```
from google.colab import drive
drive.mount('/content/drive')
```

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

```
!git clone https://github.com/ultralytics/yolov5
%cd yolov5
!pip install -r requirements.txt
```

Show hidden output

```
data_yaml_content = """
train: /content/drive/MyDrive/Datasets/WeedCrop.v1i.yolov5pytorch/train/images
val: /content/drive/MyDrive/Datasets/WeedCrop.v1i.yolov5pytorch/valid/images

nc: 2
names: ['crop', 'weed']
"""
with open('/content/yolov5/data.yaml', 'w') as f:
    f.write(data_yaml_content)
```

```
!python /content/yolov5/train.py --img 320 --batch 32 --epochs 14 --data /content/yolov5/data.yaml --weights yolov5s.pt --cache --name we
```

```
      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       5/13      1.83G     0.1086    0.03266   0.002478         32        320: 100% 78/78 [00:19<00:00,  3.91it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 4/4 [00:01<00:00,  2.00it/s]
                   all        235       1605       0.37       0.32      0.316      0.103

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       6/13      1.83G     0.1059    0.03433   0.001896         60        320: 100% 78/78 [00:19<00:00,  3.96it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 4/4 [00:02<00:00,  1.54it/s]
                   all        235       1605      0.153       0.36       0.18     0.0524

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       7/13      1.83G     0.1042    0.03487   0.001678         41        320: 100% 78/78 [00:19<00:00,  3.90it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 4/4 [00:02<00:00,  1.54it/s]
                   all        235       1605      0.598      0.326      0.335      0.114

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       8/13      1.83G     0.1011    0.03495   0.001566         47        320: 100% 78/78 [00:19<00:00,  3.96it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 4/4 [00:01<00:00,  2.15it/s]
                   all        235       1605      0.461      0.543      0.454      0.185

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
       9/13      1.83G     0.1007    0.03612   0.001251         26        320: 100% 78/78 [00:19<00:00,  3.96it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 4/4 [00:01<00:00,  2.39it/s]
                   all        235       1605      0.766      0.352      0.406      0.163

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      10/13      1.83G      0.098    0.03568   0.001239         49        320: 100% 78/78 [00:19<00:00,  3.99it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 4/4 [00:01<00:00,  2.37it/s]
                   all        235       1605      0.509      0.391      0.435      0.171

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      11/13      1.83G    0.09729    0.03525   0.001171         50        320: 100% 78/78 [00:20<00:00,  3.87it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 4/4 [00:02<00:00,  1.79it/s]
                   all        235       1605       0.51      0.498      0.465      0.185

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      12/13      1.83G    0.09565    0.03564   0.001196         33        320: 100% 78/78 [00:19<00:00,  3.96it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 4/4 [00:02<00:00,  1.56it/s]
                   all        235       1605      0.552      0.417      0.438      0.178

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      13/13      1.83G    0.09453    0.03477   0.001265         33        320: 100% 78/78 [00:19<00:00,  4.03it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 4/4 [00:02<00:00,  1.89it/s]
