array([[    0.99992,   0.0098378,   -0.007445,
0],
       [ -0.0098698,     0.99994,  -0.0042785,           0],
       [  0.0074025,   0.0043516,     0.99996,           0],
       [          0,           0,           0,           1]]),
'calib/Tr_velo_to_cam': array([[  0.0075337,    -0.99997,  -0.0006166,
-0.0040698],
       [   0.014802,  0.00072807,    -0.99989,   -0.076316],
       [    0.99986,   0.0075238,    0.014808,    -0.27178],
       [          0,           0,           0,           1]]),
'calib/Tr_imu_to_velo': array([[          1,  0.00075531,  -0.0020358,
-0.80868],
       [ -0.0007854,     0.99989,   -0.014823,     0.31956],
       [  0.0020244,    0.014825,     0.99989,    -0.79972],
       [          0,           0,           0,           1]]),
'annos': {'name': array(['Car', 'Car', 'Car', 'Cyclist', 'DontCare',
'DontCare'], dtype='<U8'), 'truncated': array([          0,
0,           0,           0,          -1,          -1]), 'occluded':
array([ 0,  0,  0,  0, -1, -1]), 'alpha': array([      -1.56,
1.71,        1.64,        1.89,         -10,         -10]), 'bbox':
array([[     564.62,      174.59,      616.43,      224.74],
       [     481.59,      180.09,      512.55,      202.42],
       [     542.05,      175.55,      565.27,      193.79],
       [      330.6,      176.09,      355.61,       213.6],
       [     753.33,      164.32,         798,      186.74],
       [      738.5,      171.32,      753.27,      184.42]]),
'dimensions': array([[        3.2,        1.61,        1.66],
       [        3.7,         1.4,        1.51],
       [       4.05,        1.46,        1.66],
       [       1.95,        1.72,         0.5],
```

```
       [         -1,          -1,          -1],
       [         -1,          -1,          -1]]), 'location': array([[
-0.69,        1.69,       25.01],
       [      -7.43,        1.88,       47.55],
       [      -4.71,        1.71,       60.52],
       [     -12.63,        1.88,       34.09],
       [      -1000,       -1000,       -1000],
       [      -1000,       -1000,       -1000]]), 'rotation_y':
array([      -1.59,        1.55,        1.56,        1.54,          -
10,         -10]), 'score': array([          0,           0,
0,           0,           0,           0]), 'index': array([ 0,  1,
2,  3, -1, -1], dtype=int32), 'group_ids': array([0, 1, 2, 3, 4, 5],
dtype=int32), 'difficulty': array([ 0, -1, -1,  1, -1, -1],
dtype=int32), 'num_points_in_gt': array([182,  20,   5,  25,  -1,  -
1], dtype=int32)}}, {'image_idx': 9, 'pointcloud_num_features': 4,
'velodyne_path': 'training/velodyne/000009.bin', 'img_path':
'training/image_2/000009.png', 'img_shape': array([ 375, 1242],
dtype=int32), 'calib/P0': array([[     721.54,           0,
609.56,           0],
       [          0,      721.54,      172.85,           0],
       [          0,           0,           1,           0],
       [          0,           0,           0,           1]]),
'calib/P1': array([[     721.54,           0,      609.56,     -
387.57],
       [          0,      721.54,      172.85,           0],
       [          0,           0,           1,           0],
       [          0,           0,           0,           1]]),
'calib/P2': array([[     721.54,           0,      609.56,
44.857],
       [          0,      721.54,      172.85,     0.21638],
       [          0,           0,           1,   0.0027459],
       [          0,           0,           0,           1]]),
'calib/P3': array([[     721.54,           0,      609.56,     -
339.52],
       [          0,      721.54,      172.85,      2.1999],
       [          0,           0,           1,   0.0027299],
       [          0,           0,           0,           1]]),
'calib/R0_rect': array([[    0.99992,   0.0098378,   -0.007445,
0],
       [ -0.0098698,     0.99994,  -0.0042785,           0],
       [  0.0074025,   0.0043516,     0.99996,           0],
       [          0,           0,           0,           1]]),
'calib/Tr_velo_to_cam': array([[  0.0075337,    -0.99997,  -0.0006166,
-0.0040698],
       [   0.014802,  0.00072807,    -0.99989,   -0.076316],
       [    0.99986,   0.0075238,    0.014808,    -0.27178],
       [          0,           0,           0,           1]]),
'calib/Tr_imu_to_velo': array([[          1,  0.00075531,  -0.0020358,
-0.80868],
```

