Lab Assignment No.3

Name :- Om Santosh Bhutkar

PRN :- 202201040111

Batch :- A-2

Deep Learning

Lab Assignment NO.03: Object Detection and Multi Object Classification (4 HOURS)-YOLO11 Model

Analyze, design, and optimize an object detection and multi-object classification model by integrating detection, segmentation, and recognition tasks. Evaluate model performance on real-world data and explore potential applications.

Github Repository Link :- https://github.com/OmBhutkar/Object_Detection_and_Multi_Object_Classification_using_YOLO11_Model.

Google Colab Link :-

https://colab.research.google.com/drive/1pdvJbhBBvL658PrbAcU-AnsSYV7i5Xeu?usp=sharing#scrollTo=iZKMHGmWs_DP

Google Colab Lab Assignment -YOLO 11 Model

Course Name: PEC Deep Learning

Lab Title: YOLOv11 for object detection on the COCO dataset

Student Name: Om Bhutkar

Student ID: 202201040111

Date of Submission: 17/03/2025

Group Members:

1. Bhavesh Bagul
2. Yash Mali

Objective The purpose of this lab is to understand and implement YOLOv11 for real-time object detection. Students will perform dataset preparation, model implementation, inference, and performance evaluation.

Task 1: Environment Setup and YOLOv11 Installation

Objective:
Set up the required libraries and dependencies to run YOLOv11.

Steps:

1. **Install Python Libraries:**
   Install required libraries using pip: `roboflow` and `ultralytics` (which includes PyTorch, OpenCV, etc.).

```
# Install roboflow and ultralytics
!pip install roboflow
!pip install ultralytics
```

Show hidden output

Task 2: Dataset Preparation & Preprocessing

Objective:

Load and preprocess a dataset for object detection.

Steps:

1. **Dataset Acquisition:**
    - Use Roboflow to download the COCO dataset (version 34) in YOLOv11 format.
    - Utilize your API key and select the Microsoft workspace.

```
from roboflow import Roboflow

# Initialize Roboflow with your API key
rf = Roboflow(api_key="sLpQp9tNRxVlPd1zmIqo")

# Load COCO dataset (version 34) from Microsoft workspace
project = rf.workspace("microsoft").project("coco")
version = project.version(34)
dataset = version.download("yolov11")
```

```
loading Roboflow workspace...
loading Roboflow project...
```

