YOLOv8 OBJECT DETECTION - PERSONS AND PPE

Raghavendra Dabral (raghavendradabral@gmail.com | +918297139711 | https://github.com/raghav-i/yolov8-obj)

ABSTRACT
The purpose of this report is to explain in detail the working logic of the project, the results achieved, constraints and future improvements for the project.

While person-detection using YOLOv8 models is a fairly easy task, PPE detection with minimal data is challenging. A larger compute resource for models like YOLOv8L/YOLOv8x, or a larger dataset would be much more ideal. However, the combined inference of the 2 produces fairly decent results.

ABOUT THE DATA
The normal dataset consists of:

- Original Images: Full-sized images with multiple persons.
- Annotations: PascalVOC format for original images, converted to YOLOv8 format for training.

The cropped person dataset consists of:

- Person Cropping: Persons were cropped from full images using the person detection model.
- PPE Annotations: Annotations were adjusted for cropped images using coordinate mapping logic.

pascalVOC_to_yolo.py
The pascalVOC_to_yolo.py script converts PascalVOC annotations (.xml) to YOLOv8 format (.txt). This is necessary because YOLOv8 requires annotations in a specific format for training.

Input: Directory containing PascalVOC .xml files and corresponding images.
Output: Directory with YOLOv8 .txt annotation files.
Error Handling:

- Skips invalid or incomplete .xml files.
- Deletes corresponding images if annotations are invalid.

split_data.py
The split_data.py script divides the dataset into training and validation sets. This ensures the model can be evaluated on unseen data during training.

Input: Directory containing images and corresponding annotations.
Output: Two directories (train and val) for images and labels.
Random Shuffling: Ensures a random and unbiased split.

train_person_model.py

The train_person_model.py script trains a YOLOv8 model for person detection using a custom dataset. The model is trained to detect only the person class in images.

Model: Uses the YOLOv8n (nano) variant as the base model.
Training Data: Requires a YAML file (dataset.yaml) specifying the dataset paths and class names.dataset.yaml file is correctly configured with 1 class (“person”) and the path to the training and validation images which were created using the split_data.py file.
Single-Class Detection: The single_cls=True flag ensures the model is trained to detect only the person class.
Epochs: 50
Image Size: 640x640 pixels
Batch Size: 8
Output Directory: Results are saved in /person_detection_results and the model weights are in /person_detection_results/weighs/best.pt

crop_person.py

The crop_person.py script is used to crop person regions from full images using a pre-trained person detection model. These cropped images are then used for training the PPE detection model. The script also generates a crop_mapping.json file to store metadata about the cropped regions.

Person Detection: Uses best.pt from /person_detection_results/weighs to locate persons in full images.
Cropping: Crops detected person regions and saves them as separate images.
Metadata: Generates a crop_mapping.json file containing:

- Original image name.
- Cropped image coordinates (xmin, ymin, xmax, ymax).
- Original image dimensions (original_width, original_height).
- Cropped image dimensions (crop_width, crop_height).

crop_mapping.json

The crop_mapping.json file is a metadata file generated by the crop_person.py script. It contains information about cropped person regions extracted from full images. This file is essential for:

- Adjusting PPE Annotations: Mapping PPE annotations from the original full image coordinates to the cropped person image coordinates.
- Tracking Crops: Maintaining a record of which cropped regions belong to which original image and their respective dimensions.

The crop_mapping.json file is a JSON dictionary where each key represents a unique crop ID, and the value is a dictionary containing metadata about the crop.

- Maintains Consistency: Ensures PPE annotations are correctly mapped to cropped images.
- Enables Training: Provides the necessary metadata for training the PPE detection model on cropped person images.
- Tracks Relationships: Links cropped images back to their original images for debugging and analysis.

This file is a critical component of the PPE Detection System, enabling accurate annotation adjustment and model training.

adjust_ppe_annotations.py

The adjust_ppe_annotations.py script adjusts PPE annotations from the original full images to the cropped person images. This is necessary because the PPE detection model is trained on cropped person images, and the annotations must match the new coordinate system of the cropped images.

Input:

- Original annotations (.txt files) for full images.
- crop_mapping.json file generated by crop_person.py.

Output:

- Adjusted annotations (.txt files) for cropped person images.

Coordinate Adjustment:

- Converts annotations from the original image coordinates to the cropped image coordinates.
- Only annotations within the cropped region are retained.