```
       [ -0.0007854,     0.99989,   -0.014823,     0.31956],
       [  0.0020244,    0.014825,     0.99989,    -0.79972],
       [          0,           0,           0,           1]]),
'annos': {'name': array(['Car', 'Car', 'Car', 'DontCare', 'DontCare'],
dtype='<U8'), 'truncated': array([          0,           0,
0,          -1,          -1]), 'occluded': array([ 0,  2,  0, -1, -
1]), 'alpha': array([       -1.5,         1.75,         1.78,          -
10,         -10]), 'bbox': array([[     601.96,      177.01,
659.15,      229.51],
       [     600.14,      177.09,      624.65,      193.31],
       [     574.98,      178.64,      598.45,      194.01],
       [      710.6,      167.73,      736.68,      182.35],
       [     758.52,      156.27,      782.52,      179.23]]),
'dimensions': array([[        3.2,        1.61,         1.66],
       [       3.66,        1.44,        1.61],
       [       3.37,        1.41,        1.53],
       [         -1,          -1,          -1],
       [         -1,          -1,          -1]]), 'location': array([[
0.7,        1.76,       23.88],
       [       0.24,        1.84,       66.37],
       [      -2.19,        1.96,       68.25],
       [      -1000,       -1000,       -1000],
       [      -1000,       -1000,       -1000]]), 'rotation_y':
array([      -1.48,        1.76,        1.75,          -10,          -
10]), 'score': array([          0,           0,           0,
0,           0]), 'index': array([ 0,  1,  2, -1, -1], dtype=int32),
'group_ids': array([0, 1, 2, 3, 4], dtype=int32), 'difficulty':
array([ 0, -1, -1, -1, -1], dtype=int32), 'num_points_in_gt':
array([215,   4,   1,  -1,  -1], dtype=int32)}}, {'image_idx': 10,
'pointcloud_num_features': 4, 'velodyne_path':
'training/velodyne/000010.bin', 'img_path':
'training/image_2/000010.png', 'img_shape': array([ 375, 1242],
dtype=int32), 'calib/P0': array([[     721.54,           0,
609.56,           0],
       [          0,      721.54,      172.85,           0],
       [          0,           0,           1,           0],
       [          0,           0,           0,           1]]),
'calib/P1': array([[     721.54,           0,      609.56,     -
387.57],
       [          0,      721.54,      172.85,           0],
       [          0,           0,           1,           0],
       [          0,           0,           0,           1]]),
'calib/P2': array([[     721.54,           0,      609.56,
44.857],
       [          0,      721.54,      172.85,     0.21638],
       [          0,           0,           1,   0.0027459],
       [          0,           0,           0,           1]]),
'calib/P3': array([[     721.54,           0,      609.56,     -
339.52],
```

```
       [          0,      721.54,      172.85,      2.1999],
       [          0,           0,           1,   0.0027299],
       [          0,           0,           0,           1]]),
'calib/R0_rect': array([[    0.99992,   0.0098378,   -0.007445,
0],
       [ -0.0098698,     0.99994,  -0.0042785,           0],
       [  0.0074025,   0.0043516,     0.99996,           0],
       [          0,           0,           0,           1]]),
'calib/Tr_velo_to_cam': array([[  0.0075337,    -0.99997,  -0.0006166,
-0.0040698],
       [   0.014802,  0.00072807,    -0.99989,   -0.076316],
       [    0.99986,   0.0075238,    0.014808,    -0.27178],
       [          0,           0,           0,           1]]),
'calib/Tr_imu_to_velo': array([[          1,  0.00075531,  -0.0020358,
-0.80868],
       [ -0.0007854,     0.99989,   -0.014823,     0.31956],
       [  0.0020244,    0.014825,     0.99989,    -0.79972],
       [          0,           0,           0,           1]]),
'annos': {'name': array(['Car', 'Car', 'Pedestrian', 'Car', 'Car',
'Car', 'Car', 'Car', 'Car', 'DontCare', 'DontCare', 'DontCare',
'DontCare'], dtype='<U10'), 'truncated': array([        0.8,
0,           0,           0,           0,           0,           0,
0,           0,          -1,          -1,          -1,          -1]),
'occluded': array([ 0,  0,  2,  0,  2,  0,  2,  1,  1, -1, -1, -1, -
1]), 'alpha': array([      -2.09,        1.95,        1.41,       -
1.78,       -1.69,         1.8,        1.77,       -1.67,        1.92,
-10,         -10,         -10,         -10]), 'bbox':
array([[     1013.4,      182.46,        1241,         374],
       [     354.43,      185.52,      549.52,      294.49],
       [     859.54,       159.8,      879.68,       221.4],
       [     819.63,      178.12,      926.85,      251.56],
       [     800.54,      178.06,      878.75,      230.56],
       [     558.55,      179.04,      635.05,      230.61],
       [      598.3,      178.68,      652.25,      218.17],
       [     784.59,      178.04,      839.98,       220.1],
       [     663.74,      175.36,      707.21,      204.15],
       [     737.69,      163.56,      790.86,      197.98],
       [      135.6,      185.44,      196.06,      202.15],
       [     796.02,      162.52,      862.73,       183.4],
       [     879.35,      165.65,      931.48,      182.36]]),
'dimensions': array([[       3.35,        1.57,        1.65],
       [       3.95,        1.43,         1.7],
       [       1.09,        1.96,        0.72],
       [       3.24,        1.51,         1.6],
       [        4.1,        1.45,        1.74],
       [       3.79,        1.54,        1.68],
       [       3.35,        1.49,        1.52],
       [       4.37,        1.53,        1.65],
       [       3.48,        1.64,        1.45],
```