2. **Dataset Structure and Preprocessing:**
    - Verify that the dataset has been downloaded with the expected directory structure (`train/`, `valid/`, and `test/` folders containing images and labels).
    - Confirm that annotations are in the correct YOLO format.

```
import os

# List files to confirm dataset download
!ls -R /content/COCO-Dataset-34
```

```
000000009400_jpg.rf.1926c92326b48f77b8b081169668c44e.txt
000000009845_jpg.rf.e229e70f09239f18efbf68ae064b7247.txt
000000011122_jpg.rf.bfc38911a4f880cee0cffa41f275837a.txt
000000012933_jpg.rf.e98d33ed492106c2d80d711bef83d5ce.txt
000000013524_jpg.rf.62e7b283211f3fd55bcf1e1364cc3812.txt
000000014044_jpg.rf.705d3b529479a1a99ddc4bbc78f156ae.txt
000000015002_jpg.rf.81315262e3826c263eeb98454c9915ea.txt
000000015690_jpg.rf.3adbdfb64b57e251c0339af16d13a075.txt
000000017778_jpg.rf.dea0f3c966f432e6e0b2198ddb78640f.txt
000000018290_jpg.rf.0beefa821825a6188aaa43dc0bccb94c.txt
000000018614_jpg.rf.3bb733baf94efba3e208c14748f687ef.txt
000000018728_jpg.rf.5be59d76b01d5c9ed7919c7919ada9c8.txt
000000020291_jpg.rf.86880a9523e87511712bc976ff7eade7.txt
000000021248_jpg.rf.768f85b6c4bf2f060d08d1d5bf676a48.txt
000000021353_jpg.rf.ac4c8d046e14d5baca46987ce66f3756.txt
000000022199_jpg.rf.336aff9c0ebafa13dfbae4efbebe9763.txt
000000022229_jpg.rf.63c40b56ea6ca8600bd0d301d7143a25.txt
000000022526_jpg.rf.8f54e5a73f964df75a7b06d772fa2a50.txt
000000024023_jpg.rf.3a5cda5ea8eedddcbd1d90999ccf2321.txt
000000024980_jpg.rf.5beded38714f45bc2f04d51417b552d0.txt
000000026310_jpg.rf.e333707ea808eae6bc759e7a45e32bd9.txt
000000027246_jpg.rf.2470a0fe8d3deaa9647327a4601ba80b.txt
000000029482_jpg.rf.f37a043f6006625b4a189a2d2196da8d.txt
000000029715_jpg.rf.1539158c462c1ec6a3494478c803cd64.txt
000000030519_jpg.rf.469e348276c997d2c67f1d0e16286e09.txt
000000031373_jpg.rf.7092d7fbf231700030682dd72b7b1ab0.txt
000000032720_jpg.rf.13aa2ce4375761c6a534e259ef419695.txt
000000032990_jpg.rf.bdc9882221cf6630d933b3b48cd7d511.txt
000000034489_jpg.rf.656d672eff71374a5577fd086f4ba724.txt
000000034702_jpg.rf.d91a68a75b16b3ae997e72b5f1411d68.txt
000000035318_jpg.rf.583328d907b37e5b4c297b0b9d911baa.txt
000000035351_jpg.rf.dada74c3812da496ac2cd96746f7bec1.txt
000000037437_jpg.rf.c5d2023789cc50088a4402e52c1e0422.txt
000000039468_jpg.rf.ae709a23f600bbd9e6b23defce534bdb.txt
000000039993_jpg.rf.832d479e79fd38925415503344b6b9e1.txt
000000040658_jpg.rf.ce3b384940c0dd675926320a52d2c336.txt
000000043270_jpg.rf.9e594cfd7829a43be8d233bf6279c3ce.txt
000000043813_jpg.rf.cc5e5901986576a4746d9b3edb2079bb.txt
000000044946_jpg.rf.af6c86f6999b30246bdced6d684a50ce.txt
000000045148_jpg.rf.e551e6c88648e955043cba5143a3d31a.txt
000000047619_jpg.rf.4ba17653fd252aea9d043ebdbea40f29.txt
000000049135_jpg.rf.cd7b7ef54ac9a9cf445bb06753e53966.txt
000000050727_jpg.rf.924ab9fa11edc6d0a091e9747b51cf7d.txt
000000126137_jpg.rf.8a875933888aa097e28a4beed3773aa6.txt
```

Outcome:

- A well-organized dataset (COCO in YOLOv11 format) ready for training.

Task 3: Training YOLOv11 Model

Objective:

Train YOLOv11 on the prepared dataset.

Steps:

1. **Model Initialization:**
   - Load the YOLOv11 model using the pre-trained weights file (e.g., `yolo11n.pt`).

```
from ultralytics import YOLO

# Load YOLOv11 model with pretrained weights
model = YOLO('yolo11n.pt')  # Load YOLOv11 pretrained model

# training parameters
batch_size = 16
epochs = 50
learning_rate = 0.001
```

