PPE Detection Using Machine Learning For Enhanced Safety in Construction

Darin Verduzco, Fatimat Atanda, Victor Hugo Germano

Group 4

Shiley-Marcos School of Engineering, University of San Diego

AAI-501: Introduction to Artificial Intelligence

Andrew Van Benschoten, PhD

December 9, 2024

Abstract

Construction work professionals are subject to multiple risks in the work environment that are mitigated using protection, and relying on safety procedures to ensure usage. Using computer vision technologies we created a model capable of identifying people wearing Personal Protective Equipment (PPE). The model leverages a computer vision model, YOLOv8 (You Only Look Once), convolutional neural network and a dataset of images to train a model using two different approaches; custom manual-labelled approach and hybrid approach (combining manually annotated and auto-annotated images).

Introduction

Construction practitioners are at a disproportionately higher risk of fatal and nonfatal injuries compared to practitioners from other industries. The absence of and inappropriate use of personal protective equipment (PPE), hereinafter referred to as PPE non-compliance, are major causes of fatal and nonfatal injuries at construction workplaces (Jalil Al-Bayati et al., 2023) One strategy to mitigate these incidents is implementing a strict policy of wearing personal protective equipment (PPE) when working in hazardous environments. This equipment can include protective glasses, high-visibility vests, gloves, and helmets to ensure the workers remain safe and productive in the long run.

However, manually monitoring PPE compliance is challenging as it involves the daily supervision of an extensive workforce. Traditional safety inspections rely heavily on human intervention, which is prone to error and delays (Kursunoglu et al., 2022).

An alternative is using the latest computer vision (CV)-based solutions and frameworks to determine whether workers wear PPE according to safety protocols. By integrating computer vision PPE detection technology, industries can automate and enhance the compliance monitoring process. (Vukicevic et al., 2024).

Objectives

- Develop a YOLOv8-based object detection model for PPE compliance monitoring.
- Combine team manually annotated datasets with labelled Kaggle datasets for model training.
- Combine manually annotated and auto-annotated datasets for model training (hybrid approach).
- Evaluate the system's performance on a diverse dataset to ensure reliability.

Methodology

1. Data Collection and Preparation

Manually-annotated datasets included images created and labeled by the team to ensure high-quality bounding boxes and class annotations for PPE object detection. This dataset was used for the custom model training approach. It serves as a gold standard for initial training, providing a reliable foundation for the model to learn essential detection patterns in the combined dataset used for hybrid training. This ensures accuracy, eliminates labeling noise, and provides clear, consistent annotations for critical PPE categories.

The dataset for the custom model includes 2,801 images and labels from a Kaggle "Construction Site Safety Image Dataset Roboflow" dataset and 58 images extracted from 3 personal videos filmed by Darin Verduzco which were manually annotated using a free open

source labeling software, Computer Vision Annotation Tool (CVAT), by first defining the 10 class labels in the software, then drawing bounding boxes around each object in the images. There was no confirmation found that all annotations in the Kaggle dataset were created manually, such as in the images from our personal video source, so “initial annotation count” is the terminology used for annotations in Table 1.

Auto-annotated datasets used pre-trained YOLOv8 model to automatically label datasets. Annotations include bounding boxes and class labels for PPE items, with a confidence threshold of 0.5 applied to filter predictions. This approach allows for rapid scale up of dataset size, reducing the time and cost of manual labeling. It introduces variability in data to improve the model’s generalization capability.

Combined Dataset: The manually annotated and auto-annotated datasets were merged to form the final training dataset for the hybrid training approach. This datasets includes 63 images sourced from google in addition to custom datasets. Of the total datasets, 178 images were manually annotated and the rest of the images (2,746) were annotated using a pre-trained model for efficiency and scalability.

Dataset Labels/Classes: The 10 class labels from this dataset include the following: “Hardhat”, “Mask”, “NO-Hardhat”, “NO-Mask”, “NO-Safety Vest”, “Person”, “Safety Cone”, “Safety Vest”, “Machinery”, and “Vehicle”. The inclusion of “NO-” before the objects was desirable because it will be useful for providing counts when an object is not detected on a person. However, the focus is on 7 main objects identification; “Hardhat”, “NO-Hardhat”, “Mask”, “NO-Mask”, “Safety Vest”, “NO-Safety Vest”, “Person,”.

Data Combination: The custom datasets were randomly split into training (70%), validation (20%), and test (10%) subsets. For the hybrid datasets, the manually annotated datasets (178)

were split into training (80%), validation (20%) subsets while the final combined datasets were split into training (70%), validation (20%), and test (10%).

Dataset Properties: The personal video source and Kaggle dataset images contained image resolutions of 640 x 640 pixels. Image sizes of the google search images varied, but a conversion to the size of 640 x 640 will be done automatically by default when training an image dataset on a YOLO model. 27 filenames from the Kaggle dataset larger than 100 characters were also truncated manually to eliminate errors.

	Trained from Custom Dataset		Hybrid Model	
Dataset Description	**Image Count**	**Initial Annotation Count**	**Image Count**	**Initial Annotation Count**
Kaggle dataset: "Construction Site Safety Image Dataset Roboflow "	2801	2801	2801	120
Personal video source - Darin Verduzco	58	58	58	58
Google search images	0	0	63	0
Total	2859	2859	2922	178

Table 1: The summary of counts for the dataset of each model

2. Model Training

Custom Training Parameters

Model training was conducted on all annotated datasets (kaggle and custom) with the following parameters and lasted approximately 8.78 hours: epochs: 30, seed: 88 (for reproducibility), nms: True (Non-Maximum Suppression for predictions), optimizer : AdamW (via auto), and batch: 0.7 (determines batch size based on 70% memory usage). Prediction parameters used included iou: 0.5 (Intersection over Union threshold, for reducing duplicates) and conf: 0.4 (confidence threshold to remove low confidence predictions).

Hybrid Training Parameters

Initial training was conducted with manually annotated datasets first. The pretrained model was then used to auto-annotate the rest of the datasets by generating bounding boxes and class labels, as described in Table 2, leveraging prior knowledge from related datasets to bootstrap the dataset. Reducing the dependence on manual annotations saves time and resources while maintaining reasonably good annotation quality. The final training was then conducted on combined datasets (manual and auto-annotated).

Parameter	Initial Training	Final Training
Dataset	Manually Annotated Dataset	Combined Dataset
Epochs	50	30
Image Size	640	640
Batch Size	16	20
Learning Rate(lr0)	1e-4	1e-3
Optimizer	AdamW(via “auto)	AdamW(via “auto)
Mixed Precision(amp)	Not Used	Enabled (for faster training)
Patience (early stopping)	10	10

Table 2: Hybrid Model Training Parameters

Results

Evaluation Metrics

- **Precision (P):** Measures how many of the model’s predictions are correct.
- **Recall (R):** Measures how many true instances the model captures.
- **Mean Average Precision (mAP50):** Evaluates detection accuracy at an IoU threshold of 50%.
- **Mean Average Precision (mAP50-95):** Evaluates detection accuracy across multiple IoU thresholds.

The tables 3 and 4 show the class-specific and overall model performance on custom dataset training and hybrid dataset training.

Class	Precision	Recall	mAP50	mAP50-95
Mask	0.933	0.911	0.94	0.743
Safety Vest	0.918	0.759	0.834	0.601
Hardhat	0.918	0.724	0.807	0.529
Person	0.923	0.812	0.892	0.681
NO-Mask	0.875	0.657	0.722	0.399
NO-Safety Vest	0.908	0.762	0.834	0.588
NO-Hardhat	0.93	0.702	0.806	0.529
Overall (10 classes)	0.904	0.752	0.824	0.581

Table 3: Custom Datasets Results

Class	Precision	Recall	mAP50	mAP50-95
Mask	0.829	0.946	0.929	0.782
Safety Vest	0.785	0.842	0.89	0.709
Hardhat	0.863	0.956	0.972	0.83
Person	0.885	0.92	0.956	0.825
NO-Mask	0.772	0.83	0.841	0.56
NO-Safety Vest	0.722	0.73	0.808	0.62
NO-Hardhat	0.833	0.875	0.915	0.737
Overall (10 classes)	0.805	0.868	0.89	0.715

Table 4: Hybrid Datasets Results

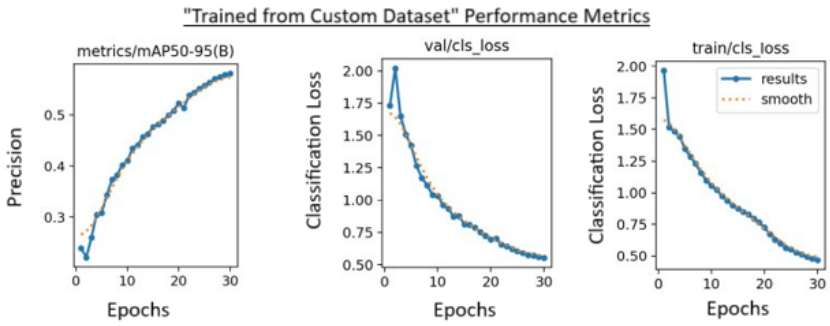


Figure 1: Custom Dataset Performance Metrics

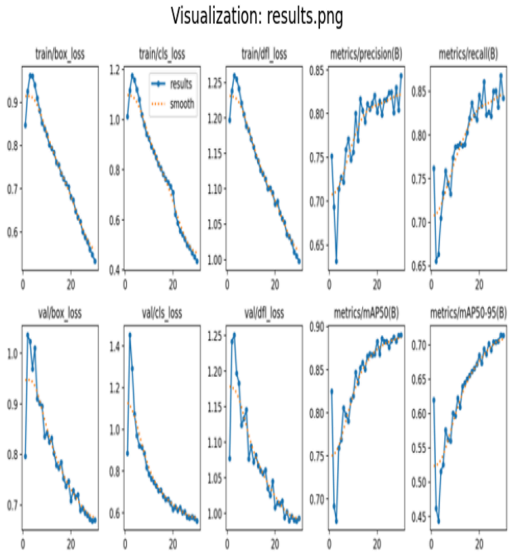


Figure 2: Hybrid Training Performance Metric

Analysis

The model demonstrated consistent improvement across epochs in both training approaches, with an increasing mAP50 score and steadily decreasing losses (box loss, class loss, and DFL loss), indicating good convergence. High precision and recall values highlight the model's reliability, while validation images confirm its ability to detect multiple objects in complex construction scenes. Additionally, the model showed strong generalization to unseen images, suggesting its potential for real-world construction site safety applications. However, the significant gap

between mAP50 and mAP50-95 scores indicates a decline in performance at higher IoU thresholds. While the model performed well across most classes, it struggled slightly with "NO-Mask" detection presented in Table 4, revealing room for improvement. Some misclassifications and missed detections were observed, which could be mitigated by addressing class imbalances, increasing the dataset size for underperforming classes, and employing techniques such as transfer learning or ensemble methods.

Real Life Example

Figure 3 - FUTEK HD Modern Basement Construction Technology

Testing the model on a real construction site's video stream provides insight into its readiness for production. Figure 3 highlights key challenges such as image quality, camera distance, non-workers on-site, and non-fixed cameras. Addressing these issues involves establishing standard operations for video capture and image processing, including fixed camera positions and geofencing to exclude non-site entities (e.g., spectators, vehicles, animals) and reduce false positives.

The next step is enhancing the model's categorization capabilities through a combination of refinement techniques, performance optimization, and bias mitigation. The first step is model and dataset refinement: fine tuning the hyperparameters for training, exploring different architectures and amplifying the use of augmentation techniques. Balancing the class distribution and expanding the dataset, reducing the risk of model bias. Techniques like Layer-wise Relevance Propagation (LRP) can be used to minimize the influence of background bias on classifiers.

Performance Optimization could play a significant role. Applying techniques like filter pruning or low-rank approximation to reduce model size without significantly impacting performance and investigating bias in this dataset can also prove to be a good decision: Using methods like Hybrid Sample Synthesis to identify likely bias-conflicting samples and creating new training data to reduce the impact of bias-aligned samples.

Conclusion

The development of a PPE detection model using the YOLOv8 framework demonstrated high precision and recall in detecting critical PPE categories, showcasing its potential for improving workplace safety. The hybrid training approach effectively combined manually and auto-annotated datasets, achieving robust detection performance.

Both models performed to a satisfactory degree, with the hybrid approach having a much larger training time performance. As the datasets increase in the number of images not annotated, hybrid would be the preferred option, understanding the possibility of bias and unbalanced classes represented in the dataset.

Key recommendations for improvement include expanding the dataset to enhance generalizability, adopting active learning for continuous refinement, prioritizing real-time deployment with alert systems, and implementing robust performance monitoring during deployment. These enhancements aim to ensure sustained accuracy, scalability, and adaptability in diverse environments.

The project underscores the transformative potential of automated PPE detection systems in mitigating workplace risks, supporting safety compliance, and promoting safer work environments. It highlights the intersection of technology and occupational safety, advancing automation in compliance management while safeguarding workers.

References

- Computer Vision Annotation Tool. (2024). Accessed November 25, 2024, from https://www.cvat.ai/
- Hady, M. F. A., & Schwenker, F. (2013). Semi-supervised learning. In M. Bianchini, M. Maggini, & L. Jain (Eds.), *Handbook on neural information processing* (Vol. 49, pp. 215\u2013244). Springer. https://doi.org/10.1007/978-3-642-36657-4_7
- Jalil Al-Bayati, A., Rener, A. T., Listello, M. P., & Mohamed, M. (2023). PPE non-compliance among construction workers: An assessment of contributing factors utilizing fuzzy theory. In Journal of Safety Research (85), 242–253. https://doi.org/10.1016/j.jsr.2023.02.008
- Sanyal, S. (2022). Construction Site Safety Image Dataset Roboflow, Version 1. Retrieved November 2, 2024 from https://www.kaggle.com/datasets/snehilsanyal/construction-site-safety-image-dataset-roboflow/data
- OpenAI. (2024). *ChatGPT* (v2.0) [Large Language Model]. https://chat.openai.com/chat
- Ultralytics. (2024). *YOLO performance metrics: A guide*. Retrieved December 1, 2024, from https://docs.ultralytics.com/guides/yolo-performance-metrics/
- Verduzco, D. (2024) Construction PPE Personal Videos, OneDrive. Retrieved November 25, 2024 from https://uofsandiego-my.sharepoint.com/:f:/g/personal/dverduzco_sandiego_edu/EpWCwg-E6jZGltY-j_EcIUkBIgGk1fMxfnBQKqBTSfaVTg?e=42Zkn7
- *Workers at construction sites wearing PPE*. (n.d.). Retrieved November 25, 2024, from Google Images: https://www.google.com

Train from Custom Dataset

December 8, 2024

1 Construction PPE Object Detection:

1.0.1 Train From Custom Dataset

AAI-501 Group 4: Fatimat Atanda, Victor Hugo Germano, Darin Verduzco GitHub: https://github.com/victorhg/aai-501-final-project

2 Dataset setup:

download custom "datasets" folder into "train_from_custom" folder (will be ignored to git). Dataset Summary: construction_ppe: 2801 images custom dataset: 58 images totals: (2859 raw_images and 2859 raw_labels) (note: file names larger than 99 characters may not be read, update manually)

2.0.1 Import Modules and Libraries

[5]:
```
import os
import sys
import cv2 # Opencv for images
import time
import glob
import yaml
import torch
import shutil
import random
import warnings
import numpy as np
import pandas as pd
from PIL import Image
from pathlib import Path
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import albumentations as A # For image augmentation
from ultralytics import YOLO # For obeject detection
from albumentations.pytorch import ToTensorV2 # For image formating
from torchvision.transforms import ColorJitter
```

[6]:
```
warnings.filterwarnings("ignore") # Suppress all warnings
# Import msaai from parent folder
module_path = os.path.abspath(os.path.join('..'))
if module_path not in sys.path:
    sys.path.append(module_path+"/")
import msaai
```

2.0.2 Set path directories

[5]:
```
dataset_base_dir = "datasets"
image_base_dir = dataset_base_dir + "/images"
labels_base_dir = dataset_base_dir + "/labels"

# Define directories to reset
image_dirs = {
    "train": f"{image_base_dir}/train",
    "valid": f"{image_base_dir}/valid",
    "test": f"{image_base_dir}/test",
}
label_dirs = {
    "train": f"{labels_base_dir}/train",
    "valid": f"{labels_base_dir}/valid",
    "test": f"{labels_base_dir}/test",
}
```

2.0.3 Custom_data.yaml file creation in dataset dir

[31]:
```
ppe_classes = [
    'Hardhat',
    'Mask',
    'NO-Hardhat',
    'NO-Mask',
    'NO-Safety Vest',
    'Person',
    'Safety Cone',
    'Safety Vest',
    'Machinery',
    'Vehicle'
]
number_classes = len(ppe_classes)
output_dir = 'datasets'

dict_file = {
    'train': 'images/train',
    'val': 'images/valid',
    'test': 'images/test',
    'nc': number_classes,
```

```
    'names': ppe_classes
}

with open(os.path.join(dataset_base_dir, 'custom_data.yaml'), 'w+') as file:
    yaml.dump(dict_file, file)
# Full yaml path to prevent path error
yaml_path = str(Path('datasets/custom_data.yaml').resolve())
```

2.0.4 Clear train/test/valid image folders from "datasets" folder

[23]:
```
directories_to_reset = list(image_dirs.values()) + list(label_dirs.values())

# Reset directories
msaai.reset_directories(directories_to_reset)

print("Directories reset and ready for use.")
```

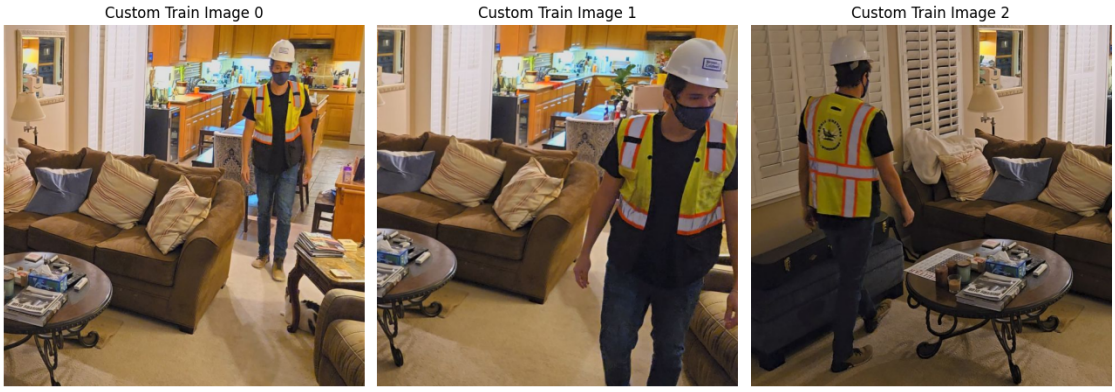
```
Deleted existing directory: datasets/images/train
Recreated directory: datasets/images/train
Deleted existing directory: datasets/images/valid
Recreated directory: datasets/images/valid
Deleted existing directory: datasets/images/test
Recreated directory: datasets/images/test
Deleted existing directory: datasets/labels/train
Recreated directory: datasets/labels/train
Deleted existing directory: datasets/labels/valid
Recreated directory: datasets/labels/valid
Deleted existing directory: datasets/labels/test
Recreated directory: datasets/labels/test
Directories reset and ready for use.
```

2.0.5 Split into train/valid/test folders from "raw_image" and "raw_label" folders

[24]:
```
source_image_dir = dataset_base_dir + "/raw_images"
source_label_dir = dataset_base_dir + "/raw_labels"

# Combine and assign to DataFrame
img_split_df = msaai.img_train_test_split(source_image_dir)

# Copy files from source to folders
msaai.arrange_image_and_label_files(
    img_split_df,
    source_image_dir,
    source_label_dir,
    image_dirs,
    label_dirs
)
```

2.0.6 Show custom image dataset

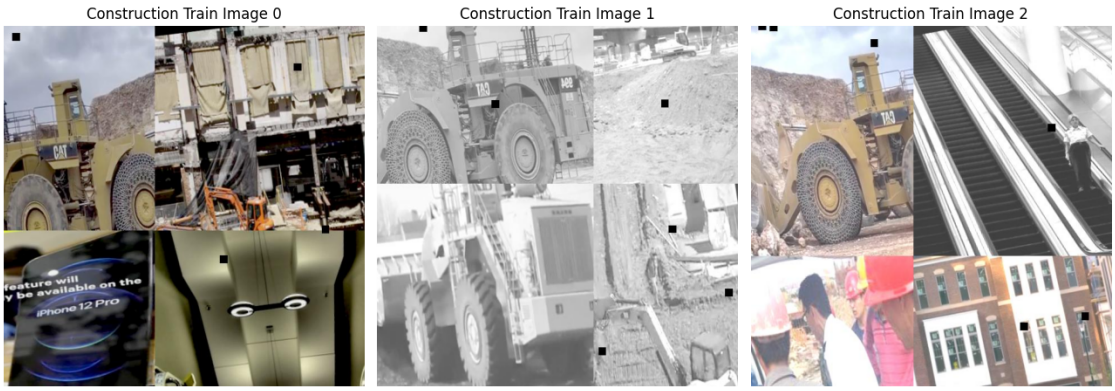
[22]:
```
# Get the paths to the first three custom images
image_paths = glob.glob(f"{image_dirs["train"]}/scene*.jpg")[:3]

# Check if there are enough images
if len(image_paths) < 3:
    print("Not enough images found!")
else:
    # Create a subplot with 1 row and 3 columns
    fig, axes = plt.subplots(1, 3, figsize=(13, 7))  # Width, height

    for i, ax in enumerate(axes):
        # Load and display each image
        image = mpimg.imread(image_paths[i])
        ax.imshow(image)
        ax.axis('off')  # Hide axes
        ax.set_title(f"Custom Train Image {i}")

    plt.tight_layout()
    plt.show()
```



2.0.7 Show construction dataset

[23]:
```
# Get the paths to the first three construction images
image_paths = glob.glob(f"{image_dirs["train"]}/construction*.jpg")[:3]

# Check if there are enough images
if len(image_paths) < 3:
    print("Not enough images found!")
else:
    # Create a subplot with 1 row and 3 columns
```

```
    fig, axes = plt.subplots(1, 3, figsize=(13, 7))  # Adjust the figsize as␣
 ↪needed

    for i, ax in enumerate(axes):
        # Load and display each image
        image = mpimg.imread(image_paths[i])
        ax.imshow(image)
        ax.axis('off')  # Hide axes
        ax.set_title(f"Construction Train Image {i}")

    plt.tight_layout()
    plt.show()
```