```
       [         -1,          -1,          -1],
       [         -1,          -1,          -1],
       [         -1,          -1,          -1],
       [         -1,          -1,          -1]]), 'location': array([[
4.43,        1.65,         5.2],
       [      -2.39,        1.66,        11.8],
       [       8.33,        1.55,       23.51],
       [       5.85,        1.64,        16.5],
       [       6.87,        1.62,       22.05],
       [      -0.38,        1.76,       23.64],
       [       0.64,        1.74,       29.07],
       [       7.88,        1.75,       28.53],
       [        4.5,         1.8,       42.85],
       [      -1000,       -1000,       -1000],
       [      -1000,       -1000,       -1000],
       [      -1000,       -1000,       -1000],
       [      -1000,       -1000,       -1000]]), 'rotation_y':
array([      -1.42,        1.76,        1.75,       -1.44,       -
1.39,        1.78,        1.79,        -1.4,        2.02,         -10,
-10,         -10,         -10]), 'score': array([          0,
0,           0,           0,           0,           0,           0,
0,           0,           0,           0,           0,           0]),
'index': array([ 0,  1,  2,  3,  4,  5,  6,  7,  8, -1, -1, -1, -1],
dtype=int32), 'group_ids': array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,
9, 10, 11, 12], dtype=int32), 'difficulty': array([-1,  0,  2,  0,  2,
0,  2,  1,  1,  1, -1, -1, -1], dtype=int32), 'num_points_in_gt':
array([ 304, 1013,   23,  340,   48,  242,   50,   33,   20,   -1,   -
1,   -1,   -1], dtype=int32)}}]

Loaded: /kaggle/working/kitti/kitti_infos_val.pkl
Data Type: <class 'list'>
First 5 items: [{'image_idx': 1, 'pointcloud_num_features': 4,
'velodyne_path': 'training/velodyne/000001.bin', 'img_path':
'training/image_2/000001.png', 'img_shape': array([ 375, 1242],
dtype=int32), 'calib/P0': array([[     721.54,           0,
609.56,           0],
       [          0,      721.54,      172.85,           0],
       [          0,           0,           1,           0],
       [          0,           0,           0,           1]]),
'calib/P1': array([[     721.54,           0,      609.56,     -
387.57],
       [          0,      721.54,      172.85,           0],
       [          0,           0,           1,           0],
       [          0,           0,           0,           1]]),
'calib/P2': array([[     721.54,           0,      609.56,
44.857],
       [          0,      721.54,      172.85,     0.21638],
       [          0,           0,           1,   0.0027459],
       [          0,           0,           0,           1]]),
```

