**Using Gradio for Object Detection with YOLO**

# 1. Introduction

This document explains how to create a simple object detection web app using **Gradio** and **YOLOv5**. The app allows users to upload an image, detects objects in the image, and returns the image with bounding boxes and confidence scores.

# 2. Required Libraries

import gradio as gr import torch

from torchvision import transforms from PIL import Image, ImageDraw

* **Gradio** → For creating the web UI.
* **Torch** → To load and run the YOLOv5 model.
* **PIL (Pillow)** → For handling images and drawing bounding boxes.

# 3. Loading the YOLO Model

def load\_model():

model = torch.hub.load('ultralytics/yolov5', 'yolov5s') # Load YOLOv5 model return model

* Loads the **pre-trained YOLOv5 model** from Ultralytics' repository.
* The **YOLOv5s model** is lightweight and optimized for speed.

# 4. Object Detection Function

def predict(image):

model = load\_model() results = model(image)

labels, cords = results.xyxyn[0][:, -1], results.xyxyn[0][:, :-1] # Get labels & coordinates img = Image.fromarray(image) draw = ImageDraw.Draw(img)

for i in range(len(labels)):

x1, y1, x2, y2, conf = cords[i]

x1, y1, x2, y2 = int(x1 \* img.width), int(y1 \* img.height), int(x2 \* img.width), int(y2 \* img.height)

draw.rectangle([x1, y1, x2, y2], outline="red", width=2) draw.text((x1, y1), f"{model.names[int(labels[i])]}: {conf:.2f}", fill="red") return img

* Runs object detection on the input image.
* Extracts bounding boxes, confidence scores, and class labels.
* Draws **bounding boxes and labels** on the image.

# 5. Creating the Gradio Web App

demo = gr.Interface(fn=predict, inputs=gr.Image(type="numpy"), outputs=gr.Image(type="pil")) demo.launch()

* Uses gr.Interface to create a simple web app.
* Accepts image uploads, processes them, and returns the **annotated image**.

# 6. Running in Google Colab

To run the app in **Google Colab**, follow these steps:

1. Install Gradio if not installed:
2. !pip install gradio 3. Run the above code.

4. Click on the **public Gradio link** to test the app.

**What is YOLO?**

## 1. Introduction

**YOLO (You Only Look Once)** is a real-time object detection algorithm that processes the entire image in a **single pass**, making it extremely fast.

## 2. Key Features

✅ **Real-time Speed** – Works at 30+ FPS. ✅ **End-to-End Detection** – Predicts objects in one

pass. ✅ **High Accuracy** – Detects multiple objects in an image. ✅ **Uses a Single Convolutional Network** – Unlike traditional multi-stage approaches.

**3. How YOLO Works?**

* **Grid-based Prediction** → The image is divided into a grid (e.g., 7×7), and each cell predicts objects inside it.
* **Bounding Box Prediction** → Each box contains:
  + (x, y) → Object center.
  + (w, h) → Box width and height. o **Confidence Score** → How sure the model is.
* **Class Prediction** → Each box is assigned a class label (e.g., car, person, dog).
* **Non-Maximum Suppression (NMS)** → Removes overlapping boxes and keeps the best one.

## 4. YOLO Versions

* **YOLOv1** – First version.
* **YOLOv2 & YOLOv3** – Improved accuracy.
* **YOLOv4** – Optimized for real-time tasks.
* **YOLOv5** – Open-source, highly optimized.
* **YOLOv7 & YOLOv8** – Faster and more efficient.

**5. Where is YOLO Used?**

🚗 **Autonomous Vehicles** – Detects pedestrians, cars, and road signs. 🩺 **Medical Imaging** – Identifies tumors and abnormalities. 🎥 **Security & Surveillance** – Recognizes faces and activities. 🛍️ **Retail** – Product recognition in stores.

## Conclusion

* **Gradio + YOLOv5** makes it easy to build an object detection web app.
* YOLO is **fast and accurate**, making it ideal for real-time applications.
* Running the code in **Google Colab** allows easy access without local setup. Would you like a **Streamlit version** as well? 🚀