Lab 7: Implementing Object Detection with YOLO for Telecom Infrastructure

Objective This lab focuses on implementing object detection for telecom infrastructure using YOLOv8. Students will learn to train and deploy a YOLO model for detecting different types of antennas (GSM and Microwave) in real-world scenarios. The lab demonstrates practical applications of object detection in the telecommunications industry, covering model training, evaluation, and inference on new images.

Dataset Information

Source: Tower Detection Dataset

Classes: 3 types of antennas

- GSM Antenna
- GSM Antenna (variant)
- Microwave Antenna

Dataset Split

- Training: 1350 images (99%)
- Validation: 7 images (1%)
- Testing: 5 images

Tasks Overview

1. Environment Setup and Dependencies Installation
2. Dataset Download and Preparation
3. YOLOv8 Model Configuration
4. Model Training
5. Performance Evaluation
6. Inference on Test Images
7. Model Export

Task 1: Environment Setup and Dependencies Installation

```
# Install required packages
!pip install ultralytics
!pip install roboflow
```

```
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.1.4->ultralytics) (2024.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (2.2
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.23.0->ultralytics) (202
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (3.16.1)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (4
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (2024.10.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/dist-packages (from torch>=1.8.0->ultralytics) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy==1.13.1->torch>=1.8.0->ultral
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=3.3.0->ultr
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.8.0->ultralytics) (3
Downloading ultralytics-8.3.51-py3-none-any.whl (901 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 901.3/901.3 kB 20.1 MB/s eta 0:00:00
Downloading ultralytics_thop-2.0.13-py3-none-any.whl (26 kB)
Installing collected packages: ultralytics-thop, ultralytics
Successfully installed ultralytics-8.3.51 ultralytics-thop-2.0.13
Collecting roboflow
  Downloading roboflow-1.1.50-py3-none-any.whl.metadata (9.7 kB)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from roboflow) (2024.12.14)
Collecting idna==3.7 (from roboflow)
  Downloading idna-3.7-py3-none-any.whl.metadata (9.9 kB)
Requirement already satisfied: cycler in /usr/local/lib/python3.10/dist-packages (from roboflow) (0.12.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.4.7)
```

```
Installing collected packages: filetype, python-dotenv, idna, roboflow
  Attempting uninstall: idna
    Found existing installation: idna 3.10
    Uninstalling idna-3.10:
      Successfully uninstalled idna-3.10
```

```
# Import necessary libraries
import ultralytics
from ultralytics import YOLO
from roboflow import Roboflow
import os
import cv2
import numpy as np
from google.colab import drive
```

```
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/qui
```

Task 2: Dataset Download and Preparation

```
# Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Set working directory
import os
%cd /content
```

```
Mounted at /content/drive
/content
```

Setup Data path

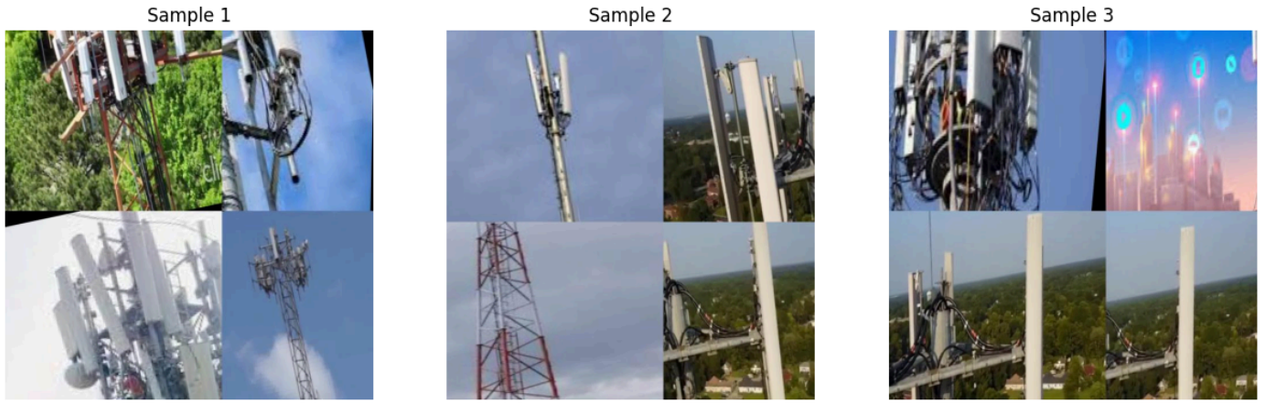
```
# Define dataset path (update this to your Google Drive path)
DATASET_PATH = '/content/drive/MyDrive/tower_dataset'  # Update this path

# Verify dataset structure
!ls {DATASET_PATH}

# Display data.yaml content
!cat {DATASET_PATH}/data.yaml

# Setup directory structure
train_path = os.path.join(DATASET_PATH, 'train/images')
valid_path = os.path.join(DATASET_PATH, 'valid/images')
test_path = os.path.join(DATASET_PATH, 'test/images')

# Verify image counts
print(f"Training images: {len(os.listdir(train_path))}")
print(f"Validation images: {len(os.listdir(valid_path))}")
print(f"Test images: {len(os.listdir(test_path))}")
```

```
best.pt  data.yaml  test  train  valid
train: ../train/images
val: ../valid/images
test: ../test/images

nc: 3
names: ['GSM Antenna', 'GSM Antenna ', 'Microwave Antenna']

roboflow:
  workspace: object-detection-yolo-c8gsd
  project: tower-detection-tff1p
  version: 3
  license: CC BY 4.0
  url: https://universe.roboflow.com/object-detection-yolo-c8gsd/tower-detection-tff1p/dataset/3Training images: 1350
Validation images: 7
Test images: 5
```

