**CNST 6308 – DATA ANALYSIS IN CONSTRUCTION MANAGEMENT**



**Semester Project Report**

**PPE Detection for Construction Safety using YOLOv8.**

Performed under the guidance of **Dr. Lu Gao, Ph. D,**

Associate Professor of Construction Management.

**Executed by:**

Aiswarya Lalichetti (UH ID 2252761)

**Contents:**

[Abstract](#Abstract)……………………………………………………………………………………………3

[Overview](#Overview)…………………………………………………………………………………………..3

[Literature review](#LiteratureReview)…………………………………………………………………………………..4

[Problem definition](#ProblemDefinition)…………………………………………………………………………………4

[Chapter1: Data Collection](#Chapter1)…………………………………………………………………………5

[1.1 Composition and Structure of Dataset](#Compositionofdataset)………………………………………………………5

[1.2 PPE Class Map](#PPEClassmap)……………………………………………………………………………...6

[1.3 Details about dataset](#Detailsaboutdataset)………………………………………………………………………...6

[Chapter 2: Methodology](#chapter2)…………………………………………………………………………...6

[Chapter 3: Implementation Of Project](#chapter3).……………………………………………………………8

[3.1 Organizing Data paths](#Organisingdatapath)………………………………………………………………………8

[3.2 Defining Class Names](#DefiningClassNames)………………………………………………………………………9

[3.3 Retrieving and Sorting Filenames](#RetrieveingFilename)…………………………………………………………..9

[3.4 Storing File Names in lists](#StoringFileNames)…………………………………………………………………10

[3.5 Retrieving and Sorting Image File Names](#RetrievingImageNames)…………………………………………………10

[3.6 Comparison of Images and Labels of Dataset](#ComparisionofImagesandlabels)……………………………………………...10

[3.7 Data Analysis](#DataAnalysis)……………………………………………………………………………...10

[3.8 Processing Annotation Files](#ProcessingAnnotationFiles)……………………………………………………………….10

[3.9 Normalization](#Normalization)……………………………………………………………………………...11

[Chapter 4: Custom Object Detection – Yolov8](#Chapter4CustomObjectDetectionModel)………………………………………………...…12

[4.1 Use of Yolo](#UseofYolo)………………………………………………………………………………...13

[4.2 Visualizing Samples](#VisualizingSamples)……………………………………………………………………….14

[4.3 Setting Up YAML Configuration File](#SettingupYAMLconfigurationfile)…………………………………………………….15

[4.4 Training a Custom Model](#TrainingCustomModel)…………………………………………………………………16

[4.5 Visualizing Results from YOLOv8 Object Detection Model](#VisualizingResultsfromYOLOv8object)……………………………..17

[Chapter 5: Prediction and it’s Results](#PredictionanditsResults)……………………………………………………………26

[Applications](#Applications) [& Challenges](#Challenges)……………………………………………………………………….27

[Conclusion](#Conclusion) [& Future Work](#FutureWork)……………………………………………………………………...28

[References](#Ref)………………………………………………………………………………………..29

**Abstract**

This project focuses on the construction industry, where the lack of proper safety equipment use is a significant concern. Our goal is to identify and locate objects within an image or a video using YOLOv8 refers to the You Only Look Once (YOLO) version 8, which is a real-time object detection algorithm. Object detection is a computer vision task where the goal is to detect the use of Personal Protective Equipment (PPE) among workers. This detection capability is a crucial step towards developing a comprehensive safety monitoring system, which could include real-time tracking and alarm triggering functionalities.

Drawing from a robust dataset provided by Roboflow, featuring 2801 annotated images in YoloV8 format, the system is trained to recognize and classify 10 key classes relevant to construction safety, including various PPE items and construction equipment. The dataset, split into training, validation, and testing sets, enables comprehensive training and evaluation of the model.

This capability is crucial not only in ensuring adherence to safety protocols but also in significantly reducing the risk of accidents and fatalities. Successful implementation has the potential to transform safety standards in construction sites globally, paving the way for similar applications across various industrial sectors.

**Overview**

The Occupational Safety and Health Administration (US Department of Labor) has reported alarming statistics: in 2020, 4,764 workers died on the job, translating to 3.4 fatalities per 100,000 full-time equivalent workers. Remarkably, jobs involving material movement, building, extraction, and transportation accounted for over half of all fatal occupational injuries. This emphasizes how important it is to upgrade safety precautions in these high-risk industries, especially construction.

The Construction Site Safety Image Dataset serves as a vital tool in the realm of AI-driven safety monitoring in construction areas. Accidents in construction sites, often attributed to the non-compliance with safety measures such as the use of Personal Protective Equipment (PPE), are a significant concern. This dataset specifically addresses the need for automated monitoring of safety equipment usage among workers, which is crucial for enhancing on-site safety standards and preventing accidents. We used YOLOv8 for object detection. The YOLO algorithm family is known for its speed and efficiency in object detection tasks. YOLO divides an image into a grid and predicts bounding boxes and class probabilities for each grid cell. YOLOv8 is an evolution of the YOLO series, with each version incorporating improvements and optimizations.

**Literature Review**

There is a growing consensus on the critical role of AI in enhancing workplace safety, particularly in high-risk environments like construction sites. Studies indicate that while traditional PPE monitoring methods are necessary, they often fall short due to their reactive nature and human resource limitations. The integration of AI and computer vision technologies, specifically advanced object detection systems like YOLOv8, has been identified as a game-changer, offering real-time analysis and increased accuracy in PPE compliance monitoring.

Recent research points toward the effectiveness of deep learning models in identifying and classifying objects with high precision, even in complex visual scenes typical of construction sites. However, there remains a significant gap in the deployment of these models in practical, on-site applications. Our project addresses this gap by not only fine-tuning YOLOv8 for the specific demands of construction safety but also by creating a dataset tailored to the nuanced requirements of PPE detection.

In conclusion, while the potential of AI in construction safety is well-documented, our project stands out by providing a targeted solution that bridges the gap between theoretical research and practical, real-world application. We aim to set a new benchmark for PPE compliance and worker safety through the deployment of our YOLOv8-based model.

**Problem Definition**

The problem we're addressing with the "PPE Detection for Construction Safety using YOLOv8" project revolves around enhancing the safety of workers in construction environments. Specifically, the project aims to:

1. Automate the Monitoring of Safety Compliance:

Traditional methods rely on manual supervision to ensure that workers are wearing the correct Personal Protective Equipment (PPE) such as helmets, vests, and safety goggles. This manual process is time-consuming and prone to human error.

2. Reduce Workplace Accidents:

Construction sites are inherently risky, and the absence of proper PPE significantly increases the likelihood of injuries. A system that can promptly identify non-compliance can intervene more rapidly to prevent potential accidents.

3. Increase Efficiency and Coverage of Safety Checks:

Manual checks are limited by the number of supervisors and their ability to observe every worker at all times. An automated system can continuously monitor all areas of a site simultaneously.

The YOLOv8-powered AI solution seeks to tackle these challenges by providing a reliable, efficient, and real-time monitoring system that ensures adherence to safety protocols and ultimately reduces the risk of injuries on construction sites.