2.1 Train YOLOv8 from construction and custom dataset

[32]:
```
proj_name = input('Please enter run folder description: ')
# Count in minutes
start = time.time()
for i in range(10000000):
    pass

# Train
model = YOLO('../../models/yolov8m.pt')
model.train(
    data = yaml_path,
    epochs = 30, # Dataset iterations
    patience = 10, # Epochs to wait to discontinue after no val metric␣
 ↪improvement
    seed = 88, # Repeatability
    plots = True, # produce plots
    nms = True, # Non-Maximum Suppression (NMS) improves post-processing␣
 ↪detection
```

```
    batch = 0.7, # Determine batch size using defined fraction of memory usage
    name = proj_name # input for run description
)

# Time end
end = time.time()
elapsed_seconds = end - start
elapsed_minutes = elapsed_seconds / 60
print(f"Total model train time: {elapsed_minutes:.2f} minutes")
```

```
Please enter run folder description:  new m lower batch
```

```
Ultralytics 8.3.32  Python-3.12.4 torch-2.5.1+cpu CPU (Intel Core(TM)
i9-14900KF)
engine\trainer: task=detect, mode=train,
model=../../models/yolov8m.pt, data=C:\Users\DARiN\Documents\Python-
JL\Github\aai-501-final-
project\notebooks\train_from_custom\datasets\custom_data.yaml, epochs=30,
time=None, patience=10, batch=0.7, imgsz=640, save=True, save_period=-1,
cache=False, device=None, workers=8, project=None, name=new m lower batch,
exist_ok=False, pretrained=True, optimizer=auto, verbose=True, seed=88,
deterministic=True, single_cls=False, rect=False, cos_lr=False, close_mosaic=10,
resume=False, amp=True, fraction=1.0, profile=False, freeze=None,
multi_scale=False, overlap_mask=True, mask_ratio=4, dropout=0.0, val=True,
split=val, save_json=False, save_hybrid=False, conf=None, iou=0.7, max_det=300,
half=False, dnn=False, plots=True, source=None, vid_stride=1,
stream_buffer=False, visualize=False, augment=False, agnostic_nms=False,
classes=None, retina_masks=False, embed=None, show=False, save_frames=False,
save_txt=False, save_conf=False, save_crop=False, show_labels=True,
show_conf=True, show_boxes=True, line_width=None, format=torchscript,
keras=False, optimize=False, int8=False, dynamic=False, simplify=True,
opset=None, workspace=4, nms=True, lr0=0.01, lrf=0.01, momentum=0.937,
weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1,
box=7.5, cls=0.5, dfl=1.5, pose=12.0, kobj=1.0, label_smoothing=0.0, nbs=64,
hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1, scale=0.5,
shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, bgr=0.0, mosaic=1.0,
mixup=0.0, copy_paste=0.0, copy_paste_mode=flip, auto_augment=randaugment,
erasing=0.4, crop_fraction=1.0, cfg=None, tracker=botsort.yaml,
save_dir=runs\detect\new m lower batch
Overriding model.yaml nc=80 with nc=10

                   from  n    params  module
arguments
  0                  -1  1      1392  ultralytics.nn.modules.conv.Conv
[3, 48, 3, 2]
  1                  -1  1     41664  ultralytics.nn.modules.conv.Conv
[48, 96, 3, 2]
  2                  -1  2    111360  ultralytics.nn.modules.block.C2f
```

```
[96, 96, 2, True]
  3                  -1  1    166272  ultralytics.nn.modules.conv.Conv
[96, 192, 3, 2]
  4                  -1  4    813312  ultralytics.nn.modules.block.C2f
[192, 192, 4, True]
  5                  -1  1    664320  ultralytics.nn.modules.conv.Conv
[192, 384, 3, 2]
  6                  -1  4   3248640  ultralytics.nn.modules.block.C2f
[384, 384, 4, True]
  7                  -1  1   1991808  ultralytics.nn.modules.conv.Conv
[384, 576, 3, 2]
  8                  -1  2   3985920  ultralytics.nn.modules.block.C2f
[576, 576, 2, True]
  9                  -1  1    831168  ultralytics.nn.modules.block.SPPF
[576, 576, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample
[None, 2, 'nearest']
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat
[1]
 12                  -1  2   1993728  ultralytics.nn.modules.block.C2f
[960, 384, 2]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample
[None, 2, 'nearest']
 14             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat
[1]
 15                  -1  2    517632  ultralytics.nn.modules.block.C2f
[576, 192, 2]
 16                  -1  1    332160  ultralytics.nn.modules.conv.Conv
[192, 192, 3, 2]
 17            [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat
[1]
 18                  -1  2   1846272  ultralytics.nn.modules.block.C2f
[576, 384, 2]
 19                  -1  1   1327872  ultralytics.nn.modules.conv.Conv
[384, 384, 3, 2]
 20             [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat
[1]
 21                  -1  2   4207104  ultralytics.nn.modules.block.C2f
[960, 576, 2]
 22        [15, 18, 21]  1   3781486  ultralytics.nn.modules.head.Detect
[10, [192, 384, 576]]
Model summary: 295 layers, 25,862,110 parameters, 25,862,094 gradients, 79.1
GFLOPs

Transferred 469/475 items from pretrained weights
TensorBoard: Start with 'tensorboard --logdir runs\detect\new m
lower batch', view at http://localhost:6006/
Freezing layer 'model.22.dfl.conv.weight'
```

```
train: Scanning C:\Users\DARiN\Documents\Python-
JL\Github\aai-501-final-project\notebooks\train_from_custom\datasets\la

train: WARNING  C:\Users\DARiN\Documents\Python-
JL\Github\aai-501-final-project\notebooks\train_from_custom\datasets\images\trai
n\004720_jpg.rf.afc486560a4004c7cfd67910af31a29c.jpg: 1 duplicate labels removed
train: WARNING  C:\Users\DARiN\Documents\Python-
JL\Github\aai-501-final-project\notebooks\train_from_custom\datasets\images\trai
n\construction-813-_jpg.rf.b085952261fd98f2e76b8065de149b5f.jpg: 1 duplicate
labels removed
train: New cache created: C:\Users\DARiN\Documents\Python-
JL\Github\aai-501-final-
project\notebooks\train_from_custom\datasets\labels\train.cache
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01,
blur_limit=(3, 7)), ToGray(p=0.01, num_output_channels=3,
method='weighted_average'), CLAHE(p=0.01, clip_limit=(1.0, 4.0),
tile_grid_size=(8, 8))
AutoBatch: Computing optimal batch size for imgsz=640 at 70.0% CUDA
memory utilization.
AutoBatch:   intended for CUDA devices, using default batch-size 16

train: Scanning C:\Users\DARiN\Documents\Python-
JL\Github\aai-501-final-project\notebooks\train_from_custom\datasets\la

train: WARNING  C:\Users\DARiN\Documents\Python-
JL\Github\aai-501-final-project\notebooks\train_from_custom\datasets\images\trai
n\004720_jpg.rf.afc486560a4004c7cfd67910af31a29c.jpg: 1 duplicate labels removed
train: WARNING  C:\Users\DARiN\Documents\Python-
JL\Github\aai-501-final-project\notebooks\train_from_custom\datasets\images\trai
n\construction-813-_jpg.rf.b085952261fd98f2e76b8065de149b5f.jpg: 1 duplicate
labels removed
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01,
blur_limit=(3, 7)), ToGray(p=0.01, num_output_channels=3,
method='weighted_average'), CLAHE(p=0.01, clip_limit=(1.0, 4.0),
tile_grid_size=(8, 8))


val: Scanning C:\Users\DARiN\Documents\Python-
JL\Github\aai-501-final-project\notebooks\train_from_custom\datasets\labe

val: New cache created: C:\Users\DARiN\Documents\Python-
JL\Github\aai-501-final-
project\notebooks\train_from_custom\datasets\labels\valid.cache


Plotting labels to runs\detect\new m lower batch\labels.jpg…
optimizer: 'optimizer=auto' found, ignoring 'lr0=0.01' and
'momentum=0.937' and determining best 'optimizer', 'lr0' and 'momentum'
automatically…
optimizer: AdamW(lr=0.000714, momentum=0.9) with parameter groups
```

```
77 weight(decay=0.0), 84 weight(decay=0.0005), 83 bias(decay=0.0)
TensorBoard: model graph visualization added
Image sizes 640 train, 640 val
Using 0 dataloader workers
Logging results to runs\detect\new m lower batch
Starting training for 30 epochs…

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       1/30         0G      1.299      1.966      1.479        262        640:
100%|      | 125/125 [16:54<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:46
                   all        569       7670      0.538      0.452      0.448
0.24


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       2/30         0G      1.239      1.515      1.428        275        640:
100%|      | 125/125 [16:04<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:51
                   all        569       7670      0.592      0.416      0.436
0.221


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       3/30         0G      1.253      1.483      1.422        231        640:
100%|      | 125/125 [16:20<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:51
                   all        569       7670      0.625      0.446      0.487
0.26


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       4/30         0G      1.242      1.444      1.415        326        640:
100%|      | 125/125 [15:45<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:48
                   all        569       7670      0.656      0.496      0.537
0.304
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       5/30         0G      1.202      1.346      1.382        331        640:
100%|      | 125/125 [15:11<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:44
                   all        569       7670      0.673      0.501      0.539
0.308
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       6/30         0G      1.173      1.287      1.366        275        640:
100%|      | 125/125 [15:18<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:45
                   all        569       7670      0.736      0.522      0.592
0.343
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       7/30         0G       1.15      1.233      1.349        255        640:
100%|      | 125/125 [15:12<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:46
                   all        569       7670      0.761      0.566      0.636
0.373
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       8/30         0G      1.106      1.158      1.327        325        640:
100%|      | 125/125 [15:16<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:49
                   all        569       7670      0.773      0.571       0.65
0.382
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
```

```
       9/30         0G      1.089      1.097      1.304        244        640:
100%|      | 125/125 [16:22<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:56
                   all        569       7670      0.782      0.594      0.668
0.402


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      10/30         0G      1.067      1.056      1.283        424        640:
100%|      | 125/125 [17:21<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:50
                   all        569       7670      0.805      0.591      0.679
0.411


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      11/30         0G      1.037      1.023      1.267        274        640:
100%|      | 125/125 [16:16<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:49
                   all        569       7670      0.818      0.622      0.705
0.434


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      12/30         0G      1.022     0.9724      1.257        376        640:
100%|      | 125/125 [16:11<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:51
                   all        569       7670      0.828       0.63      0.712
0.441


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      13/30         0G     0.9995     0.9381      1.239        556        640:
100%|      | 125/125 [16:14<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:52
```

```
                 all        569       7670      0.827       0.64      0.723
0.457

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      14/30         0G     0.9771      0.899      1.225        276        640:
100%|      | 125/125 [16:13<00:00,
                Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:51
                 all        569       7670      0.828      0.652       0.73
0.462

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      15/30         0G     0.9546     0.8733      1.216        308        640:
100%|      | 125/125 [16:33<00:00,
                Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:54
                 all        569       7670       0.85      0.666      0.744
0.476

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      16/30         0G     0.9424     0.8487      1.203        308        640:
100%|      | 125/125 [16:52<00:00,
                Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:51
                 all        569       7670      0.855       0.66      0.745
0.481

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      17/30         0G     0.9331     0.8252      1.195        292        640:
100%|      | 125/125 [16:32<00:00,
                Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:50
                 all        569       7670      0.869      0.665      0.755
0.488
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      18/30         0G     0.9093     0.7923      1.184        275        640:
100%|      | 125/125 [16:46<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:50
                   all        569       7670       0.85      0.684      0.764
0.5
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      19/30         0G     0.9053     0.7639       1.17        257        640:
100%|      | 125/125 [16:32<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:54
                   all        569       7670      0.866      0.691      0.772
0.508
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      20/30         0G     0.8683     0.7246      1.155        349        640:
100%|      | 125/125 [15:01<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:47
                   all        569       7670      0.877      0.698      0.783
0.523
```

```
Closing dataloader mosaic
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01,
blur_limit=(3, 7)), ToGray(p=0.01, num_output_channels=3,
method='weighted_average'), CLAHE(p=0.01, clip_limit=(1.0, 4.0),
tile_grid_size=(8, 8))
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      21/30         0G     0.8941     0.6708      1.167        195        640:
100%|      | 125/125 [14:52<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:45
                   all        569       7670      0.864      0.706      0.783
0.514
```

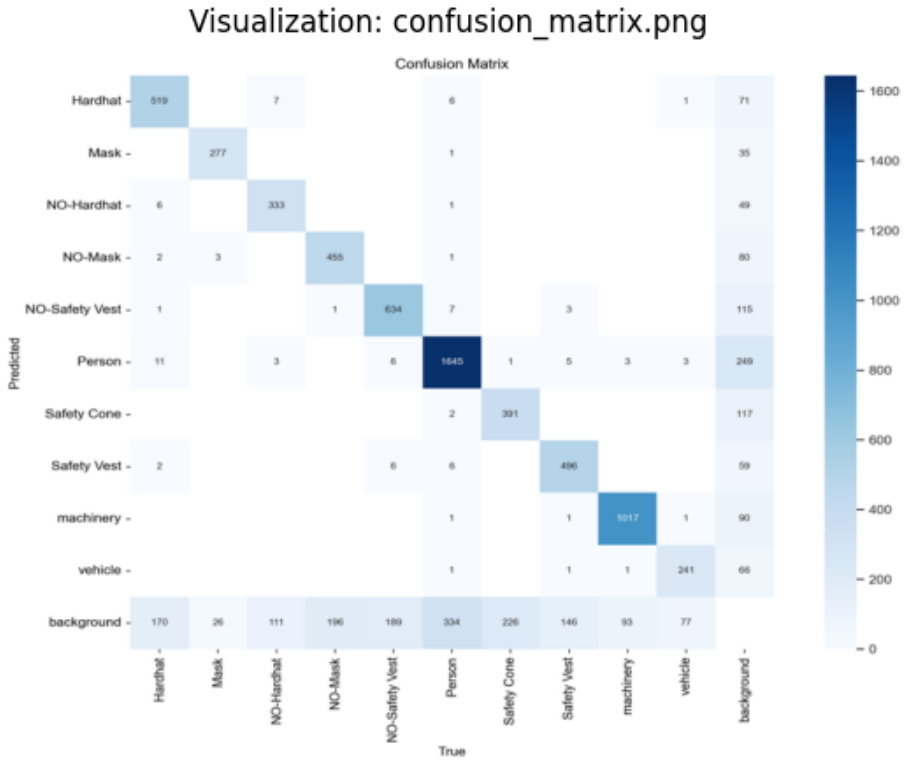
```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      22/30         0G      0.869     0.6296      1.152        221        640:
100%|      | 125/125 [14:51<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:46
                   all        569       7670      0.875      0.718      0.796
0.538

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      23/30         0G     0.8431     0.5999      1.133        173        640:
100%|      | 125/125 [14:50<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:45
                   all        569       7670      0.883       0.72      0.796
0.543

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      24/30         0G      0.813     0.5655      1.108        225        640:
100%|      | 125/125 [14:53<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:44
                   all        569       7670      0.879       0.73      0.806
0.551

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      25/30         0G     0.7991     0.5535      1.107        213        640:
100%|      | 125/125 [14:51<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:43
                   all        569       7670      0.893      0.735      0.812
0.557

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
```

```
      26/30         0G     0.7768     0.5278       1.09        162        640:
100%|      | 125/125 [14:54<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:44
                   all        569       7670      0.891      0.737      0.815
0.563


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      27/30         0G     0.7561     0.5144      1.078        208        640:
100%|      | 125/125 [14:53<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:45
                   all        569       7670      0.892       0.74      0.817
0.571


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      28/30         0G     0.7305     0.4915      1.058        184        640:
100%|      | 125/125 [14:54<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:45
                   all        569       7670      0.897       0.75      0.819
0.574


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      29/30         0G     0.7129     0.4776      1.052        225        640:
100%|      | 125/125 [14:51<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:43
                   all        569       7670        0.9      0.748      0.823
0.579


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      30/30         0G     0.6986     0.4668      1.047        114        640:
100%|      | 125/125 [14:54<00:00,
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 18/18 [01:43
```

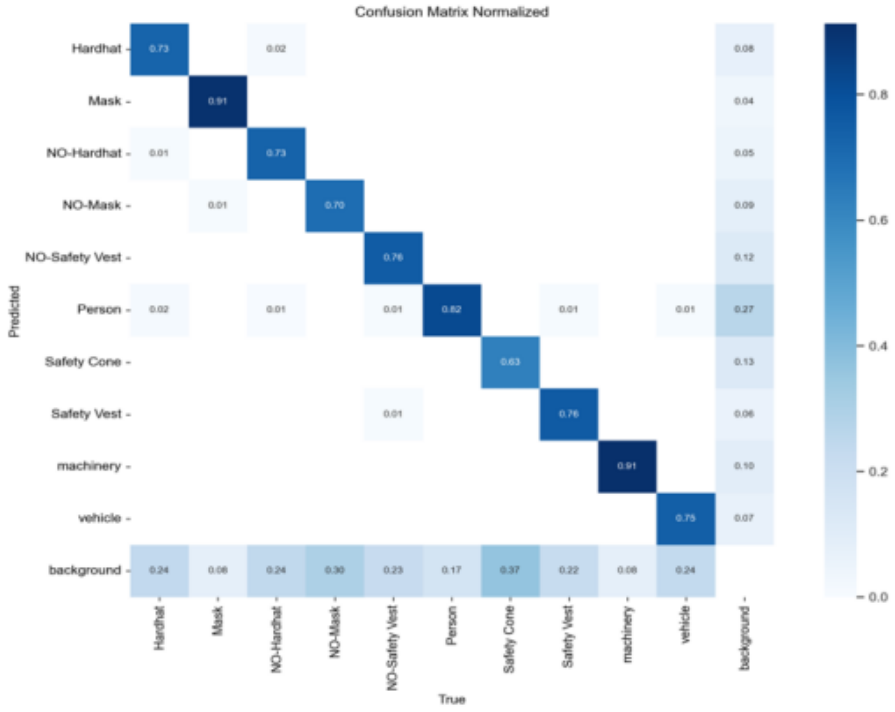
```
                   all        569       7670      0.904      0.752      0.824
0.581



30 epochs completed in 8.776 hours.
Optimizer stripped from runs\detect\new m lower batch\weights\last.pt, 52.0MB
Optimizer stripped from runs\detect\new m lower batch\weights\best.pt, 52.0MB

Validating runs\detect\new m lower batch\weights\best.pt…
Ultralytics 8.3.32  Python-3.12.4 torch-2.5.1+cpu CPU (Intel Core(TM)
i9-14900KF)
Model summary (fused): 218 layers, 25,845,550 parameters, 0 gradients, 78.7
GFLOPs

                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|     | 18/18 [01:29

                   all        569       7670      0.904      0.752      0.824
0.581
               Hardhat        271        711      0.918      0.724      0.807
0.529
                  Mask        225        306      0.933      0.911       0.94
0.743
            NO-Hardhat        283        454       0.93      0.702      0.806
0.529
               NO-Mask        320        652      0.875      0.657      0.722
0.399
        NO-Safety Vest        399        835      0.908      0.762      0.834
0.588
                Person        542       2005      0.923      0.812      0.892
0.681
           Safety Cone        125        618      0.835      0.563      0.656
0.349
           Safety Vest        273        652      0.918      0.759      0.834
0.601
             machinery        440       1114       0.94        0.9      0.945
0.805
                vehicle        166        323      0.862      0.733      0.802
0.591
Speed: 0.9ms preprocess, 152.0ms inference, 0.0ms loss, 0.3ms postprocess per
image
Results saved to runs\detect\new m lower batch
Total model train time: 528.16 minutes
```

3 Model Performance Metrics

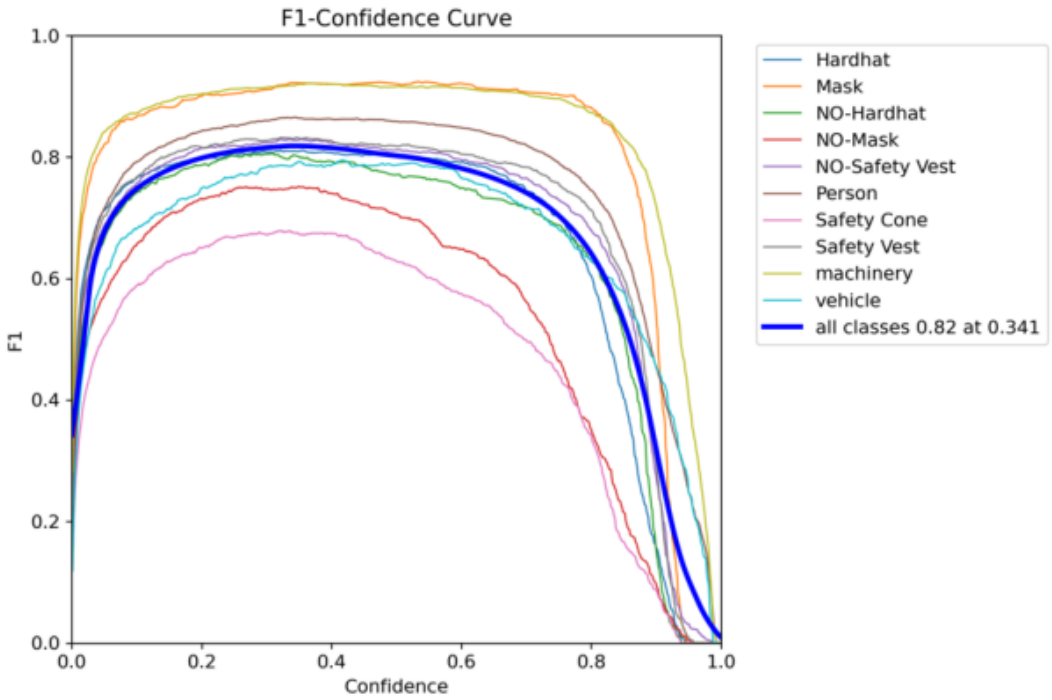
[7]:
```
run_path = "runs/detect/train"
# Loop through images, do not include images for val and train
img_files = [
    f for f in os.listdir(run_path)
    if f.endswith((".png", ".jpg")) and not f.startswith(("train", "val"))
]

# Loop through each image file and display it
for img_file in img_files:
    file_path = os.path.join(run_path, img_file)
    image = Image.open(file_path).resize((900, 600)) # width, height
    # Display the image
    plt.figure(figsize=(8, 6))
    plt.imshow(image)
    plt.axis("off")  # Hide axes
    plt.title(f"Visualization: {img_file}")
    plt.show()
```



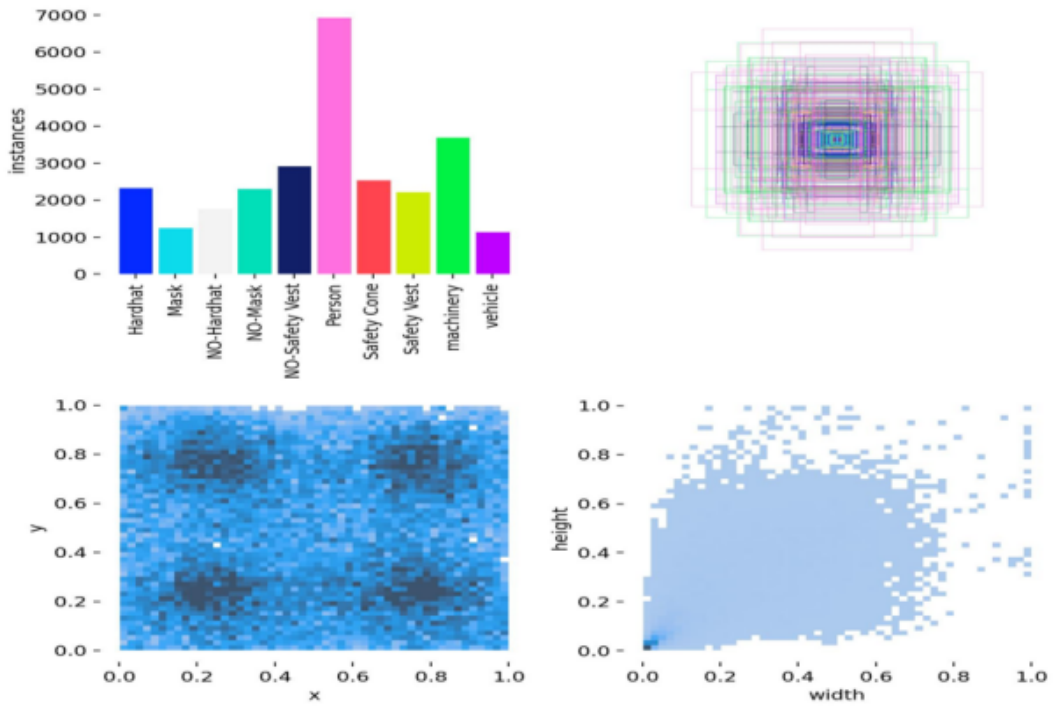
Visualization: confusion_matrix_normalized.png



