Computer Vision PPE Detection Project-Document

Submitted by

Name: Shardul More

Contact: 7775004907

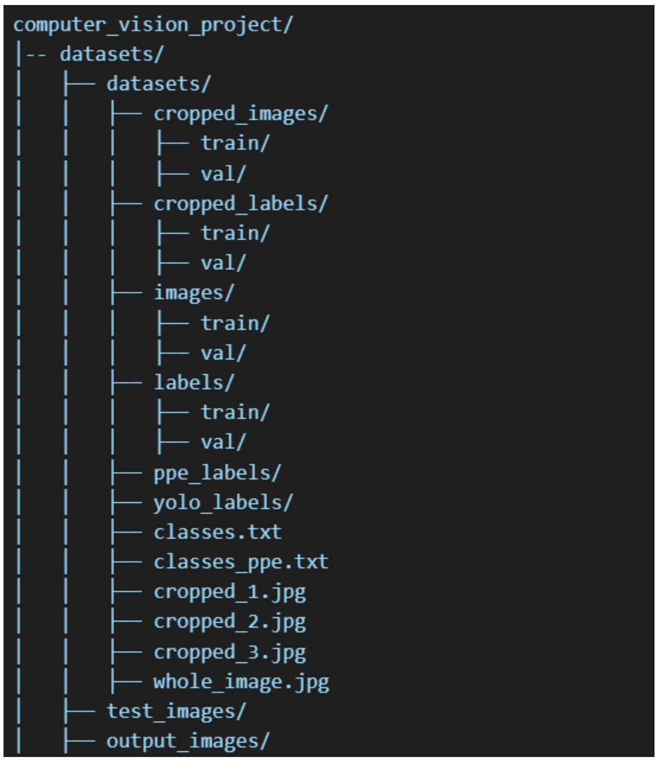
Mail: morepatilshardul@gmail.com

1. Introduction

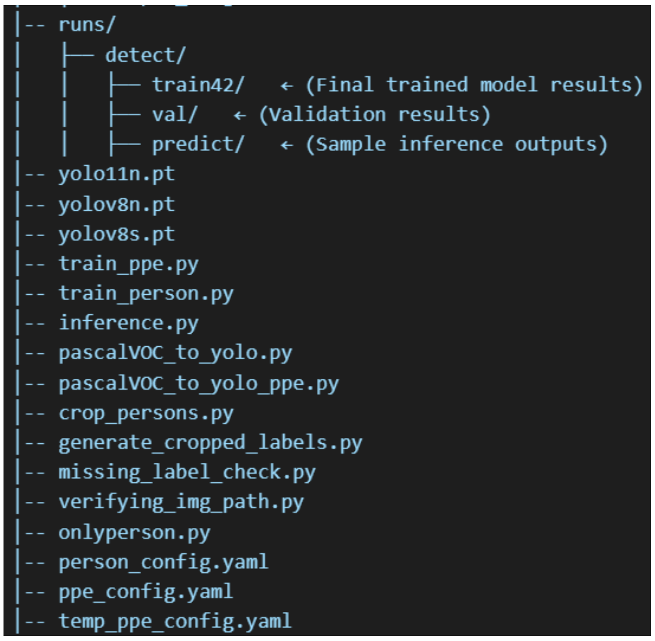
This project focuses on computer vision techniques to detect and analyze personal protective equipment (PPE) and human presence in images. The project uses YOLO models for object detection and includes scripts for data processing, training, validation, and inference. All necessary configurations, scripts, and model weights are organized within the project directory.

2. Project Structure

The project is organized as follows:



```
computer_vision_project/
|-- datasets/
|   ├── datasets/
|   |   ├── cropped_images/
|   |   |   ├── train/
|   |   |   ├── val/
|   |   ├── cropped_labels/
|   |   |   ├── train/
|   |   |   ├── val/
|   |   ├── images/
|   |   |   ├── train/
|   |   |   ├── val/
|   |   ├── labels/
|   |   |   ├── train/
|   |   |   ├── val/
|   |   ├── ppe_labels/
|   |   ├── yolo_labels/
|   |   ├── classes.txt
|   |   ├── classes_ppe.txt
|   |   ├── cropped_1.jpg
|   |   ├── cropped_2.jpg
|   |   ├── cropped_3.jpg
|   |   ├── whole_image.jpg
|   ├── test_images/
|   ├── output_images/
```



```
|-- runs/
|   ├── detect/
|   |   ├── train42/   ← (Final trained model results)
|   |   ├── val/  ← (Validation results)
|   |   ├── predict/   ← (Sample inference outputs)
|-- yolo11n.pt
|-- yolov8n.pt
|-- yolov8s.pt
|-- train_ppe.py
|-- train_person.py
|-- inference.py
|-- pascalVOC_to_yolo.py
|-- pascalVOC_to_yolo_ppe.py
|-- crop_persons.py
|-- generate_cropped_labels.py
|-- missing_label_check.py
|-- verifying_img_path.py
|-- onlyperson.py
|-- person_config.yaml
|-- ppe_config.yaml
|-- temp_ppe_config.yaml
```

