Springboard Construction

123 Anywhere St., Any City
+123-456-7890
www.spriingboardconstruction.com

Automatic Safety Gear Detection Model

May 2024

Presented By
Karma Gurung

Company Name

Table of Contents

Introduction

The improper wearing of safety gear in industrial construction environments is a common issue, posing a significant risk to worker safety and compliance. Despite the importance of safety gear, many workers fail to wear it properly, putting themselves and others at risk. This problem is particularly prevalent in industries where safety gear is not always mandatory, such as construction sites.

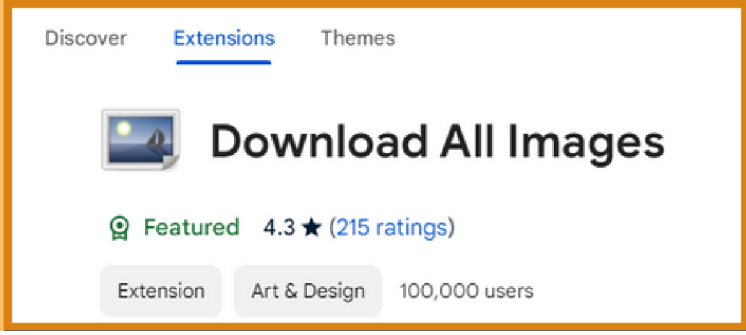
Model Overview

The developed model is an image processing system designed to analyze images captured in different environments and detect the presence and proper wearing of safety gear.

Tasks

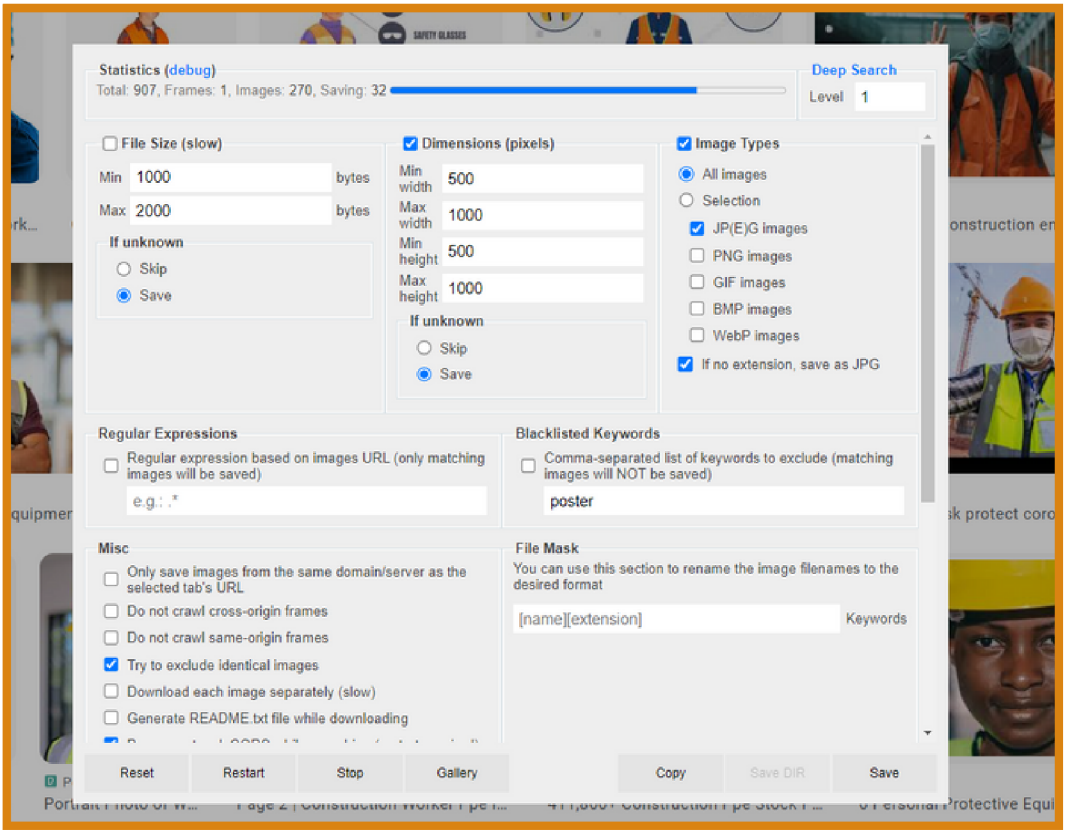
Develop a system to automatically detect safety gear usage in images captured at industrial sites, promoting worker safety and compliance with regulations.