2. **Set Training Parameters:**
   - Configure key parameters such as `epochs`, `batch` size, and `lr0` (initial learning rate).
3. **Monitoring Training:**
   - Watch for improvements in loss, mAP, and other metrics as the training progresses.
   - Save the best model weights for further inference.

```
results = model.train(
    data='/content/COCO-Dataset-34/data.yaml',  # Path to data.yaml
    epochs=50,                   # Number of epochs
    batch=16,                    # Batch size
    lr0=0.001,                   # Learning rate
    imgsz=640                    # Input size
)
```

```
                 clock          2          2       0.18        0.5      0.126
                 couch          1          1          1          0          0
                   cup          4         10      0.708        0.3      0.372      0.
          dining table          6          8          1          0      0.114     0.0
                   dog          2          4          0          0     0.0248     0.0
                 donut          2         12          1          0          0
              elephant          2         13      0.309      0.154      0.226      0.
          fire hydrant          3          3      0.888      0.333      0.361      0.
                  fork          1          1          1          0          0
               frisbee          1          1          1          0          0
               handbag          5         11          1          0          0
                 horse          1          1      0.329          1      0.995      0.
              keyboard          2          7          1          0    0.00807    0.00
                  kite          1          3          1          0          0
                 knife          2          3          1          0          0
                laptop          4         10          1          0      0.224      0.
             microwave          1          4          1          0          0
            motorcycle          3          4          0          0      0.113      0.
                 mouse          3          4          1          0          0
                  oven          1          1          1          0          0
                person         35        157        0.5      0.516      0.524      0.
                 pizza          1          1          0          0      0.199      0.
          potted plant          1          1          1          0          0
          refrigerator          2          3          0          0          0
                remote          2          2          1          0     0.0229      0.
                  sink          1          1     0.0951      0.476      0.199      0.
            skateboard          2          8          0          0       0.01    0.00
             snowboard          2          2          1          0     0.0286    0.00
           sports ball          2          2          1          0          0
             stop sign          2          2          1          0     0.0362     0.0
              suitcase          1          1          0          0      0.014    0.00
             surfboard          2          3          1          0          0
         tennis racket          3          3          1          0          0
                   tie          2          2          1          0          0
                toilet          1          1          0          0          0
         traffic light          1          1          1          0          0
                 train          3          6          1          0     0.0786     0.0
                 truck          5          8     0.0987     0.0863      0.038     0.0
                    tv          4          8          1          0     0.0499     0.0
              umbrella          5          5          1          0          0
                  vase          1          1      0.752          1      0.995      0.
            wine glass          1          3          0          0     0.0383     0.0
Speed: 0.2ms preprocess, 2.3ms inference, 0.0ms loss, 3.1ms postprocess per image
Results saved to runs/detect/train3
```

Outcome:

- A successfully trained YOLOv11 model with improved detection accuracy and better performance metrics.

Task 4: Model Inference and Evaluation

Objective:
Test the trained model on new images and videos and evaluate its performance.

Steps:

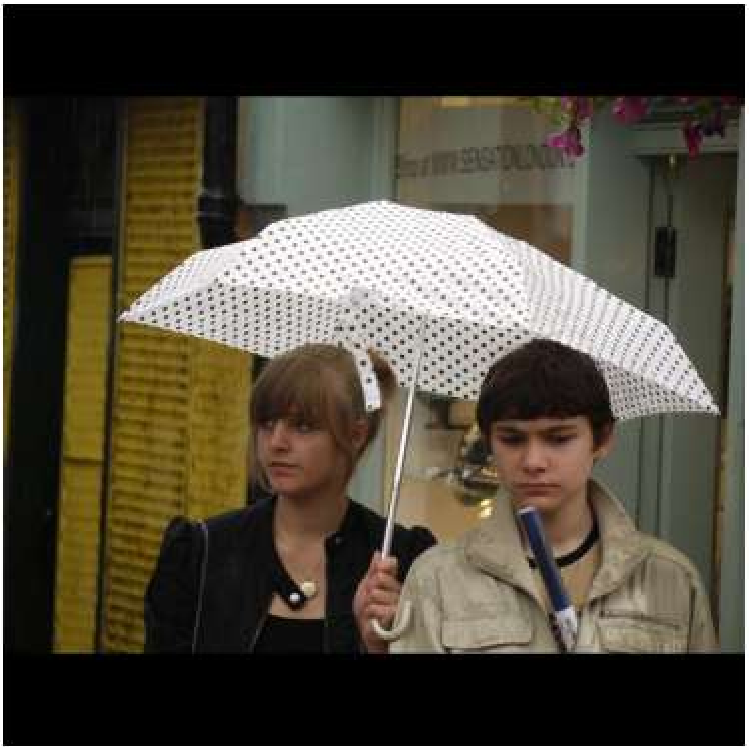
1. **Load Trained Model:**
   - Load the best-performing model weights saved during training.

```
from ultralytics import YOLO

# Load the trained model weights
model = YOLO('/content/runs/detect/train/weights/best.pt')
```

