```
%%capture
!pip install ultralytics
```

```
%%capture
!pip install transformers==4.42.0
!pip install accelerate
```

```
import cv2
import torch
import torchvision.models as models
import torchvision.transforms as transforms
from transformers import pipeline
import nltk
from nltk import word_tokenize, pos_tag, ne_chunk
from collections import Counter

print("All modules imported")
```

```
All modules imported
```

```
# Load an image
image_path = '/buisness.jpg'  # Change this to your actual image path
image = cv2.imread(image_path)
# Check if the image was successfully loaded
if image is None:
    raise FileNotFoundError(f"Cannot open/read image file: {image_path}")
```

YOLO MODEL

```
from ultralytics import YOLO

# Load a pre-trained YOLOv10n model
model = YOLO("yolov10n.pt")

# Perform object detection on an image
results = model(image_path)
detected_objects = []
# Print the detected objects
for result in results:
    boxes = result.boxes  # Boxes object for bounding box outputs
    masks = result.masks  # Masks object for segmentation masks outputs
    keypoints = result.keypoints  # Keypoints object for pose outputs
    probs = result.probs  # Probs object for classification outputs
    obb = result.obb  # Oriented boxes object for OBB outputs
    scores = result.boxes.conf
    classes = result.boxes.cls
    for box, score, cls in zip(boxes, scores, classes):
        label = model.names[int(cls)]
        detected_objects.append(f"{label} (confidence: {score:.2f})")
        print(f'Object: {label}, Confidence: {score}')
    result.save(filename="result.jpg")
```

```
Downloading https://github.com/ultralytics/assets/releases/download/v8.2.0/yolov10n.pt to 'yolov10n.pt'...
100%|██████████| 5.59M/5.59M [00:00<00:00, 83.3MB/s]

image 1/1 /buisness.jpg: 448x640 3 persons, 1 cup, 1 chair, 1 couch, 1 dining table, 1 laptop, 382.8ms
Speed: 20.2ms preprocess, 382.8ms inference, 6.8ms postprocess per image at shape (1, 3, 448, 640)
Object: person, Confidence: 0.9088335037231445
Object: person, Confidence: 0.8238939642906189
Object: laptop, Confidence: 0.6804316639900208
Object: cup, Confidence: 0.6720181703567505
Object: person, Confidence: 0.5322608351707458
Object: chair, Confidence: 0.45813655853271484
Object: dining table, Confidence: 0.36274057626724243
Object: couch, Confidence: 0.2790757417678833
```

Florence

```
%%capture
!pip install flash_attn einops timm
```

```
#generating_caption
import requests
import torch
from PIL import Image
from transformers import AutoProcessor, AutoModelForCausalLM

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

model = AutoModelForCausalLM.from_pretrained("microsoft/Florence-2-large", trust_remote_code=True)
model.to(device)  # Move model to GPU
processor = AutoProcessor.from_pretrained("microsoft/Florence-2-large", trust_remote_code=True)

prompt = "<VERY_DETAILED_CAPTION>"

image = Image.open(image_path)

inputs = processor(text=prompt, images=image, return_tensors="pt")
inputs = {k: v.to(device) for k, v in inputs.items()}  # Move inputs to GPU

generated_ids = model.generate(
    input_ids=inputs["input_ids"],
    pixel_values=inputs["pixel_values"],
    max_new_tokens=1024,
    num_beams=3,
    do_sample=False
)

# Move generated_ids back to CPU before decoding
generated_ids = generated_ids.cpu()

generated_text = processor.batch_decode(generated_ids, skip_special_tokens=False)[0]

parsed_answer = processor.post_process_generation(generated_text, task="<VERY_DETAILED_CAPTION>", image_size=(image.width, i

print(parsed_answer)


detected_objects_text = ", ".join(detected_objects)
combined_context = f"The image contains the following objects: {detected_objects_text}. The scene depicts: {parsed_answer}"
print(combined_context)
```

```
The image contains the following objects: person (confidence: 0.91), person (confidence: 0.82), laptop (confidence: 0.68
```

QUESTION - DistilBERT

```
from transformers import pipeline
question_answerer = pipeline("question-answering", model='distilbert-base-cased-distilled-squad')


result = question_answerer(question="List all the objects in scene",context=combined_context)
print(f"Answer: '{result['answer']}', score: {round(result['score'], 4)}, start: {result['start']}, end: {result['end']}")
```

```
Answer: 'couch (confidence: 0.28)', score: 0.029, start: 233, end: 257
```

```
from transformers import AutoModelForQuestionAnswering, AutoTokenizer, pipeline

while True:
    question = input("Please ask a question about the image (or type 'exit' to quit): ")
    if question.lower() == 'exit':
        break

    context = combined_context

    model_name = "deepset/roberta-base-squad2"
    model = AutoModelForQuestionAnswering.from_pretrained(model_name)
    tokenizer = AutoTokenizer.from_pretrained(model_name)

# a) Get predictions
    nlp = pipeline('question-answering', model=model_name, tokenizer=model_name)
    QA_input = {
        'question': question,
        'context': context
    }
    res = nlp(QA_input)


    print("Answer:", res)
```

```
Please ask a question about the image (or type 'exit' to quit): How many people in frame?
Answer: {'score': 0.5470255017280579, 'start': 325, 'end': 330, 'answer': 'three'}
Please ask a question about the image (or type 'exit' to quit): How many women in frame?
Answer: {'score': 0.0007227178430184722, 'start': 325, 'end': 330, 'answer': 'three'}
Please ask a question about the image (or type 'exit' to quit): How many men in frame?
Answer: {'score': 0.4921179711818695, 'start': 339, 'end': 342, 'answer': 'two'}
Please ask a question about the image (or type 'exit' to quit): What are the men holding?
Answer: {'score': 0.15993867814540863, 'start': 521, 'end': 526, 'answer': 'a pen'}
Please ask a question about the image (or type 'exit' to quit): Who is staring at the laptop?
Answer: {'score': 0.04477490857243538, 'start': 657, 'end': 679, 'answer': 'The woman on the right'}
Please ask a question about the image (or type 'exit' to quit): How is the atmosphere?
Answer: {'score': 0.5041537880897522, 'start': 852, 'end': 871, 'answer': 'serious and focused'}
Please ask a question about the image (or type 'exit' to quit): What is the man wearing?
Answer: {'score': 0.0003371036145836115, 'start': 513, 'end': 526, 'answer': 'holding a pen'}
Please ask a question about the image (or type 'exit' to quit): exit
```