**Chapter 1: Dataset collection**

This dataset is provided as a collection in Roboflow. This dataset is a great collection of images, with the labels in the following 10 key class format: 'Hardhat', 'Mask', 'NO-Hardhat', 'NO-Mask', 'NO-Safety Vest', 'Person', 'Safety Cone', 'Safety Vest', 'machinery', 'vehicle'. It is very important in tracking and monitoring applications whether a person is wearing Hardhat or NO-Hardhat.

The dataset consists of 2801 image samples with labels in YoloV8 format. These images are split into train: 2605, valid: 114 and test: 82 sets. Each folder consists of images and labels folders.



Fig: Personal Protective Equipment

To assess a dataset on Construction Site Safety Image Dataset uploaded by Roboflow. The dataset can be accessed here: [Construction Site Safety Image Dataset](https://colab.research.google.com/corgiredirector?site=https%3A%2F%2Funiverse.roboflow.com%2Froboflow-universe-projects%2Fconstruction-site-safety)

Here is the drive link to download the dataset: <https://drive.google.com/drive/folders/1Z4soLIlCFebeXm88uq9flljpgAHHLfUU?usp=sharing>

The above link consists of three folders, one with the images (source files), css\_data folder containing train, valid, test folder and the other is the results folder. To understand the images in the dataset we have visualized the images and their annotations. Since we have all the annotations in YoloV8 format we converted these annotation files to bounding box coordinates.

**1.1 Composition and Structure of Dataset:**

Number of Classes: 10 distinct classes

Label Annotation Format: YOLO (You Only Look Once) format, stored in .txt files. This format is particularly suited for real-time object detection tasks.

Metadata: Includes metadata.csv and count.csv, offering detailed insights into the dataset composition and the distribution of training, validation, and testing samples.

**1.2** **PPE Class Map:**

|  |  |  |
| --- | --- | --- |
| **No.** | **Safety Equipment Category** | **Assigned Class** |
| 1. | Hardhat | 0 |
| 2. | Mask | 1 |
| 3. | NO-Hardhat | 2 |
| 4. | NO-Mask | 3 |
| 5. | NO-Safety Vest | 4 |
| 6. | Person | 5 |
| 7. | Safety Cone | 6 |
| 8. | Safety Vest | 7 |
| 9. | Machinery | 8 |
| 10. | Vehicle | 9 |

**1.3** **Details about dataset:**

The dataset includes high-quality images capturing various construction scenarios, lighting conditions, and perspectives to ensure robust model training. What sets this data set apart is its specific focus on both the presence and absence of key safety equipment (e.g., 'Hardhat' vs. 'NO-Hardhat'). This nuanced categorization is crucial for real-world applications, as it enables the AI models to identify not just what is present but also what is missing, which is critical for safety compliance monitoring.

**Chapter 2: Methodology**

The following are the steps performed in our project:

* Exploratory Data Analysis
* Visualization Of Class Distribution
* Visualizing Samples
* Run Custom Object Detection – Yolov8
* Training Custom Model
* Visualizing Results of Trained Model
* Predictions
* Visualize Predictions

**Exploratory Data Analysis:** Exploratory Data Analysis (EDA) is an analysis approach that identifies general patterns in the data. These patterns include outliers and features of the data that might be unexpected. It is an approach to analyze datasets to summarize their main characteristics, often with unexpected visual methods. We first did an exploratory data analysis on the CSS Dataset and tried to see the distribution of train, valid and test sets.

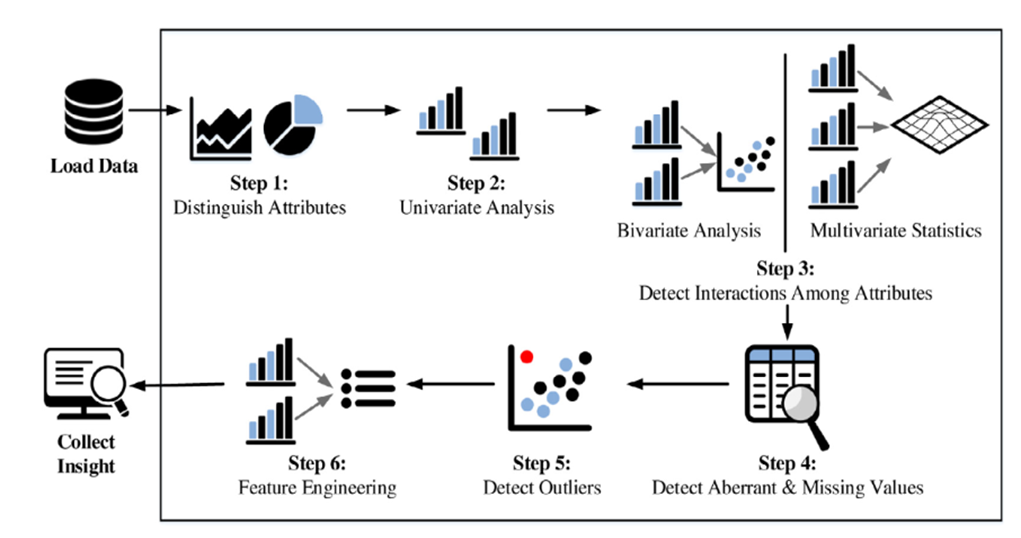


Fig: A Typical EDA Process

**Visualization of Class Distribution:**

In this dataset, each image consists of more than 1 class annotation in general. So, we need to check the total number of annotations in each image only then we can visualize overall class distribution.

There are three cases when we read the annotation file.

1. Annotation file is empty.

len(annotation) == 0 We will skip these files during training and evaluation of our object detection model.

2. Annotation file has only one annotation.

len(annotation.shape) == 1 We will reshape the array and get the annotations.

3. Annotation file has multiple annotations.

len(annotation) > 1 We will get the annotations directly.

**Visualization of Samples:**

Now, the next step is to visualize some samples from the dataset and their annotations. Since we have all the annotations in YoloV8 format in this step we will start with converting these annotation files to bounding box coordinates.

YoloV8 format for annotation has 5 entries in each line of the text file.

* Class Label (c)
* Bounding Box's center (X coordinate)
* Bounding Box's center (Y coordinate)
* Width of Bounding Box (w)
* Height of Bounding Box (h)

The bounding box can be calculated from the last 4 entries in the annotation file.

**Run Custom Object Detection:**

After the visualization of dataset, it is the time to train our custom object detection. We will use YoloV8 to train our custom object detection model.

**Training Custom Model:**

We will use the Command Line Interface (CLI) command for training a custom model using YoloV8.

**Visualize results of Trained Model:**

Here, we will visualize the results we got during the training of custom model like PR\_Curve, F1\_Curve, R\_Curve, Label\_Correlogram, etc.

**Predictions:**

We will run the custom object detection model to predict PPEs on source files. We have 3 videos and 4 images on which we have made predictions.

**Visualize Predictions:**

In this step, we will only visualize image predictions as the video files are heavy.

**Chapter 3: Implementation of Project**

**3.1 Organizing Data paths:**

The first step is setting up the paths for the data used in your project, particularly organizing the paths for training, validation, and testing datasets, as well as an output path for saving results.

We organized the data into three sets:

* Train Set: Used to train the model.
* Validation Set: Used to tune the hyperparameters and evaluate the models.
* Test Set: Used to test the model after the model has gone through initial vetting by the validation set.

