📄 DeHaDo-AI Baseline Model

🧠 Objective

This document provides a baseline script to detect **handwritten regions** in scanned document images using **YOLOv8** from Ultralytics.

📦 Requirements

Install the necessary packages with:

pip install ultralytics opencv-python

📁 Folder Structure

DeHaDo-AI_Baseline/

```
├─── handwritten_region_detection_yolov8.py   # Baseline script for detection and label saving
├─── images/                    # Folder containing input scanned document images
├─── results/                   # Folder for output (annotated images and labels)
│    └─── labels/                # Subfolder containing label files (.txt) per image
└─── yolov8n.pt                 # (Optional) Pretrained YOLOv8 model for inference
```

💉 Baseline Inference Script

Save the following as *handwritten_region_detection_yolov8.py:*

```
"""
DiHaDo'25 Challenge - Baseline Script for Handwritten Region Detection using YOLOv8

This script demonstrates how to use YOLOv8 (Ultralytics) for detecting handwritten
regions in scanned document images.
Requirements:
- Python 3.8+
- Ultralytics library (pip install ultralytics)
- OpenCV (pip install opencv-python)
Usage:
1. Place test images in the 'images/' folder.
2. Run the script.
3. Results with detected handwritten regions will be saved in 'results/'.

Author: DiHaDo'25 Organizers
"""
```

```
import os
from ultralytics import YOLO
import cv2

# --- CONFIGURATION ---
MODEL_PATH = 'yolov8n.pt'  # Use a trained handwritten region detection model if available
IMAGE_DIR = 'images'
RESULT_DIR = 'results'
LABEL_DIR = os.path.join(RESULT_DIR, 'labels')
CONFIDENCE_THRESHOLD = 0.25

# --- PREPARE DIRECTORIES ---
os.makedirs(RESULT_DIR, exist_ok=True)
os.makedirs(LABEL_DIR, exist_ok=True)

# --- LOAD MODEL ---
model = YOLO(MODEL_PATH)

# --- PROCESS IMAGES ---
for filename in os.listdir(IMAGE_DIR):
    if filename.lower().endswith(('.jpg', '.jpeg', '.png')):
        image_path = os.path.join(IMAGE_DIR, filename)
        print(f"Processing: {image_path}")

        # Run inference
        results = model(image_path, conf=CONFIDENCE_THRESHOLD)

        for result in results:
            # Save annotated image
            annotated_img = result.plot()
            output_img_path = os.path.join(RESULT_DIR, filename)
            cv2.imwrite(output_img_path, annotated_img)

            # Save labels
            label_path = os.path.join(LABEL_DIR, filename.rsplit('.', 1)[0] + '.txt')
            with open(label_path, 'w') as f:
                for box in result.boxes:
                    cls_id = int(box.cls[0])
                    conf = float(box.conf[0])
                    x1, y1, x2, y2 = map(int, box.xyxy[0])  # Bounding box
                    f.write(f'{cls_id} {conf:.2f} {x1} {y1} {x2} {y2}\n')
            print(f"Saved image to: {output_img_path}")
            print(f"Saved labels to: {label_path}")
```

The *handwriting.yaml* file should look like:

```
path: /path/to/dataset
train: images/train
val: images/val
names:
  0: handwritten_region
```

✅ **Output**

For every image in the *images/* folder, the script generates:

- **Annotated image** with bounding boxes saved to the *results/* folder.
- **Label file** in text format saved to results/labels/, containing:
    - Detected class ID
    - Confidence score
    - Bounding box coordinates (x_min, y_min, x_max, y_max)