```
'calib/P3': array([[     721.54,           0,      609.56,     -
339.52],
       [          0,      721.54,      172.85,      2.1999],
       [          0,           0,           1,   0.0027299],
       [          0,           0,           0,           1]]),
'calib/R0_rect': array([[    0.99992,   0.0098378,   -0.007445,
0],
       [ -0.0098698,     0.99994,  -0.0042785,           0],
       [  0.0074025,   0.0043516,     0.99996,           0],
       [          0,           0,           0,           1]]),
'calib/Tr_velo_to_cam': array([[  0.0075337,    -0.99997,  -0.0006166,
-0.0040698],
       [   0.014802,  0.00072807,    -0.99989,   -0.076316],
       [    0.99986,   0.0075238,    0.014808,    -0.27178],
       [          0,           0,           0,           1]]),
'calib/Tr_imu_to_velo': array([[          1,  0.00075531,  -0.0020358,
-0.80868],
       [ -0.0007854,     0.99989,   -0.014823,     0.31956],
       [  0.0020244,    0.014825,     0.99989,    -0.79972],
       [          0,           0,           0,           1]]),
'annos': {'name': array(['Truck', 'Car', 'Cyclist', 'DontCare',
'DontCare', 'DontCare', 'DontCare'], dtype='<U8'), 'truncated':
array([          0,           0,           0,          -1,          -
1,          -1,          -1]), 'occluded': array([ 0,  0,  3, -1, -1,
-1, -1]), 'alpha': array([      -1.57,        1.85,       -1.65,
-10,         -10,         -10,         -10]), 'bbox':
array([[     599.41,       156.4,      629.75,      189.25],
       [     387.63,      181.54,      423.81,      203.12],
       [      676.6,      163.95,      688.98,      193.93],
       [     503.89,      169.71,      590.61,      190.13],
       [     511.35,      174.96,      527.81,      187.45],
       [     532.37,      176.35,      542.68,      185.27],
       [     559.62,      175.83,       575.4,      183.15]]),
'dimensions': array([[      12.34,        2.85,        2.63],
       [       3.69,        1.67,        1.87],
       [       2.02,        1.86,         0.6],
       [         -1,          -1,          -1],
       [         -1,          -1,          -1],
       [         -1,          -1,          -1],
       [         -1,          -1,          -1]]), 'location': array([[
0.47,        1.49,       69.44],
       [     -16.53,        2.39,       58.49],
       [       4.59,        1.32,       45.84],
       [      -1000,       -1000,       -1000],
       [      -1000,       -1000,       -1000],
       [      -1000,       -1000,       -1000],
       [      -1000,       -1000,       -1000]]), 'rotation_y':
array([      -1.56,        1.57,       -1.55,         -10,          -
10,         -10,         -10]), 'score': array([          0,
```

```
0,           0,           0,           0,           0,           0]),
'index': array([ 0,  1,  2, -1, -1, -1, -1], dtype=int32),
'group_ids': array([0, 1, 2, 3, 4, 5, 6], dtype=int32), 'difficulty':
array([ 1, -1, -1, -1, -1, -1, -1], dtype=int32), 'num_points_in_gt':
array([71,  9, 18, -1, -1, -1, -1], dtype=int32)}}, {'image_idx': 2,
'pointcloud_num_features': 4, 'velodyne_path':
'training/velodyne/000002.bin', 'img_path':
'training/image_2/000002.png', 'img_shape': array([ 375, 1242],
dtype=int32), 'calib/P0': array([[     721.54,           0,
609.56,           0],
       [          0,      721.54,      172.85,           0],
       [          0,           0,           1,           0],
       [          0,           0,           0,           1]]),
'calib/P1': array([[     721.54,           0,      609.56,     -
387.57],
       [          0,      721.54,      172.85,           0],
       [          0,           0,           1,           0],
       [          0,           0,           0,           1]]),
'calib/P2': array([[     721.54,           0,      609.56,
44.857],
       [          0,      721.54,      172.85,     0.21638],
       [          0,           0,           1,   0.0027459],
       [          0,           0,           0,           1]]),
'calib/P3': array([[     721.54,           0,      609.56,     -
339.52],
       [          0,      721.54,      172.85,      2.1999],
       [          0,           0,           1,   0.0027299],
       [          0,           0,           0,           1]]),
'calib/R0_rect': array([[    0.99992,   0.0098378,   -0.007445,
0],
       [ -0.0098698,     0.99994,  -0.0042785,           0],
       [  0.0074025,   0.0043516,     0.99996,           0],
       [          0,           0,           0,           1]]),
'calib/Tr_velo_to_cam': array([[  0.0075337,    -0.99997,  -0.0006166,
-0.0040698],
       [   0.014802,  0.00072807,    -0.99989,   -0.076316],
       [    0.99986,   0.0075238,    0.014808,    -0.27178],
       [          0,           0,           0,           1]]),
'calib/Tr_imu_to_velo': array([[          1,  0.00075531,  -0.0020358,
-0.80868],
       [ -0.0007854,     0.99989,   -0.014823,     0.31956],
       [  0.0020244,    0.014825,     0.99989,    -0.79972],
       [          0,           0,           0,           1]]),
'annos': {'name': array(['Misc', 'Car'], dtype='<U4'), 'truncated':
array([          0,           0]), 'occluded': array([0, 0]), 'alpha':
array([      -1.82,       -1.67]), 'bbox': array([[     804.79,
167.34,      995.43,      327.94],
       [     657.39,      190.13,      700.07,      223.39]]),
'dimensions': array([[       2.37,        1.63,        1.48],
```