3. Installation & Setup

Prerequisites

Ensure you have the following installed:

- Python (>=3.8), PyTorch
- CUDA (if using GPU acceleration)
- Required libraries from requirements.txt

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

4. Usage Instructions

Training the Model

Run the following command to train the PPE detection model:

```
python train_ppe.py
```

Running Inference

To perform inference on test images:

```
python inference.py --source datasets/test_images/
```

5. Annotation Conversion

Convert Pascal VOC annotations to YOLO format using:

```
python pascalVOC_to_yolo.py
```

```
python pascalVOC_to_yolo_ppe.py
```

6. Preprocessing & Data Handling

Cropping Persons from Images

```
python crop_persons.py
```

Generating Cropped Labels

```
python generate_cropped_labels.py
```

Checking for Missing Labels

```
python missing_label_check.py
```

7. Results & Outputs

- *runs/detect/train42/* contains final trained model results.
- *runs/detect/val/* stores validation outputs.
- *runs/detect/predict/* holds sample inference results.

8. Model Weights

Trained model weights:

- *weights/person_model.pt*
- *weights/ppe_model.pt*
- *yolo11n.pt*
- *yolov8n.pt*
- *yolov8s.pt*

9. Troubleshooting & Common Issues

- **CUDA out of memory**: Reduce batch size or use CPU mode.
- **Dataset path errors**: Ensure paths in scripts match your file locations.

10. Conclusion & Future Work

Future improvements may include:

- Expanding dataset diversity.
- Enhancing model accuracy with additional augmentations.
- Deploying the model in a real-time video processing system.

Question 1:

Please name your script pascalVOC_to_yolo.py which will take two paths, first path is base input directory path and second path is output directory where YOLOv8 annotations will be saved. Please use Python's argparse library as we will be running the script with the command line.

Response:

I have implemented pascalVOC_to_yolo.py which correctly takes two paths:

1. **Base input directory** (where Pascal VOC XML annotation files are stored).
2. **Output directory** (where converted YOLO format annotations will be saved).

The script uses argparse to take command-line arguments:

python pascalVOC_to_yolo.py --input_dir datasets/datasets/labels --output_dir datasets/datasets/yolo_labels

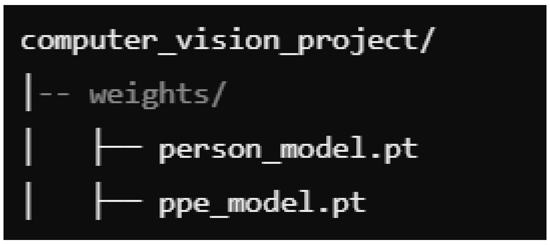
This script extracts bounding box coordinates from Pascal VOC annotations, converts them into YOLO format, and saves the new labels in the specified output directory.

Question 2 & 3:

Put weight files in the weights named directory.

Response:

I have stored the model weight files inside the weights/ directory as required:



```
computer_vision_project/
|-- weights/
|   ├── person_model.pt
|   ├── ppe_model.pt
```

These weights are used for inference in our detection pipeline.

Question 4 & 5:

Please name your script inference.py and use argparse library. This script will take the following 4 arguments: input_dir, output_dir, person_det_model, and ppe_detection_model. Use OpenCV to draw the bounding boxes and put texts.

Response:

inference.py script correctly takes the required arguments using argparse:

python inference.py --input_dir datasets/test_images/ --output_dir datasets/output_images/ \--person_det_model weights/person_model.pt --ppe_detection_model weights/ppe_model.pt

Logic Implemented:

1. Load the **person detection model** (weights/person_model.pt).
2. Detect persons in images from input_dir.
3. Crop detected persons and run the **PPE detection model** (weights/ppe_model.pt).
4. Map the PPE detections back to the original image.
5. **Draw bounding boxes** with class labels using OpenCV.
6. Save the processed images in output_dir.

Question 6:

Report must contain the logic used in question 2.

Response:

The logic for handling model weights (*weights/person_model.pt* and *weights/ppe_model.pt*) is as follows:

- The **person detection model** detects individuals in the image.
- The **PPE detection model** is applied to cropped person images to classify PPE compliance.
- These weights are used dynamically during inference and are stored in the weights/ directory for modularity and ease of access.