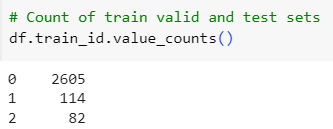


Fig: Display Count of Train-valid-test sets



Fig: Representative of data set split

The graph calculates the frequency of each unique value in the DataFrame df, where the x-axis represents the dataset splits (train, validation, test), and the y-axis represents the count of samples in each split These values correspond to the different dataset splits (“0 for train”, “1 for valid”, and “2 for test”).

**3.2** **Defining Class Names**

We classified a total of 10 classes from our dataset.

**[Hardhat', 'Mask', 'NO-Hardhat', 'NO-Mask', 'NO-Safety Vest', 'Person', 'Safety Cone', 'Safety Vest', 'machinery', 'vehicle']**

class\_names is a list of strings representing the classes that are used for object detection tasks. This includes various safety equipment and other relevant objects like 'Hardhat', 'Mask', 'Safety Vest' etc.

The next step is to get a list of filenames for the images and their associated labels in each dataset (train, valid, and test). Sorting these filenames is crucial to ensure that each image aligns with its correct label.

**3.3** **Retrieving and Sorting Filenames**

The filenames are sorted using Python's built-in sorted () function. This is an essential step because it ensures that the order of the image files matches the order of the label files.

Consistency in ordering is particularly important in object detection tasks, where each image must be associated with the correct label file.

**3.4** **Storing File Names in Lists**

The sorted filenames for images and labels are stored in separate lists: train\_filenames, valid\_filenames, test\_filenames for image files, and train\_labels, valid\_labels, test\_labels for label files. These lists are specific to each dataset split (training, validation, and testing).

We have organized lists of image and label filenames for each of your dataset splits. This organization is fundamental for the next steps in our project, where we load these images and labels for processing, model training, and evaluation. The alignment of images with their labels is critical for ensuring the accuracy and effectiveness of your object detection model.

List comprehension is a concise way to create lists. It consists of brackets containing an expression followed by a for clause, then zero or more for or if clauses. In this case, list comprehension is used to condense the multiple lines of code needed to retrieve and sort filenames into single lines for each of images and labels.

**3.5** **Retrieving and Sorting Image Filenames**

Similarly, t\_l, v\_l, te\_l are lists for the labels in the train, validation, and test datasets.

The process is the same as we have done for images, but we use folders (which correspond to the 'labels' folder).

**3.6** **Comparision of Images and labels of data set**

The code compares the filenames of the images and labels in each dataset split (train, valid, test) to ensure they are aligned correctly. For each filename in the image and label lists, it strips the file extension (like .jpg for images or .txt for labels) and compares the base names.

This is done by using a list comprehension ([item.split('.')[0] for item in train\_filenames]) that iterates through each list, splits the filename at the period (which typically starts the file extension), and takes the first part of the split (which is the base filename without the extension).The comparison is then made between these lists of base filenames for images and labels.

Importance of This Check:

Matching Images and Labels: In object detection tasks, it's critical that each image is associated with the correct label. This check ensures that the ordering is consistent, so that when the model is being trained, it learns from the correct annotations for each image.

**3.7** **Data Analysis**

Data analysis is a process where a DataFrame (df) is used to manage and inspect your dataset. Let's understand the steps involved in this process:

1. Checking for Duplicate Entries:

The purpose is to ensure that there are no duplicate images or label filenames in the dataset, as duplicates could lead to issues in training or evaluation of the model.

2. Counting Entries in Train, Validation, and Test Sets:

This is useful for understanding the distribution of data across the different sets, which is important to ensure that the model is trained on a representative sample of data and validated/tested on appropriate unseen data.

3.Visualizing the dataset distribution:

We represent the distribution of dataset across the different splits (training, validation, and test sets) using a bar chart. It provides a clear visual representation of how many samples are in each of the trains, validation, and test sets. This is an effective way to quickly grasp the size and distribution of the dataset.

**3.8** **Processing Annotation files**

It focuses on loading the files, checking their validity, and updating the Data Frame with this information. Validating the annotation files of our dataset ensures that they are correctly formatted and matched with their respective images, and thus preparing the Data Frame for subsequent steps in our object detection project.

Summarizing the counts of different classes in each of the annotation files is a bit complex, involving several steps to extract, organize, and store this information.

**3.9** **Normalization**

Normalizing the counts in each column (train, valid, test) by dividing each count by the total count in that column. lambda x: x/df\_count.train.sum(), divides each count in the train column by the total sum of counts in the train column.

This normalization is repeated for the valid and test columns as well. We normalized the counts to get a correct visualization, since there is huge difference between the counts of train valid and test splits

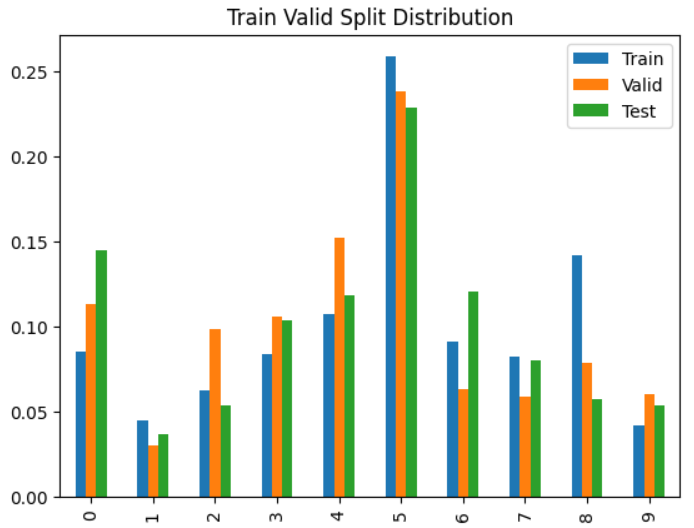


Fig: Train Valid Split Distribution

From the distribution graph, it can be observed that each class has been distributed along train, valid, test splits. Each class range is normalized in the range between 0 and 1. With Class "person" having higher frequency compared to other classes.

We can see the normalized counts of all the classes are balanced, this will ensure that we are not seeing some classes more often than others during training as the distribution is almost same in all classes