```
       [       4.36,        1.41,        1.58]]), 'location': array([[
3.23,       1.59,       8.55],
       [       3.18,        2.27,       34.38]]), 'rotation_y':
array([      -1.47,       -1.58]), 'score': array([          0,
0]), 'index': array([0, 1], dtype=int32), 'group_ids': array([0, 1],
dtype=int32), 'difficulty': array([0, 1], dtype=int32),
'num_points_in_gt': array([1349,   67], dtype=int32)}}, {'image_idx':
4, 'pointcloud_num_features': 4, 'velodyne_path':
'training/velodyne/000004.bin', 'img_path':
'training/image_2/000004.png', 'img_shape': array([ 375, 1242],
dtype=int32), 'calib/P0': array([[     721.54,           0,
609.56,           0],
       [          0,      721.54,      172.85,           0],
       [          0,           0,           1,           0],
       [          0,           0,           0,           1]]),
'calib/P1': array([[     721.54,           0,      609.56,     -
387.57],
       [          0,      721.54,      172.85,           0],
       [          0,           0,           1,           0],
       [          0,           0,           0,           1]]),
'calib/P2': array([[     721.54,           0,      609.56,
44.857],
       [          0,      721.54,      172.85,     0.21638],
       [          0,           0,           1,   0.0027459],
       [          0,           0,           0,           1]]),
'calib/P3': array([[     721.54,           0,      609.56,     -
339.52],
       [          0,      721.54,      172.85,      2.1999],
       [          0,           0,           1,   0.0027299],
       [          0,           0,           0,           1]]),
'calib/R0_rect': array([[    0.99992,   0.0098378,   -0.007445,
0],
       [ -0.0098698,     0.99994,  -0.0042785,           0],
       [  0.0074025,   0.0043516,     0.99996,           0],
       [          0,           0,           0,           1]]),
'calib/Tr_velo_to_cam': array([[  0.0075337,    -0.99997,  -0.0006166,
-0.0040698],
       [   0.014802,  0.00072807,    -0.99989,   -0.076316],
       [    0.99986,   0.0075238,    0.014808,    -0.27178],
       [          0,           0,           0,           1]]),
'calib/Tr_imu_to_velo': array([[          1,  0.00075531,  -0.0020358,
-0.80868],
       [ -0.0007854,     0.99989,   -0.014823,     0.31956],
       [  0.0020244,    0.014825,     0.99989,    -0.79972],
       [          0,           0,           0,           1]]),
'annos': {'name': array(['Car', 'Car', 'DontCare', 'DontCare',
'DontCare', 'DontCare', 'DontCare'], dtype='<U8'), 'truncated':
array([          0,           0,          -1,          -1,          -
1,          -1,          -1]), 'occluded': array([ 0,  0, -1, -1, -1,
-1, -1]), 'alpha': array([       1.96,        1.88,         -10,
```