Data Source

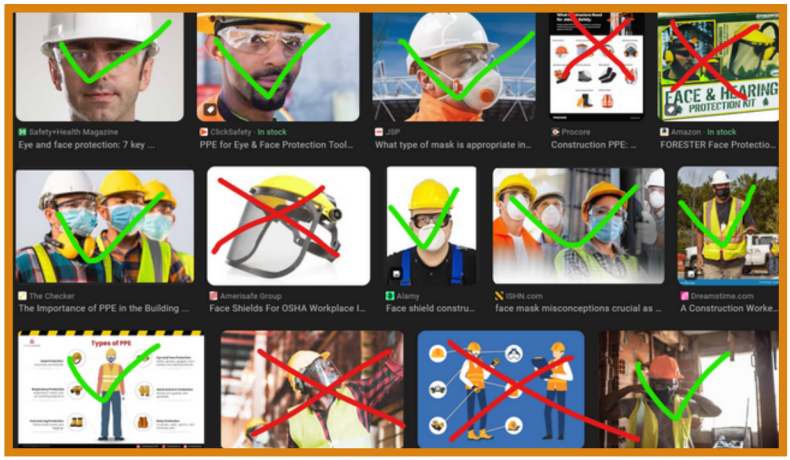


Used ‘Download all Images’ extension for download.

Collected Images



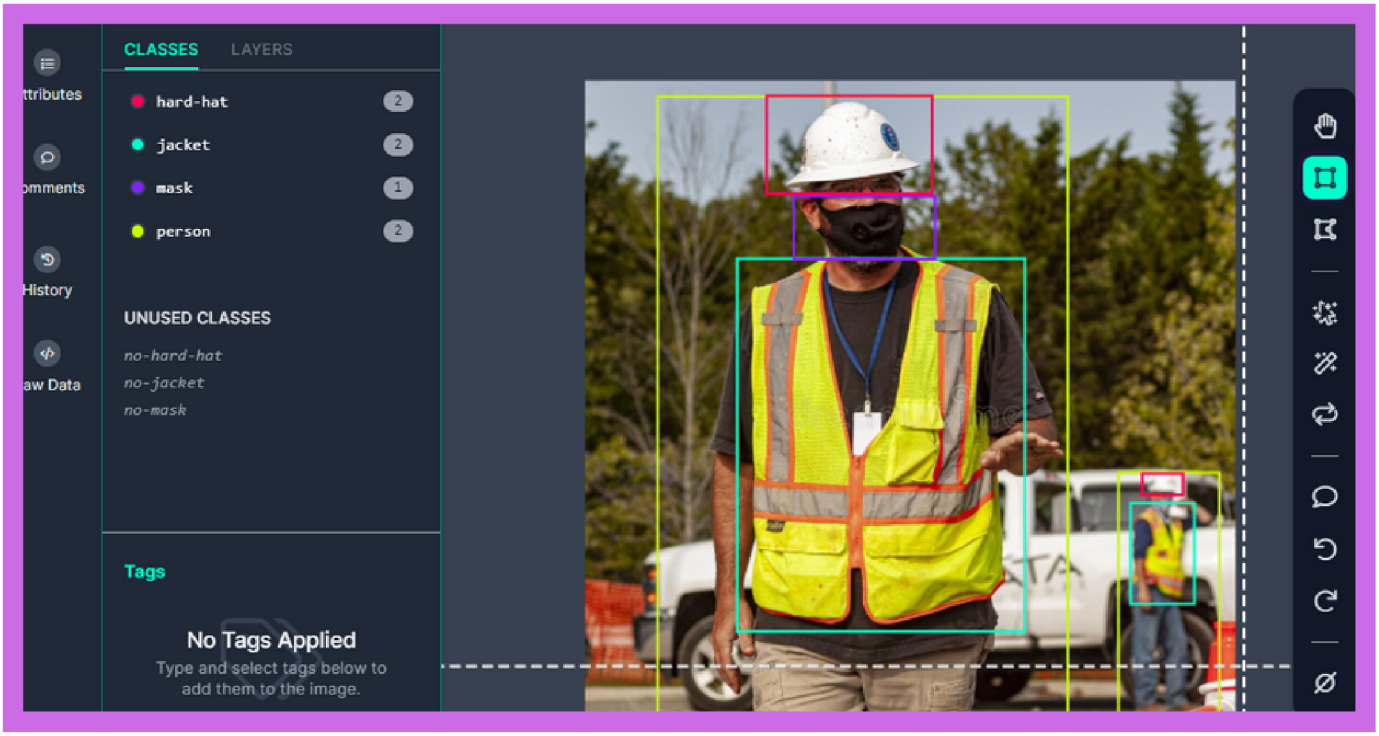
Filter only relevant images.