**Chapter 4:** **Custom Object Detection Model**

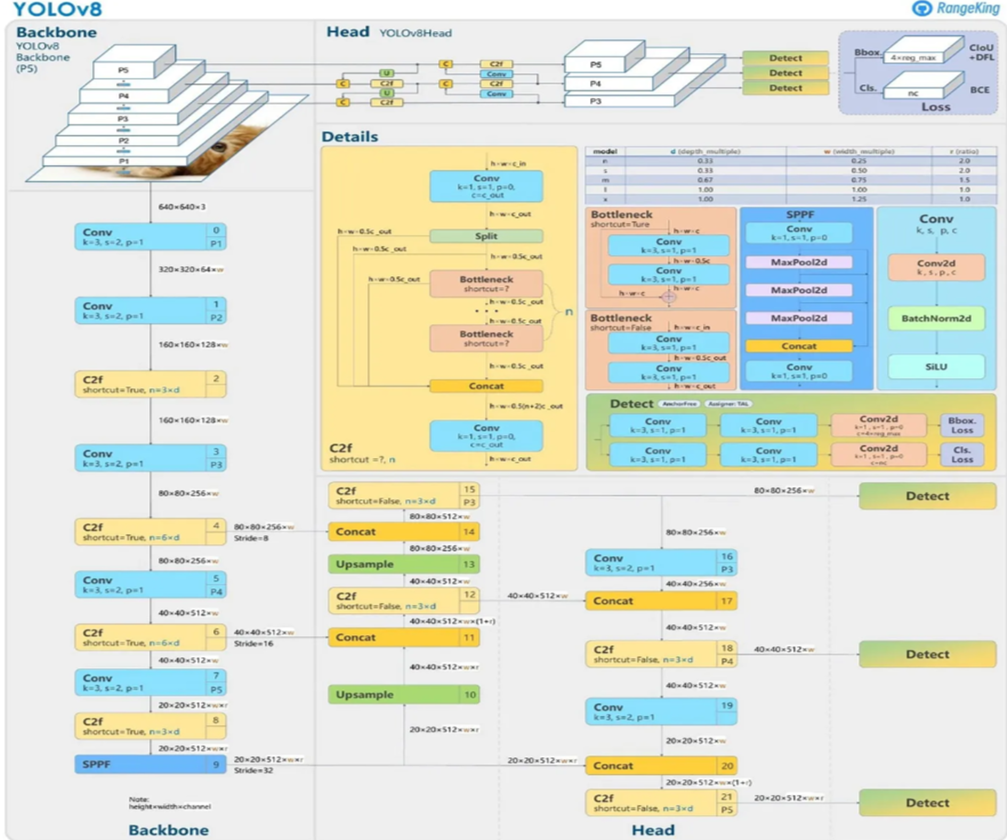


Fig: YOLOv8 Architecture

YOLOv8 has better accuracy than previous YOLO models. It supports object detection, instance segmentation, and image classification. Training of YOLOv8 will be probably faster than the other two-stage object detection models.

YOLOv8 has a simple annotation format which is the same as the YOLOv5 PyTorch TXT annotation format, a modified version of the Darknet annotation format. Every image sample has one .txt file with one line for each bounding box.

**4.1** **Use of Yolo:**

The function yolo\_annotation\_to\_bbox is designed to convert YOLO format annotations into bounding box coordinates. The YOLO format typically defines objects in an image via bounding boxes specified by the center coordinates, width, and height relative to the image size. Let's go through the function used in our code:

Function Definition:

def yolo\_annotation\_to\_bbox(annotation, img\_height, img\_width):

The function takes three arguments:

annotation: The YOLO annotation data for an image. This is expected to be a NumPy array where each row corresponds to an object and contains the class identifier and the normalized center coordinates (x, y), width, and height of the bounding box.

img\_height: The height of the image the annotations are for.

img\_width: The width of the image.

A function visualize\_samples is used for plotting a specified number of sample images from the train, validation, or test set of the dataset. Each image is displayed with its bounding boxes and class labels as annotated.

Plotting Process:

For each sample, the function loads the image and its annotation file. It converts the YOLO annotations to bounding box coordinates using the yolo\_annotation\_to\_bbox function. It then draws each bounding box on the image and annotates it with the corresponding class label.

YoloV8 format for annotation has 5 entries in each line of the text file.

* Class Label (c): Represents the category or class of the object in the image.
* Bounding Box's center (X, Y coordinate): Specifies the coordinates of the center of the bounding box.
* Width of Bounding Box (w): Represents the width of the bounding box.
* Height of Bounding Box (h): Indicates the height of the bounding box.

To calculate the bounding box coordinates:

The top-left corner coordinates (x1, y1) can be calculated as follows:

x1 = X - (w / 2)

y1 = Y - (h / 2)

The bottom-right corner coordinates (x2, y2) can be calculated as follows:

x2 = X + (w / 2)

y2 = Y + (h / 2)

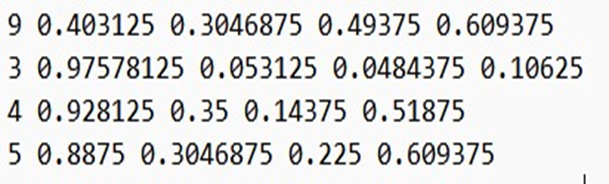


Fig: Annotated labeled text file, where each line corresponds to a different object in the image.

Explanation of annotated txt file

•Class Label (c): 9 (9 is the class for “person”)

•Bounding Box's Center (X, Y coordinates): (0.4031, 0.3047)

•Width of Bounding Box (w): 0.4938

•Height of Bounding Box (h): 0.6094

Therefore, the YOLOv8 annotation format encodes object information with a class label and the bounding box's center coordinates, width, and height. The actual bounding box coordinates (top-left and bottom-right corners) can be derived from this information.

**4.2** **Visualizing Samples**

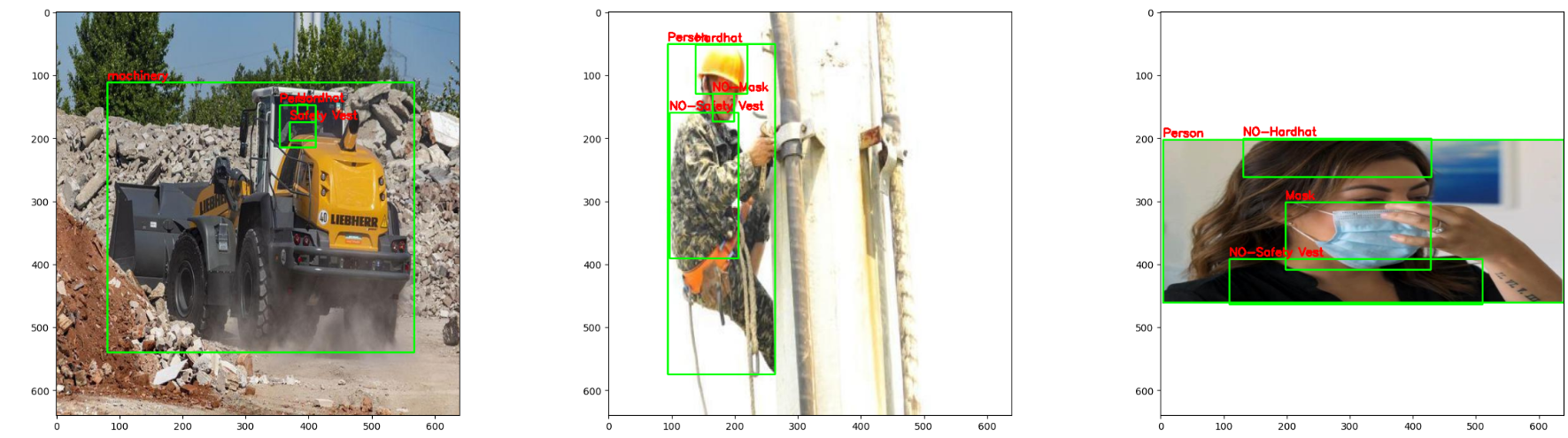
The images are displayed in a grid layout (4x3 in this case, as there are 12 samples in each split visualization). Each subplot shows one image with its bounding boxes and labels.