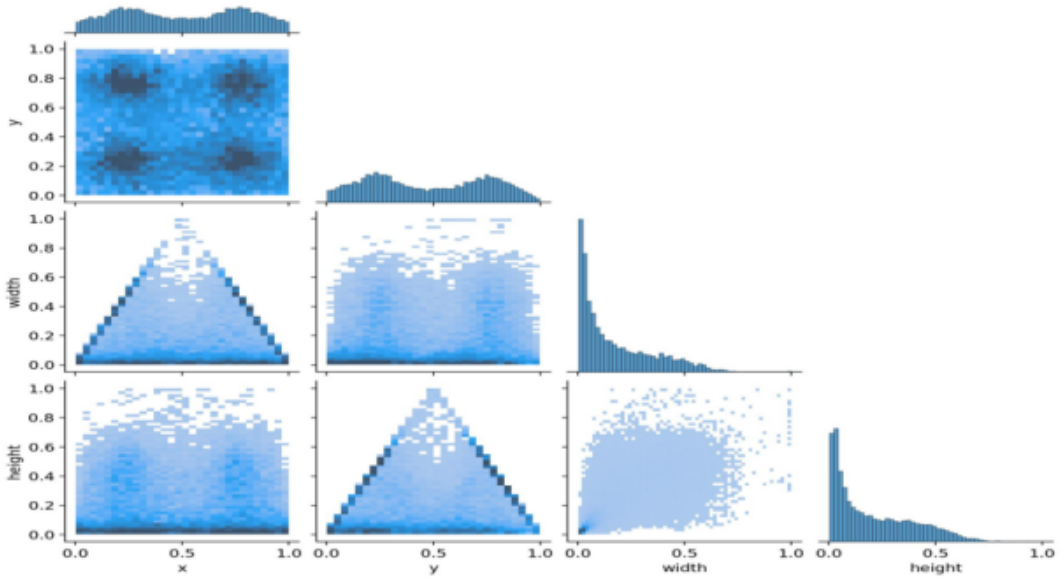
Visualization: F1_curve.png



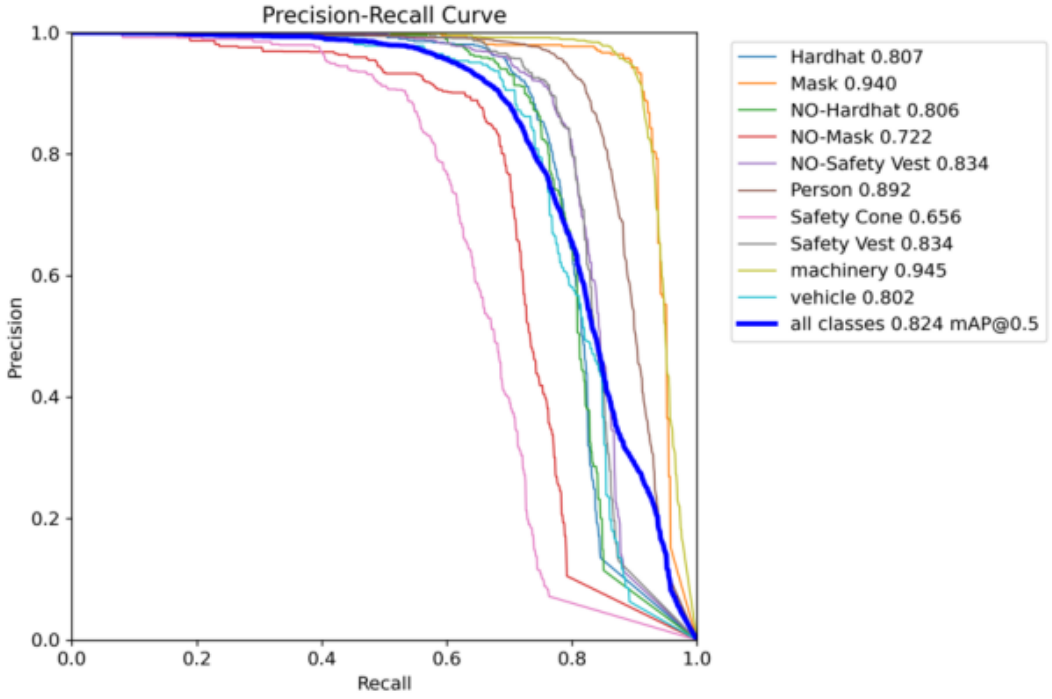
Visualization: labels.jpg



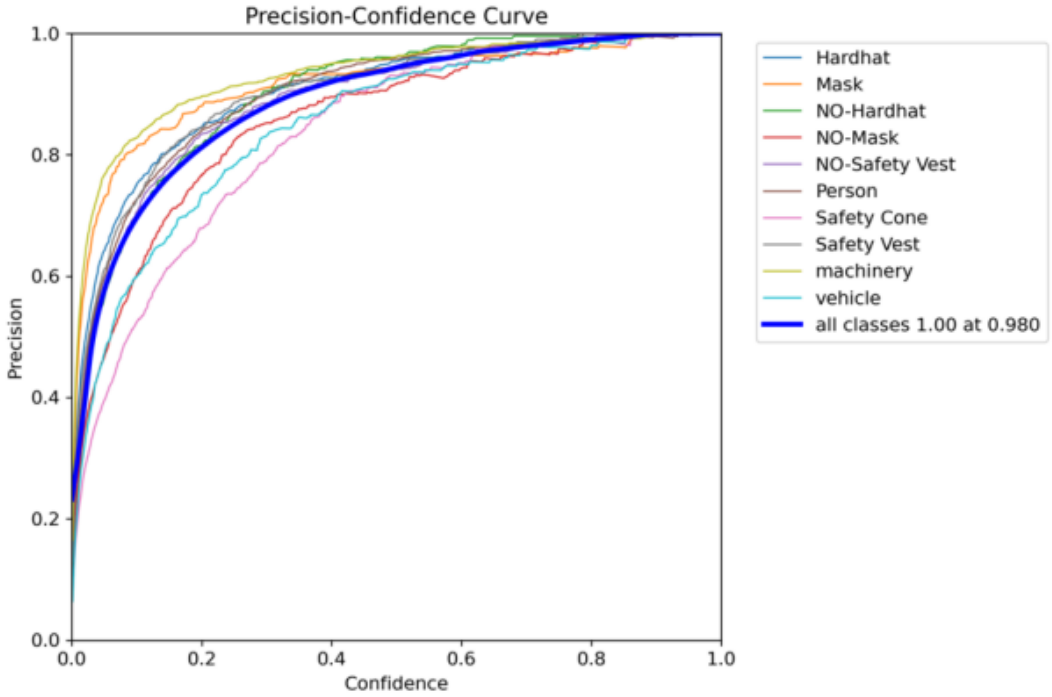
Visualization: labels_correlogram.jpg



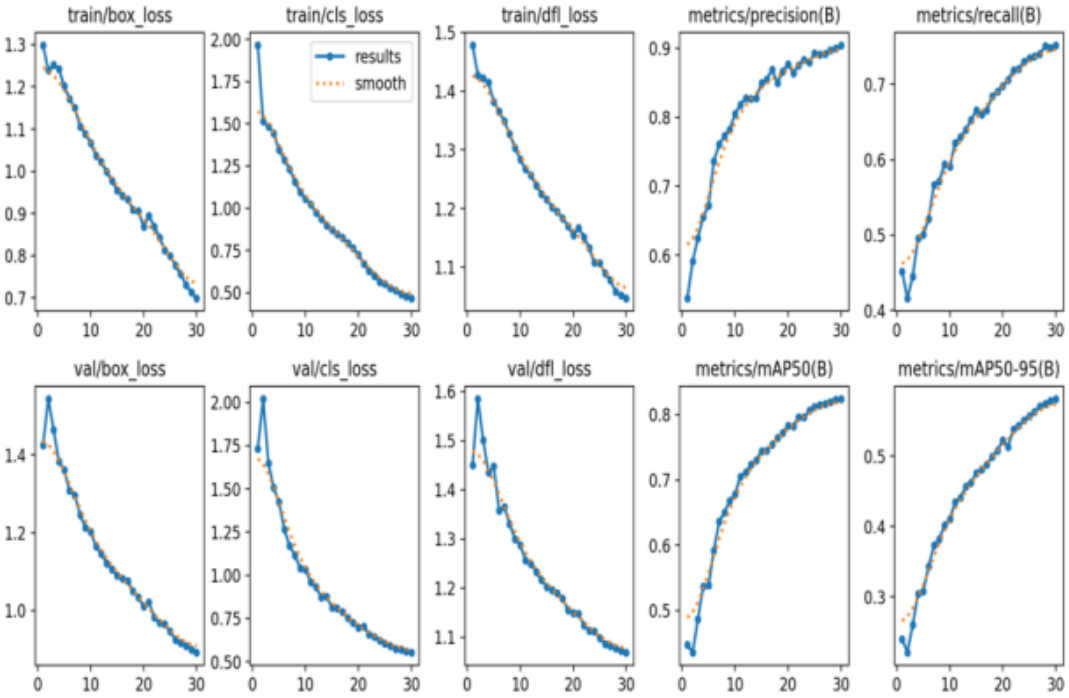
Visualization: PR_curve.png



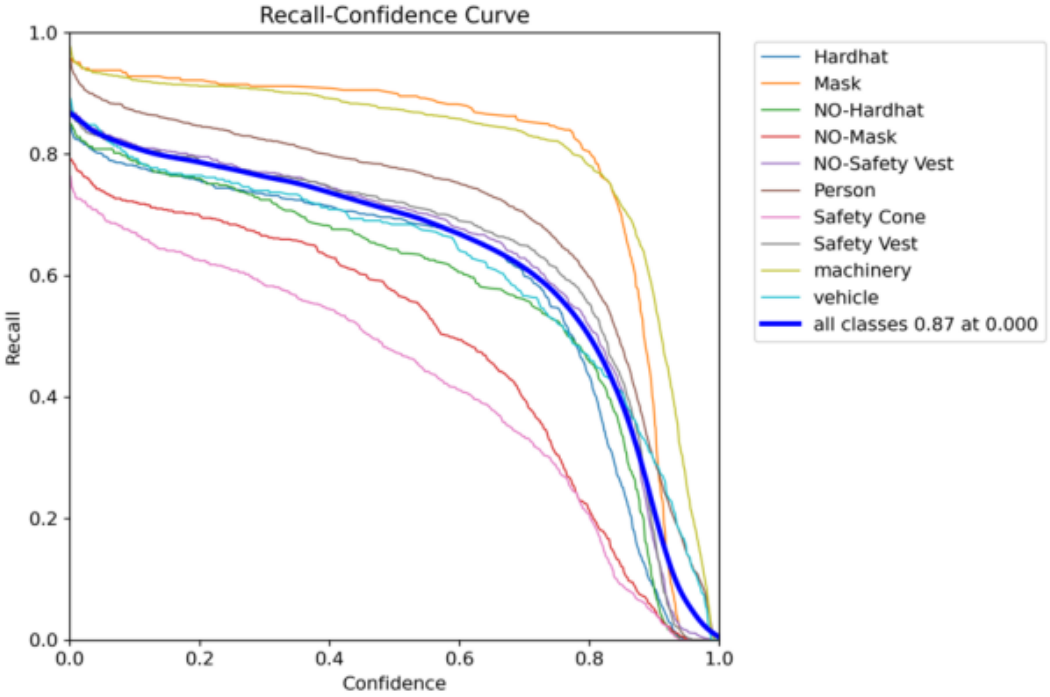
Visualization: P_curve.png



Visualization: results.png

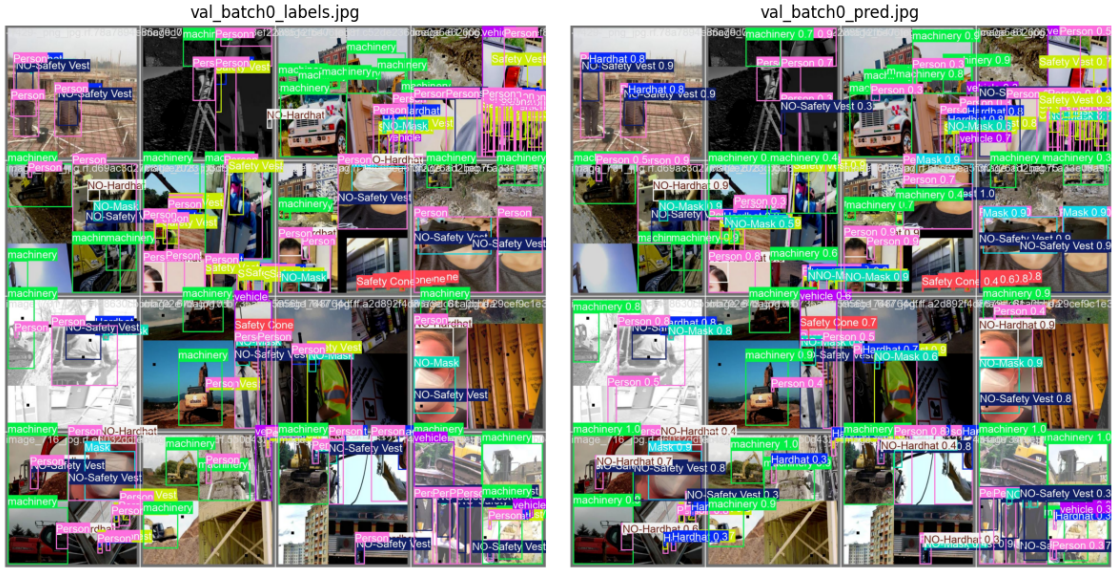


Visualization: R_curve.png



3.0.1 Show validation image labels vs predictions

[28]:
```
# Get val label and pred image
pred_image_path = glob.glob(os.path.join(run_path, "val_batch0*pred.jpg"))
labels_image_path = glob.glob(os.path.join(run_path, "val_batch0*labels.jpg"))

# Assign first matching image if found
pred_image_path = pred_image_path[0] if pred_image_path else None
labels_image_path = labels_image_path[0] if labels_image_path else None

# Error if not found
if not pred_image_path or not labels_image_path:
    print("Not enough images found!")
else:
    # Plot the images side by side
    fig, axes = plt.subplots(1, 2, figsize=(12, 6))  # 1 row, 2 columns
    for ax, image_path in zip(axes, [labels_image_path, pred_image_path]):
        image = mpimg.imread(image_path)
        ax.imshow(image)
        ax.axis('off')
        ax.set_title(os.path.basename(image_path))  # Title is the image name

    plt.tight_layout()
    plt.show()
```



3.0.2 Construction and Custom test images

[36]:
```
# Get the paths to the first three "construction" and "scene" test images
construction_images = glob.glob(f"{image_dirs['test']}/auto*.jpg")[:3]
scene_images = glob.glob(f"{image_dirs['test']}/scene*.jpg")[:3]

# Combine the image paths
image_paths = construction_images + scene_images

# Check if there are enough images
if len(image_paths) < 6:
    print("Not enough images found!")
else:
    # Create a subplot with 2 rows and 3 columns
    fig, axes = plt.subplots(2, 3, figsize=(11, 8))  # height, width

    for i, ax in enumerate(axes.flatten()):
        # Load and display each image
        image = mpimg.imread(image_paths[i])
        ax.imshow(image)
        ax.axis('off')  # Hide axes
        if i < 3:
            ax.set_title(f"Construction Image {i + 1}")  # Titles for␣
 ↪construction images
        else:
            ax.set_title(f"Custom Image {i - 2}")  # Titles for scene images

    plt.tight_layout()
    plt.show()
```

3.0.3 Predict test images with fine tuned model (best.pt)

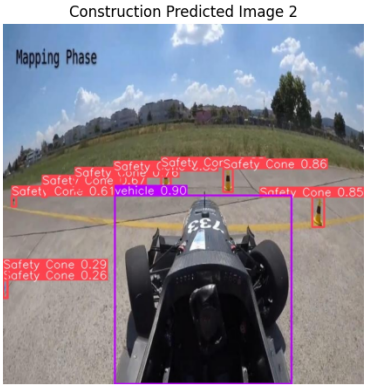
[37]:
```
# Remove previous predictions
detect_path = "runs/detect/"
for folder in os.listdir(detect_path):
    folder_path = os.path.join(detect_path, folder)
    if os.path.isdir(folder_path) and folder.startswith("predict"):
        shutil.rmtree(folder_path, ignore_errors=True)

# Load fine tuned model
model = YOLO("runs/detect/train/weights/best.pt")

# Make predictions on "scene" and "construction" images
for prefix in ["scene", "auto"]:
    model.predict(
        f"{image_dirs['test']}/{prefix}*.jpg",
        save=True,         # Enable saving predictions
        conf=0.25,         # Minimum confidence threshold
        agnostic_nms=True,  # Merge overlapping boxes across classes
        iou=0.7,           # Intersection over Union threshold
        verbose=False      # Silence prediction output
    )
```

```python
# Get the paths to the first three "scene" and "construction" predicted images
scene_predicted_paths = glob.glob("runs/detect/predict/scene*.jpg")[:3]
construction_predicted_paths = glob.glob("runs/detect/predict/auto*.jpg")[:3]

# Combine the image paths, construction on top row
all_predicted_paths =  construction_predicted_paths + scene_predicted_paths

# Check if there are enough predicted images
if len(all_predicted_paths) < 6:
    print("Not enough predictions found!")
else:
    # Create a subplot with 2 rows and 3 columns
    fig, axes = plt.subplots(2, 3, figsize=(13, 10)) # Width, height

    for i, ax in enumerate(axes.flatten()):
        # Load and display each predicted image
        image = mpimg.imread(all_predicted_paths[i])
        ax.imshow(image)
        ax.axis('off')  # Hide axes
        # Set titles based on the type of image
        if i < 3:
            ax.set_title(f"Construction Predicted Image {i + 1}")
        else:
            ax.set_title(f"Custom Predicted Image {i - 2}")

    plt.tight_layout()
    plt.show()
```

```
Results saved to runs\detect\predict
Results saved to runs\detect\predict
```

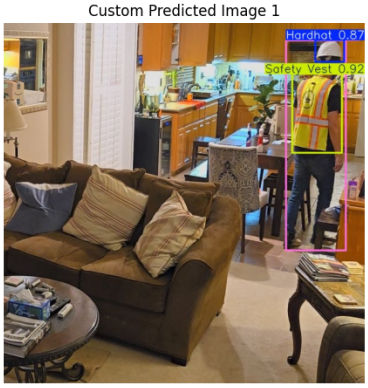
Construction Predicted Image 1

Construction Predicted Image 2

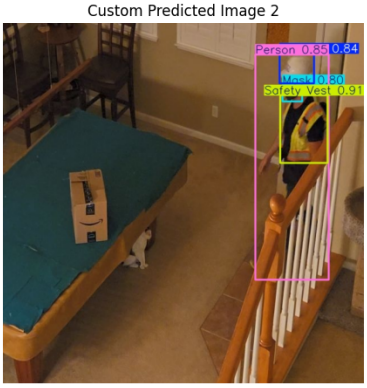


Construction Predicted Image 3

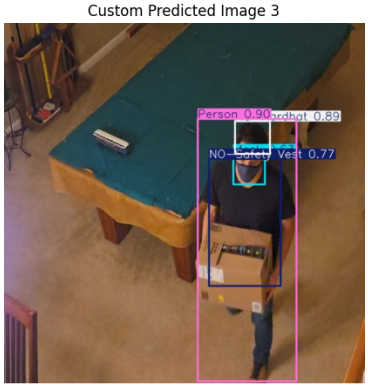
Custom Predicted Image 1



Custom Predicted Image 2



Custom Predicted Image 3



~/Development/aai-501-final-project/notebooks/msaai.py

```python
# import modules
import os  # file
import cv2
import shutil
import random
from numpy import split
import pandas as pd


# Function to get class colors
def getColours(cls_num):
    base_colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255)]
    color_index = cls_num % len(base_colors)
    increments = [(1, -2, 1), (-2, 1, -1), (1, -1, 2)]
    color = [
        base_colors[color_index][i]
        + increments[color_index][i] * (cls_num // len(base_colors)) % 256
        for i in range(3)
    ]
    return tuple(color)


def drawPredictionBox(img, class_name, box):
    # get coordinates
    [x1, y1, x2, y2] = box.xyxy[0]
    # convert to int
    x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
    vertex1 = (x1, y1)
    vertex2 = (x2, y2)

    confidence = box.conf[0]
    colour = getColours(int(box.cls[0]))

    # draw prediction retangle
    cv2.rectangle(img, vertex1, vertex2, colour, 2)

    # put the class name and confidence on the image
    cv2.putText(
        img,
        f"{class_name} {confidence:.2f}",
        vertex1,
        cv2.FONT_HERSHEY_SIMPLEX,
        1,
        colour,
        2,
    )
    return img


def drawPredictions(predictions, img, class_names, treshold=0.4):
    summary = ""
```

```
    for p in predictions:
        summary = p.verbose()
        boxes = p.boxes
        for box in boxes:
            confidence = box.conf[0]
            if confidence > treshold:
                cls = int(box.cls[0])  # get the class index
                # get the class name
                class_name = class_names[cls]
                drawPredictionBox(img, class_name, box)
    return summary


# create directories to organize images and cleanup for a new to avoid duplicate images
def reset_directories(directories):
    """
    Check if the specified directories exist. If they do, delete them and recreate them.
    Ensures the directories are clean before use.

    Parameters:
        directories (list): List of directories to reset.
    """
    for path in directories:
        if os.path.exists(path):
            # delete the directory and all its contents
            try:
                shutil.rmtree(path)
                print(f"Deleted existing directory: {path}")
            except Exception as e:
                print(f"Failed to delete {path}. Reason: {e}")

        # Recreate the directory
        try:
            os.makedirs(path, exist_ok=True)
            print(f"Recreated directory: {path}")
        except Exception as e:
            print(f"Failed to create directory {path}. Reason: {e}")


# SPLIT IMAGE DATASET AND FOLDERS
def arrange_image_and_label_files(
    df, source_image_dir, source_label_dir, image_dirs, label_dirs
):
    for _, row in df.iterrows():
        img_file = row["filename"]
        split = row["split"]

        # Source paths
        img_src = os.path.join(source_image_dir, img_file)
        label_src = os.path.join(
            source_label_dir, os.path.splitext(img_file)[0] + ".txt"
        )
```

```
104
105         # Destination paths
106         img_dest = os.path.join(image_dirs[split], img_file)
107         label_dest = os.path.join(
108             label_dirs[split], os.path.splitext(img_file)[0] + ".txt"
109         )
110
111         # Copy image file
112         if os.path.exists(img_src):
113             shutil.copy(img_src, img_dest)
114
115         # Copy corresponding label file
116         if os.path.exists(label_src):
117             shutil.copy(label_src, label_dest)
118
119
120 def img_train_test_split(source_image_dir):
121     """
122     Base on files in source_image_dir, split images following rule
123     Parameters:
124         source_image_dir (str): Original Images Directory to split
125     """
126     # Set random seed for reproducibility
127     random.seed(42)
128
129     # extract the image files
130     image_files = [
131         f for f in os.listdir(source_image_dir) if f.endswith((".jpg", ".jpeg",
    ".png"))
132     ]
133     random.shuffle(image_files)
134
135     # Create DataFrame with file paths and dataset split assignments
136     split_files = pd.DataFrame({"filename": image_files})
137
138     # dynamically split dataset into train, valid, and test
139     # split sizes
140     train_size = int(0.7 * len(split_files))  # 70% for training
141     valid_size = int(0.2 * len(split_files))  # 20% for validation
142     test_size = len(split_files) - train_size - valid_size  # remaining 10% for
    testing
143
144     # split labels
145     train_labels = ["train"] * train_size
146     valid_labels = ["valid"] * valid_size
147     test_labels = ["test"] * test_size
148
149     split_files["split"] = train_labels + valid_labels + test_labels
150     return split_files
151
152
153 def setup_image_directories(base_dir):
154     return
155
```

hybrid_model_train

December 8, 2024

1 Construction PPE Object Detection:

1.0.1 Hybrid Model Training

AAI-501 Group 4: Fatimat Atanda, Victor Hugo Germano, Darin Verduzco GitHub: https://github.com/victorhg/aai-501-final-project