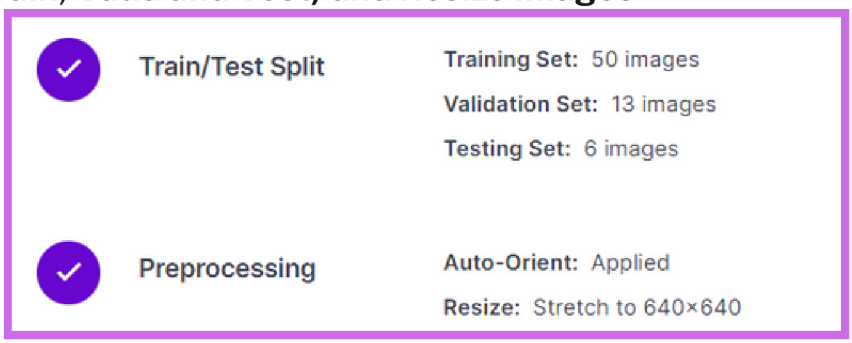
Prepare Data

roboflow

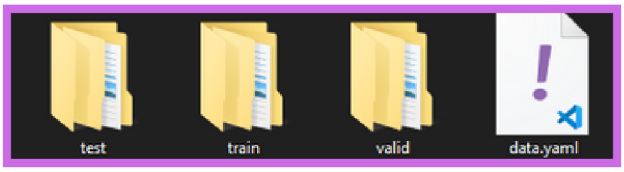
- Used ‘Roboflow’ web application.
- Annotations / Labelling using Roboflow web application.