```
-10,         -10,         -10,         -10]), 'bbox':
array([[     280.38,       185.1,       344.9,      215.59],
       [     365.14,      184.54,      406.11,       205.2],
       [     402.27,      166.69,      477.31,      197.98],
       [     518.53,      177.31,      531.51,      187.17],
       [     1207.5,      233.35,        1240,      333.39],
       [     535.06,      177.65,      545.26,      185.82],
       [     558.03,      177.88,       567.5,      184.65]]),
'dimensions': array([[       4.01,        1.49,        1.76],
       [       3.41,        1.38,         1.8],
       [         -1,          -1,          -1],
       [         -1,          -1,          -1],
       [         -1,          -1,          -1],
       [         -1,          -1,          -1],
       [         -1,          -1,          -1]]), 'location': array([[
-15.71,        2.16,       38.26],
       [     -15.89,        2.23,       51.17],
       [      -1000,       -1000,       -1000],
       [      -1000,       -1000,       -1000],
       [      -1000,       -1000,       -1000],
       [      -1000,       -1000,       -1000],
       [      -1000,       -1000,       -1000]]), 'rotation_y':
array([       1.57,        1.58,         -10,         -10,         -
10,         -10,         -10]), 'score': array([          0,
0,           0,           0,           0,           0,           0]),
'index': array([ 0,  1, -1, -1, -1, -1, -1], dtype=int32),
'group_ids': array([0, 1, 2, 3, 4, 5, 6], dtype=int32), 'difficulty':
array([ 1, -1,  1, -1,  0, -1, -1], dtype=int32), 'num_points_in_gt':
array([78, 26, -1, -1, -1, -1, -1], dtype=int32)}}, {'image_idx': 5,
'pointcloud_num_features': 4, 'velodyne_path':
'training/velodyne/000005.bin', 'img_path':
'training/image_2/000005.png', 'img_shape': array([ 375, 1242],
dtype=int32), 'calib/P0': array([[     721.54,           0,
609.56,           0],
       [          0,      721.54,      172.85,           0],
       [          0,           0,           1,           0],
       [          0,           0,           0,           1]]),
'calib/P1': array([[     721.54,           0,      609.56,     -
387.57],
       [          0,      721.54,      172.85,           0],
       [          0,           0,           1,           0],
       [          0,           0,           0,           1]]),
'calib/P2': array([[     721.54,           0,      609.56,
44.857],
       [          0,      721.54,      172.85,     0.21638],
       [          0,           0,           1,   0.0027459],
       [          0,           0,           0,           1]]),
'calib/P3': array([[     721.54,           0,      609.56,     -
339.52],
```

```
       [          0,      721.54,      172.85,      2.1999],
       [          0,           0,           1,   0.0027299],
       [          0,           0,           0,           1]]),
'calib/R0_rect': array([[    0.99992,   0.0098378,   -0.007445,
0],
       [ -0.0098698,     0.99994,  -0.0042785,           0],
       [  0.0074025,   0.0043516,     0.99996,           0],
       [          0,           0,           0,           1]]),
'calib/Tr_velo_to_cam': array([[  0.0075337,    -0.99997,  -0.0006166,
-0.0040698],
       [   0.014802,  0.00072807,    -0.99989,   -0.076316],
       [    0.99986,   0.0075238,    0.014808,    -0.27178],
       [          0,           0,           0,           1]]),
'calib/Tr_imu_to_velo': array([[          1,  0.00075531,  -0.0020358,
-0.80868],
       [ -0.0007854,     0.99989,   -0.014823,     0.31956],
       [  0.0020244,    0.014825,     0.99989,    -0.79972],
       [          0,           0,           0,           1]]),
'annos': {'name': array(['Pedestrian', 'DontCare', 'DontCare',
'DontCare', 'DontCare'], dtype='<U10'), 'truncated':
array([          0,          -1,          -1,          -1,          -
1]), 'occluded': array([ 0, -1, -1, -1, -1]), 'alpha':
array([       1.94,         -10,         -10,         -10,         -
10]), 'bbox': array([[     330.06,      178.74,      360.77,
238.64],
       [     606.64,      170.67,      621.06,      184.34],
       [        606,      170.91,      621.35,      184.28],
       [     605.68,      171.21,      620.77,      184.34],
       [     566.39,      168.89,      585.07,      184.56]]),
'dimensions': array([[       0.65,        1.87,        0.96],
       [         -1,          -1,          -1],
       [         -1,          -1,          -1],
       [         -1,          -1,          -1],
       [         -1,          -1,          -1]]), 'location': array([[
-8.5,        2.07,       23.02],
       [      -1000,       -1000,       -1000],
       [      -1000,       -1000,       -1000],
       [      -1000,       -1000,       -1000],
       [      -1000,       -1000,       -1000]]), 'rotation_y':
array([       1.59,         -10,         -10,         -10,         -
10]), 'score': array([          0,           0,           0,
0,           0]), 'index': array([ 0, -1, -1, -1, -1], dtype=int32),
'group_ids': array([0, 1, 2, 3, 4], dtype=int32), 'difficulty':
array([ 0, -1, -1, -1, -1], dtype=int32), 'num_points_in_gt':
array([70, -1, -1, -1, -1], dtype=int32)}}, {'image_idx': 6,
'pointcloud_num_features': 4, 'velodyne_path':
'training/velodyne/000006.bin', 'img_path':
'training/image_2/000006.png', 'img_shape': array([ 374, 1238],
dtype=int32), 'calib/P0': array([[     718.34,           0,
```