2 Dataset setup:

Download hybrid "datasets" folder into the current "hybrid_model_train" directory: Manually-annotated datasets: For images annotated manually ./datasets/manually_annotated/raw_images (178 images) ./datasets/manually_annotated/raw_labels (178 labels) Auto-annotated datasets: For images annotated with pre-trained model - Raw Images Datasets: Contains images not annotated ./datasets/raw_images (2746 images) (note: file names larger than 99 characters may not be read, update manually)

2.0.1 Import Modules and Libraries

[55]:
```
import os # File
import cv2 # Opencv for images
import yaml
import glob
import torch
import shutil
import random
import numpy as np
import pandas as pd
from PIL import Image
import seaborn as sns
from pathlib import Path
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
from ultralytics import YOLO # For obeject detection
import albumentations as A # For image augmentation
from albumentations.pytorch import ToTensorV2 # For image formating
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_recall_curve
# Suppress all warnings
```

```
import warnings
warnings.filterwarnings("ignore")
```

2.0.2 Create directories

[56]:
```
# Create direcories to organize images and cleanup for to avoid duplicate␣
 ↪images
dataset_base_dir = "datasets/"
# Source directory containing all raw manually annotated images
manual_source_images_dir = os.path.join(dataset_base_dir, "manually_annotated/
 ↪raw_images")
manual_source_labels_dir = os.path.join(dataset_base_dir, "manually_annotated/
 ↪raw_labels")
# Source directory containing all raw unannotated images
raw_images_dir = os.path.join(dataset_base_dir, "raw_images")

# Create auto-annotated directories
auto_annotated_images_dir = os.path.join(dataset_base_dir, "auto_annotated/
 ↪images")
auto_annotated_labels_dir = os.path.join(dataset_base_dir, "auto_annotated/
 ↪labels")
# Create manually annotated directories
manual_train_images_dir = os.path.join(dataset_base_dir, "manually_annotated/
 ↪images/train")
manual_train_labels_dir = os.path.join(dataset_base_dir, "manually_annotated/
 ↪labels/train")
manual_val_images_dir = os.path.join(dataset_base_dir, "manually_annotated/
 ↪images/val")
manual_val_labels_dir = os.path.join(dataset_base_dir, "manually_annotated/
 ↪labels/val")
```

2.0.3 Reset directories

[57]:
```
def reset_directories(directories):
    """
    Check if directories exist. Delete and recreate them if they do.
    Ensures the directories are clean before use.

    Parameters:
        directories (list): List of directories to reset.
    """
    for path in directories:
        if os.path.exists(path):
            # delete the directory and all its contents
            try:
                shutil.rmtree(path)
                print(f"Deleted existing directory: {path}")
```

```
            except Exception as e:
                print(f"Failed to delete {path}. Reason: {e}")

        # Recreate the directory
        try:
            os.makedirs(path, exist_ok=True)
            print(f"Recreated directory: {path}")
        except Exception as e:
            print(f"Failed to create directory {path}. Reason: {e}")

# Define directories to reset
directories_to_reset = [
    auto_annotated_images_dir,
    auto_annotated_labels_dir,


    # manually annotated directories
    manual_train_images_dir,
    manual_train_labels_dir,
    manual_val_images_dir,
    manual_val_labels_dir
]

# Reset directories
reset_directories(directories_to_reset)
print("Directories reset and ready for use.")

# Set random seed for reproducibility
random.seed(42)
```

```
Deleted existing directory: datasets/auto_annotated/images
Recreated directory: datasets/auto_annotated/images
Deleted existing directory: datasets/auto_annotated/labels
Recreated directory: datasets/auto_annotated/labels
Deleted existing directory: datasets/manually_annotated/images/train
Recreated directory: datasets/manually_annotated/images/train
Deleted existing directory: datasets/manually_annotated/labels/train
Recreated directory: datasets/manually_annotated/labels/train
Deleted existing directory: datasets/manually_annotated/images/val
Recreated directory: datasets/manually_annotated/images/val
Deleted existing directory: datasets/manually_annotated/labels/val
Recreated directory: datasets/manually_annotated/labels/val
Directories reset and ready for use.
```

2.0.4 Create manual_data.yaml

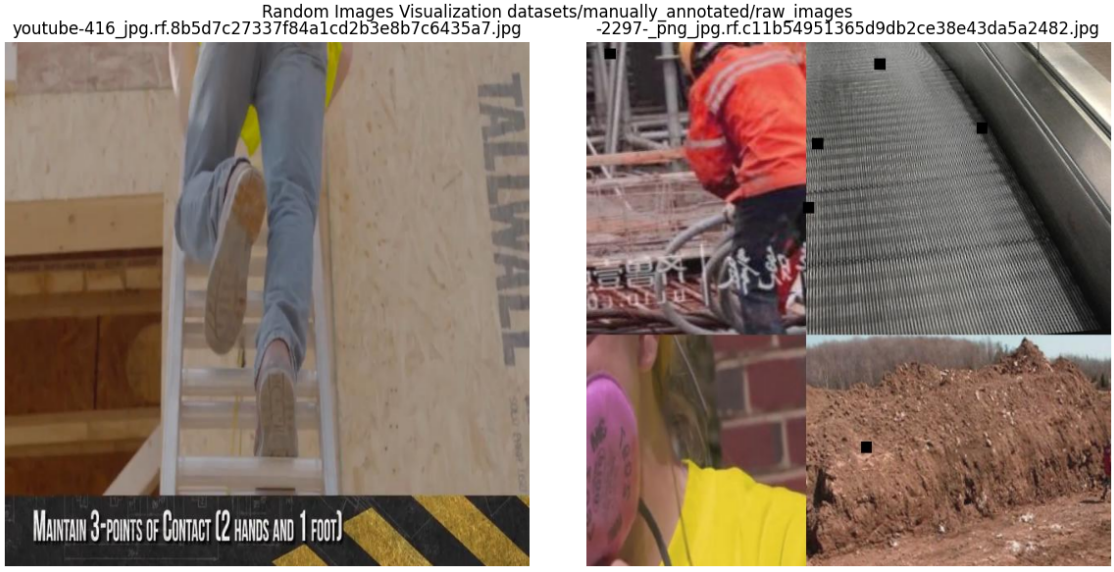
[58]:
```
# create manual_data.yaml in local datasets folder
ppe_classes = [
    'Hardhat',
    'Mask',
    'NO-Hardhat',
    'NO-Mask',
    'NO-Safety Vest',
    'Person',
    'Safety Cone',
    'Safety Vest',
    'Machinery',
    'Vehicle'
]
number_classes = len(ppe_classes)
output_dir = 'datasets'

dict_file = {
    'train': 'manually_annotated/images/train',
    'val': 'manually_annotated/images/val',
    'nc': number_classes,
    'names': ppe_classes
}

with open(os.path.join(dataset_base_dir, 'manual_data.yaml'), 'w+') as file:
    yaml.dump(dict_file, file)
# Full yaml path to prevent path error
manual_yaml_path = str(Path('datasets/manual_data.yaml').resolve())
```

2.0.5 Create final_hybrid_data.yaml

[59]:
```
# create final yaml in datasets
dict_file = {
    'train': 'images/train',
    'val': 'images/val',
    'test': 'images/test',
    'nc': number_classes,
    'names': ppe_classes
}

with open(os.path.join(dataset_base_dir, 'final_hybrid_data.yaml'), 'w+') as␣
 ↪file:
    yaml.dump(dict_file, file)
# Full yaml path to prevent path error
final_yaml_path = str(Path('datasets/final_hybrid_data.yaml').resolve())
```

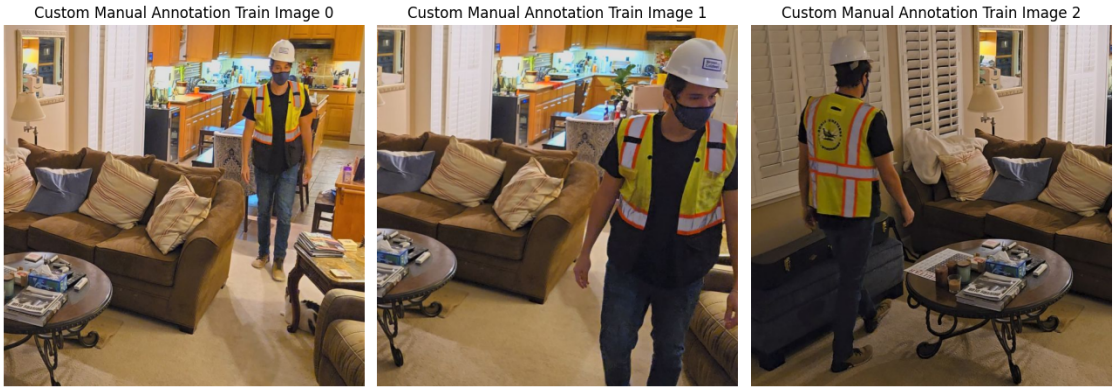
2.0.6 Random Image Visualization

[60]:
```
# define function to visualize random images
def visualize_images(images_dir, num_images=5):
    random.seed(42)
    image_files = [f for f in os.listdir(images_dir) if f.endswith(('.jpg', '.
 ↪jpeg', '.png'))]
    selected_files = random.sample(image_files, min(num_images,␣
 ↪len(image_files)))

    plt.figure(figsize=(12, 6))
    for i, image_file in enumerate(selected_files):
        image_path = os.path.join(images_dir, image_file)
        image = cv2.imread(image_path)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        plt.subplot(1, num_images, i + 1)
        plt.imshow(image)
        plt.axis("off")
        plt.title(image_file)
    plt.suptitle(f"Random Images Visualization {images_dir}")
    plt.tight_layout()
    plt.show()

visualize_images(manual_source_images_dir, num_images=2)
```



2.0.7 Custom images dataset

[61]:
```
# Get the paths to the first three custom images
image_paths = glob.glob(f"{manual_source_images_dir}/scene*.jpg")[:3]

# Check if there are enough images
if len(image_paths) < 3:
    print("Not enough images found!")
else:
    # Create a subplot with 1 row and 3 columns
    fig, axes = plt.subplots(1, 3, figsize=(13, 7))  # Width, height

    for i, ax in enumerate(axes):
        # Load and display each image
        image = mpimg.imread(image_paths[i])
        ax.imshow(image)
        ax.axis('off')  # Hide axes
        ax.set_title(f"Custom Manual Annotation Train Image {i}")

    plt.tight_layout()
    plt.show()
```



2.0.8 Google images dataset

[62]:
```
# Get the paths to the first three custom images
image_paths = glob.glob(f"{raw_images_dir}/images*.jpeg")[:3]

# Check if there are enough images
if len(image_paths) < 3:
    print("Not enough images found!")
else:
    # Create a subplot with 1 row and 3 columns
    fig, axes = plt.subplots(1, 3, figsize=(13, 7))  # Width, height
```

```
    for i, ax in enumerate(axes):
        # Load and display each image
        image = mpimg.imread(image_paths[i])
        ax.imshow(image)
        ax.axis('off')  # Hide axes
        ax.set_title(f"Custom Manual Annotation Train Image {i}")

    plt.tight_layout()
    plt.show()
```

2.0.9 Image Properties

[34]:
```
# Define function to analyze image dimensions, aspect ratio
def analyze_image_properties(images_dir):
    properties = {"filename": [], "width": [], "height": [], "aspect_ratio": []}

    for image_file in os.listdir(images_dir):
        if image_file.endswith(('.jpg', '.jpeg', '.png')):
            image_path = os.path.join(images_dir, image_file)
            image = cv2.imread(image_path)
            height, width = image.shape[:2]
            aspect_ratio = width / height
            properties["filename"].append(image_file)
            properties["width"].append(width)
            properties["height"].append(height)
            properties["aspect_ratio"].append(aspect_ratio)

    # Convert to pandas DataFrame for analysis
    df = pd.DataFrame(properties)
    return df
print("Image Properties Analysis Of Manually Labeled Images")
analyze_image_properties(manual_source_images_dir)
```

```
Image Properties Analysis Of Manually Labeled Images
```

[34]:
```
                                                filename  width  height  \
0    -1670-_png_jpg.rf.0463edb430019e01ec79eed27a63…     640     640
1    -1670-_png_jpg.rf.3cb172ea2c4165c19ae2dd498b38…     640     640
2    -1670-_png_jpg.rf.7da967f9aeaa62defc36543b9e60…     640     640
3    -1670-_png_jpg.rf.b42b26d784545ce1a033679674a4…     640     640
4    -1670-_png_jpg.rf.dd5cb0a4d6da02d34f1dc003fb4e…     640     640
..                                                 …     …     …
173  youtube-619_jpg.rf.93749f0e33b2f7beb302dd2e176…     640     640
174  youtube-697_jpg.rf.950b34f23e196850ec51a4add2e…     640     640
175  youtube-727_jpg.rf.cb4c039b4ffea84835e67e2e3d2…     640     640
176  youtube-804_jpg.rf.c4250b3c54e1cca8d5862297088…     640     640
177  youtube-836_jpg.rf.df6fc9e569e7b6b850ecb75c377…     640     640

     aspect_ratio
0             1.0
1             1.0
2             1.0
3             1.0
4             1.0
..            …
173           1.0
174           1.0
175           1.0
176           1.0
177           1.0

[178 rows x 4 columns]
```

2.0.10 Split manually annotated datasets to train and val

[35]:
```
# Define function to split manually annotated dataset into train and valation␣
 ↪sets
def split_manual_dataset(images_dir, labels_dir, split_ratio=0.8):
    image_files = [
        f for f in os.listdir(images_dir) if f.lower().endswith(('.jpg', '.
 ↪jpeg', '.png'))
    ]
    random.shuffle(image_files)

    train_size = int(split_ratio * len(image_files))
    train_files = image_files[:train_size]
    val_files = image_files[train_size:]

    # Copy train files and corresponding labels
    for file in train_files:
        label_file = os.path.splitext(file)[0] + ".txt"  # Match label with␣
 ↪image file
```

```
        image_src = os.path.join(images_dir, file)
        label_src = os.path.join(labels_dir, label_file)
        shutil.copy(image_src, os.path.join(manual_train_images_dir, file))
        shutil.copy(label_src, os.path.join(manual_train_labels_dir,␣
↪label_file))

    # Copy val files and corresponding labels
    for file in val_files:
        label_file = os.path.splitext(file)[0] + ".txt"  # Match label with␣
↪image file
        image_src = os.path.join(images_dir, file)
        label_src = os.path.join(labels_dir, label_file)
        shutil.copy(image_src, os.path.join(manual_val_images_dir, file))
        shutil.copy(label_src, os.path.join(manual_val_labels_dir, label_file))

    print("Manual dataset split completed. Images and labels copied to train␣
↪and val directories.")

split_manual_dataset(manual_source_images_dir, manual_source_labels_dir)
```

```
Manual dataset split completed. Images and labels copied to train and val
directories.
```

