YOLO + SAM Object Detection and Segmentation in Colab

Name : Shehryar Khan

```python
import torch
import numpy as np
import cv2
import matplotlib.pyplot as plt
from segment_anything import sam_model_registry, SamPredictor
from PIL import Image
from ultralytics import YOLO
```

These are the necessary imports:
- PyTorch for running models
- OpenCV and matplotlib for image processing and visualization
- Segment Anything and Ultralytics YOLO libraries for detection and segmentation.

```python
# Load trained YOLO model
yolo_model = YOLO("runs/detect/train/weights/best.pt")
```

Loads the YOLOv11 model that was trained on the brain tumor dataset. This will be used to detect tumors in input images.

```python
# Upload your image (or provide path)
from google.colab import files
```

```python
uploaded = files.upload()
image_path = list(uploaded.keys())[0]
```

Prompts the user to upload an image in Google Colab and captures the uploaded file path.

```python
# Run detection
results = yolo_model(image_path)
detections = results[0].boxes.xyxy.cpu().numpy().astype(int)
```

Runs object detection on the uploaded image using the YOLO model. The bounding boxes are extracted in (x1, y1, x2, y2) format.

```python
# Download SAM weights if not already present
!wget https://dl.fbaipublicfiles.com/segment_anything/sam_vit_b_01ec64.pth
```

Downloads the pre-trained SAM (Segment Anything Model) checkpoint file.

```python
# Load the SAM model
sam_checkpoint = "sam_vit_b_01ec64.pth"
sam = sam_model_registry["vit_b"](checkpoint=sam_checkpoint)
sam.to("cuda" if torch.cuda.is_available() else "cpu")
predictor = SamPredictor(sam)
```

Loads the SAM model and prepares the predictor. It is moved to GPU if available, otherwise CPU is used.

```python
# Load image and set it for SAM
image_bgr = cv2.imread(image_path)
image_rgb = cv2.cvtColor(image_bgr, cv2.COLOR_BGR2RGB)
predictor.set_image(image_rgb)
```

Reads and converts the uploaded image to RGB format, and sets it as the input image for the SAM predictor.

```python
# Loop through YOLO bounding boxes and apply SAM
for i, box in enumerate(detections):
    input_box = np.array(box)  # x1, y1, x2, y2
    masks, scores, logits = predictor.predict(box=input_box, multimask_output=True)

    # Plot the best mask (highest confidence)
    plt.figure(figsize=(5, 5))
    plt.imshow(image_rgb)
    plt.imshow(masks[0], alpha=0.6)  # semi-transparent mask
    plt.title(f"Object {i+1} (Confidence: {scores[0]:.2f})")
    plt.axis("off")
    plt.show()
```

For each object detected by YOLO, SAM is used to generate segmentation masks.
The best mask (with highest confidence) is overlayed on the original image and displayed using matplotlib.

Output :

Choose files 41598_202...g1_HTML.jpg

- **41598_2023_41576_Fig1_HTML.jpg**(image/jpeg) - 54666 bytes, last modified: 22/04/2025 - 100% done

```
Saving 41598_2023_41576_Fig1_HTML.jpg to 41598_2023_41576_Fig1_HTML (1).jpg

image 1/1 /content/41598_2023_41576_Fig1_HTML (1).jpg: 640x640 1 meningioma, 25.5ms
Speed: 8.9ms preprocess, 25.5ms inference, 3.1ms postprocess per image at shape (1, 3, 640, 640)
--2025-04-22 11:07:33--  https://dl.fbaipublicfiles.com/segment_anything/sam_vit_b_01ec64.pth
Resolving dl.fbaipublicfiles.com (dl.fbaipublicfiles.com)... 108.157.254.124, 108.157.254.121, 108.157.254.102, ..
Connecting to dl.fbaipublicfiles.com (dl.fbaipublicfiles.com)|108.157.254.124|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 375042383 (358M) [binary/octet-stream]
Saving to: ‘sam_vit_b_01ec64.pth’

sam_vit_b_01ec64.pt 100%[===================>] 357.67M  34.4MB/s    in 5.3s

2025-04-22 11:07:38 (67.7 MB/s) - ‘sam_vit_b_01ec64.pth’ saved [375042383/375042383]
```

Object 1 (Confidence: 0.96)