Data Visualization

```
import matplotlib.pyplot as plt
import random

def display_sample_images(image_path, num_samples=3):
    """Display random sample images from the dataset"""
    images = os.listdir(image_path)
    samples = random.sample(images, num_samples)

    plt.figure(figsize=(15, 5))
    for i, img_name in enumerate(samples, 1):
        img = cv2.imread(os.path.join(image_path, img_name))
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        plt.subplot(1, num_samples, i)
        plt.imshow(img)
        plt.title(f'Sample {i}')
        plt.axis('off')
    plt.show()

# Display sample training images
print("Sample Training Images:")
display_sample_images(train_path)
```

```
Sample Training Images:
```



Task 3: YOLOv8 Model Configuration

```
import torch

# Check CUDA availability
print(f"CUDA is available: {torch.cuda.is_available()}")
if torch.cuda.is_available():
    print(f"CUDA Device: {torch.cuda.get_device_name(0)}")
    print(f"Number of CUDA devices: {torch.cuda.device_count()}")
    print(f"CUDA Version: {torch.version.cuda}")
```

```
CUDA is available: True
CUDA Device: Tesla T4
Number of CUDA devices: 1
CUDA Version: 12.1
```

```
# Load a pre-trained YOLOv8 model
model = YOLO('yolov8n.pt')

# Move model to GPU if available
device = 'cuda' if torch.cuda.is_available() else 'cpu'
print(f"\nUsing device: {device}")

# Display model information
print("\nModel Information:")
print(model.info())

# Display model parameters
print(f"\nTotal Parameters: {sum(p.numel() for p in model.parameters())}")
```

```
Downloading https://github.com/ultralytics/assets/releases/download/v8.3.0/yolov8n.pt to 'yolov8n.pt'...
100%|██████████| 6.25M/6.25M [00:00<00:00, 362MB/s]

Using device: cuda

Model Information:
YOLOv8n summary: 225 layers, 3,157,200 parameters, 0 gradients, 8.9 GFLOPs
(225, 3157200, 0, 8.8575488)

Total Parameters: 3157200
```

Task 4: Model Training

Configure Training Parameters

```
# Define training configuration
train_config = {
    'data': os.path.join(DATASET_PATH, 'data.yaml'),
    'epochs': 50,
    'imgsz': 640,
    'batch': 32,
    'name': 'tower_detection_model',
    'patience': 20,         # Early stopping patience
    'save': True,          # Save best model
    'device': 0 if torch.cuda.is_available() else 'cpu',  # Use GPU if available
    'workers': 8,          # Number of worker threads
    'optimizer': 'Adam',   # Optimizer (SGD, Adam, AdamW)
    'lr0': 0.01,          # Initial learning rate
    'weight_decay': 0.0005,# Weight decay
    'exist_ok': True,      # Overwrite existing experiment
    'pretrained': True,    # Use pretrained backbone
    'amp': True,          # Automatic Mixed Precision
}

# Adjust batch size based on available GPU memory
if torch.cuda.is_available():
    gpu_mem = torch.cuda.get_device_properties(0).total_memory / 1e9  # Memory in GB
    print(f"\nGPU Memory Available: {gpu_mem:.2f} GB")

    # Adjust batch size based on GPU memory
    if gpu_mem < 8:
```

```
        train_config['batch'] = 8
    elif gpu_mem < 16:
        train_config['batch'] = 16
    else:
        train_config['batch'] = 32

print("\nTraining Configuration:")
for key, value in train_config.items():
    print(f"{key}: {value}")
```

```
GPU Memory Available: 15.84 GB

Training Configuration:
data: /content/drive/MyDrive/tower_dataset/data.yaml
epochs: 50
imgsz: 640
batch: 16
name: tower_detection_model
patience: 20
save: True
device: 0
workers: 8
optimizer: Adam
lr0: 0.01
weight_decay: 0.0005
exist_ok: True
pretrained: True
amp: True
```