                   all        235       1605      0.438      0.573      0.472      0.221

14 epochs completed in 0.089 hours.
Optimizer stripped from yolov5/runs/train/weed_detection/weights/last.pt, 14.3MB
Optimizer stripped from yolov5/runs/train/weed_detection/weights/best.pt, 14.3MB

Validating yolov5/runs/train/weed_detection/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 4/4 [00:03<00:00,  1.10it/s]
                   all        235       1605      0.442      0.573      0.472      0.222
                  crop        235         47      0.434       0.34      0.309      0.166
                  weed        235       1558       0.45      0.806      0.635      0.278
Results saved to yolov5/runs/train/weed_detection
```

```
!python /content/yolov5/val.py --weights /content/yolov5/runs/train/weed_detection/weights/best.pt --data /content/yolov5/data.yaml --ir
```

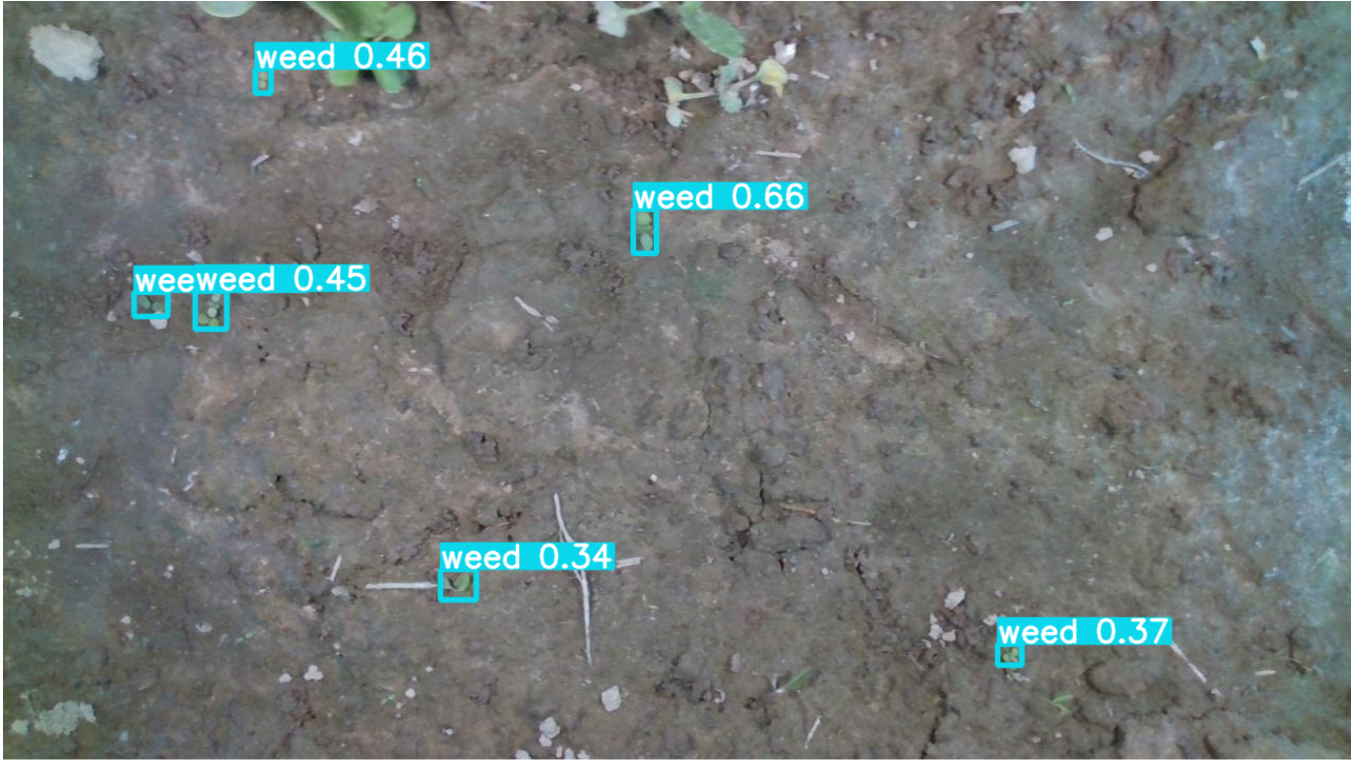
```
val: data=/content/yolov5/data.yaml, weights=['/content/yolov5/runs/train/weed_detection/weights/best.pt'], batch_size=32, imgsz=32
YOLOv5 🚀 v7.0-350-g6096750f Python-3.10.12 torch-2.3.1+cu121 CUDA:0 (Tesla T4, 15102MiB)

Fusing layers...
Model summary: 157 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
val: Scanning /content/drive/.shortcut-targets-by-id/1IFuoxcFB8PONXZPG6-fYuhYglnfGxmBG/Datasets/WeedCrop.v1i.yolov5pytorch/valid/lab
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 8/8 [00:06<00:00,  1.33it/s]
                   all        235       1605      0.451      0.572      0.486      0.228
                  crop        235         47      0.444       0.34      0.337      0.177
                  weed        235       1558      0.458      0.804      0.634      0.279
Speed: 0.0ms pre-process, 2.6ms inference, 7.5ms NMS per image at shape (32, 3, 320, 320)
Results saved to yolov5/runs/val/exp
```

```
import torch
from PIL import Image
from pathlib import Path
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score, accuracy_score

# Load the model
model = torch.hub.load('ultralytics/yolov5', 'custom', path='/content/yolov5/runs/train/weed_detection/weights/best.pt')

# Define test image directory
test_img_dir = Path('/content/drive/MyDrive/Datasets/WeedCrop.v1i.yolov5pytorch/test/images')

# Load test images
test_imgs = list(test_img_dir.glob('*.jpg'))

y_true = []
y_pred = []

# Defining the classes
classes = ['crop', 'weed']

for img_path in test_imgs:
    img = Image.open(img_path)
    results = model(img)

    # Assuming binary classification: weed (1) and crop (0)
    # Adjust threshold as needed
    weed_detected = any([True for x in results.xyxy[0] if x[5] == 1])
    y_pred.append(1 if weed_detected else 0)

    # Load the corresponding label
    label_path = img_path.with_suffix('.txt').as_posix().replace('images', 'labels')
    with open(label_path, 'r') as f:
        anns = f.read().strip().split('\n')
        # Now 'classes' is defined and can be used here
        is_weed = any([True for ann in anns if classes[int(ann.split()[0])] == 'weed'])
        y_true.append(1 if is_weed else 0)

# Calculate metrics
cm = confusion_matrix(y_true, y_pred)
precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)
f1 = f1_score(y_true, y_pred)
accuracy = accuracy_score(y_true, y_pred)

print("Confusion Matrix:\n", cm)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("Accuracy:", accuracy)
```

```
Using cache found in /root/.cache/torch/hub/ultralytics_yolov5_master
YOLOv5 🚀 2024-8-6 Python-3.10.12 torch-2.3.1+cu121 CUDA:0 (Tesla T4, 15102MiB)

Fusing layers...