However, for the convenience of report presentation the first three images from the visualization output are displayed in each train, valid and test sets.

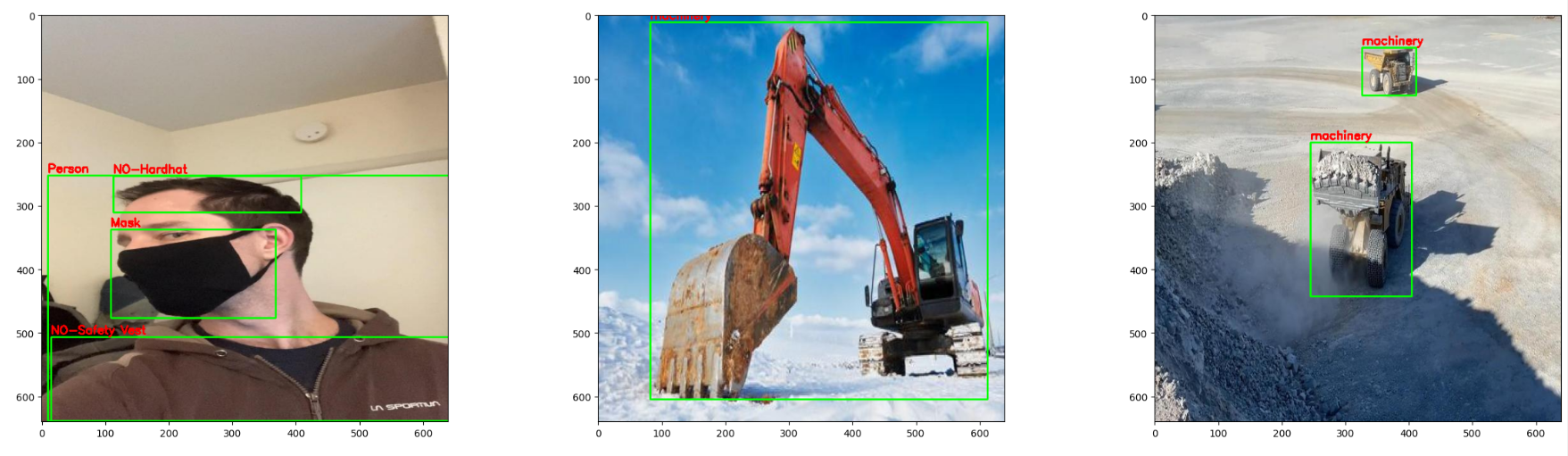
**Visualizing sample images of trained set:**



**Visualizing sample images of valid set:**



**Visualizing sample images of test set:**



Purpose and Visualization Insights:

Visual Check of Data Quality: This visualization is particularly useful for checking the quality of the annotations. It helps ensure that the objects in the images are correctly labeled and that the bounding boxes are accurately drawn.

Understanding Dataset Characteristics: It provides a quick way to understand the types of images and objects, which can be crucial for tuning models and understanding their performance.

Debugging and Presentation: If there are any issues with the annotations or if certain types of objects are more prevalent, this function will help highlight those aspects. It's also useful for presenting examples of our dataset.

**4.3** **Setting up YAML configuration file:**

We have to setup YAML configuration file for a YOLOv8 object detection task, particularly for a dataset related to Personal Protective Equipment (PPE). Here's a breakdown of each step:

Installing the ultralytics Package:

!pip install -q ultralytics

This command installs the ultralytics Python package, to use Ultralytics YOLOv8 for object detection tasks.

Creating the YAML Configuration Dictionary:

ppe\_data = dict(train = train\_path, val = valid\_path, test = test\_path, nc = len(class\_names), names = class\_names)

This line creates a dictionary named ppe\_data with several key-value pairs:

Writing the Configuration to a YAML File:

with open ('ppe\_data.yaml', 'w') as output: yaml.dump(ppe\_data, output, default\_flow\_style = True)

This code opens a file named ppe\_data.yaml in write mode and uses the yaml.dump function to serialize the ppe\_data dictionary into this file.

default\_flow\_style = True controls how the data is formatted in the YAML file.

Displaying the Contents of the YAML File:

%cat "/content/drive/MyDrive/archive (1)/results\_yolov8n\_100e/kaggle/working/ppe\_data.yaml"

This command is used to display the contents of the YAML file created. Purpose and

Importance:

Configuration for YOLOv8: This YAML file serves as a configuration for training a YOLOv8 model. It specifies where the model can find the data and what classes it should be looking for. This configuration file is a key component in setting up a YOLOv8 training pipeline.

Organizing Model Training: This setup is essential for organizing and standardizing the training process, making it easier to manage, especially for complex tasks like object detection.

Ease of Use: YAML files are human-readable and are commonly used in machine learning and data processing tasks for configuration due to their simplicity and clarity.

## **4.4** **Training a custom model:**

We used the Command Line Interface (CLI) command for training a custom model using YoloV8.

! yolo task=detect mode=train epochs=2 data='/content/ppe\_data.yaml' model=yolov8n.pt imgsz=640 patience=10

The command uses YOLO (You Only Look Once) for an object detection task, specifically to train a model using the YOLOv8 framework.

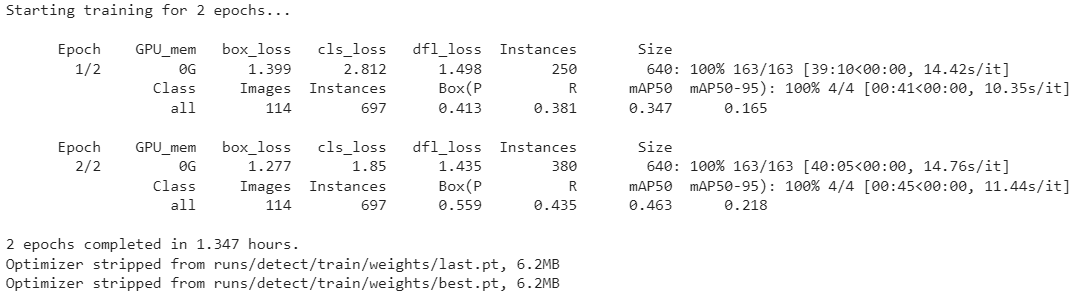


Fig: Using epochs for training custom model using Yolov8

Command Parameters used:

task=detect: Specifies that the task is object detection.

mode=train: Indicates that the model should be trained.

epochs=2: Sets the number of training epochs to 2. An epoch is a full training cycle on the entire dataset.

data='/content/ppe\_data.yaml': Points to the YAML file that contains the dataset configuration (paths and class information).

model=yolov8n.pt: Specifies the pre-trained model to use, in this case, 'yolov8n.pt'. This might be a path to a model file.

imgsz=640: Sets the image size for training to 640x640 pixels.

patience=10: Likely refers to the number of epochs with no improvement after which training will be stopped.

Purpose and Importance:

Accessing Model Training Results: It is crucial to access and review the results of our YOLOv8 model training. The results include metrics like loss, accuracy, precision, recall, or other relevant performance indicators.

Initial Data Inspection: Viewing the first few rows of the results DataFrame is a standard practice for initial data inspection, ensuring that the results are loaded correctly and formatted as expected.

