PPE Detection Challenge - Report

This report summarizes the end-to-end implementation of a two-stage object detection pipeline designed to detect persons and their associated PPE (Personal Protective Equipment) items in images. The solution includes data preprocessing, model training, custom annotation transformation, and inference using two proposed YOLOv8 models.

 # Problem Statement Overview

- **Q1**: Convert annotations from Pascal VOC to YOLOv8 format.
- **Q2**: Train a YOLOv8 model for person detection only.
- **Q3**: Train another YOLOv8 model for PPE detection on cropped person images.
- **Q4 & Q5**: Build a pipeline (inference.py) to perform inference using both models and convert cropped predictions back to full images.
- **Q6**: Submit a report detailing approaches, logic, learnings, and evaluation metrics.

Q1: Annotation Conversion

A custom Python script pascalVOC_to_yolo.py was developed using **argparse** to:

- ✓ Parse XML files from Pascal VOC format.
- ✓ Normalize bounding box coordinates relative to image dimensions.
- ✓ Save YOLOv8 format labels (.txt) preserving folder structure.

Q2: Person Detection Model Training

- The dataset had full annotations including person and PPE classes.
- Since we needed PPE annotations with respect to full images in the next step, the person detection model was trained on the full annotations dataset.
- However, during inference, only person predictions were used to crop individual persons from full images.

Training Configuration

- **Model**: YOLOv8s
- **Epochs**: 100
- **Image size**: 640x640
- **Loss function**: Default YOLOv8 loss (objectness + classification + bbox regression)
- **Data YAML**: data.yaml (with all 10 classes)

Results:

Metric	Value
Precision (person)	**0.7581**
Recall (person)	**0.7258**
mAP@0.5 (person)	**0.2380**
mAP@0.5:0.95 (person)	**0.4508**

Inference Speed

Stage	Time per image (ms)
Preprocessing	5.25 ms
Inference	11.21 ms
Postprocess	3.82 ms

Q3: PPE Detection on Cropped Images

Dataset Preparation

- Full images were used to detect persons using the trained model from Q2.
- Each detected person was cropped and saved as a separate image.
- PPE annotations from the full image were converted to match cropped coordinates using bounding box IOU and center overlap logic.

Annotation Conversion Logic

Let a person bounding box in the full image be represented as:

• B_p = [x_p, y_p, w_p, h_p], where (x_p, y_p) is the top-left coordinate, and w_p, h_p are width and height.

Let a PPE annotation in the full image be:

• B_ppe = [x_a, y_a, w_a, h_a]

Step 1: Check overlap using center-point inside logic

• Compute the center of the PPE box: c_x = x_a + w_a / 2, c_y = y_a + h_a / 2

• If the center of PPE lies within person box: x_p < c_x < x_p + w_p, y_p < c_y < y_p + h_p then the PPE is assigned to this person crop.

Step 2: Convert annotation to cropped coordinates

• Shift the PPE box relative to the top-left of person box: x' = x_a - x_p, y' = y_a - y_p

• Crop size is w_p × h_p, so normalize: x'_center = (x' + w_a / 2) / w_p, y'_center = (y' + h_a / 2) / h_p

• Final normalized width and height: w' = w_a / w_p, h' = h_a / h_p

PPE Classes Detected

- hard-hat, gloves, mask, glasses, boots, vest, ppe-suit, ear-protector, safety-harness

Training Configuration

- **Model**: YOLOv8s
- **Epochs**: 100
- **Image size**: 640x640
- **Loss function**: Default YOLOv8 loss (objectness + classification + bbox regression)
- **Data YAML**: ppe.yaml (with 9 PPE classes)
- Observations: Some PPE classes (e.g., safety-harness, ear-protector) were underrepresented, affecting performance slightly.

Overall Evaluation Metrics

The PPE detection model was trained using YOLOv8 for detecting various PPE items on cropped images of persons. The evaluation metrics are as follows:

Metric	Value
Precision	0.6547
Recall	0.5908
mAP@0.5	0.6133
mAP@0.5:0.95	0.4146
Fitness	0.4345

PPE Class	mAP@0.5
Hard-hat	0.6855
Mask	0.6965
Glasses	0.4146
Ear Protector	0.4146
Safety Harness	0.4146
PPE Suit	0.3722

Model Speed

Phase	Time (ms/image)
Preprocessing	0.139 ms
Inference	36.889 ms
Postprocessing	2.131 ms

Q4 & Q5: Inference Pipeline

A complete pipeline inference.py was created with the following logic:

Arguments via argparse

- input_dir: Directory with test images
- output_dir: Directory to save results
- person_det_model: Path to person detection weights
- ppe_detection_model: Path to PPE detection weights

Pipeline Steps

1. Load full images.
2. Run YOLOv8 person detection.
3. Crop detected persons.
4. Run PPE detection on cropped images.
5. Convert PPE predictions from cropped to full image coordinates.
6. Use cv2.rectangle() and cv2.putText() to draw all detections.
7. Save results to output_dir.

Learnings & Challenges

- **Annotation Conversion & Quality Control**
  Converting VOC to YOLO format required precise normalization of bounding boxes. Errors in this step directly impacted training quality. Keeping all PPE labels even when training only for person detection turned out useful for downstream logic.
- **Class Imbalance in PPE Dataset**
  The PPE dataset showed significant class imbalance — items like hard-hat and mask appeared frequently, while ppe suit, ear protector, and gloves were rare. This led to lower performance for underrepresented classes and highlighted the importance of augmentation or sampling strategies.
- **Annotation Logic for Cropped Images**
  Associating PPE annotations to individual cropped persons was challenging. Initial IOU-based matching resulted in many missed or incorrect labels. An improved approach using **cropping + re-inference** greatly enhanced label accuracy, producing ~880 valid cropped labels from 1080 crops.
- **Model Training Strategy**
  The person detection model was trained on all classes but used only person detections during inference. This allowed reuse of existing annotations for later steps. PPE model training on person crops helped reduce background noise but still struggled with rare items.
- **Inference & Coordinate Mapping**
  Mapping PPE detections from cropped images back to the full image required careful handling of offsets and dimensions. Ensuring spatial consistency was key for accurate visualization.
- **Visualization & Debugging**
  Using OpenCV instead of YOLO's built-in plotting gave better control for bounding box drawing, label formatting, and result clarity — especially when overlaying multiple stages of predictions.

Directory Structure for Submission

Syook_CV_assignment/

├── pascalVOC_to_yolo.py

├── inference.py

├── weights/

│ ├── person.pt

│ └── ppe.pt

├── report.pdf

Conclusion

The entire pipeline was successfully built and tested. Person detection is accurate, and PPE detection shows good performance despite class imbalance. The logic and custom scripts make the system easily extendable and interpretable.

Some Output images : RESULTS 😊 (by inference pipeline):