The split_data.py is used again to train-val split the cropped data for ppe training

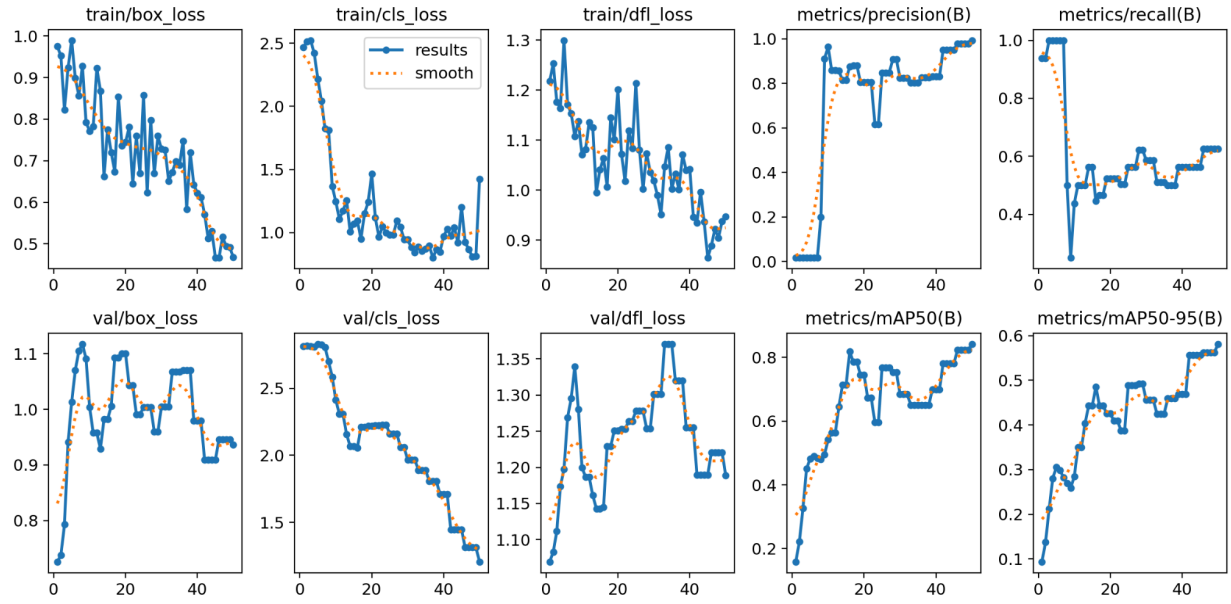
train_ppe_model.py

The train_ppe_model.py script trains a YOLOv8 model for person detection using a custom dataset. The model is trained to detect 9 classes in images.

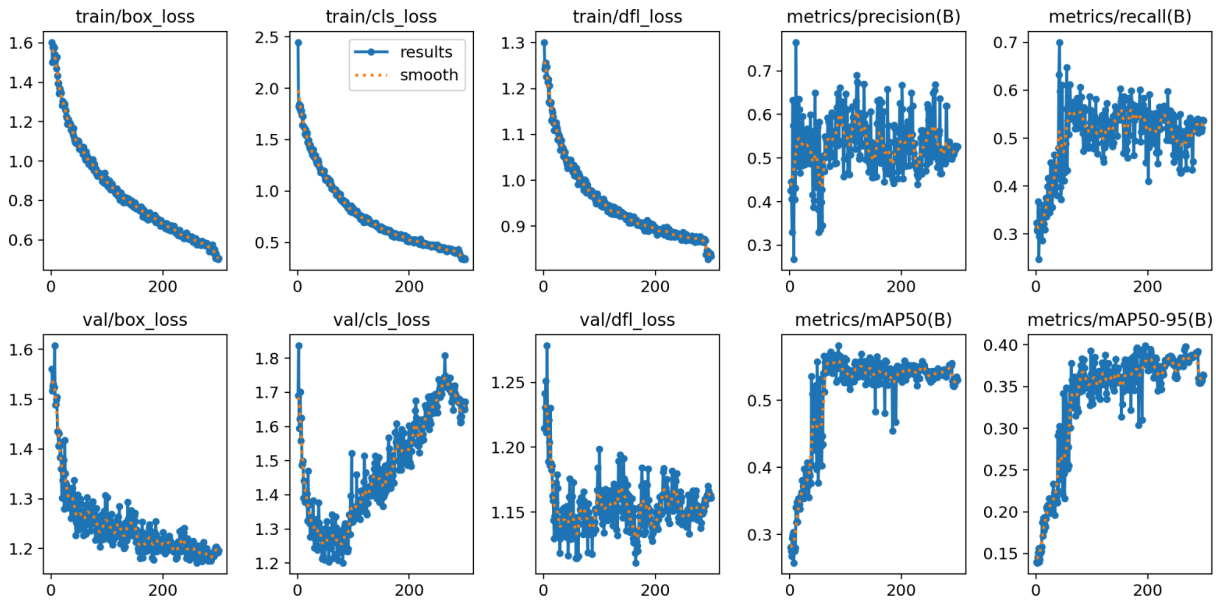
Model: Uses the YOLOv8m (medium) variant as the base model.
Training Data: Requires a YAML file (ppe_dataset.yaml) specifying the dataset paths and class names. ppe_dataset.yaml file is correctly configured with 9 classes ("safety-harness", "hard-hat", "gloves", "glasses", "mask", "boots" , "vest", "ppe-suit", "ear-protector") and the path to the training and validation images which were created using the split_data.py file.
Epochs: 300
Image Size: 150
Batch Size: 8
Output Directory: Results are saved in /ppe_detection_results and the model weights are in /ppe_detection_results/weighs/best.pt

