Object Detection Using YOLOS-Tiny

1. Project Overview

This documentation outlines the setup, execution, and testing of an object detection system using the YOLOS-Tiny model, chosen for its lightweight design, fast inference times, and reliable accuracy. The system processes images, detects objects, annotates them with bounding boxes, and outputs results in JSON format.

---

2. Prerequisites

System Requirements

- Python 3.10 or later
- Dockerfile (optional, if running in a container)

Dependencies

The required dependencies are listed in the `requirements.txt` file:

- PyTorch
- Transformers (with timm)
- FastAPI
- Uvicorn
- Matplotlib
- Pillow

Install them using:

```
pip install -r requirements.txt
```

---

3. Project Structure

The project files are organized as follows:

```
yolo_object_detection/
├── api.py                # FastAPI backend for image upload and detection
├── main.py               # Standalone script for object detection
├── requirements.txt      # Python dependencies
├── Dockerfile            # Docker container setup
├── samples/              # Sample input images (e.g., cats.jpg)
```

---

4. Setup Instructions

A. Run Locally

1. Go to project directory
2. Install dependencies:
   `pip install -r requirements.txt`
3. Test the standalone script:
   - Place an input image in the `samples/` folder.

Run standalone script by command:

```
python main.py
```

   - Output will be saved as:
     - Annotated image: `output_detected.jpg`
     - JSON results: `detection_results.json`

Run the FastAPI backend:

```
python api.py
```

   - Access the API at `http://127.0.0.1:8000`.

B. Run with Docker

Build the Docker image:

```
docker build -t object-detection-api .
```

1. Build the Docker image:
   `docker build -t object-detection-api .`

2. Run the container:
   `docker run -d -p 8000:8000 object-detection-api`

3. Access the API at `http://127.0.0.1:8000`.

---

5. Using the FastAPI Backend

Upload an Image

1. Use the `/upload/` endpoint to upload an image:
   - `curl -X POST "http://127.0.0.1:8000/upload/" -F "file=@samples/cats.jpg"`

2. The response includes:
   - `filename`: Name of the uploaded file.
   - `json_result`: JSON output with detected objects.
   - `image_path`: Path to the annotated image.
   - `image_base64`: Base64-encoded annotated image.

---

6. Output Details

A. Annotated Image

- Bounding boxes are drawn on the detected objects.
- Labels and confidence scores are displayed.

B. JSON Output

Example JSON structure:

```
{
    "filename": "cats.jpg",
    "json_result": {
        "Detected Objects": [
            {
                "label": "cat",
                "score": 98.7,
                "bounding_box": {
                    "x1": 15,
                    "y1": 30,
                    "x2": 120,
                    "y2": 150
                }
            }
        ]
    },
    "image_path": "/absolute/path/to/output.jpg",
    "image_base64":
"/9j/4AAQSkZJRgABAQEAAAAAAAD/2wCEAAEBAQEBAQEBAQEBAQEBA..."
}
```

---

7. Implementation Details

A. YOLOS-Tiny

- Model: `hustvl/yolos-tiny` from Hugging Face Transformers.
- Processes input images and outputs object detection results.

B. FastAPI Backend

- Handles image upload, inference, and response generation.
- Returns results in JSON format and provides the annotated image.

C. Standalone Script

- `main.py` processes a single input image for testing and development purposes.