Start Training

```
# Start training with configured parameters
results = model.train(**train_config)

# Save training results
model.save(os.path.join(DATASET_PATH, 'best.pt'))
```

```
      42/50      2.56G      1.679      1.476      1.648         76        640: 100%|██████████| 85/85 [00:29<00:00,  2.87it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,  3.34it/s

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      43/50      2.54G       1.68      1.462      1.651         59        640: 100%|██████████| 85/85 [00:32<00:00,  2.58it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,  6.04it/s

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      44/50      2.59G      1.672      1.448       1.65         69        640: 100%|██████████| 85/85 [00:33<00:00,  2.54it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,  6.46it/s
                   all          7         21      0.814      0.525      0.663      0.354

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      45/50      2.58G      1.672      1.441      1.644         59        640: 100%|██████████| 85/85 [00:32<00:00,  2.59it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,  3.62it/s

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      46/50      2.59G      1.638      1.406      1.616         71        640: 100%|██████████| 85/85 [00:30<00:00,  2.81it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,  6.04it/s
                   all          7         21      0.814       0.58      0.723      0.374

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      47/50      2.59G      1.634      1.409      1.617         54        640: 100%|██████████| 85/85 [00:30<00:00,  2.76it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,  4.86it/s

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      48/50      2.58G       1.61      1.388      1.606         54        640: 100%|██████████| 85/85 [00:33<00:00,  2.55it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,  5.73it/s

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      49/50      2.61G      1.605       1.37      1.593         77        640: 100%|██████████| 85/85 [00:36<00:00,  2.30it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,  4.52it/s

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      50/50      2.57G      1.583      1.365      1.594         97        640: 100%|██████████| 85/85 [00:30<00:00,  2.78it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,  2.89it/s


50 epochs completed in 0.503 hours.
Optimizer stripped from runs/detect/tower_detection_model/weights/last.pt, 6.2MB
Optimizer stripped from runs/detect/tower_detection_model/weights/best.pt, 6.2MB

Validating runs/detect/tower_detection_model/weights/best.pt...
Ultralytics 8.3.51 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 3,006,233 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,  8.02it/s
                   all          7         21      0.749      0.656       0.75      0.402
           GSM Antenna          6         16      0.721      0.312      0.505       0.18
     Microwave Antenna          4          5      0.778          1      0.995      0.624
Speed: 0.2ms preprocess, 3.5ms inference, 0.0ms loss, 4.2ms postprocess per image
Results saved to runs/detect/tower_detection_model
```

Task 5: Performance Evaluation

```
# Plot training metrics
from ultralytics.utils.plotting import plot_results
```

```
# Plot the results using the static method
plot_results(file='/content/runs/detect/tower_detection_model/results.png')  # plot results.txt as results.png
plt.show()

# Display final metrics
print("\nTraining Results:")

# Load and display metrics from results.csv
import pandas as pd
try:
    results_df = pd.read_csv('/content/runs/detect/tower_detection_model/results.csv')
    print("\nFinal Metrics:")
    print(f"Best mAP50: {results_df['metrics/mAP50(B)'].max():.4f}")
    print(f"Best mAP50-95: {results_df['metrics/mAP50-95(B)'].max():.4f}")
    print(f"Final Epoch: {len(results_df)}")
except Exception as e:
    print(f"Could not load results.csv: {e}")
```

```
Training Results:

Final Metrics:
Best mAP50: 0.7504
Best mAP50-95: 0.4021
Final Epoch: 50
```

Performance Evaluation

```
# Validate the model
metrics = model.val()

# Display metrics
print(f"mAP50: {metrics.box.map50}")
print(f"mAP50-95: {metrics.box.map}")
```