```
600.39,           0],
       [          0,      718.34,      181.51,           0],
       [          0,           0,           1,           0],
       [          0,           0,           0,           1]]),
'calib/P1': array([[     718.34,           0,      600.39,     -
385.88],
       [          0,      718.34,      181.51,           0],
       [          0,           0,           1,           0],
       [          0,           0,           0,           1]]),
'calib/P2': array([[     718.34,           0,      600.39,
44.504],
       [          0,      718.34,      181.51,    -0.59511],
       [          0,           0,           1,   0.0026163],
       [          0,           0,           0,           1]]),
'calib/P3': array([[     718.34,           0,      600.39,     -
336.31],
       [          0,      718.34,      181.51,      3.1599],
       [          0,           0,           1,   0.0053238],
       [          0,           0,           0,           1]]),
'calib/R0_rect': array([[    0.99995,   0.0097917,  -0.0029253,
0],
       [ -0.0098069,     0.99994,  -0.0052387,           0],
       [  0.0028738,   0.0052671,     0.99998,           0],
       [          0,           0,           0,           1]]),
'calib/Tr_velo_to_cam': array([[  0.0077554,    -0.99997,  -0.0010143,
-0.0072755],
       [  0.0022941,   0.0010321,          -1,   -0.063241],
       [    0.99997,   0.0077531,    0.002302,    -0.26704],
       [          0,           0,           0,           1]]),
'calib/Tr_imu_to_velo': array([[          1,  0.00075531,  -0.0020358,
-0.80868],
       [ -0.0007854,     0.99989,   -0.014823,     0.31956],
       [  0.0020244,    0.014825,     0.99989,    -0.79972],
       [          0,           0,           0,           1]]),
'annos': {'name': array(['Car', 'Car', 'Car', 'Car', 'DontCare',
'DontCare'], dtype='<U8'), 'truncated': array([          0,
0,           0,           0,          -1,          -1]), 'occluded':
array([ 2,  0,  0,  1, -1, -1]), 'alpha': array([      -1.55,       -
1.21,        0.15,        2.05,         -10,         -10]), 'bbox':
array([[        548,      171.33,       572.4,      194.42],
       [     505.25,      168.37,      575.44,      209.18],
       [       49.7,      185.65,      227.42,      246.96],
       [     328.67,      170.65,      397.24,      204.16],
       [     603.36,      169.62,      631.06,      186.56],
       [     578.97,      168.88,      603.78,      187.56]]),
'dimensions': array([[       3.62,        1.48,        1.56],
       [       4.32,        1.67,        1.64],
       [       3.88,         1.5,        1.62],
       [       4.29,        1.68,        1.67],
```

```
       [         -1,          -1,          -1],
       [         -1,          -1,          -1]]), 'location': array([[
-2.72,        0.82,       48.22],
       [      -2.61,        1.13,       31.73],
       [     -12.54,        1.64,       19.72],
       [     -12.66,        1.13,       38.44],
       [      -1000,       -1000,       -1000],
       [      -1000,       -1000,       -1000]]), 'rotation_y':
array([      -1.62,        -1.3,       -0.42,        1.73,          -
10,         -10]), 'score': array([          0,           0,
0,           0,           0,           0]), 'index': array([ 0,  1,
2,  3, -1, -1], dtype=int32), 'group_ids': array([0, 1, 2, 3, 4, 5],
dtype=int32), 'difficulty': array([-1,  0,  0,  1, -1, -1],
dtype=int32), 'num_points_in_gt': array([  9,  64, 321,  26,  -1,  -
1], dtype=int32)}}]
YOLO labels and training_presets/dataset.yaml created successfully!
```

2.2 Model Training

- **Trained YOLOv8 models:** Models trained on respective datasets for domain-specific improvements.

Training Parameters:

- Data augmentation techniques applied.
- Trained for 50 epochs at an image resolution of 640x640.