2.0.11 Train base model with manually annotated datasets

[ ]:
```
# Train the generic model on manually labeled data
def train_model(manual_train_images, manual_val_images, epochs=50):

    # Train the model
    print("Training model on manually labeled dataset...")
    model = YOLO("../../models/yolov8m.pt")
    model.train(
        data=manual_yaml_path,
        epochs=50, # Iterations
        imgsz=640, # Image size
        batch=16, # Batch size
        seed=88, # Train Repeatability
        lr0=1e-4,  # Learning rate
        optimizer="auto", # Automatic speed, stability optimization
        patience=10 # Num of epochs to discontinue unimproved train
    )
    print("Model training completed.")
    return model

# Train the model on the manual dataset
trained_model = train_model(manual_train_images_dir, manual_val_images_dir)
```

```
Training model on manually labeled dataset…
```

```
Downloading
https://github.com/ultralytics/assets/releases/download/v8.3.0/yolov8m.pt to
'yolov8m.pt'…

100%|      | 49.7M/49.7M [00:01<00:00, 36.9MB/s]

New https://pypi.org/project/ultralytics/8.3.37 available  Update with 'pip
install -U ultralytics'
Ultralytics 8.3.33  Python-3.11.9 torch-2.5.1 CPU (Apple M3 Pro)
engine/trainer: task=detect, mode=train, model=yolov8m.pt,
data=../models/manual_datasets.yaml, epochs=50, time=None, patience=10,
batch=16, imgsz=640, save=True, save_period=-1, cache=False, device=None,
workers=8, project=None, name=train, exist_ok=False, pretrained=True,
optimizer=auto, verbose=True, seed=0, deterministic=True, single_cls=False,
rect=False, cos_lr=False, close_mosaic=10, resume=False, amp=True, fraction=1.0,
profile=False, freeze=None, multi_scale=False, overlap_mask=True, mask_ratio=4,
dropout=0.0, val=True, split=val, save_json=False, save_hybrid=False, conf=None,
iou=0.7, max_det=300, half=False, dnn=False, plots=True, source=None,
vid_stride=1, stream_buffer=False, visualize=False, augment=False,
agnostic_nms=False, classes=None, retina_masks=False, embed=None, show=False,
save_frames=False, save_txt=False, save_conf=False, save_crop=False,
show_labels=True, show_conf=True, show_boxes=True, line_width=None,
format=torchscript, keras=False, optimize=False, int8=False, dynamic=False,
simplify=True, opset=None, workspace=4, nms=False, lr0=0.0001, lrf=0.01,
momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8,
warmup_bias_lr=0.1, box=7.5, cls=0.5, dfl=1.5, pose=12.0, kobj=1.0,
label_smoothing=0.0, nbs=64, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0,
translate=0.1, scale=0.5, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5,
bgr=0.0, mosaic=1.0, mixup=0.0, copy_paste=0.0, copy_paste_mode=flip,
auto_augment=randaugment, erasing=0.4, crop_fraction=1.0, cfg=None,
tracker=botsort.yaml, save_dir=runs/detect/train
Overriding model.yaml nc=80 with nc=10

                   from  n    params  module
arguments
  0                  -1  1      1392  ultralytics.nn.modules.conv.Conv
[3, 48, 3, 2]
  1                  -1  1     41664  ultralytics.nn.modules.conv.Conv
[48, 96, 3, 2]
  2                  -1  2    111360  ultralytics.nn.modules.block.C2f
[96, 96, 2, True]
  3                  -1  1    166272  ultralytics.nn.modules.conv.Conv
[96, 192, 3, 2]
  4                  -1  4    813312  ultralytics.nn.modules.block.C2f
[192, 192, 4, True]
  5                  -1  1    664320  ultralytics.nn.modules.conv.Conv
[192, 384, 3, 2]
  6                  -1  4   3248640  ultralytics.nn.modules.block.C2f
[384, 384, 4, True]
```

```
  7                  -1  1   1991808  ultralytics.nn.modules.conv.Conv
[384, 576, 3, 2]
  8                  -1  2   3985920  ultralytics.nn.modules.block.C2f
[576, 576, 2, True]
  9                  -1  1    831168  ultralytics.nn.modules.block.SPPF
[576, 576, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample
[None, 2, 'nearest']
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat
[1]
 12                  -1  2   1993728  ultralytics.nn.modules.block.C2f
[960, 384, 2]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample
[None, 2, 'nearest']
 14             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat
[1]
 15                  -1  2    517632  ultralytics.nn.modules.block.C2f
[576, 192, 2]
 16                  -1  1    332160  ultralytics.nn.modules.conv.Conv
[192, 192, 3, 2]
 17            [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat
[1]
 18                  -1  2   1846272  ultralytics.nn.modules.block.C2f
[576, 384, 2]
 19                  -1  1   1327872  ultralytics.nn.modules.conv.Conv
[384, 384, 3, 2]
 20             [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat
[1]
 21                  -1  2   4207104  ultralytics.nn.modules.block.C2f
[960, 576, 2]
 22        [15, 18, 21]  1   3781486  ultralytics.nn.modules.head.Detect
[10, [192, 384, 576]]
Model summary: 295 layers, 25,862,110 parameters, 25,862,094 gradients, 79.1
GFLOPs

Transferred 469/475 items from pretrained weights
TensorBoard: Start with 'tensorboard --logdir runs/detect/train',
view at http://localhost:6006/
Freezing layer 'model.22.dfl.conv.weight'

train: Scanning /Users/fatimatatanda/Library/CloudStorage/OneDrive-
Personal/Desktop/USD/Projects/aai-501-final-
project/datasets/manually_annotated/labels/train.cache… 142 images, 4
backgrounds, 0 corrupt: 100%|      | 142/142 [00:00<?, ?it/s]

albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01,
blur_limit=(3, 7)), ToGray(p=0.01, num_output_channels=3,
method='weighted_average'), CLAHE(p=0.01, clip_limit=(1.0, 4.0),
tile_grid_size=(8, 8))
```

```
val: Scanning /Users/fatimatatanda/Library/CloudStorage/OneDrive-
Personal/Desktop/USD/Projects/aai-501-final-
project/datasets/manually_annotated/labels/val.cache… 36 images, 0
backgrounds, 0 corrupt: 100%|      | 36/36 [00:00<?, ?it/s]

Plotting labels to runs/detect/train/labels.jpg…


optimizer: 'optimizer=auto' found, ignoring 'lr0=0.0001' and
'momentum=0.937' and determining best 'optimizer', 'lr0' and 'momentum'
automatically…
optimizer: AdamW(lr=0.000714, momentum=0.9) with parameter groups
77 weight(decay=0.0), 84 weight(decay=0.0005), 83 bias(decay=0.0)
TensorBoard: model graph visualization added
Image sizes 640 train, 640 val
Using 0 dataloader workers
Logging results to runs/detect/train
Starting training for 50 epochs…

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

       1/50         0G      1.629      3.637      1.681        297        640:
100%|      | 9/9 [02:28<00:00, 16.49s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:15<00:00,  7.83s/it]

                   all         36        402       0.68      0.167      0.194
0.135



      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

       2/50         0G      1.362      2.138      1.505        241        640:
100%|      | 9/9 [02:27<00:00, 16.37s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:15<00:00,  7.87s/it]

                   all         36        402      0.335      0.386      0.304
0.19



      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

       3/50         0G      1.278      1.784      1.425        193        640:
100%|      | 9/9 [02:27<00:00, 16.35s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:15<00:00,  7.75s/it]
```

```
                   all         36        402      0.567      0.384        0.4
0.248

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       4/50         0G      1.257      1.631      1.381        184        640:
100%|      | 9/9 [02:26<00:00, 16.25s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:15<00:00,  7.61s/it]
                   all         36        402      0.454      0.406      0.376
0.243

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       5/50         0G      1.218      1.443      1.351        236        640:
100%|      | 9/9 [02:27<00:00, 16.34s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:15<00:00,  7.78s/it]
                   all         36        402      0.542      0.448      0.451
0.266

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       6/50         0G      1.188      1.404      1.344        288        640:
100%|      | 9/9 [02:28<00:00, 16.54s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:15<00:00,  7.57s/it]
                   all         36        402      0.548      0.374      0.406
0.242

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       7/50         0G      1.202      1.352      1.348        188        640:
100%|      | 9/9 [02:26<00:00, 16.31s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:15<00:00,  7.60s/it]
                   all         36        402      0.423       0.48      0.414
0.241
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       8/50         0G      1.203      1.295       1.35        193        640:
100%|      | 9/9 [02:26<00:00, 16.26s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:15<00:00,  7.58s/it]
                   all         36        402      0.568      0.355      0.379
0.223

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       9/50         0G      1.244      1.372      1.387        265        640:
100%|      | 9/9 [02:26<00:00, 16.24s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:15<00:00,  7.58s/it]
                   all         36        402      0.623      0.386      0.447
0.257

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      10/50         0G      1.195      1.305      1.367        259        640:
100%|      | 9/9 [02:27<00:00, 16.35s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:15<00:00,  7.51s/it]
                   all         36        402      0.488      0.421      0.435
0.266

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      11/50         0G       1.17      1.236      1.354        220        640:
100%|      | 9/9 [02:28<00:00, 16.55s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:15<00:00,  7.59s/it]
                   all         36        402      0.582      0.407      0.454
0.262

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
```

```
      12/50         0G      1.177      1.186      1.318        210        640:
100%|      | 9/9 [02:27<00:00, 16.35s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:15<00:00,  7.51s/it]
                   all         36        402      0.637      0.414      0.455
0.274

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      13/50         0G      1.138      1.163      1.304        221        640:
100%|      | 9/9 [02:27<00:00, 16.42s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:14<00:00,  7.44s/it]
                   all         36        402      0.593      0.407      0.437
0.262

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      14/50         0G      1.091      1.071      1.254        274        640:
100%|      | 9/9 [02:27<00:00, 16.35s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:15<00:00,  7.57s/it]
                   all         36        402      0.785       0.38      0.458
0.284

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      15/50         0G      1.092      1.105       1.29        232        640:
100%|      | 9/9 [02:27<00:00, 16.44s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:14<00:00,  7.44s/it]
                   all         36        402      0.577      0.472      0.482
0.269

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      16/50         0G      1.079      1.101      1.274        169        640:
100%|      | 9/9 [02:27<00:00, 16.40s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:14<00:00,  7.46s/it]
```

```
                 all         36        402      0.582      0.547      0.542
0.318

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      17/50         0G       1.04      1.002      1.264        267        640:
100%|      | 9/9 [02:26<00:00, 16.30s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:14<00:00,  7.44s/it]
                 all         36        402      0.549      0.533      0.546
0.315

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      18/50         0G      1.075      1.072      1.278        198        640:
100%|      | 9/9 [02:25<00:00, 16.19s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:14<00:00,  7.42s/it]
                 all         36        402      0.703      0.407      0.493
0.291

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      19/50         0G      1.008      0.955      1.219        124        640:
100%|      | 9/9 [02:26<00:00, 16.24s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:14<00:00,  7.44s/it]
                 all         36        402      0.547      0.512        0.5
0.295

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      20/50         0G      1.008     0.9485       1.21        342        640:
100%|      | 9/9 [02:26<00:00, 16.24s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:14<00:00,  7.47s/it]
                 all         36        402      0.614      0.498      0.541
0.322
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      21/50         0G      1.024     0.9306      1.215        287        640:
100%|      | 9/9 [13:03<00:00, 87.10s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:14<00:00,  7.34s/it]
                   all         36        402      0.705      0.528      0.578
0.336

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      22/50         0G      0.963     0.8866      1.193        284        640:
100%|      | 9/9 [18:41<00:00, 124.64s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:15<00:00,  7.70s/it]
                   all         36        402      0.745      0.488      0.576
0.36

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      23/50         0G     0.9499     0.8987      1.187        216        640:
100%|      | 9/9 [17:55<00:00, 119.53s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:15<00:00,  7.61s/it]
                   all         36        402      0.684       0.52      0.578
0.361

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      24/50         0G     0.9755     0.8833      1.183        179        640:
100%|      | 9/9 [04:30<00:00, 30.04s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:23<00:00, 11.56s/it]
                   all         36        402      0.752      0.543      0.613
0.365

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
```

```
      25/50         0G     0.9293     0.8481      1.167        224        640:
100%|      | 9/9 [04:25<00:00, 29.47s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:23<00:00, 11.84s/it]
                   all         36        402       0.62      0.516      0.543
0.312


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      26/50         0G     0.8751     0.7917      1.148        203        640:
100%|      | 9/9 [04:47<00:00, 31.95s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:27<00:00, 13.83s/it]
                   all         36        402      0.619      0.516      0.554
0.337


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      27/50         0G     0.8967     0.7718      1.142        334        640:
100%|      | 9/9 [04:56<00:00, 32.99s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:25<00:00, 12.72s/it]
                   all         36        402      0.684      0.526      0.582
0.341


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      28/50         0G     0.8754     0.7799      1.131        171        640:
100%|      | 9/9 [04:53<00:00, 32.59s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:25<00:00, 12.92s/it]
                   all         36        402      0.605       0.63      0.608
0.353


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      29/50         0G       0.86     0.7497      1.142        183        640:
100%|      | 9/9 [04:53<00:00, 32.56s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:25<00:00, 12.93s/it]
```

```
                 all         36        402      0.718      0.594      0.639
0.385

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      30/50         0G     0.8677     0.7673      1.139        258        640:
100%|      | 9/9 [04:47<00:00, 31.92s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:25<00:00, 12.54s/it]
                 all         36        402      0.668      0.614      0.642
0.392

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      31/50         0G     0.8359     0.7204      1.103        213        640:
100%|      | 9/9 [04:42<00:00, 31.43s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [13:11<00:00, 395.78s/it]
                 all         36        402      0.654      0.598      0.639
0.405

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      32/50         0G     0.8033     0.6921      1.097        219        640:
100%|      | 9/9 [05:03<00:00, 33.71s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [08:19<00:00, 249.68s/it]
                 all         36        402      0.809      0.541      0.652
0.411

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      33/50         0G     0.8198     0.6849       1.11        247        640:
100%|      | 9/9 [02:29<00:00, 16.64s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:15<00:00,  7.75s/it]
                 all         36        402      0.649      0.641      0.655
0.424
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      34/50         0G     0.7695      0.611       1.06        194        640:
100%|      | 9/9 [03:23<00:00, 22.56s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:17<00:00,  8.89s/it]
                   all         36        402      0.725      0.634      0.665
0.42

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      35/50         0G     0.8013     0.6653      1.103        241        640:
100%|      | 9/9 [03:33<00:00, 23.71s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:22<00:00, 11.17s/it]
                   all         36        402      0.758      0.637      0.685
0.426

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      36/50         0G     0.7606     0.6264      1.071        227        640:
100%|      | 9/9 [03:56<00:00, 26.33s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:21<00:00, 10.99s/it]
                   all         36        402      0.826      0.609      0.685
0.423

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      37/50         0G     0.7505     0.6418      1.077        289        640:
100%|      | 9/9 [04:00<00:00, 26.77s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:22<00:00, 11.09s/it]
                   all         36        402      0.808      0.624       0.69
0.427

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
```

```
      38/50         0G     0.7275     0.6043      1.048        222        640:
100%|      | 9/9 [04:00<00:00, 26.77s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:21<00:00, 10.98s/it]
                   all         36        402      0.802      0.608      0.691
0.434

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      39/50         0G     0.7549     0.6074      1.051        171        640:
100%|      | 9/9 [03:59<00:00, 26.62s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:21<00:00, 10.92s/it]
                   all         36        402      0.683      0.649      0.678
0.429

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      40/50         0G     0.7245     0.5878      1.053        226        640:
100%|      | 9/9 [03:59<00:00, 26.64s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:22<00:00, 11.12s/it]
                   all         36        402      0.763      0.654      0.703
0.441

Closing dataloader mosaic
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01,
blur_limit=(3, 7)), ToGray(p=0.01, num_output_channels=3,
method='weighted_average'), CLAHE(p=0.01, clip_limit=(1.0, 4.0),
tile_grid_size=(8, 8))

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      41/50         0G      0.739     0.6002      1.068        182        640:
100%|      | 9/9 [04:03<00:00, 27.08s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:22<00:00, 11.08s/it]
                   all         36        402      0.771      0.683      0.716
0.443
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      42/50         0G     0.7407     0.5704      1.086        169        640:
100%|      | 9/9 [04:00<00:00, 26.70s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:22<00:00, 11.15s/it]
                   all         36        402      0.796      0.668      0.724
0.444

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      43/50         0G     0.7126     0.5495      1.053        131        640:
100%|      | 9/9 [04:06<00:00, 27.36s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:22<00:00, 11.33s/it]
                   all         36        402      0.762      0.633      0.701
0.447

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      44/50         0G      0.665      0.508      1.021        137        640:
100%|      | 9/9 [04:09<00:00, 27.76s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:22<00:00, 11.28s/it]
                   all         36        402      0.754      0.628      0.697
0.45

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      45/50         0G     0.6407     0.4671      1.001        134        640:
100%|      | 9/9 [04:12<00:00, 28.06s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:22<00:00, 11.41s/it]
                   all         36        402      0.754       0.64      0.699
0.449

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
```

```
      46/50         0G     0.6404     0.4704      1.005        121        640:
100%|      | 9/9 [04:12<00:00, 28.07s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:22<00:00, 11.45s/it]
                   all         36        402      0.858      0.573      0.697
0.452

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      47/50         0G     0.6175      0.448     0.9828        119        640:
100%|      | 9/9 [35:16<00:00, 235.21s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:14<00:00,  7.30s/it]
                   all         36        402      0.848      0.597      0.706
0.46

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      48/50         0G     0.6054       0.43     0.9836        144        640:
100%|      | 9/9 [02:27<00:00, 16.37s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:15<00:00,  7.62s/it]
                   all         36        402      0.838      0.607      0.713
0.469

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      49/50         0G     0.6159     0.4583          1        142        640:
100%|      | 9/9 [02:40<00:00, 17.81s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:15<00:00,  7.99s/it]
                   all         36        402      0.822      0.619      0.708
0.467

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      50/50         0G     0.6027     0.4359     0.9766        167        640:
100%|      | 9/9 [03:29<00:00, 23.25s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:18<00:00,  9.20s/it]
```

```
                   all         36        402      0.815      0.623       0.71
0.464



50 epochs completed in 4.610 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 52.0MB
Optimizer stripped from runs/detect/train/weights/best.pt, 52.0MB

Validating runs/detect/train/weights/best.pt…
Ultralytics 8.3.33  Python-3.11.9 torch-2.5.1 CPU (Apple M3 Pro)
Model summary (fused): 218 layers, 25,845,550 parameters, 0 gradients, 78.7
GFLOPs

                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 2/2 [00:18<00:00,  9.23s/it]
                   all         36        402      0.839      0.607      0.712
0.469
               Hardhat         21         46      0.905      0.621      0.788
0.525
                  Mask         17         19       0.79      0.684        0.8
0.557
            NO-Hardhat         21         34      0.852      0.588      0.601
0.36
               NO-Mask         26         57      0.701      0.404      0.427
0.191
        NO-Safety Vest         24         53      0.869        0.5      0.647
0.361
                Person         36        117      0.917      0.735      0.815
0.579
           Safety Cone          7         10      0.824        0.6      0.782
0.54
           Safety Vest         20         39      0.965      0.711      0.876
0.56
             Machinery         11         23      0.784      0.474      0.525
0.43
               Vehicle          3          4      0.784       0.75      0.856
0.588
Speed: 0.5ms preprocess, 509.7ms inference, 0.0ms loss, 0.4ms postprocess per
image
Results saved to runs/detect/train
Model training completed.
```

2.0.12 Automatically annotate datasets with base model

[26]:
```
# load the pre-trained model to auto annotated the raw_images
model_path = "runs/detect/train/weights/best.pt"  # Path to your trained YOLO␣
 ↪model
def auto_annotate_images(model_path, raw_images_dir, output_images_dir,␣
 ↪output_labels_dir, confidence_threshold=0.5):

    """
    Automatically annotate images using a pre-trained YOLO model.

    Args:
    - model_path (str): Path to the trained YOLO model.
    - raw_images_dir (str): Path to the directory containing raw images.
    - output_images_dir (str): Path to save annotated images.
    - output_labels_dir (str): Path to save annotation labels in YOLO format.
    - confidence_threshold (float): Minimum confidence score to include a␣
 ↪detection.
    """
    total_images = len([image_file for image_file in os.listdir(raw_images_dir)␣
 ↪if image_file.lower().endswith(('.jpg', '.jpeg', '.png'))])
    # load the pre-trained model
    model = YOLO(model_path)

    # Create output directories if they don't exist
    os.makedirs(output_images_dir, exist_ok=True)
    os.makedirs(output_labels_dir, exist_ok=True)

    for i, image_file in enumerate(os.listdir(raw_images_dir)):
        if image_file.lower().endswith(('.jpg', '.jpeg', '.png')):
            image_path = os.path.join(raw_images_dir, image_file)
            results = model(image_path, verbose=False)  # Perform inference,␣
 ↪silence output for each image

            # Save the original image to the output directory
            output_image_path = os.path.join(output_images_dir, image_file)
            cv2.imwrite(output_image_path, cv2.imread(image_path))

            # Prepare the YOLO format label file
            label_file = os.path.splitext(image_file)[0] + ".txt"
            output_label_path = os.path.join(output_labels_dir, label_file)

            with open(output_label_path, "w") as f:
                for result in results[0].boxes:
                    box = result.xywhn[0].cpu().numpy()  # Normalized x_center,␣
 ↪y_center, width, height
                    class_id = int(result.cls[0].cpu().numpy())
```

```
                    confidence = float(result.conf[0].cpu().numpy())

                    # Filter by confidence threshold
                    if confidence >= confidence_threshold:
                        f.write(f"{class_id} {box[0]:.3f} {box[1]:.3f} {box[2]:.
 ↪3f} {box[3]:.3f}\n")

            # Print the current progress (overwrite previous line)
            print(f"Annotating image {i+1} of {total_images}...", end='\r')

    print(f"Auto-annotation completed.\n' Annotated images and labels saved in␣
 ↪{output_images_dir} and {output_labels_dir}.")

auto_annotate_images(model_path, raw_images_dir, auto_annotated_images_dir,␣
 ↪auto_annotated_labels_dir)
```

```
Auto-annotation completed. Annotated images and labels saved in
datasets/auto_annotated/images and datasets/auto_annotated/labels.
```

2.0.13 Visualize auto-annotated datasets for model accuracy

[39]:
```
# visualize auto-annotation to ensure correct bounding
def visualize_annotations(image_dir, label_dir, ppe_classes, num_images=5):
    """
    Visualize YOLO-format annotations on images.

    Args:
    - image_dir (str): Path to the directory containing annotated images.
    - label_dir (str): Path to the directory containing YOLO-format label files.
    - ppe_classes (list): List of class names corresponding to class IDs.
    - num_images (int): Number of images to visualize.
    """

    num_classes = len(ppe_classes)
    colors = [
        (random.randint(0, 255),
         random.randint(0, 255),
         random.randint(0, 255)) for _ in range(num_classes)
    ]

    # Get list of image files
    image_files = [f for f in os.listdir(image_dir) if f.lower().endswith(('.
 ↪jpg', '.jpeg', '.png'))]

    # Select a subset of images to visualize
    image_files = random.sample(image_files, min(num_images, len(image_files)))
```

```
    for image_file in image_files:
        # Read the image
        image_path = os.path.join(image_dir, image_file)
        image = cv2.imread(image_path)
        if image is None:
            print(f"Could not read image: {image_file}")
            continue

        # Convert the image to RGB for visualization
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

        # Read the corresponding label file
        label_file = os.path.splitext(image_file)[0] + ".txt"
        label_path = os.path.join(label_dir, label_file)
        if not os.path.exists(label_path):
            print(f"Label file not found for image: {image_file}")
            continue

        with open(label_path, "r") as f:
            labels = f.readlines()

        # Overlay bounding boxes and class names on the image
        for label in labels:
            label_data = label.strip().split()
            class_id = int(label_data[0])
            x_center, y_center, width, height = map(float, label_data[1:])

            # Convert normalized YOLO coordinates to image pixel coordinates
            img_height, img_width, _ = image.shape
            x_center *= img_width
            y_center *= img_height
            width *= img_width
            height *= img_height

            x1 = int(x_center - width / 2)
            y1 = int(y_center - height / 2)
            x2 = int(x_center + width / 2)
            y2 = int(y_center + height / 2)

            # Draw the bounding box
            #color = (255, 0, 0)  # Blue color for bounding box
            color = colors[class_id]
            cv2.rectangle(image, (x1, y1), (x2, y2), color, 2)

            # Put the class name text
            label_text = f"{ppe_classes[class_id]}"
```

```
            cv2.putText(image, label_text, (x1, y1 - 10), cv2.
 ↪FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

        # Display the image
        plt.figure(figsize=(10, 8))
        plt.imshow(image)
        plt.axis("off")
        plt.title(f"Annotations for {image_file}")
        plt.show()

visualize_annotations(auto_annotated_images_dir, auto_annotated_labels_dir,␣
 ↪ppe_classes, num_images=2)
```

2.0.14 Combine manual and auto annotated datasets

[40]:
```
# Function to combine datasets (manual + auto-annotated)
# combined directory path

combined_images_dir = os.path.join(dataset_base_dir, "images")
combined_labels_dir = os.path.join(dataset_base_dir, "labels")

# check if path exist otherwise create it
os.makedirs(combined_images_dir, exist_ok=True)
os.makedirs(combined_labels_dir, exist_ok=True)


def copy_files(src_images_dir, src_labels_dir):
    for file in os.listdir(src_images_dir):
        if file.lower().endswith(('.jpg', '.jpeg', '.png')):
            # Copy image file
            src_image_path = os.path.join(src_images_dir, file)
            dest_image_path = os.path.join(combined_images_dir, file)
            shutil.copy(src_image_path, dest_image_path)

            # Copy corresponding label file
            label_file = os.path.splitext(file)[0] + ".txt"
            src_label_path = os.path.join(src_labels_dir, label_file)
            dest_label_path = os.path.join(combined_labels_dir, label_file)
            shutil.copy(src_label_path, dest_label_path)

# copy manually annotated images data
print("Copying manually annotated training data...")
copy_files(manual_train_images_dir, manual_train_labels_dir)

# copy manually annotated validation data
print("Copying manually annotated validation data...")
copy_files(manual_val_images_dir, manual_val_labels_dir)
```

```
# Copy auto-annotated data
print("Copying auto-annotated data...")
copy_files(auto_annotated_images_dir, auto_annotated_labels_dir)

print("Dataset combination completed.")
print(f"Combined images stored in: {combined_images_dir}")
print(f"Combined labels stored in: {combined_labels_dir}")
```

```
Copying manually annotated training data…
Copying manually annotated validation data…
Copying auto-annotated data…
Dataset combination completed.
Combined images stored in: datasets/images
Combined labels stored in: datasets/labels
```

2.0.15 Split combined images and labels to train, test and val

[41]:
```
# extract the combined image files for random split
# split directory for detection
train_images_dir = os.path.join(dataset_base_dir, "images/train")
train_labels_dir = os.path.join(dataset_base_dir, "labels/train")

test_images_dir = os.path.join(dataset_base_dir, "images/test")
test_labels_dir = os.path.join(dataset_base_dir, "labels/test")

val_images_dir = os.path.join(dataset_base_dir, "images/val")
val_labels_dir = os.path.join(dataset_base_dir, "labels/val")

# create dataset directories for detection & prediction
for dir_path in [
    train_images_dir,
    train_labels_dir,
    test_images_dir,
    test_labels_dir,
    val_images_dir,
    val_labels_dir
]:
    os.makedirs(dir_path, exist_ok=True)

image_files = [f for f in os.listdir(combined_images_dir) if f.endswith(('.
 ↪jpg', '.jpeg', '.png'))]
random.shuffle(image_files)

# Create DataFrame with file paths and dataset split assignments
df = pd.DataFrame({"filename": image_files})
# dynamically split dataset into train, valid, and test
```

```
# split sizes
train_size = int(0.7 * len(df))  # 70% for training
val_size = int(0.2 * len(df))  # 20% for validation
test_size = len(df) - train_size - val_size  # remaining 10% for testing

# split labels
train_labels = ["train"] * train_size
val_labels = ["val"] * val_size
test_labels = ["test"] * test_size

# combine and assign to DataFrame
df["split"] = train_labels + val_labels + test_labels

# Function to move files into corresponding split directories
def move_files(df, src_images_dir, src_labels_dir, dest_images_dir,␣
 ↪dest_labels_dir):
    for _, row in df.iterrows():  # iterate over a DataFrame row by row
        filename = row["filename"]
        split = row["split"]
        image_path = os.path.join(src_images_dir, filename)
        label_path = os.path.join(src_labels_dir, os.path.splitext(filename)[0]␣
 ↪+ ".txt")

        dest_image_dir = dest_images_dir[split]
        dest_label_dir = dest_labels_dir[split]

        # move image
        shutil.move(image_path, os.path.join(dest_image_dir, filename))

        # move label
        shutil.move(label_path, os.path.join(dest_label_dir, os.path.
 ↪splitext(filename)[0] + ".txt"))


# map split directories for images and labels
dest_images_dir = {"train": train_images_dir, "val": val_images_dir, "test":␣
 ↪test_images_dir}
dest_labels_dir = {"train": train_labels_dir, "val": val_labels_dir, "test":␣
 ↪test_labels_dir}

# call move fuxn
move_files(df, combined_images_dir, combined_labels_dir, dest_images_dir,␣
 ↪dest_labels_dir)

print("Dataset split completed.")
```

```
print(f"Training images: {len(os.listdir(train_images_dir))}, Labels: {len(os.
  ↪listdir(train_labels_dir))}")
print(f"Validation images: {len(os.listdir(val_images_dir))}, Labels: {len(os.
  ↪listdir(val_labels_dir))}")
print(f"Testing images: {len(os.listdir(test_images_dir))}, Labels: {len(os.
  ↪listdir(test_labels_dir))}")
```

```
Dataset split completed.
Training images: 2043, Labels: 2043
Validation images: 584, Labels: 584
Testing images: 293, Labels: 293
```

2.0.16 Train the final model with the combined datasets

(NOTE: update model_path for the base model created)

