Project Idea: Simple Object Detection Web Application

Deadline: Hoàn thiện source code và submit source code **trước 12pm của ngày thứ 2 tuần sau**.

Objective:
Build a web application where users can upload an image, and the application will detect and display common objects (e.g., people, cars, animals) in the image using a pre-trained object detection model.

Key Features:

1. Users can upload an image through a simple web interface.
2. The application processes the image using a pre-trained object detection model.
3. The detected objects are highlighted with bounding boxes and labels on the image.
4. The result is displayed back to the user on the web page.

Dataset Recommendation

1. Dataset: Use the COCO (Common Objects in Context) dataset.
    a. Why: COCO is a widely used public dataset that contains images of everyday scenes with labeled objects (e.g., people, animals, vehicles). It includes both images and bounding box annotations, which are essential for object detection.
    b. Link: [COCO Dataset](https://cocodataset.org/)
    c. Alternative: If COCO feels too large, you can use a smaller subset of COCO or another dataset like Pascal VOC, which is also public and contains object detection annotations.

Accepted Criteria for Evaluation
To evaluate the quality of the you' projects, use the following criteria:

1. Functionality:
    a. The application allows users to upload an image.
    b. It successfully processes the image using a pre-trained object detection model.
    c. It displays the detected objects with bounding boxes and labels on the image.
2. User Interface:
    a. The web interface is simple and intuitive, with clear instructions for uploading an image.
    b. The results are displayed clearly, showing the processed image with detections.
3. Code Quality:
    a. The code is well-structured and organized.
    b. Key sections of the code are commented to explain their purpose.
    c. The project uses appropriate libraries and tools (e.g., Flask for the backend, a pre-trained model for detection).
4. Documentation:

    a. A README file is provided with clear instructions on how to set up and run the application.
    b. The README includes any necessary dependencies and a brief explanation of how the application works.
5. Handling Edge Cases:
    a. The application can handle different image formats (e.g., JPEG, PNG).
    b. It manages images of various sizes, resizing them if necessary for the model.

Additional Recommendations

1. Pre-trained Model:
    a. Suggest using a pre-trained model like YOLO (You Only Look Once) or SSD (Single Shot MultiBox Detector), which are popular for object detection and have implementations available in libraries like TensorFlow or PyTorch.
    b. Alternatively, you can use OpenCV, which provides easy-to-use pre-trained models for object detection.
2. Web Framework:
    a. For the backend, recommend Flask due to its simplicity and ease of use for beginners.
    b. For the frontend, suggest using basic HTML, CSS, and JavaScript. Optionally, they can use Bootstrap for styling to make the interface more polished.
3. Image Processing:
    a. Ensure you handle image uploads properly (e.g., saving uploads temporarily, handling different formats).
    b. They may need to resize or preprocess images to match the input requirements of the pre-trained model.
4. Deployment:
    a. For simplicity, you can run the application locally and demonstrate it on their machines.
    b. If they are comfortable, they can explore deploying it on platforms like Heroku or Google Cloud, but this is optional for beginners.

Example Workflow for you

1. Set Up the Environment:
    a. Install necessary libraries (e.g., Flask, OpenCV, TensorFlow).
    b. Download a pre-trained object detection model.
2. Build the Backend:
    a. Use Flask to create an endpoint for image uploads.
    b. Load the pre-trained model and process the uploaded image to detect objects.
    c. Draw bounding boxes and labels on the image.
3. Build the Frontend:
    a. Create a simple HTML form for image uploads.
    b. Display the processed image with detections.
4. Test the Application:

    a. Upload various images to ensure the application works with different objects and image sizes.
5. Document the Project:
    a. Write a README with setup instructions and a brief explanation of the code.