```
import os
from ultralytics import YOLO

# Initialize YOLO model (pre-trained on COCO)
yolo_model = YOLO("yolov8n.pt")
# You can change to "yolov8s.pt", "yolov8m.pt" etc.
# Train YOLOv8 on your dataset
yolo_model.train(
    data="/kaggle/working/kitti/dataset.yaml",  # Correct dataset path
    epochs=50,
    imgsz=640,
    batch=16,  # Adjust based on your available memory
    workers=4,
    device="cuda"  # Use GPU for faster training
)

print("Training completed.")

# Save the trained model
trained_model_path = "runs/detect/train/weights/best.pt"  # Update if
needed
save_path = "/kaggle/working/yolov8_trained.pt"

# Ensure the trained model exists before copying
```

```
if os.path.exists(trained_model_path):
    shutil.copy(trained_model_path, save_path)
    print(f"Model saved at: {save_path}")
else:
    print("Trained model file not found. Check the training
directory.")
```

```
Ultralytics 8.3.78 🚀 Python-3.11.11 torch-2.5.1+cu124

---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
<ipython-input-20-27819f037235> in <cell line: 0>()
      6 # You can change to "yolov8s.pt", "yolov8m.pt" etc.
      7 # Train YOLOv8 on your dataset
----> 8 yolo_model.train(
      9     data="/kaggle/working/kitti/dataset.yaml",  # Correct
dataset path
     10     epochs=50,

/usr/local/lib/python3.11/dist-packages/ultralytics/engine/model.py in
train(self, trainer, **kwargs)
    802             args["resume"] = self.ckpt_path
    803
--> 804         self.trainer = (trainer or
self._smart_load("trainer"))(overrides=args,
_callbacks=self.callbacks)
    805         if not args.get("resume"):  # manually set model only
if not resuming
    806             self.trainer.model =
self.trainer.get_model(weights=self.model if self.ckpt else None,
cfg=self.model.yaml)

/usr/local/lib/python3.11/dist-packages/ultralytics/engine/trainer.py
in __init__(self, cfg, overrides, _callbacks)
    102         self.args = get_cfg(cfg, overrides)
    103         self.check_resume(overrides)
--> 104         self.device = select_device(self.args.device,
self.args.batch)
    105         self.validator = None
    106         self.metrics = None

/usr/local/lib/python3.11/dist-packages/ultralytics/utils/torch_utils.
py in select_device(device, batch, newline, verbose)
    190                 else ""
    191             )
--> 192             raise ValueError(
    193                 f"Invalid CUDA 'device={device}' requested."
    194                 f" Use 'device=cpu' or pass valid CUDA
```

```
device(s) if available,"

ValueError: Invalid CUDA 'device=0' requested. Use 'device=cpu' or
pass valid CUDA device(s) if available, i.e. 'device=0' or
'device=0,1,2,3' for Multi-GPU.

torch.cuda.is_available(): False
torch.cuda.device_count(): 0
os.environ['CUDA_VISIBLE_DEVICES']: 0
See https://pytorch.org/get-started/locally/ for up-to-date torch
install instructions if no CUDA devices are seen by torch.
```

```python
import shutil

# Path to the best trained model
trained_model_path = "runs/detect/train/weights/best.pt"  # Update
based on the actual path

# Define where to save the trained model
save_path = "/kaggle/working/yolov8_trained.pt"

# Copy the best model to the desired location
shutil.copy(trained_model_path, save_path)

print(f"Model saved at: {save_path}")
```

```
---------------------------------------------------------------------
-----
FileNotFoundError                         Traceback (most recent call
last)
<ipython-input-40-e71da7cf70aa> in <cell line: 0>()
      8
      9 # Copy the best model to the desired location
---> 10 shutil.copy(trained_model_path, save_path)
     11
     12 print(f"Model saved at: {save_path}")

/usr/lib/python3.11/shutil.py in copy(src, dst, follow_symlinks)
    429     if os.path.isdir(dst):
    430         dst = os.path.join(dst, os.path.basename(src))
--> 431     copyfile(src, dst, follow_symlinks=follow_symlinks)
    432     copymode(src, dst, follow_symlinks=follow_symlinks)
    433     return dst

/usr/lib/python3.11/shutil.py in copyfile(src, dst, follow_symlinks)
    254         os.symlink(os.readlink(src), dst)
    255     else:
--> 256         with open(src, 'rb') as fsrc:
    257             try:
    258                 with open(dst, 'wb') as fdst:
```

```
FileNotFoundError: [Errno 2] No such file or directory:
'runs/detect/train/weights/best.pt'
```