inference.py
This script performs person and PPE detection using the 2 trained models. It processes images from an input directory (the initial images in this case), detects persons and their PPE, and saves object detection on images with bounding boxes to an output directory.
The boxes are drawn using cv2 and the argparse library is used to run it in command line.

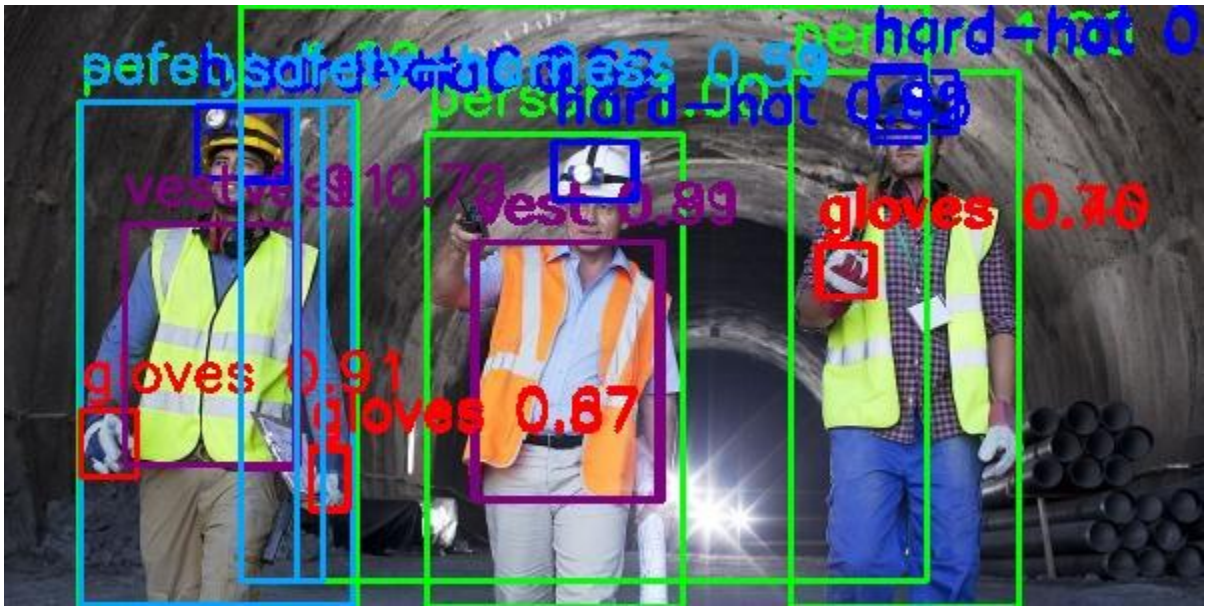
RESULTS
Person detection:



PPE detection:



Inference example:



CONCLUSION

Achieving a higher accuracy on the PPE detection model requires a lot more training time and a heavier compute due to the scarcity of the data. However, even with the current results, the inference results are very substantial. Further hyperparameter tuning for the ppe_detection_model can cause a drop in the loss by 0.10 - 0.15.