- Split Data (Train, Valid and Test) and Resize images



- Export all Labeled Images to Google Colab for Training



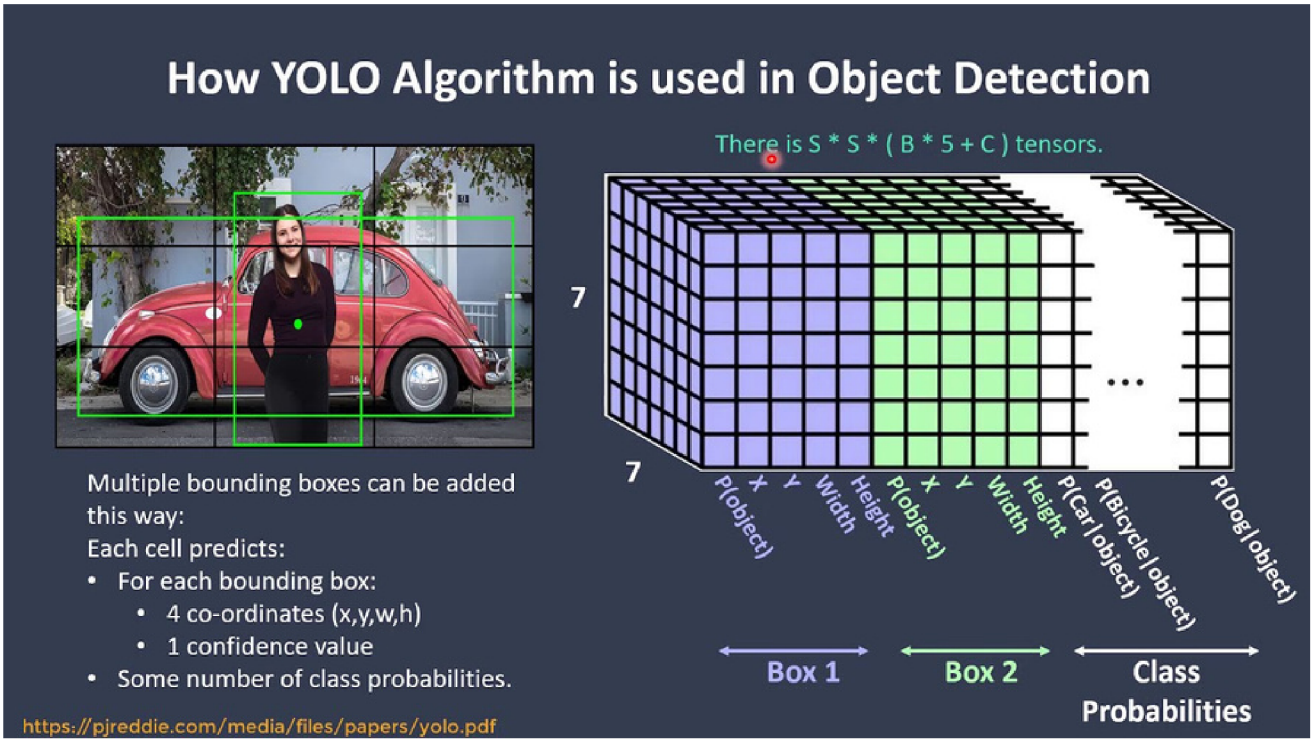
The Model

YOLO, which stands for **"You Only Look Once"**, is a popular deep learning algorithm for real-time object detection. It's known for its speed and efficiency. Here's a breakdown of what YOLO does:

- Object Detection: YOLO helps computers identify and locate objects in images and videos. This is useful for tasks like self-driving cars, security systems, and image analysis.

- Single-Stage Approach: Unlike some other methods, YOLO uses a single neural network to make predictions in one go. This makes it faster than some competing algorithms.

- Grid System and Bounding Boxes: YOLO divides an image into a grid and predicts for each cell if an object is present, the bounding box coordinates for that object (basically a box around it), and the class of the object (like car, person, etc.).



The Trainning

We choose Google Colab to train YOLO for object detection because of its ease of use, scalability, and cost-effectiveness. Colab provides pre-installed deep learning libraries, including TensorFlow, which supports YOLO. With Colab, we can quickly set up and start training our model without worrying about hardware or software configurations. Additionally, Colab offers access to high-performance computing resources, including GPUs, which can accelerate the training process. The cloud-based environment also allows us to collaborate with others and track our progress in real-time. Overall, Colab is an ideal choice for training YOLO models due to its flexibility and scalability.

We choose model ‘Yolov8n’ - ‘n’ means nano for small size

```
from ultralytics import YOLO
!yolo task=detect mode=train model=yolov8n.pt data= ../content/drive/MyDrive/ppe_data/dataset/data.yaml epochs=50 imgsz =640
```

```
Downloading https://github.com/ultralytics/assets/releases/download/v8.2.0/yolov8n.pt to 'yolov8n.pt'...
100% 6.23M/6.23M [00:00<00:00, 104MB/s]
Ultralytics YOLOv8.2.15 🚀 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
engine/trainer: task=detect, mode=train, model=yolov8n.pt, data=../content/drive/MyDrive/ppe_data/dataset/data.yaml, epochs=50, t
```

It took less than 2 min to run 50 epochs on their ‘Tesla T4’ GPU

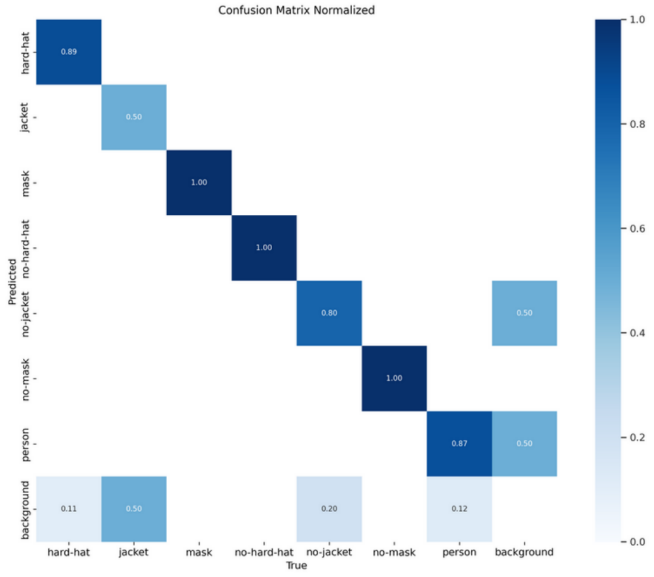
```
50 epochs completed in 0.033 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 6.3MB
Optimizer stripped from runs/detect/train/weights/best.pt, 6.3MB  our pre-train model only 6.3MB

Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.2.15 🚀 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 3007013 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  4.61it/s]
                   all         13         62      0.936      0.891       0.93      0.575
             hard-hat         13          9          1      0.972      0.995      0.617
                jacket         13          4          1      0.666      0.798      0.443
                  mask         13          4      0.981          1      0.995      0.727
          no-hard-hat         13          7      0.964          1      0.995      0.584
             no-jacket         13         10      0.801        0.8      0.836      0.462
               no-mask         13         12          1      0.988      0.995      0.452
                person         13         16      0.807      0.812      0.896      0.744
Speed: 0.2ms preprocess, 3.1ms inference, 0.0ms loss, 3.2ms postprocess per image
Results saved to runs/detect/train
💡 Learn more at https://docs.ultralytics.com/modes/train
```