[33]:
```
input_name = input('Please enter run folder description:')
# load pre-trained YOLO base model
model_path = "runs/detect/train/weights/best.pt"
model = YOLO(model_path)
# Train the model
print("Training the final model...")
model.train(
    data=final_yaml_path,
    epochs=30, # Iterations
    imgsz=640, # Image size
    batch=20, # Batch size
    lr0=1e-3, # Learning rate
    patience=10, # Num of epochs to discontinue unimproved train
    amp=True, # Reduce memore, increase speed
    plots = True, # produce plots
    seed=88, # Train repeatability
    name = input_name, # Run folder name
)
print("Model training completed.")
```

```
Training the final model…
New https://pypi.org/project/ultralytics/8.3.38 available  Update with 'pip
install -U ultralytics'
Ultralytics 8.3.33  Python-3.11.9 torch-2.5.1 CPU (Apple M3 Pro)
engine/trainer: task=detect, mode=train,
model=runs/detect/train/weights/best.pt, data=../models/datasets.yaml,
epochs=30, time=None, patience=10, batch=20, imgsz=640, save=True,
save_period=-1, cache=False, device=None, workers=8, project=None, name=train4,
exist_ok=False, pretrained=True, optimizer=auto, verbose=True, seed=0,
deterministic=True, single_cls=False, rect=False, cos_lr=False, close_mosaic=10,
resume=False, amp=True, fraction=1.0, profile=False, freeze=None,
multi_scale=False, overlap_mask=True, mask_ratio=4, dropout=0.0, val=True,
split=val, save_json=False, save_hybrid=False, conf=None, iou=0.7, max_det=300,
```

```
half=False, dnn=False, plots=True, source=None, vid_stride=1,
stream_buffer=False, visualize=False, augment=False, agnostic_nms=False,
classes=None, retina_masks=False, embed=None, show=False, save_frames=False,
save_txt=False, save_conf=False, save_crop=False, show_labels=True,
show_conf=True, show_boxes=True, line_width=None, format=torchscript,
keras=False, optimize=False, int8=False, dynamic=False, simplify=True,
opset=None, workspace=4, nms=False, lr0=0.001, lrf=0.01, momentum=0.937,
weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1,
box=7.5, cls=0.5, dfl=1.5, pose=12.0, kobj=1.0, label_smoothing=0.0, nbs=64,
hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1, scale=0.5,
shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, bgr=0.0, mosaic=1.0,
mixup=0.0, copy_paste=0.0, copy_paste_mode=flip, auto_augment=randaugment,
erasing=0.4, crop_fraction=1.0, cfg=None, tracker=botsort.yaml,
save_dir=runs/detect/train4

                   from  n    params  module
arguments
  0                  -1  1      1392  ultralytics.nn.modules.conv.Conv
[3, 48, 3, 2]
  1                  -1  1     41664  ultralytics.nn.modules.conv.Conv
[48, 96, 3, 2]
  2                  -1  2    111360  ultralytics.nn.modules.block.C2f
[96, 96, 2, True]
  3                  -1  1    166272  ultralytics.nn.modules.conv.Conv
[96, 192, 3, 2]
  4                  -1  4    813312  ultralytics.nn.modules.block.C2f
[192, 192, 4, True]
  5                  -1  1    664320  ultralytics.nn.modules.conv.Conv
[192, 384, 3, 2]
  6                  -1  4   3248640  ultralytics.nn.modules.block.C2f
[384, 384, 4, True]
  7                  -1  1   1991808  ultralytics.nn.modules.conv.Conv
[384, 576, 3, 2]
  8                  -1  2   3985920  ultralytics.nn.modules.block.C2f
[576, 576, 2, True]
  9                  -1  1    831168  ultralytics.nn.modules.block.SPPF
[576, 576, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample
[None, 2, 'nearest']
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat
[1]
 12                  -1  2   1993728  ultralytics.nn.modules.block.C2f
[960, 384, 2]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample
[None, 2, 'nearest']
 14             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat
[1]
 15                  -1  2    517632  ultralytics.nn.modules.block.C2f
```

```
[576, 192, 2]
 16                  -1  1    332160  ultralytics.nn.modules.conv.Conv
[192, 192, 3, 2]
 17            [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat
[1]
 18                  -1  2   1846272  ultralytics.nn.modules.block.C2f
[576, 384, 2]
 19                  -1  1   1327872  ultralytics.nn.modules.conv.Conv
[384, 384, 3, 2]
 20             [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat
[1]
 21                  -1  2   4207104  ultralytics.nn.modules.block.C2f
[960, 576, 2]
 22        [15, 18, 21]  1   3781486  ultralytics.nn.modules.head.Detect
[10, [192, 384, 576]]
Model summary: 295 layers, 25,862,110 parameters, 25,862,094 gradients, 79.1
GFLOPs

Transferred 475/475 items from pretrained weights
TensorBoard: Start with 'tensorboard --logdir runs/detect/train4',
view at http://localhost:6006/
Freezing layer 'model.22.dfl.conv.weight'

train: Scanning /Users/fatimatatanda/Library/CloudStorage/OneDrive-
Personal/Desktop/USD/Projects/aai-501-final-
project/datasets/labels/train.cache… 2656 images, 60 backgrounds, 0 corrupt:
100%|      | 2656/2656 [00:00<?, ?it/s]

albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01,
blur_limit=(3, 7)), ToGray(p=0.01, num_output_channels=3,
method='weighted_average'), CLAHE(p=0.01, clip_limit=(1.0, 4.0),
tile_grid_size=(8, 8))


val: Scanning /Users/fatimatatanda/Library/CloudStorage/OneDrive-
Personal/Desktop/USD/Projects/aai-501-final-project/datasets/labels/val.cache…
175 images, 1 backgrounds, 0 corrupt: 100%|      | 175/175 [00:00<?, ?it/s]

Plotting labels to runs/detect/train4/labels.jpg…


optimizer: 'optimizer=auto' found, ignoring 'lr0=0.001' and
'momentum=0.937' and determining best 'optimizer', 'lr0' and 'momentum'
automatically…
optimizer: AdamW(lr=0.000714, momentum=0.9) with parameter groups
77 weight(decay=0.0), 84 weight(decay=0.00046875), 83 bias(decay=0.0)
TensorBoard: model graph visualization added
Image sizes 640 train, 640 val
Using 0 dataloader workers
Logging results to runs/detect/train4
```

```
Starting training for 30 epochs…

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

       1/30         0G     0.8476      1.012      1.197        221        640:
100%|     | 133/133 [1:17:13<00:00, 34.84s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|     | 5/5 [01:09<00:00, 13.88s/it]

                   all        175       1409      0.751      0.762      0.825
0.62



      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

       2/30         0G     0.9253      1.116      1.239        243        640:
100%|     | 133/133 [45:30<00:00, 20.53s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|     | 5/5 [01:09<00:00, 13.86s/it]

                   all        175       1409      0.693      0.655      0.692
0.462



      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

       3/30         0G     0.9616      1.176       1.26        143        640:
100%|     | 133/133 [45:34<00:00, 20.56s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|     | 5/5 [01:09<00:00, 13.82s/it]

                   all        175       1409      0.631      0.664      0.675
0.444



      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

       4/30         0G     0.9604      1.157      1.255        200        640:
100%|     | 133/133 [45:32<00:00, 20.55s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|     | 5/5 [01:09<00:00, 13.85s/it]

                   all        175       1409      0.714      0.705      0.759
0.516



      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
```

```
       5/30         0G     0.9396       1.12      1.242        224        640:
100%|      | 133/133 [45:31<00:00, 20.54s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 5/5 [01:09<00:00, 13.85s/it]
                   all        175       1409      0.728      0.733      0.769
0.525

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       6/30         0G     0.9102      1.081      1.222        156        640:
100%|      | 133/133 [45:32<00:00, 20.54s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 5/5 [01:09<00:00, 13.81s/it]
                   all        175       1409      0.722      0.759      0.806
0.576

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       7/30         0G     0.8813       1.02      1.205        196        640:
100%|      | 133/133 [45:34<00:00, 20.56s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 5/5 [01:09<00:00, 13.88s/it]
                   all        175       1409      0.759      0.745      0.797
0.562

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       8/30         0G     0.8526     0.9793      1.189        155        640:
100%|      | 133/133 [45:31<00:00, 20.54s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 5/5 [01:09<00:00, 13.87s/it]
                   all        175       1409      0.771      0.733       0.79
0.56

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
       9/30         0G     0.8388     0.9419      1.183        200        640:
100%|      | 133/133 [45:30<00:00, 20.53s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 5/5 [01:09<00:00, 13.84s/it]
```

```
                   all        175       1409      0.747      0.773      0.815
0.601

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      10/30         0G     0.8247     0.9245       1.17        225        640:
100%|      | 133/133 [45:30<00:00, 20.53s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 5/5 [01:09<00:00, 13.86s/it]
                   all        175       1409      0.756      0.786      0.819
0.597

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      11/30         0G     0.8024     0.8924       1.16        198        640:
100%|      | 133/133 [45:32<00:00, 20.54s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 5/5 [01:09<00:00, 13.88s/it]
                   all        175       1409        0.8      0.787      0.847
0.623

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      12/30         0G      0.794     0.8722      1.146        180        640:
100%|      | 133/133 [45:33<00:00, 20.55s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 5/5 [01:08<00:00, 13.79s/it]
                   all        175       1409       0.77       0.79      0.834
0.609

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      13/30         0G     0.7835      0.848      1.139        303        640:
100%|      | 133/133 [45:33<00:00, 20.55s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 5/5 [01:09<00:00, 13.84s/it]
                   all        175       1409      0.817      0.788      0.852
0.639
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      14/30         0G     0.7612     0.8223      1.127        298        640:
100%|     | 133/133 [45:31<00:00, 20.54s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|     | 5/5 [01:09<00:00, 13.85s/it]
                   all        175       1409      0.804      0.789      0.858
0.645

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      15/30         0G     0.7546      0.806      1.122        160        640:
100%|     | 133/133 [45:32<00:00, 20.55s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|     | 5/5 [01:09<00:00, 13.86s/it]
                   all        175       1409      0.791      0.803      0.851
0.65

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      16/30         0G     0.7356     0.7834      1.114        171        640:
100%|     | 133/133 [45:33<00:00, 20.55s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|     | 5/5 [01:08<00:00, 13.79s/it]
                   all        175       1409      0.811      0.819      0.866
0.656

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      17/30         0G     0.7242     0.7642        1.1        252        640:
100%|     | 133/133 [45:32<00:00, 20.55s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|     | 5/5 [01:08<00:00, 13.78s/it]
                   all        175       1409      0.805      0.837      0.868
0.662

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
```

```
      18/30         0G     0.7128     0.7491      1.101        305        640:
100%|      | 133/133 [45:31<00:00, 20.54s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 5/5 [01:09<00:00, 13.88s/it]
                   all        175       1409      0.813      0.822      0.867
0.666


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      19/30         0G     0.7043     0.7331      1.094        190        640:
100%|      | 133/133 [45:31<00:00, 20.54s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 5/5 [01:09<00:00, 13.85s/it]
                   all        175       1409      0.821      0.818      0.868
0.672


      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      20/30         0G     0.6807     0.7081      1.079        157        640:
100%|      | 133/133 [45:32<00:00, 20.55s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 5/5 [01:09<00:00, 13.82s/it]
                   all        175       1409      0.802      0.845      0.883
0.685


Closing dataloader mosaic
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01,
blur_limit=(3, 7)), ToGray(p=0.01, num_output_channels=3,
method='weighted_average'), CLAHE(p=0.01, clip_limit=(1.0, 4.0),
tile_grid_size=(8, 8))

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      21/30         0G     0.6734     0.6241      1.082        100        640:
100%|      | 133/133 [45:28<00:00, 20.52s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 5/5 [01:09<00:00, 13.84s/it]
                   all        175       1409      0.814      0.834      0.868
0.674
```

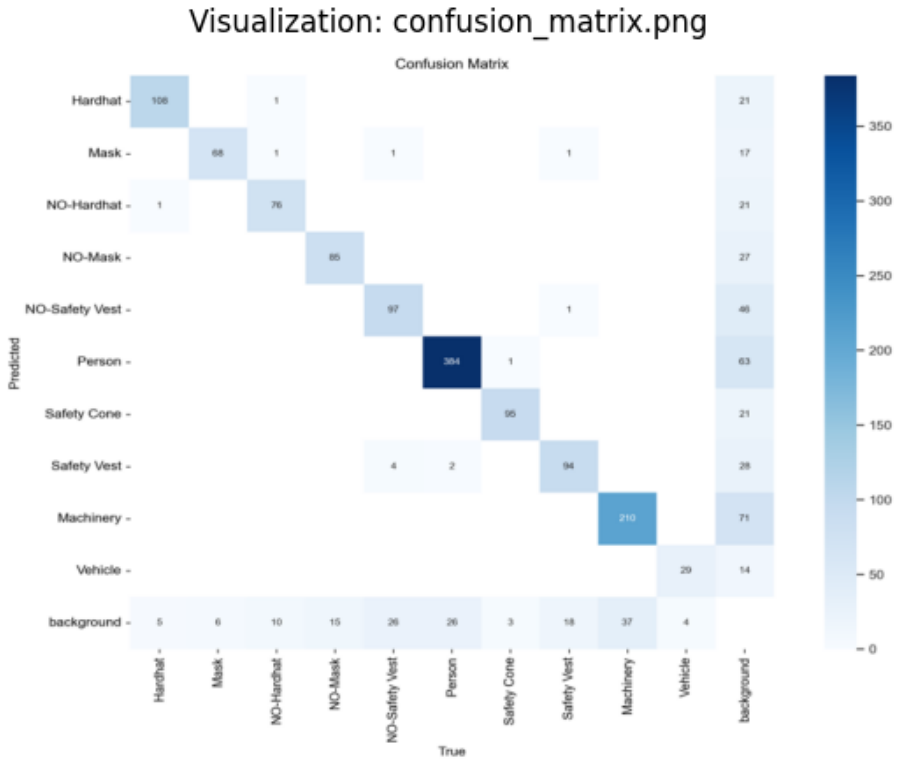
```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      22/30         0G     0.6487     0.5883      1.067         99        640:
100%|      | 133/133 [45:28<00:00, 20.51s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 5/5 [01:09<00:00, 13.87s/it]
                   all        175       1409      0.798      0.861      0.886
0.694

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      23/30         0G     0.6351     0.5559      1.058        119        640:
100%|      | 133/133 [45:36<00:00, 20.58s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 5/5 [01:09<00:00, 13.81s/it]
                   all        175       1409      0.816      0.823      0.883
0.687

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      24/30         0G     0.6246     0.5412      1.052        103        640:
100%|      | 133/133 [45:31<00:00, 20.54s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 5/5 [01:09<00:00, 13.87s/it]
                   all        175       1409      0.816      0.826      0.882
0.702

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      25/30         0G      0.601     0.5232      1.035        117        640:
100%|      | 133/133 [47:50<00:00, 21.58s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 5/5 [01:09<00:00, 13.82s/it]
                   all        175       1409      0.824      0.821      0.876
0.693

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
```

```
      26/30         0G     0.5886     0.5004      1.033         85        640:
100%|      | 133/133 [45:27<00:00, 20.51s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 5/5 [01:08<00:00, 13.73s/it]
                   all        175       1409      0.825       0.85      0.884
0.704

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      27/30         0G     0.5779     0.4901      1.025        167        640:
100%|      | 133/133 [45:27<00:00, 20.51s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 5/5 [01:09<00:00, 13.86s/it]
                   all        175       1409      0.801       0.85      0.887
0.704

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      28/30         0G     0.5602     0.4692      1.011         96        640:
100%|      | 133/133 [45:36<00:00, 20.57s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 5/5 [01:09<00:00, 13.84s/it]
                   all        175       1409       0.83      0.832      0.883
0.706

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      29/30         0G     0.5486     0.4525      1.005         77        640:
100%|      | 133/133 [45:37<00:00, 20.58s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 5/5 [01:09<00:00, 13.90s/it]
                   all        175       1409      0.804      0.868       0.89
0.715

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      30/30         0G     0.5328      0.435     0.9977        136        640:
100%|      | 133/133 [45:38<00:00, 20.59s/it]
                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|      | 5/5 [01:09<00:00, 13.88s/it]
```

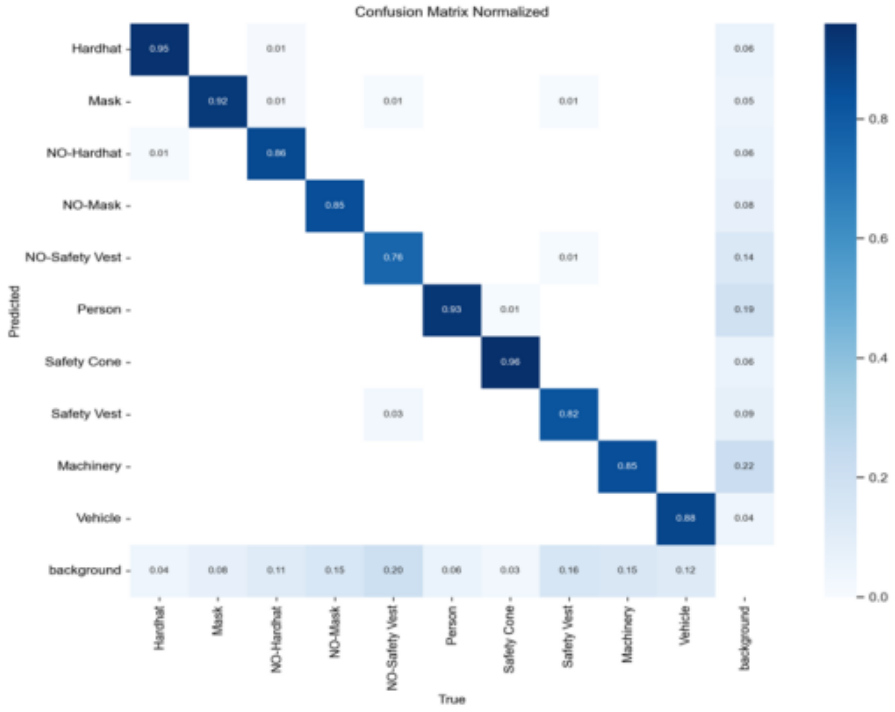
```
                   all        175       1409      0.844      0.843       0.89
0.714



30 epochs completed in 23.917 hours.
Optimizer stripped from runs/detect/train4/weights/last.pt, 52.0MB
Optimizer stripped from runs/detect/train4/weights/best.pt, 52.0MB

Validating runs/detect/train4/weights/best.pt…
Ultralytics 8.3.33  Python-3.11.9 torch-2.5.1 CPU (Apple M3 Pro)
Model summary (fused): 218 layers, 25,845,550 parameters, 0 gradients, 78.7
GFLOPs

                 Class     Images  Instances      Box(P          R      mAP50
mAP50-95): 100%|     | 5/5 [01:06<00:00, 13.37s/it]

                   all        175       1409      0.805      0.868       0.89
0.715
               Hardhat         50        114      0.863      0.956      0.972
0.83
                  Mask         60         74      0.829      0.946      0.929
0.782
            NO-Hardhat         63         88      0.833      0.875      0.915
0.737
               NO-Mask         55        100      0.772       0.83      0.841
0.56
        NO-Safety Vest         80        128      0.722       0.73      0.808
0.62
                Person        150        412      0.885       0.92      0.956
0.825
           Safety Cone         29         99       0.89       0.96      0.949
0.744
           Safety Vest         61        114      0.785      0.842       0.89
0.709
             Machinery        123        247      0.778      0.822      0.854
0.739
               Vehicle         23         33      0.688      0.801      0.783
0.601
Speed: 0.5ms preprocess, 376.9ms inference, 0.0ms loss, 0.2ms postprocess per
image
Results saved to runs/detect/train4
Model training completed.
```

2.0.17 Evalutate train results

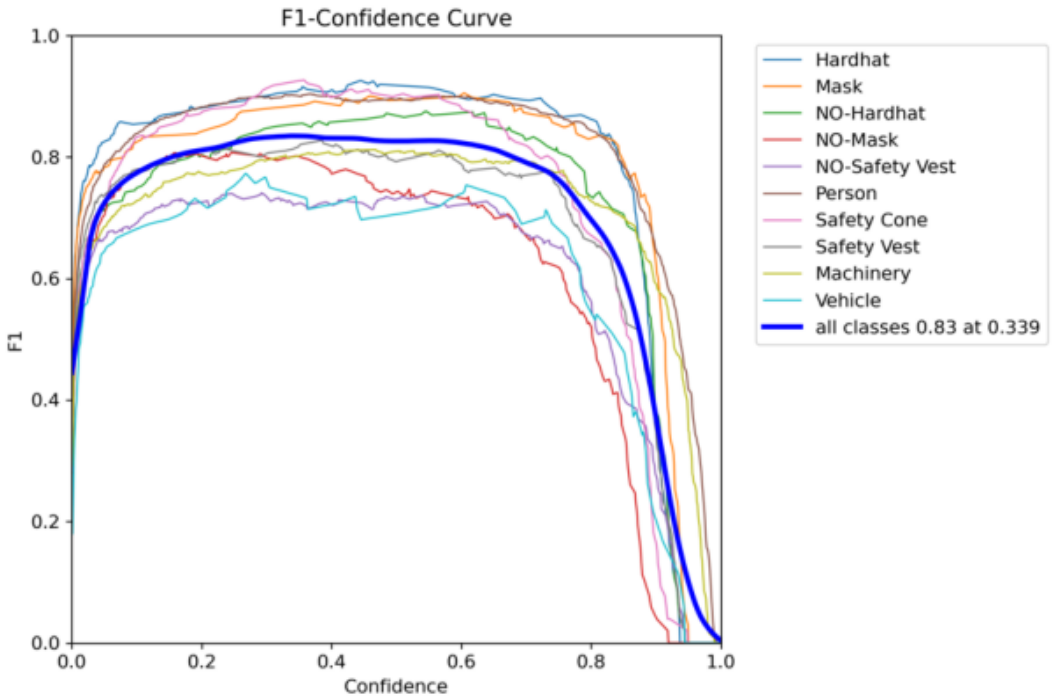
[52]:
```
run_path = "runs/detect/train4"
# loop through images, do not include images for val and train
img_files = [
    f for f in os.listdir(run_path)
    if f.endswith((".png", ".jpg")) and not f.startswith(("train", "val"))
]

# Loop through each image file and display it
for img_file in img_files:
    file_path = os.path.join(run_path, img_file)
    image = Image.open(file_path).resize((900, 600)) # Width, height
    # Display the image
    plt.figure(figsize=(8, 6))
    plt.imshow(image)
    plt.axis("off")  # Hide axes
    plt.title(f"Visualization: {img_file}")
    plt.show()
```



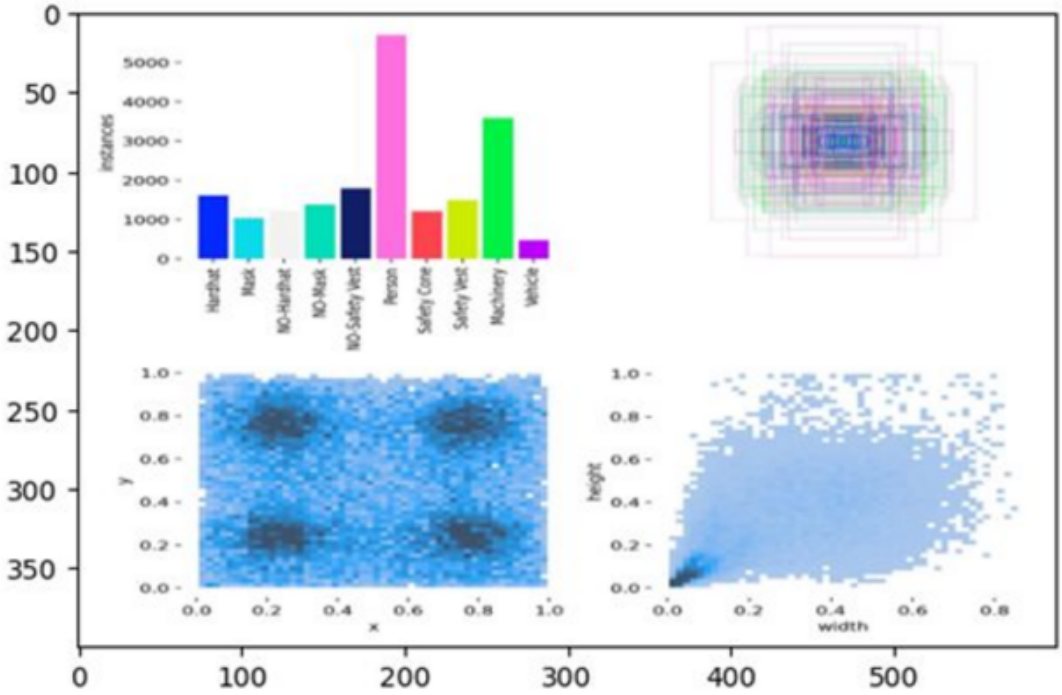
Visualization: confusion_matrix_normalized.png