2. **Run Inference:**
   - Choose a test image from the dataset and run the model's prediction.

```
import cv2
from matplotlib import pyplot as plt
import os

# Path to test images
test_image_path = '/content/COCO-Dataset-34/test/images/'

# List test images
test_images = os.listdir(test_image_path)

# Run inference on the first test image
img_path = os.path.join(test_image_path, test_images[3])

# Perform inference
results = model.predict(img_path, save=True)

# Display result
img = cv2.imread(img_path)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show()
```

```
image 1/1 /content/COCO-Dataset-34/test/images/000000025668_jpg.rf.f67afc05b355ac25bef19
Speed: 2.6ms preprocess, 19.0ms inference, 5.7ms postprocess per image at shape (1, 3, 6
Results saved to runs/detect/predict3
```



3. **Evaluate Model Performance:**
   - Compute and display key metrics such as mAP@50, mAP@50-95, Precision, Recall, and F1-Score.

```
# Evaluate model performance on the validation set
metrics = model.val()
```

```
Ultralytics 8.3.94 🚀 Python-3.11.11 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)
val: Scanning /content/COCO-Dataset-34/valid/labels.cache... 55 images, 0 backgrounds
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-
                   all         55        397      0.649      0.111      0.105      0.0
              backpack          1          1          0          0          0
                banana          1          2          1          0          0
          baseball bat          1          1          1          0          0
        baseball glove          1          4          1          0          0
                 bench          3          4          0          0          0
               bicycle          3          3          0          0          0
                  bird          2          4          1          0          0
                  boat          1          1          1          0     0.0622      0.0
                bottle          4          7      0.113      0.429     0.0654      0.0
                   bus          3          3      0.184      0.245      0.179       0.
                  cake          1          8          1          0          0
                   car          5         13       0.15      0.231       0.18       0.
                   cat          2          2      0.441          1      0.663       0.
```

```
            cell phone          4          4          1          0          0
                 chair          6         16      0.253     0.0625     0.0614     0.0
                 clock          2          2       0.18        0.5      0.126
                 couch          1          1          1          0          0
                   cup          4         10      0.707        0.3      0.373      0.
          dining table          6          8          1          0      0.105     0.0
                   dog          2          4          0          0     0.0248     0.0
                 donut          2         12          1          0          0
              elephant          2         13       0.31      0.154      0.226      0.
          fire hydrant          3          3      0.889      0.333      0.362      0.
                  fork          1          1          1          0          0
               frisbee          1          1          1          0          0
               handbag          5         11          1          0          0
                 horse          1          1       0.33          1      0.995      0.
              keyboard          2          7          1          0    0.00821    0.00
                  kite          1          3          1          0          0
                 knife          2          3          1          0          0
                laptop          4         10          1          0      0.224      0.
             microwave          1          4          1          0          0
            motorcycle          3          4          0          0      0.113      0.
                 mouse          3          4          1          0          0
                  oven          1          1          1          0          0
                person         35        157      0.498       0.51      0.516      0.
                 pizza          1          1          0          0      0.199      0.
          potted plant          1          1          1          0          0
          refrigerator          2          3          0          0          0
                remote          2          2          1          0     0.0229      0.
                  sink          1          1     0.0923      0.462      0.199      0.
            skateboard          2          8          0          0       0.01    0.00
             snowboard          2          2          1          0     0.0286    0.00
           sports ball          2          2          1          0          0
             stop sign          2          2          1          0     0.0355     0.0
              suitcase          1          1          0          0      0.014    0.00
             surfboard          2          3          1          0          0
         tennis racket          3          3          1          0          0
                   tie          2          2          1          0          0
                toilet          1          1          0          0          0
         traffic light          1          1          1          0          0
                 train          3          6          1          0     0.0786     0.0
                 truck          5          8      0.094     0.0823     0.0378     0.0
```

```
# Display key metrics
print(f"mAP@50: {metrics.box.map50:.4f}")     # Mean Average Precision at IoU 0.5
print(f"mAP@50-95: {metrics.box.map:.4f}")    # Mean Average Precision at IoU 0.5 to 0.95
print(f"Precision: {metrics.box.mp:.4f}")     # Mean Precision
print(f"Recall: {metrics.box.mr:.4f}")         # Mean Recall
```

```
mAP@50: 0.1051
mAP@50-95: 0.0839
Precision: 0.6490
Recall: 0.1107
```