Model Evaluation

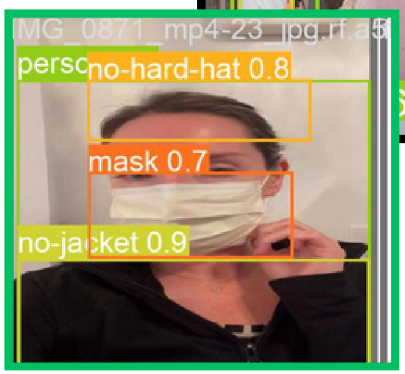
- Confusion Matrix Normalized
- Results
- Images (train and validation)

Confusion Matrix: Distribution of true positives, false positives, true negatives, and false negatives for each safety gear type.

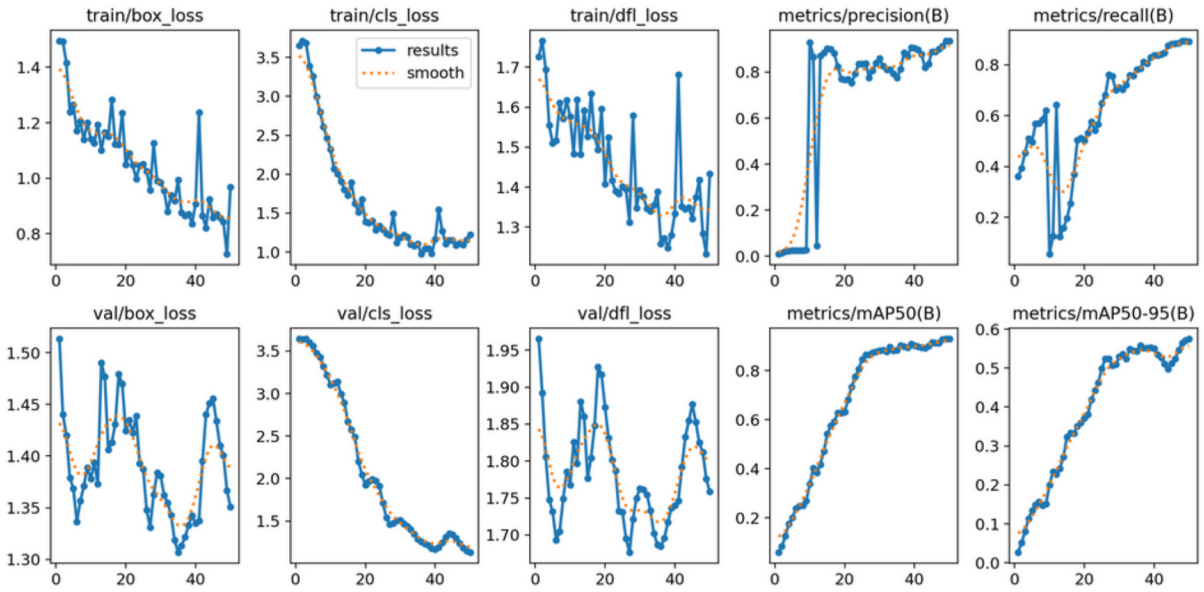


Train

Validation



Results



Accuracy: The ability to correctly detect the presence and proper wearing of safety gear. Target accuracy is at least 80%.

Final Product

Automatic Safety Gear Detection App

Scans Upper body of a person to check PPE: Hard-hat, mask and safety jacket.

***Actual photo and UI design of the product**

Conclusion

This project successfully developed an automatic safety gear detection model for industrial settings. The model demonstrates promising results in accuracy and can be deployed as a part of a comprehensive safety system.

Further work can refine the model for real-time deployment, address limitations regarding image quality and occluded equipment, and explore integration with existing security infrastructure.