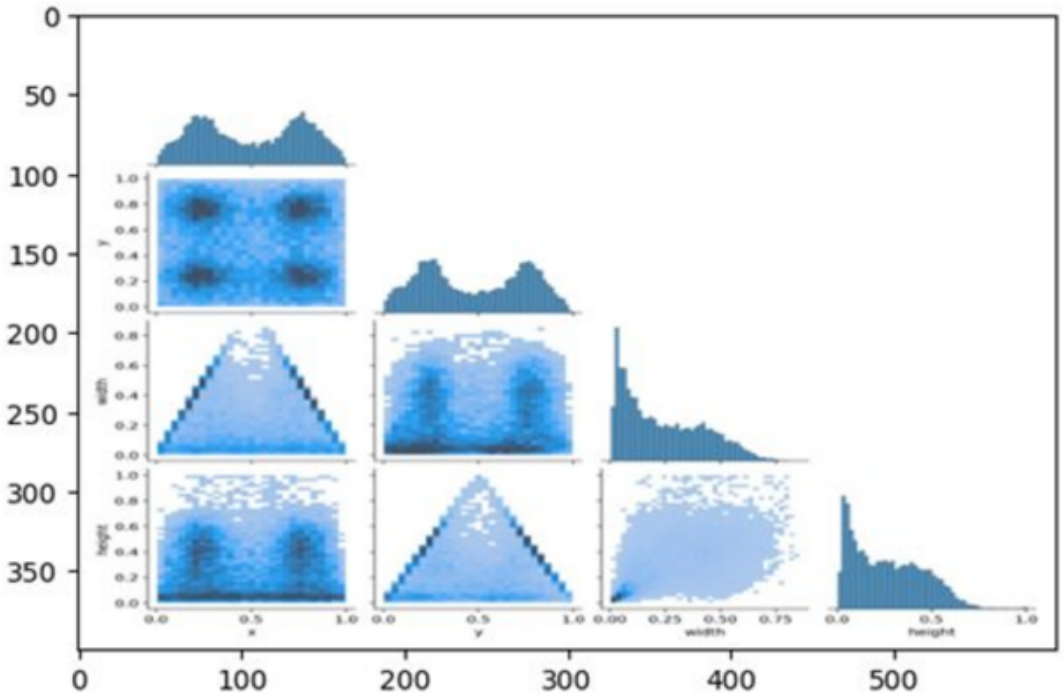
Visualization: F1_curve.png



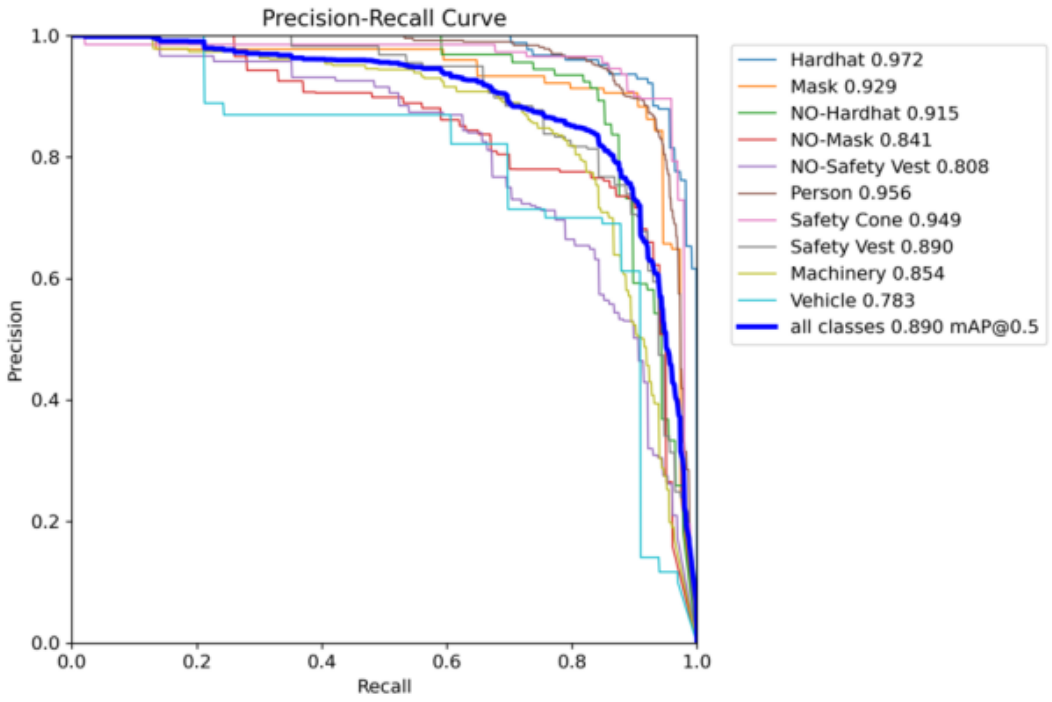
Visualization: labels.jpg



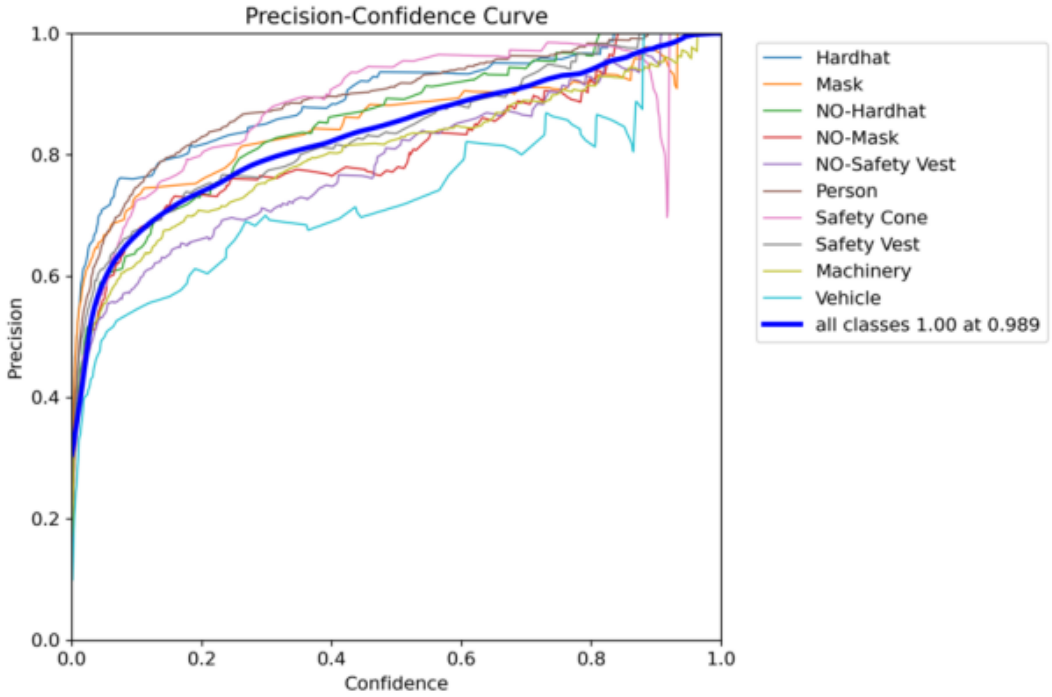
Visualization: labels_correlogram.jpg



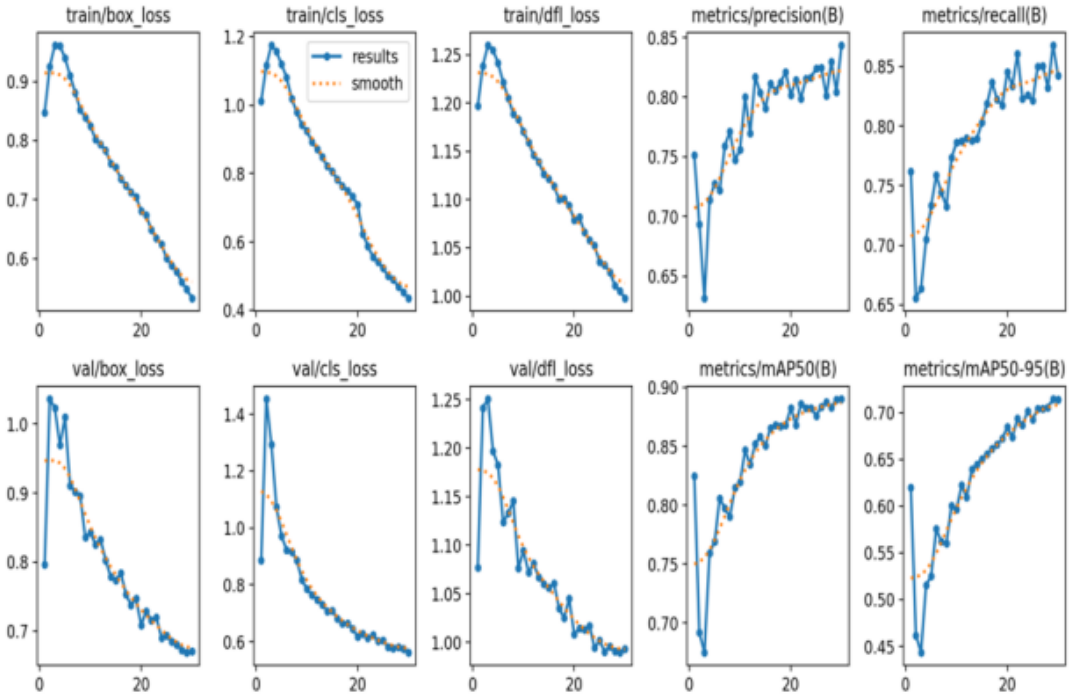
Visualization: PR_curve.png



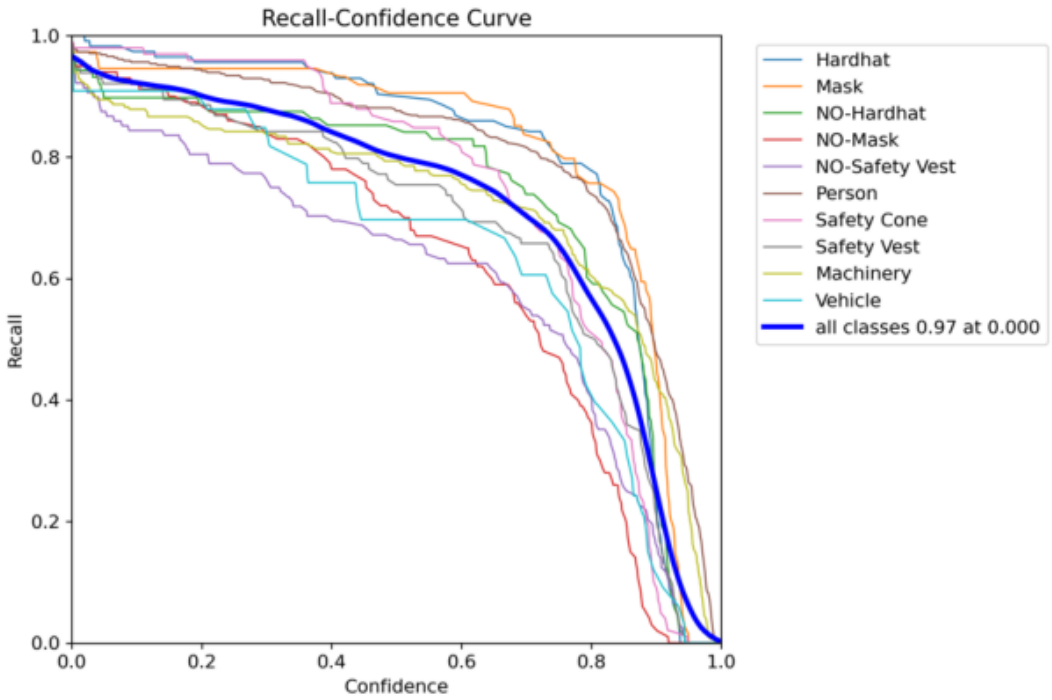
Visualization: P_curve.png



Visualization: results.png



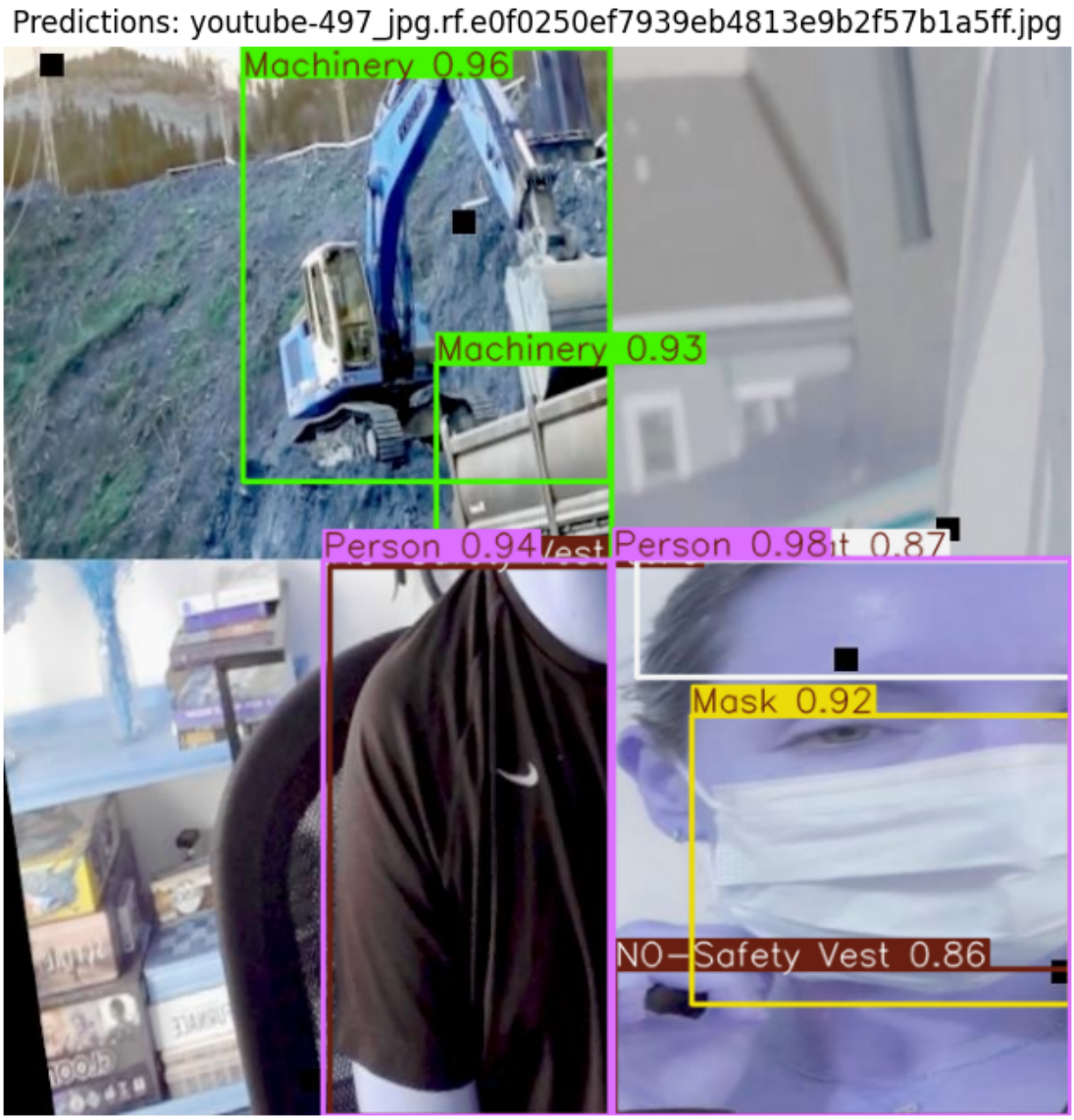
Visualization: R_curve.png



2.0.18 Visualize prediction on test sets

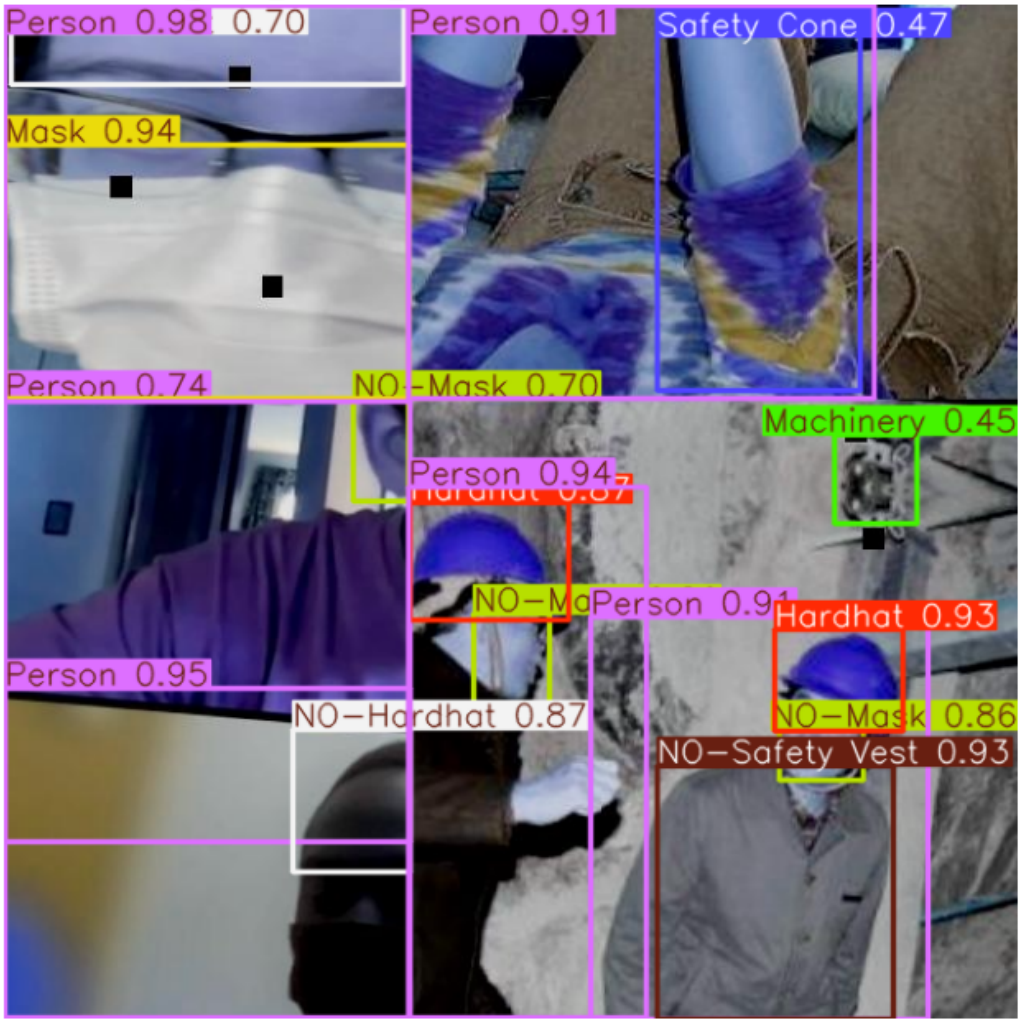
[54]:
```
# load the final trained model
model = YOLO("runs/detect/train4/weights/best.pt")

# Path to test images
test_images_dir = "datasets/images/test"

# get list of test images
test_images = [f for f in os.listdir(test_images_dir) if f.lower().endswith(('.
 ↪jpg', '.jpeg', '.png'))]

# run inference and display predictions
# shuffle the images randomly
random.shuffle(test_images)

# limit to a subset
test_images_subset = test_images[:3]

# Run inference and display predictions for random images
for test_image in test_images_subset:
    image_path = os.path.join(test_images_dir, test_image)
    results = model.predict(source=image_path, save=False)  # Run inference

    # Display the image with predictions
    img = results[0].plot()  # YOLOv8's built-in method to plot predictions
    plt.figure(figsize=(10, 8))
    plt.imshow(img)
    plt.axis("off")
    plt.title(f"Predictions: {test_image}")
    plt.show()
```

```
image 1/1 C:\Users\DARiN\Documents\Python-JL\Github\aai-501-final-project\notebo
oks\hybrid_model_train\datasets\images\test\youtube-
497_jpg.rf.e0f0250ef7939eb4813e9b2f57b1a5ff.jpg: 640x640 1 Mask, 1 NO-Hardhat, 2
NO-Safety Vests, 2 Persons, 2 Machinerys, 267.9ms
Speed: 2.0ms preprocess, 267.9ms inference, 2.0ms postprocess per image at shape
(1, 3, 640, 640)
```




```
image 1/1 C:\Users\DARiN\Documents\Python-JL\Github\aai-501-final-project\notebo
oks\hybrid_model_train\datasets\images\test\IMG_3103_mp4-
15_jpg.rf.d7e8a84fc7e9d53fe348320ea3af0e94.jpg: 640x640 2 Hardhats, 1 Mask, 2
NO-Hardhats, 3 NO-Masks, 1 NO-Safety Vest, 6 Persons, 1 Safety Cone, 1
Machinery, 253.6ms
Speed: 2.0ms preprocess, 253.6ms inference, 1.0ms postprocess per image at shape
(1, 3, 640, 640)
```

Predictions: IMG_3103_mp4-15_jpg.rf.d7e8a84fc7e9d53fe348320ea3af0e94.jpg



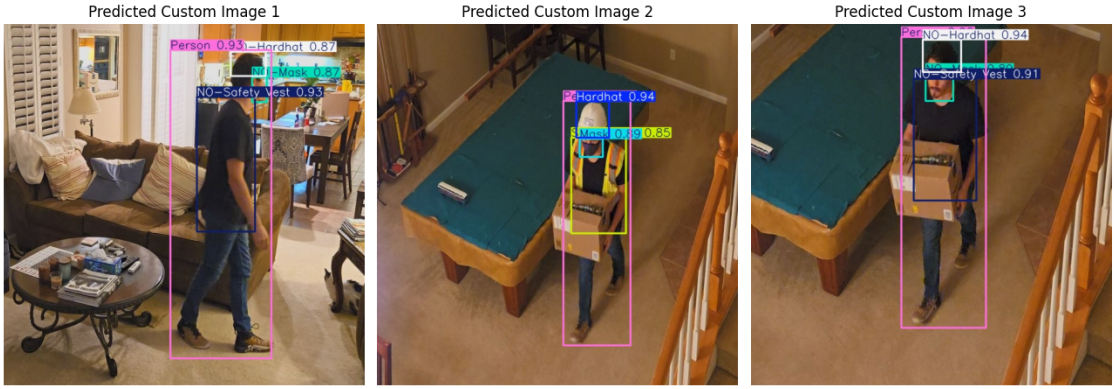

```
image 1/1 C:\Users\DARiN\Documents\Python-JL\Github\aai-501-final-project\notebo
oks\hybrid_model_train\datasets\images\test\01261_jpg.rf.ee044d42291843f1358960c
4d8c1d466.jpg: 640x640 4 Hardhats, 4 NO-Masks, 1 NO-Safety Vest, 4 Persons, 4
Safety Vests, 1 Vehicle, 235.2ms
Speed: 2.0ms preprocess, 235.2ms inference, 1.0ms postprocess per image at shape
(1, 3, 640, 640)
```

2.0.19 Predict custom images

[11]:
```
# Remove previous predictions
detect_path = "runs/detect/"
for folder in os.listdir(detect_path):
    folder_path = os.path.join(detect_path, folder)
    if os.path.isdir(folder_path) and folder.startswith("predict"):
        shutil.rmtree(folder_path, ignore_errors=True)

# Make predictions on "scene" images
model.predict(
```

```
    f"{test_images_dir}/scene*.jpg",
    save=True,
    conf=0.45,
    agnostic_nms=True,
    iou=0.8,
    verbose=False
)

# Visualize the first three predicted "scene" images
scene_predicted_paths = glob.glob("runs/detect/predict/scene*.jpg")[:3]

if len(scene_predicted_paths) < 3:
    print("Not enough predictions found!")
else:
    fig, axes = plt.subplots(1, 3, figsize=(13, 6))  # 1 row, 3 columns
    for i, ax in enumerate(axes):
        image = mpimg.imread(scene_predicted_paths[i])
        ax.imshow(image)
        ax.axis('off')
        ax.set_title(f"Predicted Custom Image {i + 1}")
    plt.tight_layout()
    plt.show()
```