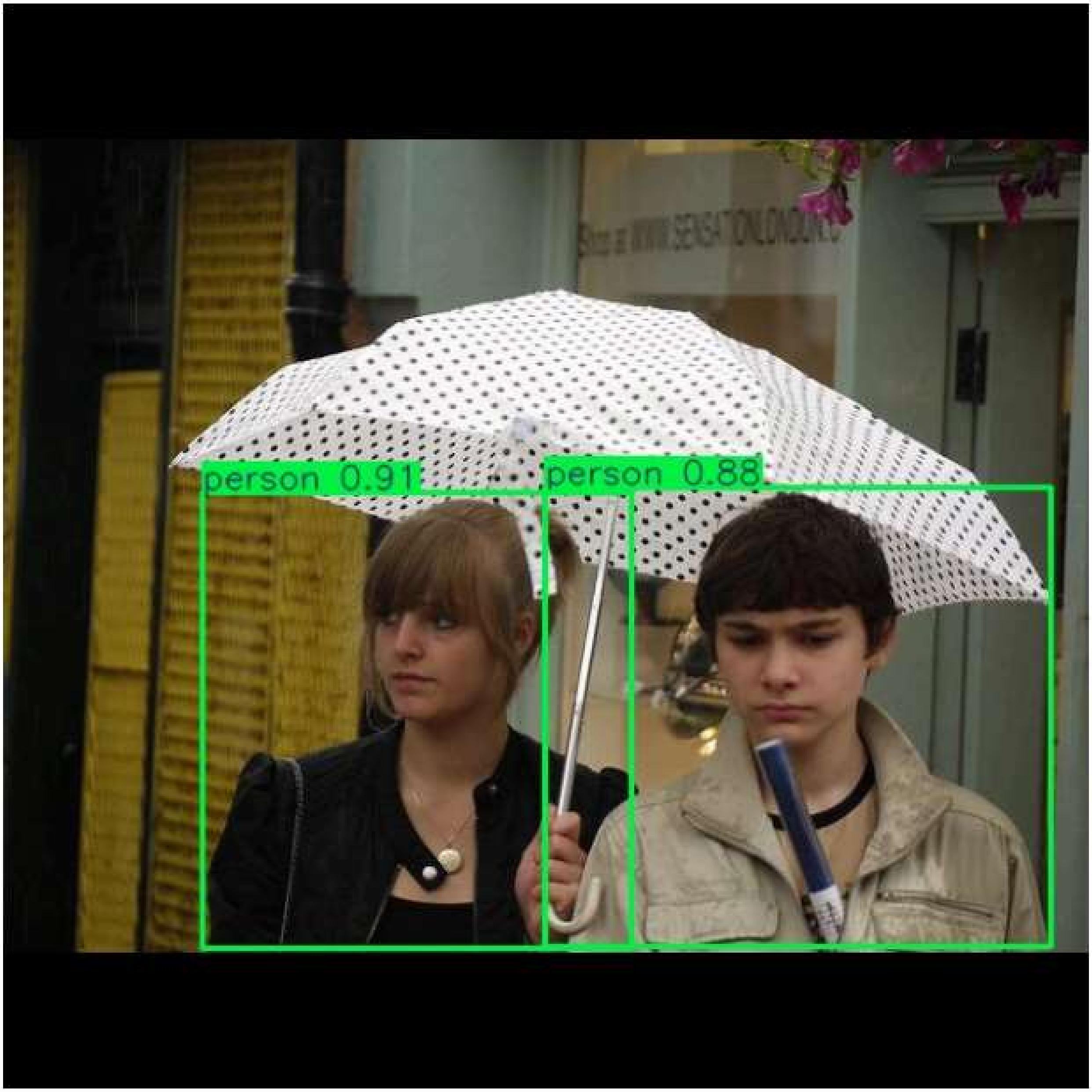
```
precision = metrics.box.mp
recall = metrics.box.mr

if precision + recall > 0:
    f1_score = 2 * (precision * recall) / (precision + recall)
    print(f"F1 Score: {f1_score:.4f}")
else:
    print("F1 Score: Undefined (precision + recall = 0)")
```

```
F1 Score: 0.1891
```