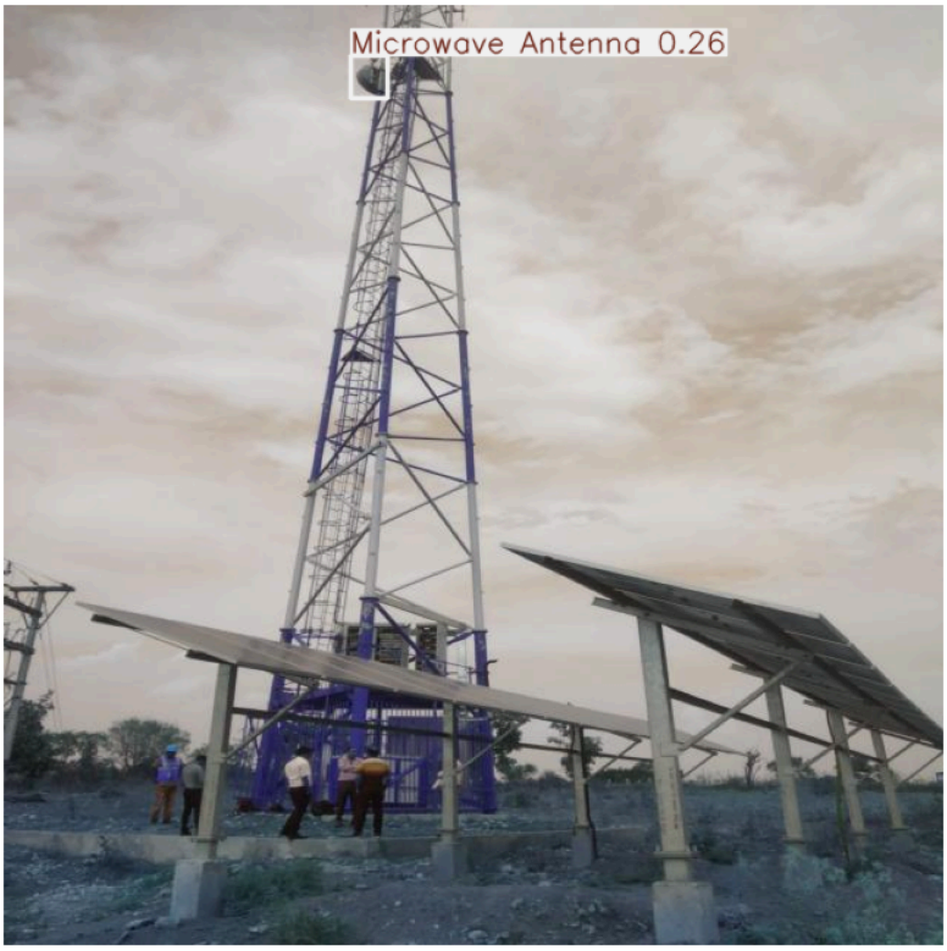
```
Ultralytics 8.3.51 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 3,006,233 parameters, 0 gradients, 8.1 GFLOPs
val: Scanning /content/drive/MyDrive/tower_dataset/valid/labels.cache... 7 images, 0 backgrounds, 0 corrupt: 100%|██████████| 7/7 [00:00
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 1/1 [00:00<00:00,  2.90it/s]
                   all          7         21      0.644      0.656        0.7      0.365
           GSM Antenna          6         16      0.658      0.312      0.505      0.179
     Microwave Antenna          4          5      0.631          1      0.895      0.552
Speed: 0.3ms preprocess, 16.9ms inference, 0.0ms loss, 2.0ms postprocess per image
Results saved to runs/detect/tower_detection_model
mAP50: 0.699988939680277
mAP50-95: 0.36516746670420464
```

Task 6: Inference on Test Images

```
def predict_image(image_path):
    # Perform prediction
    results = model.predict(image_path, conf=0.25)

    # Process and display results
    for result in results:
        boxes = result.boxes
        for box in boxes:
            # Get coordinates and class
            x1, y1, x2, y2 = box.xyxy[0]
            cls = box.cls
            conf = box.conf

            # print(f"Detected {model.names[int(cls)]} with confidence {conf:.2f}")

    return results[0].plot()

# Test on sample image
test_image = "/content/drive/MyDrive/tower_dataset/test/images/MP_3_jpeg.rf.7bded774087bd0b184d65eb93ce0eb04.jpg"
prediction = predict_image(test_image)
```

```
image 1/1 /content/drive/MyDrive/tower_dataset/test/images/MP_3_jpeg.rf.7bded774087bd0b184d65eb93ce0eb04.jpg: 640x640 1 Microwave Antenn
Speed: 3.9ms preprocess, 12.7ms inference, 1.7ms postprocess per image at shape (1, 3, 640, 640)
```

```
#Plot the show image of prediction
plt.figure(figsize=(10, 10))
plt.imshow(prediction)
plt.axis('off')
plt.show()
```



Task 7: Model Export

```
# Export model to ONNX format
model.export(format='onnx')
```