File Management: The commented zip command and the path setup indicate good practices in managing and retrieving training results, especially when working in cloud-based or remote computing environments.

**4.5** **Visualizing Results from YOLOv8 object detection model**

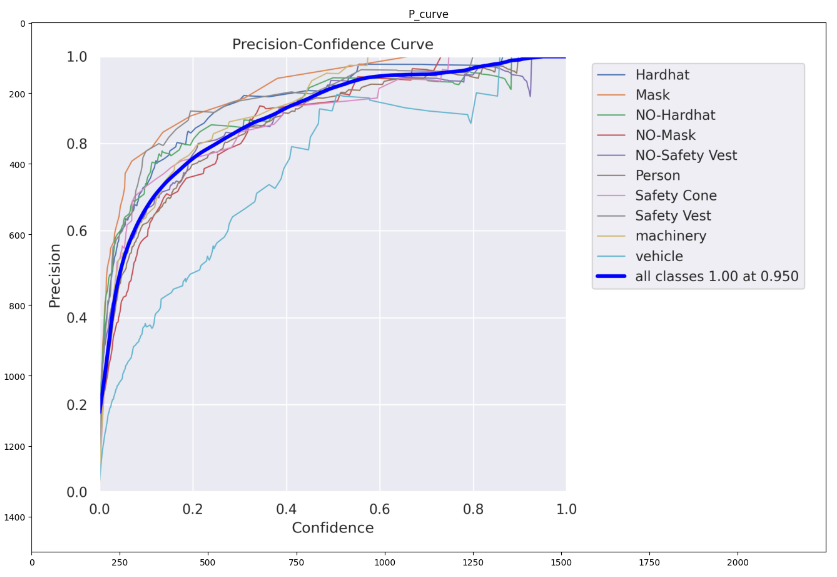


Fig: Precision - Confidence Curve

Curve Explanation:

This graph shows how the precision of the model's predictions varies with different confidence thresholds for different classes (like Hardhat, Mask, etc.).

Precision is on the y-axis, and it measures the proportion of true positives among all the instances that the model predicted as positives for a given class.

Confidence, on the x-axis, represents the model's threshold for determining whether a detection is considered positive. A higher threshold means the model is more certain about its predictions.

Class-Specific Curves:

Each line represents a different class that the model is detecting. The line's progression shows how confident the model is in its predictions for that class and the corresponding precision.

For example, if the line for 'Hardhat' is above the line for 'NO-Hardhat', it suggests that the model predicts 'Hardhat' with higher precision at the same confidence levels.

General Performance:

The line labeled "all classes" shows the overall precision of the model across all classes at different confidence thresholds. A precision of 1.00 at a confidence of 0.950 means that when the model is 95% confident, it is virtually always correct in its predictions.

Insights for PPE Detection:

The closer the lines are to the top-right corner of the graph, the better the model is at detecting those classes with high confidence and precision.

The graph provides insight into which classes are being predicted more accurately and at what confidence levels. This is crucial for setting the confidence threshold in a production environment where the model will be used for detecting PPE compliance on construction sites.

For your PPE detection project, you can use this graph to determine the optimal confidence threshold that balances the need for high precision (to avoid false positives) while maintaining a practical level of confidence for real-world application. For example, you might choose a confidence threshold that ensures a precision of at least 0.80 for all classes to ensure reliable detection.

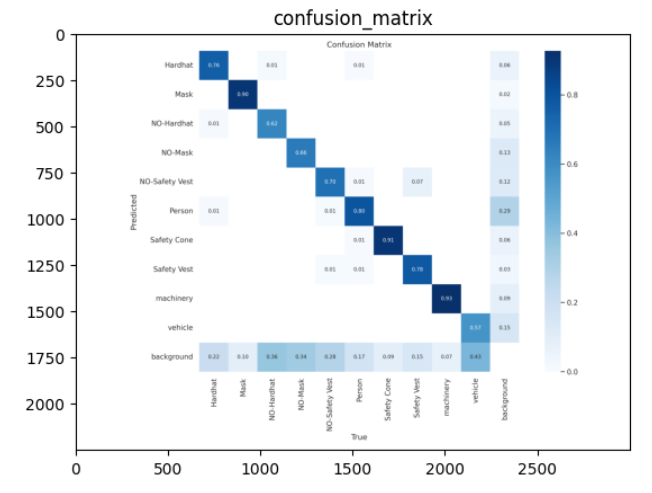


Fig: Confusion Matrix

The image likely shows a confusion matrix for your PPE detection project using YOLOv8, which is a table used to evaluate the accuracy of a classification model. Each row represents the instances in an actual class while each column represents the instances in a predicted class. The diagonal cells, which ideally should be the highest numbers in their respective rows and columns, show the number of correct predictions for each class (like 'Hardhat', 'Mask', etc.). The off-diagonal cells indicate misclassifications, where the model has predicted the wrong class. This matrix helps in identifying which classes are being confused with others, allowing for targeted improvements in the model or training data. For instance, if 'NO-Hardhat' has high numbers off the diagonal, it means the model frequently misclassifies this class, which is crucial to address for a safety-critical application like PPE detection.

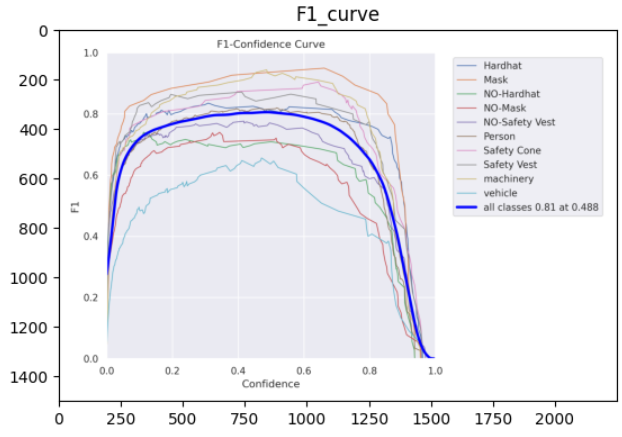


Fig: F1-Confidence Curve

F1-Confidence Curve, which is a visualization used to evaluate the performance of an object detection model like YOLOv8 across different confidence threshold levels.

F1 score is a harmonic means of precision and recall, providing a balance between the two. It's particularly useful when we need a single metric to compare models or when the class distribution is imbalanced. The F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

Confidence Threshold:

The confidence threshold is a value between 0 and 1 that determines how certain the model needs to be to classify a detection as positive. A higher threshold means the model is more certain about its predictions, while a lower threshold accepts more detections at the risk of increasing false positives.

F1-Confidence Curve:

This curve plots the F1 score (y-axis) against the confidence threshold (x-axis) for the model's predictions. Each line represents a different class that the model detects, such as 'Hardhat', 'Mask', etc. The peak of each line indicates the confidence threshold at which the F1 score is maximized for that class.

Interpreting the Curve:

A higher peak means that the model achieves a better balance of precision and recall for that particular class at a specific confidence level.

The curve generally rises with increasing confidence until it reaches an optimal point where the F1 score is highest. After that, it declines, which suggests that being too strict (requiring too high confidence) results in missing true positives (lower recall).