Model summary: 157 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
Adding AutoShape...
Confusion Matrix:
 [[  0   9]
 [  0 109]]
Precision: 0.923728813559322
Recall: 1.0
F1 Score: 0.960352422907489
Accuracy: 0.923728813559322
```

```
from PIL import Image
import torch

# Function to load and preprocess a single image
def load_image(img_path):
    img = Image.open(img_path)
    return img

# Function to make a prediction on a single image
def predict_image(model, img_path):
    img = load_image(img_path)
    results = model(img)
    return results

# Load the model
model = torch.hub.load('ultralytics/yolov5', 'custom', path='/content/yolov5/runs/train/weed_detection/weights/best.pt')

# An example image
example_img_path = '/content/drive/MyDrive/Datasets/WeedCrop.v1i.yolov5pytorch/test/images/32356_jpg.rf.a493fe0fbbb4ac1ad1b72f01bc599b52

# Make a prediction
results = predict_image(model, example_img_path)
results.show()  # Display the image with detections

# Output the prediction
weed_detected = any([True for x in results.xyxy[0] if x[5] == 1])
if weed_detected:
    print("The image is predicted to contain weed(s).")
else:
    print("The image is predicted to be free of weeds.")
```

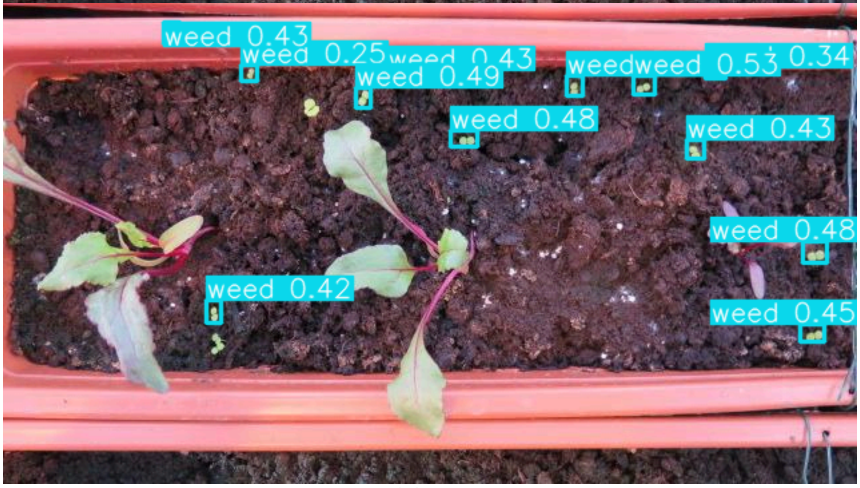
```
Using cache found in /root/.cache/torch/hub/ultralytics_yolov5_master
YOLOv5 🚀 2024-8-6 Python-3.10.12 torch-2.3.1+cu121 CUDA:0 (Tesla T4, 15102MiB)

Fusing layers...
Model summary: 157 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
Adding AutoShape...
```



```
The image is predicted to contain weed(s).
```

```
from PIL import Image
import torch

# Function to load and preprocess a single image
def load_image(img_path):
    img = Image.open(img_path)
    return img

# Function to make a prediction on a single image
def predict_image(model, img_path):
    img = load_image(img_path)
    display(img)
    results = model(img)
    return results

# Load the model
model = torch.hub.load('ultralytics/yolov5', 'custom', path='/content/yolov5/runs/train/weed_detection/weights/best.pt')

# An example image
example_img_path = '/content/drive/MyDrive/Datasets/WeedCrop.v1i.yolov5pytorch/test/images/IMG_6136_JPG.rf.660194dbd4186904e9f18afef31e4b
# Make a prediction
results = predict_image(model, example_img_path)
results.show()  # Display the image with detections

# Output the prediction
weed_detected = any([True for x in results.xyxy[0] if x[5] == 1])
if weed_detected:
    print("The image is predicted to contain weed(s).")
else:
    print("The image is predicted to be free of weeds.")
```

```
Using cache found in /root/.cache/torch/hub/ultralytics_yolov5_master
YOLOv5 🚀 2024-8-6 Python-3.10.12 torch-2.3.1+cu121 CUDA:0 (Tesla T4, 15102MiB)

Fusing layers...
Model summary: 157 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
Adding AutoShape...
```



```
The image is predicted to contain weed(s).
```