Results saved to **runs\detect\predict**



Model Prediction Comparison

December 5, 2024

1 Model Predictions Comparisons

AAI-501 Group 4: Fatimat Atanda, Victor Hugo Germano, Darin Verduzco

[1]:
```
# import modules
import cv2 # opencv for images
import torch
from ultralytics import YOLO #for obeject detection
import matplotlib.pyplot as plt

# Support methods for groupd 4
import msaai
```

2 Load Model and Model Names

Using 4 main models:

- YOLO vanilla
- YOLO community pre trained on PPEs
- YOLO trained using manual annotations
- YOLO hybrid model

[3]:
```
# load pre trained model
model_default = YOLO("../models/yolov8n.pt")
model_usd_g4 = YOLO("../models/best_custom.pt")
model_pre_trained = YOLO("../models/ppe.pt")
model_hybrid = YOLO("../models/hybrid_best.pt")

print(model_default.names)

print("===== USD Group 4 Class names =====")

print(model_usd_g4.names)

print(print("===== Community pre trained ====="))

print(model_pre_trained.names)
```

```
{0: 'person', 1: 'bicycle', 2: 'car', 3: 'motorcycle', 4: 'airplane', 5: 'bus',
6: 'train', 7: 'truck', 8: 'boat', 9: 'traffic light', 10: 'fire hydrant', 11:
```

```
'stop sign', 12: 'parking meter', 13: 'bench', 14: 'bird', 15: 'cat', 16: 'dog',
17: 'horse', 18: 'sheep', 19: 'cow', 20: 'elephant', 21: 'bear', 22: 'zebra',
23: 'giraffe', 24: 'backpack', 25: 'umbrella', 26: 'handbag', 27: 'tie', 28:
'suitcase', 29: 'frisbee', 30: 'skis', 31: 'snowboard', 32: 'sports ball', 33:
'kite', 34: 'baseball bat', 35: 'baseball glove', 36: 'skateboard', 37:
'surfboard', 38: 'tennis racket', 39: 'bottle', 40: 'wine glass', 41: 'cup', 42:
'fork', 43: 'knife', 44: 'spoon', 45: 'bowl', 46: 'banana', 47: 'apple', 48:
'sandwich', 49: 'orange', 50: 'broccoli', 51: 'carrot', 52: 'hot dog', 53:
'pizza', 54: 'donut', 55: 'cake', 56: 'chair', 57: 'couch', 58: 'potted plant',
59: 'bed', 60: 'dining table', 61: 'toilet', 62: 'tv', 63: 'laptop', 64:
'mouse', 65: 'remote', 66: 'keyboard', 67: 'cell phone', 68: 'microwave', 69:
'oven', 70: 'toaster', 71: 'sink', 72: 'refrigerator', 73: 'book', 74: 'clock',
75: 'vase', 76: 'scissors', 77: 'teddy bear', 78: 'hair drier', 79:
'toothbrush'}
===== USD Group 4 Class names =====
{0: 'Hardhat', 1: 'Mask', 2: 'NO-Hardhat', 3: 'NO-Mask', 4: 'NO-Safety Vest', 5:
'Person', 6: 'Safety Cone', 7: 'Safety Vest', 8: 'Machinery', 9: 'Vehicle'}
===== Community pre trained =====
None
{0: 'Hardhat', 1: 'Mask', 2: 'NO-Hardhat', 3: 'NO-Mask', 4: 'NO-Safety Vest', 5:
'Person', 6: 'Safety Cone', 7: 'Safety Vest', 8: 'machinery', 9: 'vehicle'}
```

[34]:
```python
# Predict the list of images and plot on a grid
def predict_ppes_grid(model, list_images, title=""):
    num_images  = len(list_images)
    num_cols    = min(num_images, 3) # minimum 2 columns
    num_rows    = int(num_images / num_cols) + (1 if num_images % num_cols != 0␣
 ↪else 0)

    # Create a grid of subplots.
    fig, axes = plt.subplots(num_rows, num_cols, figsize=(10, 9))
    list_axes = list(axes.flat)

    for i in range(num_images):
        file = list_images[i]
        # Open image file for reading and fix color grading
        img = cv2.imread(file)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

        # perform inference using model and draw predictions
        results = model.predict(img, verbose=False)
        summary = msaai.drawPredictions(results, img, model.names)
        list_axes[i].imshow(img)
        list_axes[i].axis('off')
        print(summary)

    plt.suptitle(title)
```

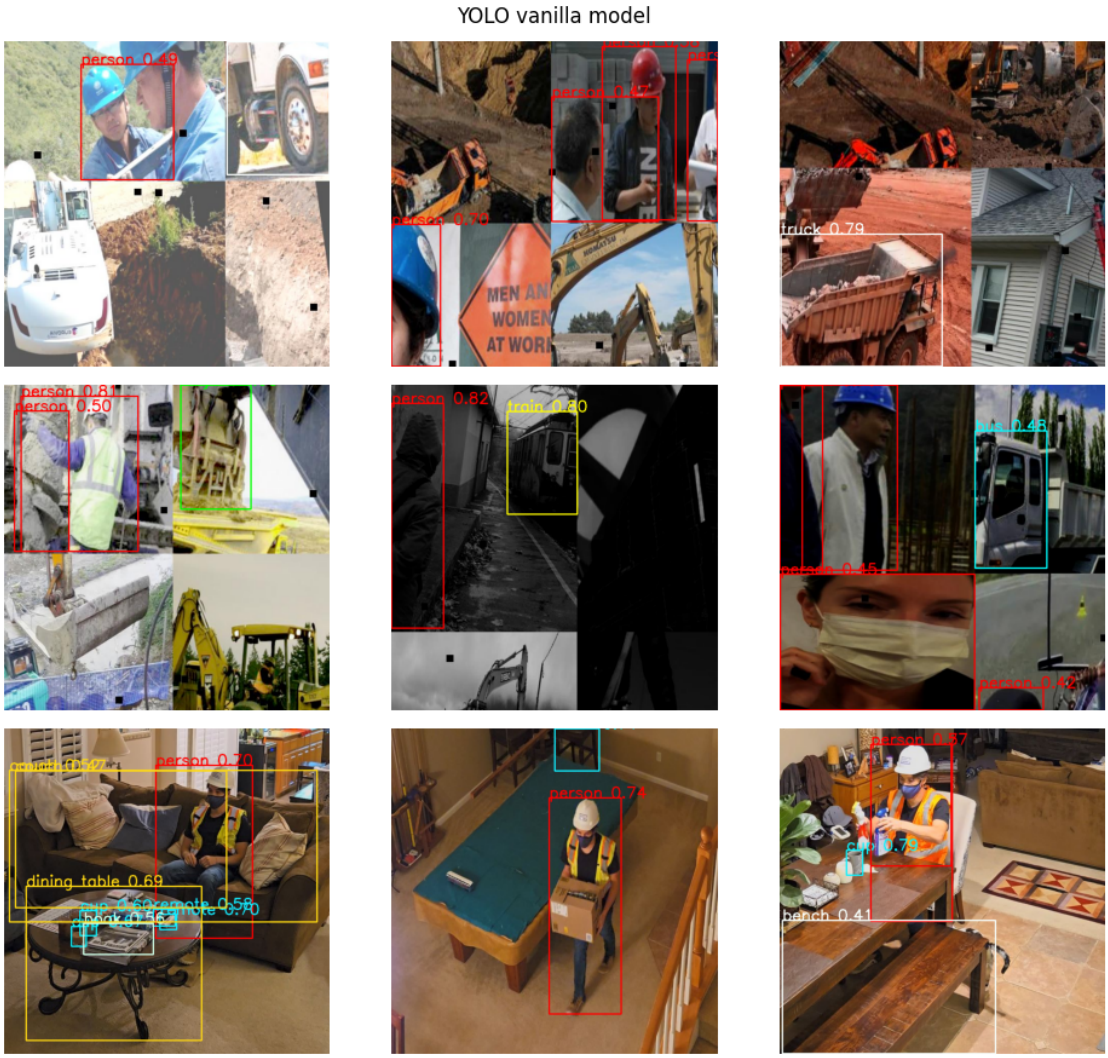
```
    plt.tight_layout()
    plt.show()
```

3 Get predictions from models

Using the loaded models and multiple images as examples, understanding how the models performed

[35]:
```
#Manually presenting annotations
# train files
base_dir="train_from_custom/datasets/raw_images/"

image_files=[f"{base_dir}01437_jpg.rf.7839171efbe212f40b2c6ce1f3c8819b.jpg",
             f"{base_dir}construction-619-_jpg.rf.
 ↪8181765b21d618233610f2c91a6057a4.jpg",
             f"{base_dir}construction-619-_jpg.rf.
 ↪dc01af03cac6c0e921cf55d5d8e8465d.jpg",
            f"{base_dir}005310_jpg.rf.83e6f0a246ede1e981765bf75cb7b067.jpg",
            f"{base_dir}2008_008382_jpg.rf.88abdb29b7db6d6218399b4adebdbe66.
 ↪jpg",
            f"{base_dir}-2270-_png_jpg.rf.4b73cc556e91bc8c587c923e348717f9.jpg",
            f"{base_dir}scene1_all_4.jpg",
            f"{base_dir}scene2_all_3.jpg",
            f"{base_dir}scene3_all_3.jpg",
]

predict_ppes_grid(model_default, image_files, title="YOLO vanilla model")
```

```
1 person, 1 truck, 1 boat,
6 persons,
1 truck,
2 persons, 1 bicycle,
1 person, 1 train,
4 persons, 1 bus,
2 persons, 3 cups, 3 couchs, 1 dining table, 2 remotes, 1 book,
1 person, 1 chair,
2 persons, 1 bench, 2 cups, 1 chair, 1 dining table,
```

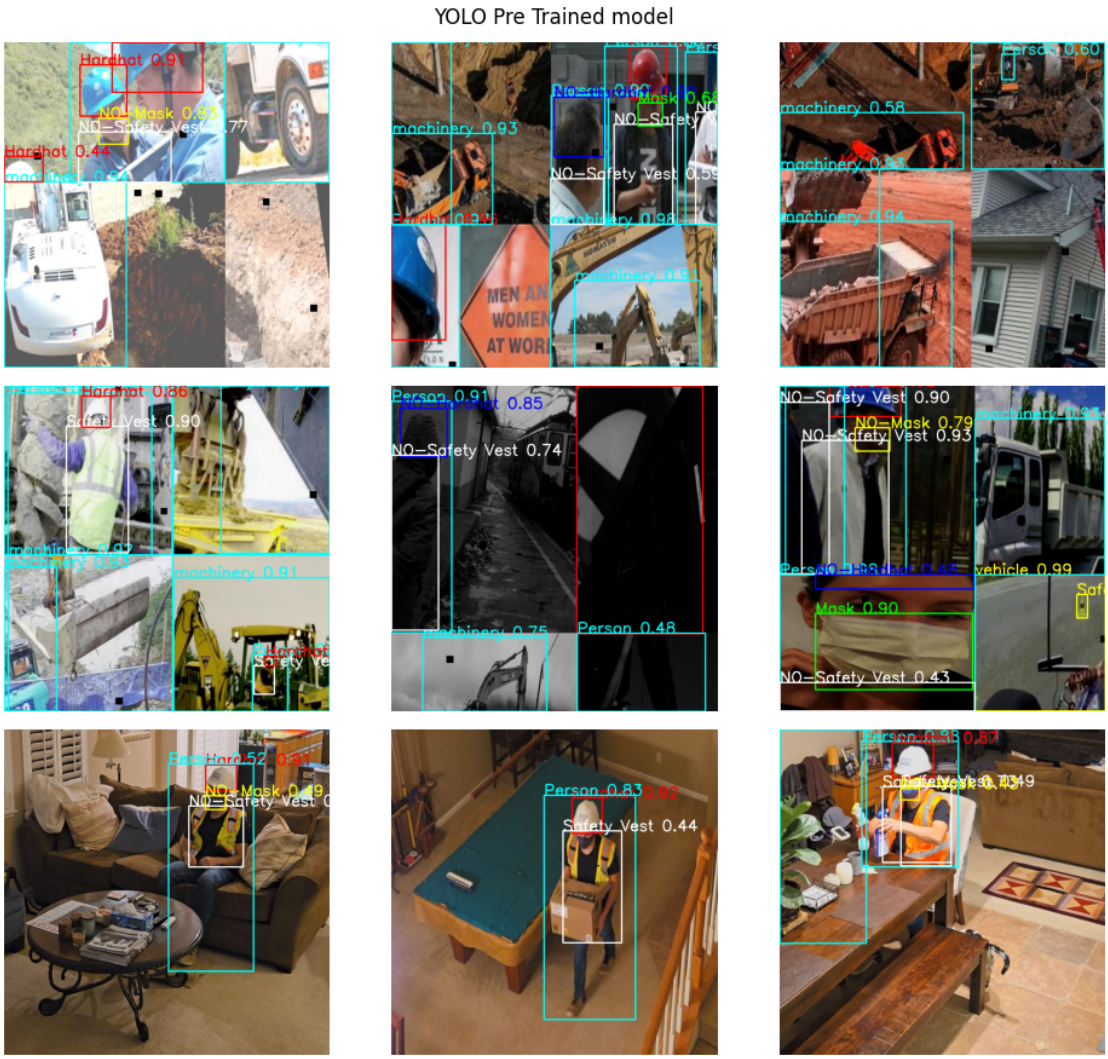


3.1 Pre-trained PEE Benchmark

[36]:
```
predict_ppes_grid(model_pre_trained, image_files, title="YOLO Pre Trained␣
↪model")
```

```
3 Hardhats, 1 NO-Mask, 1 NO-Safety Vest, 1 Person, 2 machinerys,
2 Hardhats, 1 Mask, 1 NO-Hardhat, 1 NO-Mask, 3 NO-Safety Vests, 4 Persons, 4
machinerys,
3 Persons, 5 machinerys,
2 Hardhats, 3 Persons, 2 Safety Vests, 7 machinerys,
1 Hardhat, 1 NO-Hardhat, 2 NO-Safety Vests, 2 Persons, 1 machinery,
1 Hardhat, 1 Mask, 1 NO-Hardhat, 1 NO-Mask, 3 NO-Safety Vests, 3 Persons, 1
Safety Cone, 1 machinery, 1 vehicle,
1 Hardhat, 2 NO-Masks, 1 NO-Safety Vest, 1 Person,
1 Hardhat, 1 Person, 1 Safety Vest,
```

1 Hardhat, 1 NO-Mask, 2 Persons, 2 Safety Vests,

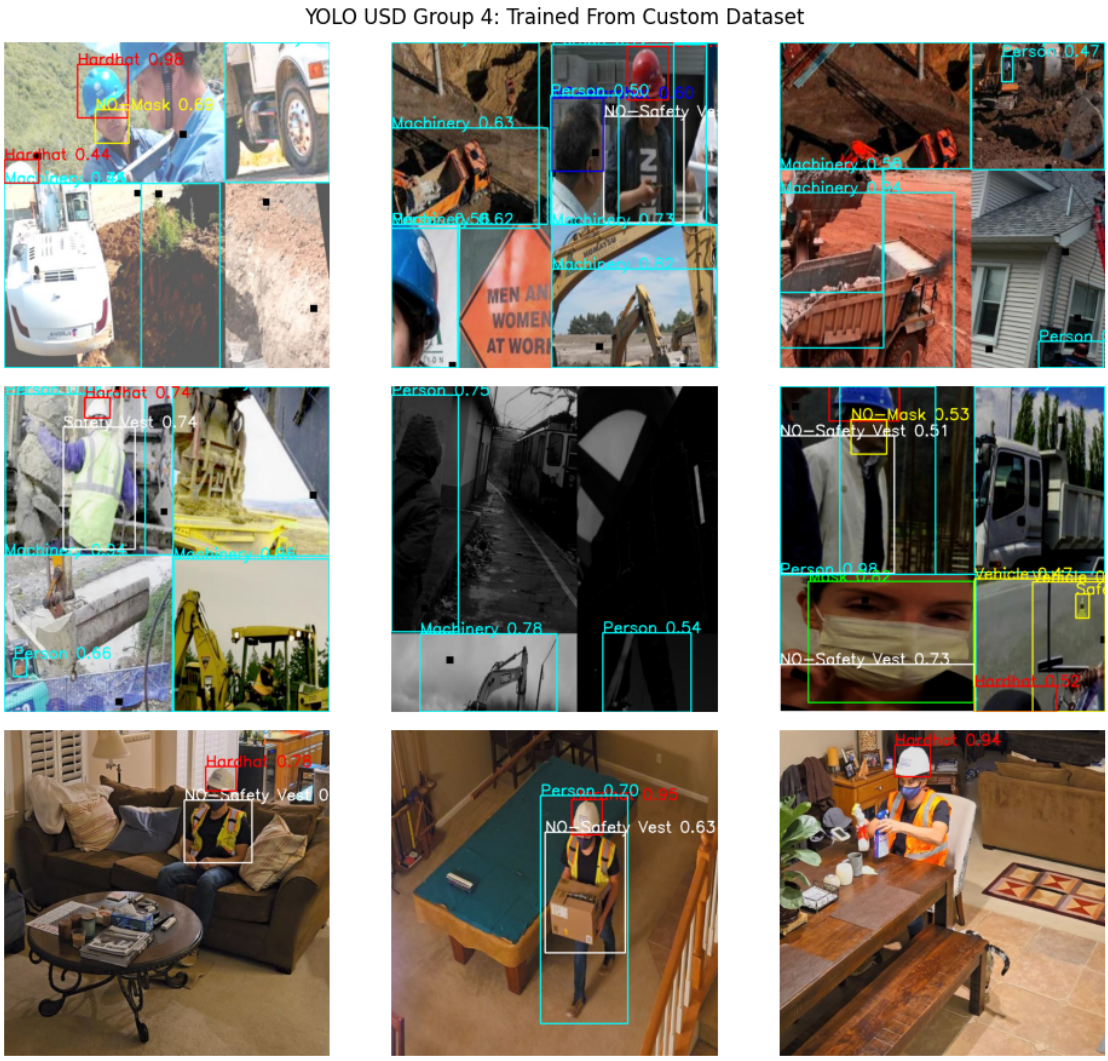


3.2 USD Group 4: Trained From Custom Dataset

[37]:
```
predict_ppes_grid(model_usd_g4, image_files, title="YOLO USD Group 4: Trained␣
↪From Custom Dataset")
```

2 Hardhats, 1 NO-Mask, 1 Person, 4 Machinerys,
1 Hardhat, 1 NO-Hardhat, 1 NO-Mask, 1 NO-Safety Vest, 4 Persons, 6 Machinerys,
1 Hardhat, 2 Persons, 6 Machinerys,
1 Hardhat, 2 Persons, 1 Safety Vest, 4 Machinerys,
1 NO-Safety Vest, 2 Persons, 1 Machinery,
2 Hardhats, 1 Mask, 3 NO-Masks, 2 NO-Safety Vests, 4 Persons, 1 Safety Cone, 1 Machinery, 2 Vehicles,
1 Hardhat, 1 NO-Safety Vest,

```
1 Hardhat, 1 NO-Safety Vest, 1 Person,
1 Hardhat,
```



3.3 USD Group 4: Hybrid Model

[38]:
```
predict_ppes_grid(model_hybrid, image_files, title="YOLO USD Group 4: Hybrid␣
↪Model")
```

```
2 Hardhats, 1 Person, 2 Machinerys,
1 Hardhat, 2 NO-Hardhats, 1 NO-Mask, 1 NO-Safety Vest, 4 Persons, 3 Machinerys,
1 Person, 6 Machinerys,
1 Hardhat, 1 Person, 1 Safety Vest,
1 NO-Hardhat, 2 Persons, 2 Machinerys,
1 Hardhat, 1 Mask, 1 NO-Hardhat, 1 NO-Mask, 3 NO-Safety Vests, 3 Persons, 1
Safety Cone, 1 Machinery, 1 Vehicle,
```

1 Hardhat, 1 Person,
1 Hardhat, 1 Mask, 1 Person, 1 Safety Vest,
1 Hardhat, 1 NO-Mask, 1 Person,

YOLO USD Group 4: Hybrid Model

Video Stream Prediction

December 5, 2024

1 Video Recognition exercise

AAI-501 Group 4: Fatimat Atanda, Victor Hugo Germano, Darin Verduzco

1.1 Configuring the environment

[1]:
```
import cv2
from ultralytics import YOLO
import math
import os # file

import msaai

# MODEL loading and config
# model
model = YOLO("../models/best_custom.pt")

class_names = ['Hardhat','Mask','NO-Hardhat',
               'NO-Mask','NO-Safety Vest','Person',
               'Safety Cone','Safety Vest','Machinery','Vehicle']

# Create output folder
os.makedirs("../datasets/stream_result", exist_ok=True)
```

[2]:
```
# DRAWING METHODS

def predict_video_stream(camera, output, verbose=False, showWindow=True,␣
 ↪flipImage=False):
    while True:
        ret, frame = camera.read()
        if not ret:
            break

        if flipImage:
            frame = cv2.flip(frame, 1)

        results = model(frame, stream=True, verbose=False)
        msaai.drawPredictions(results, frame, class_names)
```

```
        # Write the frame to the output file
        output.write(frame)
        # Display the captured frame - Performance changes when not showing
        if showWindow:
            cv2.imshow('Camera', frame)

        # Press 'q' to exit the loop
        if cv2.waitKey(1) == ord('q'):
            break
```

1.2 Capture Livestream Video from webcam

1.2.1 Use 'q' to close the window

[3]:
```
# Open the default camera
cam = cv2.VideoCapture(0)

# Get the default frame width and height
frame_width = int(cam.get(cv2.CAP_PROP_FRAME_WIDTH))
frame_height = int(cam.get(cv2.CAP_PROP_FRAME_HEIGHT))


# Define the codec and create VideoWriter object
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
out = cv2.VideoWriter('../dataset/stream_result/output_webcam.mp4', fourcc, 20.
 ↪0, (frame_width, frame_height))


# START PREDICTION AND PRESENTATION
predict_video_stream(cam, out, verbose=False, showWindow=True, flipImage=True)


# Release the capture and writer objects
cam.release()
out.release()
cv2.destroyAllWindows()
cv2.waitKey(1)
```

```
2024-11-28 18:35:26.848 python3[53673:8311993] WARNING:
AVCaptureDeviceTypeExternal is deprecated for Continuity Cameras. Please use
AVCaptureDeviceTypeContinuityCamera and add
NSCameraUseContinuityCameraDeviceType to your Info.plist.
```

[3]:
```
-1
```

2 Analyzing Video File

Video download link:

https://uofsandiego-my.sharepoint.com/:v:/g/personal/dverduzco_sandiego_edu/Ebp8SRL0ZI9Fv

[3]:
```
# Open the default camera
cam = cv2.VideoCapture("../datasets/custom_ppe_videos/scene1.mp4")

# Get the default frame width and height
frame_width = int(cam.get(cv2.CAP_PROP_FRAME_WIDTH))
frame_height = int(cam.get(cv2.CAP_PROP_FRAME_HEIGHT))

# Define the codec and create VideoWriter object
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
out = cv2.VideoWriter('../datasets/stream_result/output_custom_video.mp4',␣
 ↪fourcc, 20.0, (frame_width, frame_height))


# START PREDICTION AND PRESENTATION
predict_video_stream(cam, out, verbose=False, showWindow=True)


# Release the capture and writer objects
cam.release()
out.release()
cv2.destroyAllWindows()
cv2.waitKey(1)
```

```
2024-11-28 18:51:47.362 python3[54299:8328625] +[IMKClient subclass]: chose
IMKClient_Modern
2024-11-28 18:51:47.362 python3[54299:8328625] +[IMKInputSession subclass]:
chose IMKInputSession_Modern
```

[3]:
```
-1
```