Overall Performance:

The graph also shows an "all classes" line that represents the average F1 score across all classes at different confidence levels. The mF1 (mean F1 score) value provides a summary of the model's overall performance.

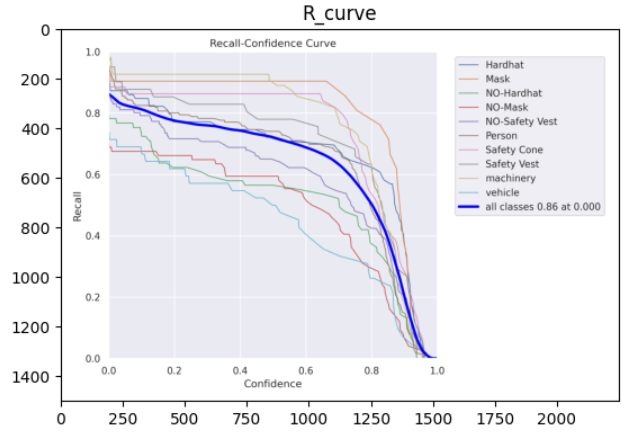


Fig: Recall- Confidence Curve

Recall: This metric indicates the model's ability to correctly identify all relevant instances of PPE in the dataset. A higher recall means the model misses fewer actual instances.

Confidence Threshold: This is the probability threshold the model uses to decide if a detection is considered positive. A higher threshold could lead to fewer false positives but might also miss more actual positives.

Curve Analysis: Each line represents a different PPE category such as 'Hardhat', 'Mask', etc. The curves show how recall changes as the confidence threshold is varied. The goal is to choose a threshold that balances recall without incurring too many false positives.

All Classes Curve: The dark blue line represents the average recall across all classes at different confidence levels. The text "all classes 0.86 at 0.000" suggests that when the confidence threshold is set to 0, the model achieves an average recall of 0.86 across all PPE categories.

For your PPE detection project, you would want a high recall to ensure all PPE items are detected, considering the safety implications of missing any item. However, setting the threshold too low could result in many false positives, which can undermine the system's usability. The graph helps in finding the sweet spot for the confidence threshold.

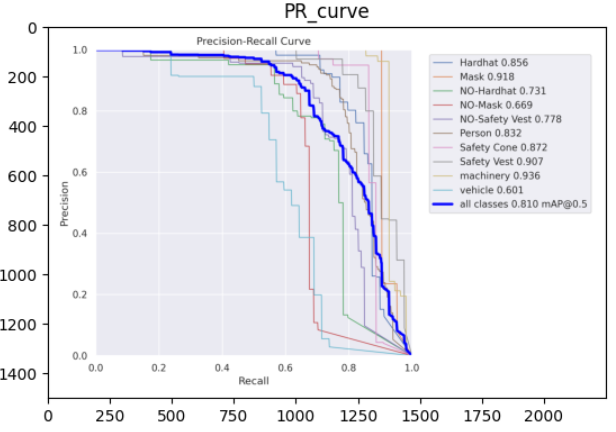


Fig: Precision-Recall Curve

As shown in the graph, we represented all the classes with different colors. The area under each curve can give you an indication of the overall performance of the model for that class; the closer the area is to 1, the better the model's performance.

Performance per Class:

The legend on the right side of the graph provides an Average Precision (AP) score for each class. The higher AP score means better performance.

Classes with higher AP scores, such as "Safety Cone" and "Machinery," are detected more accurately by the model, while classes like "NO-Hardhat" and "Vehicle" have lower AP scores, indicating room for improvement.

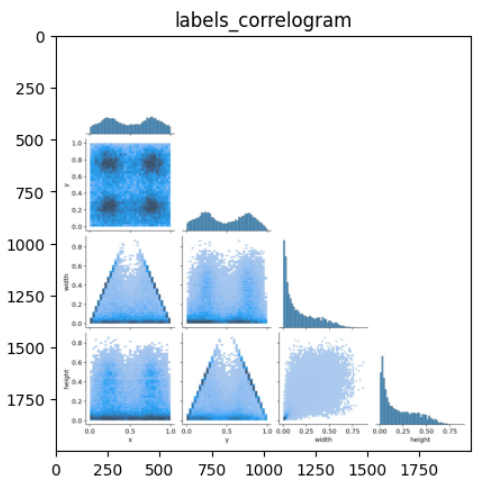


Fig: Correlogram

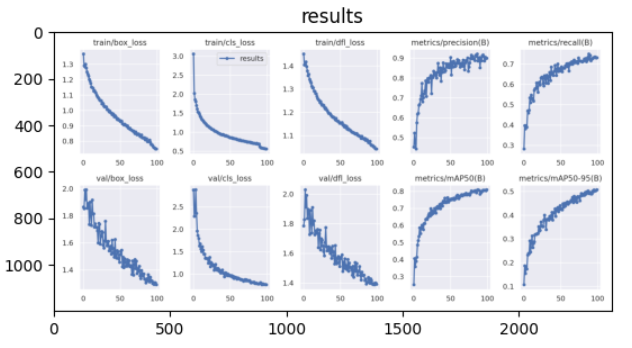
A correlogram, also known as a pair plot. This type of visualization is used to show the pairwise relationships between multiple variables in a dataset.

Here Variables used are normalized properties of bounding boxes x, y, (centered box co-ordinates), width, height. We observe scatter plots and histograms in this pair plot

Correlation Insights:

From graph, there's a triangular shape in the scatter plots involving width and height, suggests that there are common aspect ratios in the bounding boxes.

The x and y variables seem to be uniformly distributed, suggesting that the bounding boxes are spread evenly across the image area, which would be expected in a well-constructed dataset.



It shows a series of plots tracking various metrics

Loss Plots (Top and Bottom Left Panels):

show the model's loss metrics during training. The top plot typically represents training loss, while the bottom plot shows validation loss.

Box Loss: Reflects the model's performance on localizing objects (i.e., drawing the correct bounding box around each object).

Class Loss: Indicates how well the model is classifying objects within the bounding boxes.

Object Loss (obj loss): Represents the model's performance in detecting objects' presence within the bounding boxes.

A downward trend in these graphs is a good sign, indicating that the model is learning and improving its predictions over time.

Precision and Recall (Top and Bottom Middle-Right Panels):

An upward trend in these plots is desirable, showing that the model is becoming more accurate and comprehensive in its predictions as training progresses.

Mean Average Precision (Top and Bottom Right Panels):

[mAP@0.5](mailto:mAP@0.5), mAP@0.5:0.95 thresholds. It's a more stringent metric, providing a better picture of the model's performance across different levels of prediction difficulty. Higher mAP values indicate a better performing model.

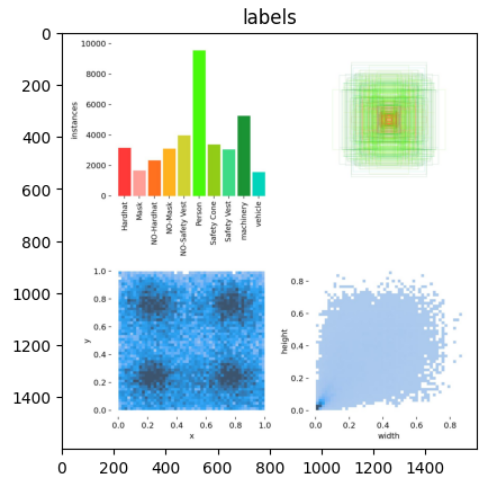


Fig: Composite Visualization

A composite visualization containing three different types of plots, used to analyze the characteristics of labeled data in an object detection dataset.