4. **Visualize Inference Results:**
    - Use `glob` to locate the saved prediction image, then display it using `matplotlib` or `PIL`.

```
import cv2
import matplotlib.pyplot as plt
from PIL import Image
import glob

# Run inference
results = model.predict(img_path, save=True, show=False)

# Find the saved prediction file
result_img_path = glob.glob('runs/detect/predict*/*.jpg')[3]

# Load and display the result using PIL and matplotlib
img = Image.open(result_img_path)
plt.figure(figsize=(8, 8))
plt.imshow(img)
plt.axis('off')
plt.show()
```

```
image 1/1 /content/COCO-Dataset-34/test/images/000000025668_jpg.rf.f67afc05b355ac25bef19
Speed: 2.3ms preprocess, 10.4ms inference, 1.3ms postprocess per image at shape (1, 3, 6
Results saved to runs/detect/predict3
```



Discussion and Conclusion

After running inference and visualizing the detection results on the COCO test images, the following performance metrics were observed:

- **mAP@50:** 0.1051
- **mAP@50-95:** 0.0839
- **Precision:** 0.6490
- **Recall:** 0.1107

- **F1 Score:** 0.1891

Discussion:

1. **Precision vs. Recall:**
   - The model achieves a relatively high precision (~0.65), indicating that when it predicts an object, it is often correct.
   - However, the recall is notably low (~0.11), meaning that the model is missing a large number of objects present in the images. This imbalance suggests that while the model is cautious in its predictions, it is not sensitive enough to detect all relevant objects.
2. **Training Considerations:**
   - The current training setup, although a good starting point, appears to be insufficient for achieving robust detection performance on the COCO dataset.
   - Increasing training epochs, applying more extensive data augmentation, and further hyperparameter tuning (such as adjusting the learning rate schedule and modifying anchor boxes) are potential strategies to improve recall without compromising precision.
3. **Visual Inspection:**
   - The visualizations show that detected objects have correctly drawn bounding boxes and appropriate confidence scores. However, many objects are still missed, which is consistent with the low recall metric.
   - The visualization reinforces the notion that while the model is reliable when it makes a detection, its overall sensitivity is low.

Conclusion:

- **Strengths:** The model demonstrates reliable detections when it does identify an object, as evidenced by the high precision. This is promising for applications where false positives are particularly problematic.
- **Weaknesses:** The low recall and overall mAP highlight the need for improvement in detecting all relevant objects in a scene.

Overall, this experiment provides valuable insights into the strengths and limitations of using YOLOv11 for object detection on the COCO dataset. With further refinements, the model can be optimized to achieve a more balanced performance, which is crucial for real-world applications.

Declaration

I, Om Bhutkar, confirm that the work submitted in this assignment is my own and has been completed following academic integrity guidelines. The code is uploaded on my GitHub repository account, and the repository link is provided below:

GitHub Repository Link: https://github.com/OmBhutkar/Object_Detection_and_Multi_Object_Classification_using_YOLO11_Model.

Signature: Om Santosh Bhutkar