* Bar Chart (Top-Left):

This bar chart likely represents the frequency of each label (or class) in the dataset.

* Scatter Plot (Bottom-Left):

This scatter plot seems to show the distribution of bounding box center. If the points are densely packed in the center, it suggests that most objects are centrally located in the images.

A uniform distribution would suggest that the labeled objects are spread evenly across the images.

* Box Plot Overlay (Top-Right):

The overlay of box plots appears to show the distribution of bounding box sizes (width and height), Each layer of box plot might represent the statistical summary (median, quartiles, and extremes) of one class.

* Density Plot (Bottom-Right):

This plot shows the density of the width and height of bounding boxes. It gives an idea of the most common aspect ratios and sizes of labeled objects.

# **Chapter 5: Prediction and its Results**

We executed the custom object detection model to predict PPEs on source files. We have 3 videos and 4 images on which we run predictions. The predictions will be saved in the working directory. It utilizes the ultralytics package to load a trained YOLO model and then applies this model to predict objects in a set of test files.





Fig: Prediction Results

Purpose and Importance:

Object Detection on New Data: The code is set up to apply trained YOLO model to new, unseen data. This is a crucial step in evaluating the model's performance in real-world scenarios.

Handling Different Media Types: The code handles both image and video files, demonstrating the versatility of YOLO models in processing different types of media.

Automated Processing: The loop allows for automated processing of multiple files, which is efficient for testing the model on a large set of data.

The results of this prediction can be used to evaluate the model's performance qualitatively by inspecting the detected objects in the test images/videos.

**Applications**

Here are a few use cases for this project:

**1.Compliance Monitoring:** The Construction Site Safety model can be used by construction site managers, safety officers, or regulatory agencies to monitor and ensure that workers are adhering to safety protocols, such as wearing appropriate personal protective equipment (PPE).

**2.Accident Detection and Prevention:** The model can be integrated with surveillance or monitoring systems on construction sites to detect potentially hazardous situations, such as a person not wearing a hardhat or safety vest near heavy machinery, allowing for real-time intervention and accident prevention.

**3.Construction Site Access Control:** The model can be employed at entry and exit points of construction sites to identify and grant access only to authorized personnel wearing the proper safety gear, helping to maintain a safe working environment and prevent unauthorized access.

**4.Equipment and Vehicle Tracking:** The Construction Site Safety model can be used to automatically track the movement and usage of construction vehicles and machinery within the construction site, enabling better project management, fleet optimization, and maintenance scheduling.

**5. Safety Monitoring**

Automated monitoring of PPE usage, identifying workers without essential safety gear like hard hats, safety vests, and masks. This can significantly enhance on-site safety protocols.

**6. Intelligent Surveillance System**

Integrating the model into an intelligent surveillance system allows for real-time monitoring of PPE compliance. Such systems could be set up in construction sites, factories, or any workplace where safety equipment is mandatory.

Here are some general challenges that might be faced during project:

**Challenges:**

1.Determining the ideal number of epochs for training a custom YOLOv8 model through the CLI can be challenging. Too few epochs may lead to underfitting, while too many can result in overfitting and are time-consuming, necessitating a careful balance for optimal performance.

2. Implementing a YOLOv8 model for PPE detection in construction safety have specific challenges:

* Accurate annotation of PPE in images is time-consuming and requires attention to detail. Inconsistent or incorrect annotations can significantly affect model performance.
* Determining the ideal number of epochs for training a custom YOLOv8 model.
* Crowded scenes can create confusion in precise object detection as bounding box coordinates overlap, making it challenging to accurately detect specific objects based on class labels. Achieving a high level of precision is necessary to ensure safety compliance.

Each of these challenges needs to be addressed through thoughtful design, robust data preparation, careful model tuning, and rigorous testing to ensure the YOLOv8-based PPE detection system is effective and reliable.

**Conclusion**

Our project is a prime example of how computer vision can be leveraged to enhance safety and compliance in the workplace. The Construction Site Safety Image dataset is a critical resource for developing AI solutions focused on improving safety standards in construction sites. Its unique structure, comprehensive labeling, and focus on both presence and absence of safety equipment make it an invaluable tool for researchers, AI developers, and safety professionals in the construction industry. By leveraging this dataset, AI can play a pivotal role in reducing accidents and enhancing safety compliance in one of the most hazardous work environments.

## **Future Work**

Comparing the performance of all five YoloV8 models (Nano to Extra Large) will give insights into the trade-offs between speed, accuracy, and model size. This can help in choosing the right model based on the specific requirements of deployment environments (e.g., real-time detection on edge devices vs. offline analysis).

Implementing ID tracking and saving boxes of people who miss out on wearing hardhats, masks or vests will be a great use case for triggering alarms and monitoring.

Implementing a feedback mechanism where the model's detections are periodically reviewed and used for further training can continually improve its accuracy. This is particularly important in dynamic environments where PPE types or conditions may change over time.

**References**

[1] Aldossary, R. S., Almutairi, M. N., Alotaibi, N. M., & Serkan, D. (2023). Personal Protective Equipment Detection Using Computer Vision Techniques. DOI: 10.2118/216253-MS. Retrieved from https://www.researchgate.net/publication/374383063.

[2] Anka, A. (2020, May 17). YOLO v8: Optimal Speed & Accuracy for Object Detection. Retrieved from https://towardsdatascience.com/79896ed47b50#a112.

[3] Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020, April 23). YOLOv8: Optimal Speed and Accuracy of Object Detection. Retrieved from https://arxiv.org/abs/2004.10934.

[4] Chao, Y.-W., Vijayanarasimhan, S., Seybold, B., Ross, D. A., Deng, J., & Sukthankar, R. (2018). Rethinking the Faster R-CNN architecture for temporal action localization. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1130–1139). IEEE.

[5] Solawetz, J., Nelson, J., & Sahoo, S. (2020, May 21). How to Train YOLOv4 on a Custom Dataset. Retrieved from https://blog.roboflow.com/training-yolov4-on-a-custom-dataset/.

[6] Vinh, T. Q., & Anh, N. T. N. (2020). Real-Time Face Mask Detector Using YOLOv3 Algorithm and Haar Cascade Classifier. In 2020 International Conference on Advanced Computing and Applications (ACOMP), Quy Nhon, Vietnam (pp. 146-149). Retrieved from https://doi.org/10.1109/ACOMP50827.2020.00029.

[7] World Health Organization (WHO). (2020, November 13). Technical specifications of personal protective equipment for COVID-19. Retrieved from https://www.who.int/publications/i/item/WHO-2019-nCoV-PPE\_specifications-2020.1.

[8] Zheng, Z., et al. (2020, September 16). Enhancing Geometric Factors in Model Learning and Inference for Object Detection and Instance Segmentation. Retrieved from https://arxiv.org/abs/2005.03572.